

Server Administration Guide

服务器管理指南

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/overview.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/overview.adoc)

Keycloak is a single sign on solution for web apps and RESTful web services. The goal of Keycloak is to make security simple so that it is easy for application developers to secure the apps and services they have deployed in their organization. Security features that developers normally have to write for themselves are provided out of the box and are easily tailor able to the individual requirements of your organization. Keycloak provides customizable user interfaces for login, registration, administration, and account management. You can also use Keycloak as an integration platform to hook it into existing LDAP and Active Directory servers. You can also delegate authentication to third party identity providers like Facebook and Google.

Key斗篷是针对 Web 应用程序和 RESTful Web 服务的解决方案的单一标志。Keycloak 的目标是使安全变得简单，这样应用程序开发人员就可以很容易地保护他们在组织中部署的应用程序和服务。开发人员通常必须为自己编写的安全特性是开箱即用的，可以很容易地根据组织的个别需求进行调整。Key斗篷提供用于登录、注册、管理和帐户管理的可定制用户界面。您还可以使用 Keycon 作为集成平台，将其挂接到现有的 LDAP 和 ActiveDirectory 服务器。你也可以将身份认证委托给第三方身份认证提供商，比如 Facebook 和 Google。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/overview/features.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/overview/features.adoc)

Keycloak provides the following features:

钥匙斗篷提供了以下特性：

- Single-Sign On and Single-Sign Out for browser applications.

浏览器应用程序的单点登录和单点退出。

- OpenID Connect support.

支持 OpenID 连接。 (2)

- OAuth 2.0 support.

2.0支持。 (2)
- SAML support.

SAML 支持。
- Identity Brokering - Authenticate with external OpenID Connect or SAML Identity Providers.

身份代理-使用外部 OpenID 连接或 SAML 身份提供者进行身份验证。
- Social Login - Enable login with Google, GitHub, Facebook, Twitter, and other social networks.

社交登录-允许使用 Google, GitHub, Facebook, Twitter 和其他社交网络登录。
- User Federation - Sync users from LDAP and Active Directory servers.

用户联盟-从 LDAP 和 ActiveDirectory 服务器同步用户。
- Kerberos bridge - Automatically authenticate users that are logged-in to a Kerberos server.

Kerberos 桥——自动验证登录到 Kerberos 服务器的用户。
- Admin Console for central management of users, roles, role mappings, clients and configuration.

管理控制台，用于集中管理用户、角色、角色映射、客户端和配置。
- Account Management console that allows users to centrally manage their account.

帐户管理控制台，允许用户集中管理他们的帐户。
- Theme support - Customize all user facing pages to integrate with your applications and branding.

主题支持-自定义所有用户面临的网页，以集成与您的应用程序和品牌。
- Two-factor Authentication - Support for TOTP/HOTP via Google Authenticator or FreeOTP.

双因素身份验证-通过 Google Authenticator 或 FreeOTP 支持 TOTP/HOTP。
- Login flows - optional user self-registration, recover password, verify email, require password update, etc.

登录流程-可选的用户自我注册，恢复密码，验证电子邮件，要求密码更新等。
- Session management - Admins and users themselves can view and manage user sessions.

会话管理——管理员和用户本身可以查看和管理用户会话。
- Token mappers - Map user attributes, roles, etc. how you want into tokens and statements.

令牌映射器-如何映射用户属性、角色等到令牌和语句。
- Not-before revocation policies per realm, application and user.

每个领域、应用程序和用户的 Not-before 撤销策略。
- CORS support - Client adapters have built-in support for CORS.

CORS 支持-客户端适配器内置了对 CORS 的支持。

- Service Provider Interfaces (SPI) - A number of SPIs to enable customizing various aspects of the server. Authentication flows, user federation providers, protocol mappers and many more.
服务提供者接口(Service Provider Interfaces, SPI)——许多 SPI 支持定制服务器的各个方面。身份验证流、用户联合提供程序、协议映射程序等等。
- Client adapters for JavaScript applications, WildFly, JBoss EAP, Tomcat, Jetty, Spring, etc.
用于 JavaScript 应用程序、WildFly、JBoss EAP、Tomcat、Jetty、Spring 等的客户端适配器。
- Supports any platform/language that has an OpenID Connect Relying Party library or SAML 2.0 Service Provider library.
支持任何具有 OpenID 连接依赖方库或 SAML 2.0 服务提供者库的平台/语言。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/overview/how.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/overview/how.adoc)

Keycloak is a separate server that you manage on your network. Applications are configured to point to and be secured by this server. Keycloak uses open protocol standards like [OpenID Connect](#) (<https://openid.net/connect/>) or [SAML 2.0](#) (<http://saml.xml.org/saml-specifications>) to secure your applications. Browser applications redirect a user's browser from the application to the Keycloak authentication server where they enter their credentials. This redirection is important because users are completely isolated from applications and applications never see a user's credentials. Applications instead are given an identity token or assertion that is cryptographically signed. These tokens can have identity information like username, address, email, and other profile data. They can also hold permission data so that applications can make authorization decisions. These tokens can also be used to make secure invocations on REST-based services.

密钥斗篷是您在网络上管理的一个单独的服务器。应用程序被配置为指向此服务器并由此服务器保护。

Keycloak 使用 OpenID Connect 或 SAML 2.0 等开放协议标准来保护应用程序的安全。浏览器应用程序将用户的浏览器从应用程序重定向到键斗身份验证服务器，在该服务器上输入用户的凭据。这种重定向非常重要，因为用户与应用程序完全隔离，而应用程序永远不会看到用户的凭据。相反，应用程序会得到一个经过加密签名的标识令牌或断言。这些令牌可以包含身份信息，如用户名、地址、电子邮件和其他配置文件数据。它们还可以保存权限数据，以便应用程序可以做出授权决策。这些令牌还可用于对基于 REST 的服务进行安全调用。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/overview/concepts.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/overview/concepts.adoc)

Consider these core concepts and terms before attempting to use Keycloak to secure your web applications and REST services.

在尝试使用 Keycover 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

Users are entities that are able to log into your system. They can have attributes associated with themselves like email, username, address, phone number, and birthday. They can be assigned group membership and have specific roles assigned to them.

用户是能够登录到您的系统的实体。它们可以具有与自身相关联的属性，如电子邮件、用户名、地址、电话号码和生日。可以为它们分配组成员资格，并为它们分配特定的角色。

authentication 认证

The process of identifying and validating a user.

识别和验证用户的过程。

authorization 授权

The process of granting access to a user.

授予用户访问权限的过程。

credentials 证件

Credentials are pieces of data that Keycloak uses to verify the identity of a user. Some examples are passwords, one-time-passwords, digital certificates, or even fingerprints.

凭据是 Keycover 用来验证用户身份的数据片段。有些例子是密码、一次性密码、数字证书，甚至是指纹。

roles 角色

Roles identify a type or category of user. Admin, user, manager, and employee are all typical roles that may exist in an organization. Applications often assign access and permissions to specific roles rather than individual users as dealing with users can be too fine-grained and hard to manage.

角色标识用户的类型或类别。管理员、用户、经理和员工都是组织中可能存在的典型角色。应用程序通常将访问权限和权限分配给特定的角色，而不是分配给单个用户，因为处理用户可能过于细粒度和难以管理。

user role mapping 用户角色映射用户角色映射

A user role mapping defines a mapping between a role and a user. A user can be associated with zero or more roles. This role mapping information can be encapsulated into tokens and assertions so that applications can decide access permissions on various resources they manage.

用户角色映射定义角色和用户之间的映射。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对其管理的各种资源的访问权限。

composite roles 混合角色

A composite role is a role that can be associated with other roles. For example a `superuser` composite role could be associated with the `sales-admin` and `order-entry-admin` roles. If a user is mapped to the `superuser` role they also inherit the `sales-admin` and `order-entry-admin` roles.

复合角色是可以与其他角色关联的角色。例如，超级用户组合角色可以与 `sales-admin` 和 `order-entry-admin` 角色相关联。如果用户被映射到超级用户角色，那么它们还将继承 `sales-admin` 和 `order-entry-admin` 角色。

groups 团体

Groups manage groups of users. Attributes can be defined for a group. You can map roles to a group as well. Users that become members of a group inherit the attributes and role mappings that group defines.

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定义的属性和角色映射。

realms 国度

A realm manages a set of users, credentials, roles, and groups. A user belongs to and logs into a realm. Realms are isolated from one another and can only manage and authenticate the users that they control.

领域管理一组用户、凭据、角色和组。用户属于并登录到领域。领域彼此隔离，只能管理和验证它们控制的用户。

clients 客户

Clients are entities that can request Keycloak to authenticate a user. Most often, clients are applications and services that want to use Keycloak to secure themselves and provide a single sign-on solution. Clients can also be entities that just want to request identity information or an access token so that they can securely invoke other services on the network that are secured by Keycloak.

客户端是可以请求 Keycloak 对用户进行身份验证的实体。大多数情况下，客户是那些希望使用 Keycloak 保护自己并提供单点登录解决方案的应用程序和服务。客户端也可以是只想请求身份信息或访问令牌的实体，这样它们就可以安全地调用网络上其他服务，而这些服务是由钥匙斗篷保护的。

client adapters 客户端适配器

Client adapters are plugins that you install into your application environment to be able to communicate and be secured by Keycloak. Keycloak has a number of adapters for different platforms that you can download. There are also third-party adapters you can get for environments that we don't cover.

客户端适配器是安装到应用程序环境中的插件，它们能够通信，并且由“钥匙斗篷”保护。Key斗篷有许多适用于不同平台的适配器，您可以下载。您还可以为我们不涉及的环境获得第三方适配器。

consent 同意

Consent is when you as an admin want a user to give permission to a client before that client can participate in the authentication process. After a user provides their credentials, Keycloak will pop up a screen identifying the client requesting a login and what identity information is requested of the user. User can decide whether or not to grant the request.

同意是指作为管理员的您希望用户在客户端可以参与身份验证过程之前向该客户端授予权限。在用户提供了他们的凭证之后，Keycloak将弹出一个屏幕，标识请求登录的客户端以及请求用户的身份信息。用户可以决定是否授予请求。

client scopes 客户端范围

When a client is registered, you must define protocol mappers and role scope mappings for that client. It is often useful to store a client scope, to make creating new clients easier by sharing some common settings. This is also useful for requesting some claims or roles to be conditionally based on the value of `scope` parameter. Keycloak provides the concept of a client scope for this.

注册客户端时，必须为该客户端定义协议映射器和角色范围映射。存储客户端范围通常很有用，可以通过共享一些公共设置来简化创建新客户端的工作。这对于请求基于作用域参数值的有条件的声明或角色也很有用。为此提供了客户端作用域的概念。

client role 客户角色

Clients can define roles that are specific to them. This is basically a role namespace dedicated to the client.

客户端可以定义特定于它们的角色。这基本上是专用于客户端的角色命名空间。

identity token 身份令牌

A token that provides identity information about the user. Part of the OpenID Connect specification.

提供用户身份信息的令牌。OpenIDConnect 规范的一部分。

access token 访问令牌

A token that can be provided as part of an HTTP request that grants access to the service being invoked on. This is part of the OpenID Connect and OAuth 2.0 specification.

可以作为 HTTP 请求的一部分提供的令牌，该请求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 断言

Information about a user. This usually pertains to an XML blob that is included in a SAML authentication response that provided identity metadata about an authenticated user.

关于用户的信息。这通常适用于 SAML 身份验证响应中包含的 XML blob，SAML 身份验证响应提供了关于经过身份验证的用户的身份元数据。

service account 服务帐户

Each client has a built-in service account which allows it to obtain an access token.

每个客户端都有一个内置的服务帐户，允许它获取访问令牌。

direct grant 直接拨款

A way for a client to obtain an access token on behalf of a user via a REST invocation.

客户端通过 REST 调用代表用户获取访问令牌的一种方法。

protocol mappers 协议映射器

For each client you can tailor what claims and assertions are stored in the OIDC token or SAML assertion. You do this per client by creating and configuring protocol mappers.

对于每个客户机，您可以调整 OIDC 令牌或 SAML 断言中存储的声明和断言。通过创建和配置协议映射程序，可以为每个客户机执行此操作。

session 会议

When a user logs in, a session is created to manage the login session. A session contains information like when the user logged in and what applications have participated within single-sign on during that session. Both admins and users can view session information.

当用户登录时，将创建一个会话来管理登录会话。会话包含诸如用户何时登录以及在该会话期间哪些应用程序在单点登录中参与等信息。管理员和用户都可以查看会话信息。

user federation provider 用户联合提供程序

Keycloak can store and manage users. Often, companies already have LDAP or Active Directory services that store user and credential information. You can point Keycloak to validate credentials from those external stores and pull in identity information.

钥匙斗篷可以存储和管理用户。通常，公司已经有存储用户和凭证信息的 LDAP 或 ActiveDirectory 服务。您可以指向“钥匙斗篷”来验证来自这些外部存储的凭据，并获取标识信息。

identity provider 身份证明文件提供者

An identity provider (IDP) is a service that can authenticate a user. Keycloak is an IDP.

身份提供者(IDP)是一种可以验证用户身份的服务。

identity provider federation 身份提供者联盟

Keycloak can be configured to delegate authentication to one or more IDPs. Social login via Facebook or Google+ is an example of identity provider federation. You can also hook Keycloak to delegate authentication to any other OpenID Connect or SAML 2.0 IDP.

可以配置密钥斗篷将身份验证委托给一个或多个 IDP。通过 Facebook 或 Google + 进行社交登录是身份提供者联盟的一个例子。您还可以钩住 Keycloak，将身份验证委托给任何其他 OpenID Connect 或 SAML 2.0 IDP。

identity provider mappers 身份提供者地图

When doing IDP federation you can map incoming tokens and assertions to user and session attributes. This helps you propagate identity information from the external IDP to your client requesting authentication.

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。这有助于您将身份信息从外部 IDP 传播到请求身份验证的客户端。

required actions 必须采取的行动

Required actions are actions a user must perform during the authentication process. A user will not be able to complete the authentication process until these actions are complete. For example, an admin may schedule users to reset their passwords every month. An `update password` required action would be set for all these users.

所需操作是用户在身份验证过程中必须执行的操作。在完成这些操作之前，用户将无法完成身份验证过程。例如，管理员可能会安排用户每月重置他们的密码。将为所有这些用户设置更新密码所需的操作。

authentication flows 身份验证流

Authentication flows are work flows a user must perform when interacting with certain aspects of the system. A login flow can define what credential types are required. A registration flow defines what profile information a user must enter and whether something like reCAPTCHA must be used to filter out bots. Credential reset flow defines what actions a user must do before they can reset their password.

身份验证流是用户在与系统的某些方面进行交互时必须执行的工作流。登录流可以定义所需的凭据类型。注册流定义用户必须输入的配置文件信息，以及是否必须使用 reCAPTCHA 之类的东西来过滤机器人。凭据重置流定义用户在重置密码之前必须执行的操作。

events 事件

Events are audit streams that admins can view and hook into.

事件是管理员可以查看和挂钩到的审计流。

themes 主题

Every screen provided by Keycloak is backed by a theme. Themes define HTML templates and stylesheets which you can override as needed.

钥匙斗篷提供的每个屏幕都有一个主题支持。主题定义您可以根据需要重写的 HTML 模板和样式表。

[Report an issue](#) 报告一个问题

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/assembly-creating-first-admin.adoc)

After installing Keycloak, you need an administrator account that can act as a *super admin* with full permissions to manage Keycloak. With this account, you can log in to the Keycloak Admin Console where you create realms and users and register applications that are secured by Keycloak.

在安装 Keycloak 之后，您需要一个管理员帐户，该帐户可以充当具有完全权限的超级管理员来管理 Keycloak。有了这个帐户，您就可以登录到用于创建领域和用户以及注册由 Keycloak 保护的应用程序的钥匙斗篷管理控制台。

Creating the account on the local host 在本地主机上创建帐户

If your server is accessible from `localhost`, perform these steps.

如果服务器可以从本地主机访问，请执行以下步骤。

Procedure 程序

1. In a web browser, go to the `http://localhost:8080` URL.

在浏览器中，访问 `http://localhost:8080` 的网址。

2. Supply a username and password that you can recall.

提供您能够回忆起来的用户名和密码。

Welcome page 欢迎页

Welcome to Keycloak

The screenshot shows the Keycloak welcome page. On the left, there is a form for creating an initial admin user. It includes fields for 'Username' and 'Password', and a 'Create' button. A message at the top of the form says: 'Please create an initial admin user to get started.' To the right of the form are four links: 'Documentation >' (with a document icon), 'Keycloak Project >' (with a hexagon icon), 'Mailing List >' (with an envelope icon), and 'Report an issue >' (with a bug icon). The 'Report an issue' link is highlighted with a blue border.

Creating the account remotely 远程创建帐户

If you cannot access the server from a `localhost` address or just want to start Keycloak from the command line, use the `KEYCLOAK_ADMIN` and `KEYCLOAK_ADMIN_PASSWORD` environment variables to create an initial admin account.

如果您无法从本地主机地址访问服务器，或者只是想从命令行启动 Keycon，请使用 `KEYCLOAK_ADMIN` 和 `KEYCLOAK_ADMIN_PASSWORD` 环境变量来创建一个初始管理帐户。

For example:

例如：

```
export KEYCLOAK_ADMIN=<username>
export KEYCLOAK_ADMIN_PASSWORD=<password>

bin/kc.[sh|bat] start
```

BASH

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/admin-console.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/admin-console.adoc)

Once you have an administrative account for the Admin Console, you can configure realms. A realm is a space where you manage objects, including users, applications, roles, and groups. A user belongs to and logs into a realm. One Keycloak deployment can define, store, and manage as many realms as there is space for in the database.

一旦您有了管理控制台的管理帐户，就可以配置领域。领域是您管理对象(包括用户、应用程序、角色和组)的空间。用户属于并登录到领域。一个 Keycon 部署可以定义、存储和管理尽可能多的领域，只要数据库中有足够的空间。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/realms/proc-using-admin-console.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/realms/proc-using-admin-console.adoc)

You configure realms and perform most administrative tasks in the Keycloak Admin Console.

您可以配置领域，并在 KeycloakAdminConsole 中执行大多数管理任务。

Prerequisites 先决条件

- You need an administrator account. See Creating the first administrator.

您需要一个管理员帐户。请参阅创建第一个管理员。

Procedure 程序

1. Go to the URL for the Admin Console.

转到管理控制台的 URL。

For example, for localhost, use this URL: <http://localhost:8080/admin/>

例如，对于 localhost，使用这个 URL: <http://localhost:8080/admin/>

Login page 登录页面

The screenshot shows a login form titled "Sign in to your account". It has two input fields: "Username or email" and "Password", both represented by empty text boxes. Below the password field is a blue "Sign In" button. The entire form is contained within a light gray box.

2. Enter the username and password you created on the Welcome Page or the `add-user-keycloak` script in the bin directory. This action displays the Admin Console.

输入您在欢迎页面上创建的用户名和密码，或者在 bin 目录中输入 add-user-keycloak 脚本。此操作显示管理控制台。

Admin Console 管理控制台

Master

Realm settings are settings that control the options for users, applications, roles, and more.

Master

Realm ID *	master
Display name	Keycloak
HTML Display name	<div class="kc-logo-text">Keycloak</div>
Frontend URL ⓘ	External requests
User-managed access ⓘ	<input type="checkbox"/> Off
User Profile Enabled ⓘ	<input type="checkbox"/> Off
Endpoints ⓘ	OpenID Endpoint Configuration ⓘ SAML 2.0 Identity Provider Metadata ⓘ

Save **Revert**

3. Note the menus and other options that you can use:

请注意您可以使用的菜单和其他选项:

- Click the menu labeled **Master** to pick a realm you want to manage or to create a new one.

单击标记为 Master 的菜单以选择要管理的域或创建新的域。

- Click the top right list to view your account or log out.

单击右上方的列表查看您的帐户或注销。

- Hover over a question mark ? icon to show a tooltip text that describes that field. The image above shows the tooltip in action.

在一个问号上徘徊? 图标以显示描述该字段的工具提示文本。上图显示了正在运行的工具提示。

- Click a question mark ? icon to show a tooltip text that describes that field. The image above shows the tooltip in action.

点击一个问号? 图标以显示描述该字段的工具提示文本。上图显示了正在运行的工具提示。

在管理控制台中，存在两种类型的领域：

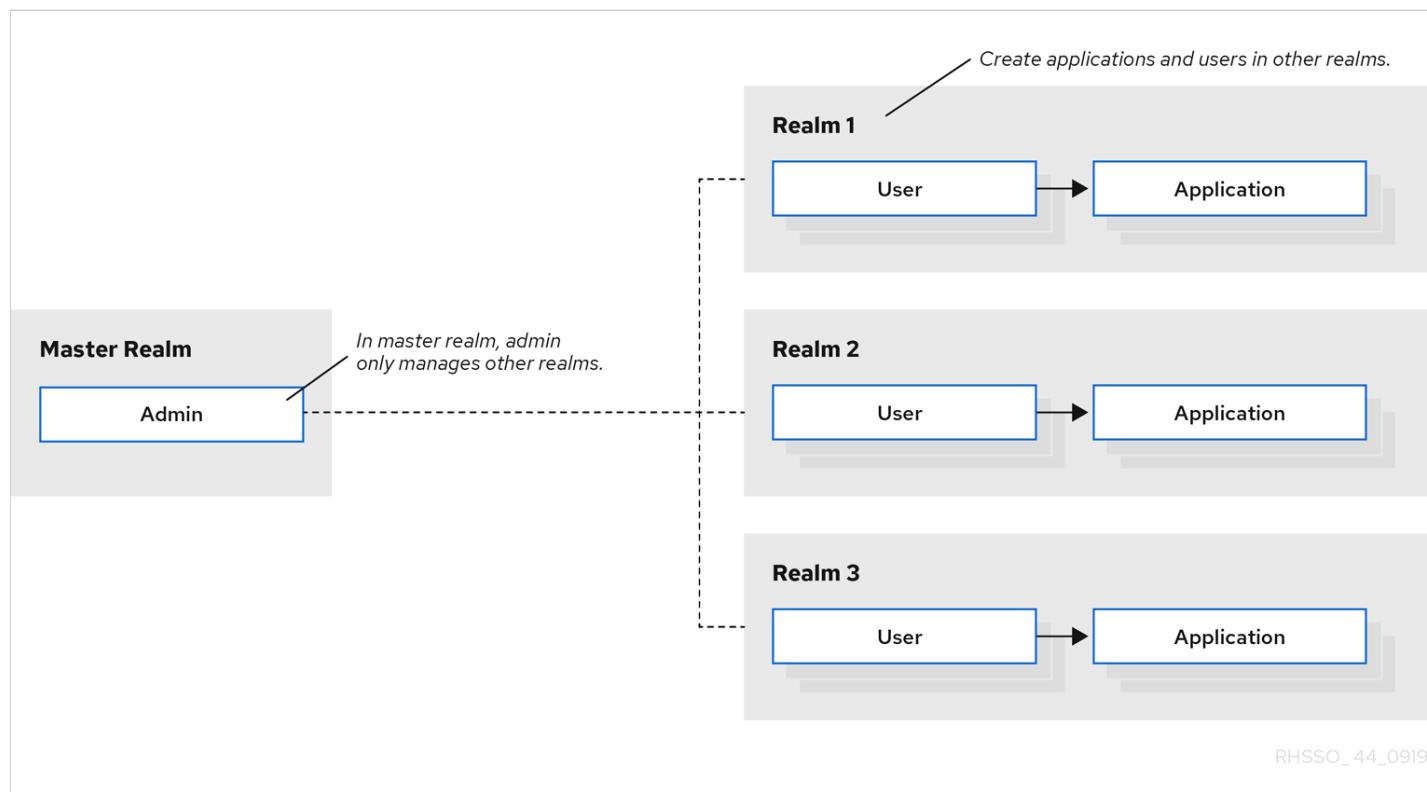
- **Master realm** - This realm was created for you when you first started Keycloak. It contains the administrator account you created at the first login. Use the *master* realm only to create and manage the realms in your system.

主领域-这个领域是在你第一次启动钥匙斗篷时为你创建的。它包含您在第一次登录时创建的管理员帐户。仅使用主域来创建和管理系统中的域。

- **Other realms** - These realms are created by the administrator in the master realm. In these realms, administrators manage the users in your organization and the applications they need. The applications are owned by the users.

其他领域——这些领域是由管理员在主领域中创建的。在这些领域中，管理员管理组织中的用户和他们需要的应用程序。应用程序由用户拥有。

Realms and applications 领域和应用



Realms are isolated from one another and can only manage and authenticate the users that they control. Following this security model helps prevent accidental changes and follows the tradition of permitting user accounts access to only those privileges and powers necessary for the successful completion of their current task.

领域彼此隔离，只能管理和验证它们控制的用户。遵循此安全模型有助于防止意外更改，并遵循只允许用户帐户访问成功完成当前任务所必需的特权和权限的传统。

Additional resources 额外资源

- See Dedicated Realm Admin Consoles if you want to disable the *master* realm and define administrator accounts within any new realm you create. Each realm has its own dedicated Admin Console that you can log into with local accounts.

如果要禁用主域并在创建的任何新域中定义管理员帐户，请参见专用域管理控制台。每个领域都有自己的专用管理控制台，您可以使用本地帐户登录该控制台。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/realms/proc-creating-a-realm.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priorty=3&description=File:%20server_admin/topics/realms/proc-creating-a-realm.adoc)

You create a realm to provide a management space where you can create users and give them permissions to use applications. At first login, you are typically in the *master* realm, the top-level realm from which you create other realms.

您创建一个领域来提供一个管理空间，您可以在其中创建用户并授予他们使用应用程序的权限。首先登录时，您通常处于主领域中，这是您创建其他领域的顶级领域。

When deciding what realms you need, consider the kind of isolation you want to have for your users and applications. For example, you might create a realm for the employees of your company and a separate realm for your customers. Your employees would log into the employee realm and only be able to visit internal company applications. Customers would log into the customer realm and only be able to interact with customer-facing apps.

在决定您需要什么领域时，请考虑您希望为您的用户和应用程序提供哪种隔离。例如，您可以为公司的雇员创建一个领域，为客户创建一个单独的领域。您的雇员将登录到雇员领域，并且只能访问公司内部的应用程序。客户将登录到客户领域，并且只能与面向客户的应用程序进行交互。

Procedure 程序

1. Point to the top of the left pane.

指向左窗格的顶部。

2. Click Create Realm.

单击“创建领域”。

Add realm menu 添加领域菜单

The screenshot shows the Keycloak Admin UI. On the left, a sidebar menu lists realms: 'Master' (selected), 'Master' (disabled), and 'Test'. Below the realms are links for 'Create Realm', 'Realm roles', 'Users', 'Groups', 'Sessions', 'Events', 'Configure', 'Realm settings' (selected), 'Authentication', 'Identity providers', and 'User federation'. The main content area is titled 'Create Realm' and contains tabs for 'Themes', 'Keys', 'Events', 'Localization', and 'Security defenses'. A large text input field is present, with placeholder text 'Enter realm name' and a note: 'The options for users, applications, roles, and groups in the current realm will be available here.' Below the input field are sections for 'Realm logo' (with a placeholder 'Keycloak'), 'Realm URL' (placeholder 'http://localhost:8080/auth'), and 'Realm public client' (checkbox). At the bottom, there are 'Save' and 'Cancel' buttons.

3. Enter a name for the realm.

输入领域的名称。

4. Click Create.

单击 Create。

Create realm 创建领域

Create realm

A realm manages a set of users, credentials, roles, and groups. A user belongs to and logs into a realm. Realms are isolated from one another and can only manage and authenticate the users that they control.

Resource file Drag a file here or browse to upload

1

Upload a JSON file

Realm name *

Enabled On

The current realm is now set to the realm you just created. You can switch between realms by clicking the realm name in the menu.

当前领域现在设置为您刚刚创建的领域。通过单击菜单中的领域名称，可以在领域之间切换。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/realms/ssl.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/realms/ssl.adoc)

Each realm has an associated SSL Mode, which defines the SSL/HTTPS requirements for interacting with the realm. Browsers and applications that interact with the realm honor the SSL/HTTPS requirements defined by the SSL Mode or they cannot interact with the server.

每个领域都有一个关联的 SSL 模式，它定义了与领域交互的 SSL/HTTPS 需求。与领域交互的浏览器和应用程序遵守由 SSL 模式定义的 SSL/HTTPS 要求，或者它们不能与服务器交互。

Procedure 程序

1. Click **Realm settings** in the menu.

单击菜单中的“域”设置。

2. Click the **General** tab.

单击“常规”选项卡。

General tab 普通标签

Master

Realm settings are settings that control the options for users, applications, roles, and groups in the current realm. [Learn more](#)

Action Enabled

General	Login	Email	Themes	Keys	Events	Localization	Security defens
Realm ID *	master						
Display name	Keycloak						
HTML Display name	<div class="kc-logo-text">Keycloak</div>						
Frontend URL							
Require SSL	External requests						
User-managed access	<input type="checkbox"/>	Off					
User Profile Enabled	<input type="checkbox"/>	Off					
Endpoints	OpenID Endpoint Configuration SAML 2.0 Identity Provider Metadata						
<input type="button" value="Save"/> <input type="button" value="Revert"/>							

3. Set **Require SSL** to one of the following SSL modes:

将 Require SSL 设置为以下 SSL 模式之一:

- **External requests** Users can interact with Keycloak without SSL so long as they stick to private IP addresses such as `localhost`, `127.0.0.1`, `10.x.x.x`, `192.168.x.x`, and `172.16.x.x`. If you try to access Keycloak without SSL from a non-private IP address, you will get an error.
外部请求用户可以在不使用 SSL 的情况下与 Keycloak 进行交互，只要他们使用的是诸如 localhost、127.0.0.1、10.x.x.x、192.168.x.x 和 172.16.x.x 等私有 IP 地址。如果您尝试从非私有 IP 地址访问没有 SSL 的钥匙斗篷，您将得到一个错误。
- **None** Keycloak does not require SSL. This choice applies only in development when you are experimenting and do not plan to support this deployment.
无钥匙斗篷不需要 SSL。这种选择仅适用于正在进行试验的开发阶段，并且不打算支持此部署。
- **All requests** Keycloak requires SSL for all IP addresses.
所有请求的钥匙斗篷需要 SSL 为所有 IP 地址。

[Edit this section](#) [Report an issue](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/realms/email.adoc)

[Report an issue](#) [报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/realms/email.adoc)

Keycloak sends emails to users to verify their email addresses, when they forget their passwords, or when an administrator needs to receive notifications about a server event. To enable Keycloak to send emails, you provide Keycloak with your SMTP server settings.

当用户忘记密码，或者当管理员需要接收服务器事件通知时，Keycloak 会向用户发送电子邮件以验证他们的电子邮件地址。若要启用“钥匙斗篷”发送电子邮件，请为“钥匙斗篷”提供 SMTP 服务器设置。

Procedure 程序

1. Click **Realm settings** in the menu.

单击菜单中的“域”设置。

2. Click the **Email** tab.

单击“电子邮件”选项卡。

Email tab 电子邮件标签

Master



Enabled

Action ▾

Realm settings are settings that control the options for users, applications, roles, and groups in the current realm. [Learn more](#)

< General Login Email Themes Keys Events Localization Security defens >

Template

From *

Sender email address

From display name ⓘ

Display name for Sender email address

Reply to

Reply to email address

Reply to display name ⓘ

Display name for "reply to" email address

Envelope from ⓘ

Sender envelope email address

Connection & Authentication

Host *

SMTP host

Port

SMTP port (defaults to 25)

Encryption

- Enable SSL
- Enable StartTLS

Authentication

Disabled

Save

Test connection

Revert

3. Fill in the fields and toggle the switches as needed.

填写字段并根据需要切换开关。

Template 模板

From 从

From denotes the address used for the From SMTP-Header for the emails sent.

From 表示用于发送电子邮件的 From SMTP-Header 的地址。

From display name 从显示名称

From display name allows to configure a user-friendly email address aliases (optional). If not set the plain **From** email address will be displayed in email clients.

从显示名称允许配置用户友好的电子邮件地址别名(可选)。如果没有设置纯从电子邮件地址将显示在电子邮件客户端。

Reply to 答复

Reply to denotes the address used for the **Reply-To** SMTP-Header for the mails sent (optional). If not set the plain **From** email address will be used.

“答复”表示用于“答复到 SMTP”的邮件头的地址(可选)。如果没有设置纯从电子邮件地址将被使用。

Reply to display name 答复显示名称

Reply to display name allows to configure a user-friendly email address aliases (optional). If not set the plain **Reply To** email address will be displayed.

答复显示名称允许配置用户友好的电子邮件地址别名(可选)。如果没有设置简单的回复电子邮件地址将显示。

Envelope from 信封来自

Envelope from denotes the Bounce Address (https://en.wikipedia.org/wiki/Bounce_address) used for the **Return-Path** SMTP-Header for the mails sent (optional).

信封从表示弹出地址用于返回路径 SMTP-头为邮件发送(可选)。

Connection & Authentication 连接及认证

Host 主持人

Host denotes the SMTP server hostname used for sending emails.

Host 表示用于发送电子邮件的 SMTP 服务器主机名。

Port 左舷

Port denotes the SMTP server port.

Port 表示 SMTP 服务器端口。

Encryption 加密

Tick one of these checkboxes to support sending emails for recovering usernames and passwords, especially if the SMTP server is on an external network. You will most likely need to change the **Port** to 465, the default port for SSL/TLS.

勾选其中一个复选框，以支持发送电子邮件恢复用户名和密码，尤其是在 SMTP 服务器位于外部网络上时。您很可能需要将 Port 更改为 465，这是 SSL/TLS 的默认端口。

Authentication 认证

Set this switch to **ON** if your SMTP server requires authentication. When prompted, supply the **Username** and **Password**. The value of the **Password** field can refer a value from an external vault.

如果 SMTP 服务器需要身份验证，则将此开关设置为 ON。当出现提示时，提供用户名和密码。Password 字段的值可以引用来自外部保险库的值。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/realms/themes.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/realms/themes.adoc)

For a given realm, you can change the appearance of any UI in Keycloak by using themes.

对于给定的领域，您可以通过使用主题来更改 Keycloak 中任何 UI 的外观。

Procedure 程序

1. Click **Realm setting** in the menu.

单击菜单中的“域”设置。

2. Click the **Themes** tab.

单击“主题”选项卡。

Themes tab 主题标签

The screenshot shows the Keycloak Admin UI with the 'Master' tab selected. At the top, there is a toggle switch labeled 'Enabled' which is set to 'On'. To the right of the switch is a 'Action' button with a dropdown arrow. Below the header, there is a navigation bar with tabs: 'gin', 'Email', 'Themes' (which is highlighted in blue), 'Keys', 'Events', 'Localization', 'Security defenses', and 'Sessions'. The main content area contains four dropdown menus for selecting themes:

- Login theme**: Set to 'keycloak'.
- Account theme**: Set to 'keycloak'.
- Admin console theme**: Set to 'keycloak'.
- Email theme**: Set to 'Select a theme'.

At the bottom of the form are two buttons: a blue 'Save' button and a white 'Revert' button.

3. Pick the theme you want for each UI category and click **Save**.

为每个 UI 类别选择所需的主题，然后单击“保存”。

Login theme 登录主题

Username password entry, OTP entry, new user registration, and other similar screens related to login.

用户名密码输入、OTP 输入、新用户注册和其他与登录相关的类似屏幕。

Account theme 帐户主题

Each user has a User Account Management UI.

每个用户都有一个用户帐户管理 UI。

Admin console theme 管理控制台主题

The skin of the Keycloak Admin Console.

钥匙斗篷管理控制台的外壳。

Email theme 电子邮件主题

Whenever Keycloak has to send out an email, it uses templates defined in this theme to craft the email.

每当钥匙斗篷必须发送电子邮件，它使用定义在这个主题的模板，以精心制作的电子邮件。

Additional resources 额外资源

- The [Server Developer Guide](https://www.keycloak.org/docs/latest/server_development/) (https://www.keycloak.org/docs/latest/server_development/) describes how to create a new theme or modify existing ones.

服务器开发人员指南描述了如何创建新主题或修改现有主题。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/realm/proc-configuring-internationalization.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/realm/proc-configuring-internationalization.adoc)

Every UI screen is internationalized in Keycloak. The default language is English, but you can choose which locales you want to support and what the default locale will be.

在 Keycloak，每个用户界面都是国际化的。默认语言是英语，但是您可以选择要支持哪种语言环境以及默认语言环境是什么。

Procedure 程序

1. Click **Realm Settings** in the menu.

单击菜单中的“域设置”。

2. Click the **Localization** tab.

单击“本地化”选项卡。

3. Enable **Internationalization**.

启用国际化。

4. Select the languages you will support.

选择您将支持的语言。

Localization tab 本地化选项卡

The screenshot shows the 'Localization' tab selected in the Keycloak admin interface. At the top, there is a toggle switch labeled 'Enabled' which is turned on. Below it, a section titled 'Internationalization' is shown with a 'Enabled' toggle switch. Under 'Supported locales', there is a list of languages: English, Français, and Italiano. A 'Select locales' button is available to add more. Under 'Default locale', a dropdown menu is set to 'English'. A note below states: 'The next time a user logs in, that user can choose a language on the login page to use for the login screens, Account Console, and Admin Console.'

The next time a user logs in, that user can choose a language on the login page to use for the login screens, Account Console, and Admin Console.

下次用户登录时，该用户可以在登录页面上选择用于登录屏幕、帐户控制台和管理控制台的语言。

Additional resources 额外资源

- The [Server Developer Guide](https://www.keycloak.org/docs/latest/server_development/) (https://www.keycloak.org/docs/latest/server_development/) explains how you can offer additional languages. All internationalized texts which are provided by the theme can be overwritten by realm-specific texts on the **Localization** tab.

服务器开发人员指南解释了如何提供其他语言。主题提供的所有国际化文本都可以被 Localization 选项卡上的特定于领域的文本覆盖。

User locale selection 用户区域设置选择

A locale selector provider suggests the best locale on the information available. However, it is often unknown who the user is. For this reason, the previously authenticated user's locale is remembered in a persisted cookie.

区域设置选择器提供程序建议可用信息的最佳区域设置。但是，通常不知道用户是谁。由于这个原因，先前经过身份验证的用户的区域设置被记录在持久化 Cookie 中。

The logic for selecting the locale uses the first of the following that is available:

选择区域设置的逻辑使用以下第一种逻辑：

- User selected - when the user has selected a locale using the drop-down locale selector

用户选择-当用户使用下拉式区域设置选择器选择区域设置时

- User profile - when there is an authenticated user and the user has a preferred locale set
用户配置文件——当有一个经过身份验证的用户并且该用户有一个首选区域设置时
- Client selected - passed by the client using for example ui_locales parameter
客户端选择-由客户端使用例如 ui_locales 参数传递
- Cookie - last locale selected on the browser
Cookie-在浏览器上选择的最后一个区域设置
- Accepted language - locale from Accept-Language header
从 Accept-Language 头中接受语言区域设置
- Realm default
领域默认
- If none of the above, fall back to English
如果以上都不是，那就回到英语上来

When a user is authenticated an action is triggered to update the locale in the persisted cookie mentioned earlier. If the user has actively switched the locale through the locale selector on the login pages the users locale is also updated at this point.

当用户通过身份验证时，会触发一个操作来更新前面提到的持久化 cookie 中的区域设置。如果用户通过登录页面上的区域设置选择器主动切换了区域设置，那么此时用户的区域设置也会被更新。

If you want to change the logic for selecting the locale, you have an option to create custom `LocaleSelectorProvider`. For details, please refer to the [Server Developer Guide](#) (https://www.keycloak.org/docs/latest/server_development/#_locale_selector).

如果要更改用于选择区域设置的逻辑，可以选择创建自定义 LocaleSelectorProvider。有关详细信息，请参阅服务器开发人员指南。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/login-settings.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/login-settings.adoc)

Keycloak includes several built-in login page features.

Key斗篷包括几个内置的登录页面特性。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/login-settings/forgot-password.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/login-settings/forgot-password.adoc)

If you enable `Forgot password`, users can reset their login credentials if they forget their passwords or lose their OTP generator.

如果启用忘记密码，用户可以在忘记密码或丢失 OTP 生成器时重置其登录凭据。

Procedure 程序

1. Click **Realm settings** in the menu.

单击菜单中的“域”设置。

2. Click the **Login** tab.

单击 Login 选项卡。

Login tab 登录标签

The screenshot shows the Keycloak administration interface. The left sidebar is titled 'Master' and contains links for Manage, Clients, Client scopes, Realm roles, Users, Groups, Sessions, Events, Configure, Realm settings (which is selected), Authentication, Identity providers, and User federation. The main content area is titled 'Master' and describes realm settings. It features a navigation bar with tabs: General (disabled), Login (selected), Email, Themes, Keys, and Events. The 'Login screen customization' section includes three toggle switches: 'User registration' (On), 'Forgot password' (Off), and 'Remember me' (On). The 'Email settings' section includes four toggle switches: 'Email as username' (Off), 'Login with email' (On), 'Duplicate emails' (Off), and 'Verify email' (Off). The 'User info settings' section includes one toggle switch: 'Edit username' (Off).

3. Toggle **Forgot password** to ON.

忘记开机密码。

A **Forgot Password?** link displays in your login pages.

在您的登录页面中显示一个“忘记密码?”链接。

Forgot password link 忘了密码链接

The screenshot shows a login form titled "Sign in to your account". It includes fields for "Username or email" and "Password", both represented by empty input boxes. Below these fields is a blue button labeled "Sign In". To the right of the "Sign In" button is a blue link labeled "Forgot Password?". At the bottom of the form, there is a grey bar containing the text "New user? [Register](#)".

4. Click this link to bring users where they can enter their username or email address and receive an email with a link to reset their credentials.

点击这个链接可以让用户输入他们的用户名或电子邮件地址，然后收到一封带有重置证书链接的电子邮件。

Forgot password page 忘了密码页

The screenshot shows a "Forgot Your Password?" page. It features a field for "Username or email" with an empty input box. Below this is a blue button labeled "Submit". At the bottom of the page, there is a grey bar containing the text "Enter your username or email address and we will send you instructions on how to create a new password." Above this bar, there is a blue link labeled "« Back to Login".

The text sent in the email is configurable. See [Server Developer Guide](#) (https://www.keycloak.org/docs/latest/server_development/) for more information.

电子邮件中发送的文本是可配置的。有关详细信息，请参阅服务器开发人员指南。

When users click the email link, Keycloak asks them to update their password, and if they have set up an OTP generator, Keycloak asks them to reconfigure the OTP generator. Depending on security requirements of your organization, you may not want users to reset their OTP generator through email.

当用户单击电子邮件链接时，钥匙斗篷要求他们更新密码，如果他们已经设置了 OTP 生成器，钥匙斗篷要求他们重新配置 OTP 生成器。根据组织的安全需求，您可能不希望用户通过电子邮件重置其 OTP 生成器。

To change this behavior, perform these steps:

要更改此行为，请执行以下步骤：

Procedure 程序

1. Click **Authentication** in the menu.

单击菜单中的“身份验证”。

2. Click the **Flows** tab.

单击 Flows 选项卡。

3. Select the **Reset Credentials** flow.

选择“重置凭据”流。

Reset credentials flow 重置凭据流

Reset credentials

Default

 Built-in

Action ▾



Add step

Add sub-flow

Steps

Requirement



Choose User

Required



Send Reset Email

Required



Reset Password

Required



Reset - Conditional OTP

Conditional



Flow to determine if the OTP
should be reset or not. Set to
REQUIRED to force.

Condition - user
configured

Required



Reset OTP

Required



If you do not want to reset the OTP, set the **Reset OTP** requirement to **Disabled**.

如果不想重置 OTP，请将“重置 OTP”要求设置为“禁用”。

4. Click **Authentication** in the menu.

单击菜单中的“身份验证”。

5. Click the **Required actions** tab.

单击“必要操作”选项卡。

6. Ensure **Update Password** is enabled.

确保启用更新密码。

Required Actions 所需行动

Authentication

Authentication is the area where you can configure and manage different credential types. [Learn more](#)

Required actions	Enabled	Set as default action
Configure OTP	<input checked="" type="checkbox"/> On	<input type="checkbox"/> Off
Terms and Conditions	<input type="checkbox"/> Off	<input checked="" type="checkbox"/> Disabled off
Update Password	<input checked="" type="checkbox"/> On	<input type="checkbox"/> Off
Update Profile	<input checked="" type="checkbox"/> On	<input type="checkbox"/> Off
Verify Email	<input checked="" type="checkbox"/> On	<input type="checkbox"/> Off
Delete Account	<input type="checkbox"/> Off	<input checked="" type="checkbox"/> Disabled off
Update User Locale	<input checked="" type="checkbox"/> On	<input type="checkbox"/> Off
Webauthn Register Passwordless	<input checked="" type="checkbox"/> On	<input checked="" type="checkbox"/> On
Webauthn Register	<input checked="" type="checkbox"/> On	<input checked="" type="checkbox"/> On
Verify Profile	<input type="checkbox"/> Off	<input checked="" type="checkbox"/> Disabled off

[Edit this section](#) 编辑这部分

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/login-settings/remember-me.adoc)

[Report an issue](#) 报告一个问题

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/login-settings/remember-me.adoc)

A logged-in user closing their browser destroys their session, and that user must log in again. You can set Keycloak to keep the user's login session open if that user clicks the *Remember Me* checkbox upon login. This action turns the login cookie from a session-only cookie to a persistence cookie.

登录用户关闭浏览器会销毁其会话，并且该用户必须再次登录。如果用户在登录时单击“记住我”复选框，则可以设置 Keycloak 以保持用户的登录会话处于打开状态。此操作将登录 Cookie 从仅会话 Cookie 转换为持久性 Cookie。

Procedure 程序

1. Click Realm settings in the menu.

单击菜单中的“域”设置。

2. Click the Login tab.

单击 Login 选项卡。

3. Toggle the Remember Me switch to On.

将“记住我”切换到“打开”。

Login tab 登录标签

The screenshot shows the 'Master' realm settings page with the 'Login' tab selected. At the top, there is a status bar with a toggle switch labeled 'Enabled' and an 'Action' dropdown. Below the status bar, a navigation bar includes tabs for General, Login (which is selected and highlighted in blue), Email, Themes, Keys, Events, Localization, and Security defens. The main content area is divided into sections: 'Login screen customization', 'Email settings', and 'User info settings'. In the 'Login screen customization' section, three switches are shown: 'User registration' (On), 'Forgot password' (Off), and 'Remember me' (On). In the 'Email settings' section, four switches are shown: 'Email as username' (Off), 'Login with email' (On), 'Duplicate emails' (Off), and 'Verify email' (Off). In the 'User info settings' section, one switch is shown: 'Edit username' (Off). The 'Remember me' switch in the 'Login screen customization' section is explicitly highlighted as being turned on.

When you save this setting, a `remember me` checkbox displays on the realm's login page.

当您保存此设置时，将在域的登录页面上显示一个“记住我”复选框。

Remember Me 记住我

Sign in to your account

Username or email

Password

Remember me

Sign In

New user? [Register](#)

[Edit this section 编辑这部分](#)
 (https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/login-settings/acr-to-loa-mapping.adoc)

[Report an issue 报告一个问题](#)
 (https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/login-settings/acr-to-loa-mapping.adoc)

In the login settings of a realm, you can define which **Authentication Context Class Reference (ACR)** value is mapped to which **Level of Authentication (LoA)**. The ACR can be any value, whereas the LoA must be numeric. The acr claim can be requested in the **claims** or **acr_values** parameter sent in the OIDC request and it is also included in the access token and ID token. The mapped number is used in the authentication flow conditions.

在域的登录设置中，可以定义将哪个身份验证上下文类引用(ACR)值映射到哪个身份验证级别(LoA)。ACR 可以是任何值，而 LoA 必须是数值。Acr 声明可以在 OIDC 请求中发送的声明或 acr_value 参数中请求，它也包含在访问令牌和 ID 令牌中。在身份验证流条件中使用映射的编号。

Mapping can be also specified at the client level in case that particular client needs to use different values than realm. However, a best practice is to stick to realm mappings.

也可以在客户端级别指定映射，以防特定客户端需要使用与域不同的值。然而，最佳实践是坚持领域映射。

ACR to LoA Mapping	silver	1	-
	gold	2	-
	ACR	LoA	+
	Save	Cancel	

For further details see Step-up Authentication and [the official OIDC specification](#)

(https://openid.net/specs/openid-connect-core-1_0.html#acrSemantics).

有关详细信息，请参阅升级身份验证和官方 OIDC 规范。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/login-settings/update-email-workflow.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/login-settings/update-email-workflow.adoc)

With this workflow, users will have to use an UPDATE_EMAIL action to change their own email address.

使用此工作流，用户将必须使用 UPDATE _ EMAIL 操作来更改自己的电子邮件地址。

The action is associated with a single email input form. If the realm has email verification disabled, this action will allow to update the email without verification. If the realm has email verification enabled, the action will send an email update action token to the new email address without changing the account email. Only the action token triggering will complete the email update.

该操作与单个电子邮件输入表单相关联。如果领域已禁用电子邮件验证，此操作将允许更新电子邮件而无需验证。如果领域已启用电子邮件验证，则操作将向新电子邮件地址发送电子邮件更新操作令牌，而不更改帐户电子邮件。只有操作令牌触发将完成电子邮件更新。

Applications are able to send their users to the email update form by leveraging UPDATE_EMAIL as an AIA (Application Initiated Action).

通过利用 UPDATE _ EMAIL 作为 AIA (应用程序启动操作)，应用程序能够将其用户发送到电子邮件更新表单。

 Please note that Update Email Workflow support is in development. Use this feature experimentally.

请注意，更新电子邮件工作流支持正在开发中。

 If you enable this feature and you are migrating from a previous version, enable the **Update Email** required action in your realms. Otherwise, users cannot update their email addresses.

如果您启用了此特性，并且正在从以前的版本迁移，请在您的领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。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/realmkeys.adoc)

[Report an issue 报告一个问题](#)

The authentication protocols that are used by Keycloak require cryptographic signatures and sometimes encryption. Keycloak uses asymmetric key pairs, a private and public key, to accomplish this.

钥匙斗篷使用的身份验证协议需要加密签名，有时还需要加密。Key斗篷使用非对称密钥对(私钥和公钥)来实现这一点。

Keycloak has a single active keypair at a time, but can have several passive keys as well. The active keypair is used to create new signatures, while the passive keypair can be used to verify previous signatures. This makes it possible to regularly rotate the keys without any downtime or interruption to users.

一次只有一个主动密钥对，但也可以有几个被动密钥。主动密钥对用于创建新的签名，而被动密钥对可用于验证以前的签名。这样就可以定期旋转按键，而不会造成用户停机或中断。

When a realm is created a key pair and a self-signed certificate is automatically generated.

在创建领域时，将自动生成一个密钥对和一个自签名证书。

Procedure 程序

1. Click **Realm settings** in the menu.

单击菜单中的“域”设置。

2. Click **Keys**.

按键。

3. Select **Passive keys** from the filter dropdown to view passive keys.

从筛选器下拉菜单中选择被动键，查看被动键。

4. Select **Disabled keys** from the filter dropdown to view disabled keys.

从筛选器下拉列表中选择禁用键以查看禁用键。

A keypair can have the status **Active**, but still not be selected as the currently active keypair for the realm. The selected active pair which is used for signatures is selected based on the first key provider sorted by priority that is able to provide an active keypair.

密钥对可以具有状态为 Active，但仍然不能被选择为该领域的当前活动密钥对。根据按优先级排序的第一个密钥提供程序选择用于签名的选定的活动密钥对，该密钥提供程序能够提供活动密钥对。

Rotating keys 旋转钥匙

We recommend that you regularly rotate keys. To do so, start by creating new keys with a higher priority than the existing active keys. Or create new keys with the same priority and making the previous keys passive.

我们建议您定期轮换按键。为此，首先创建优先级高于现有活动键的新键。或者创建具有相同优先级的新键，并使前面的键成为被动的。

Once new keys are available all new tokens and cookies will be signed with the new keys. When a user authenticates to an application the SSO cookie is updated with the new signature. When OpenID Connect tokens are refreshed new tokens are signed with the new keys. This means that over time all cookies and tokens will use the new keys and after a while the old keys can be removed.

一旦新的密钥可用，所有新的令牌和 cookie 将用新的密钥签名。当用户对应用程序进行身份验证时，SSO cookie 将使用新签名进行更新。当 OpenIDConnect 令牌被刷新时，新的令牌将使用新的密钥进行签名。这意味着，随着时间的推移，所有 cookie 和令牌将使用新的密钥，并在一段时间后，旧的密钥可以删除。

The frequency of deleting old keys is a tradeoff between security and making sure all cookies and tokens are updated. Consider creating new keys every three to six months and deleting old keys one to two months after you create the new keys. If a user was inactive in the period between the new keys being added and the old keys being removed, that user will have to re-authenticate.

删除旧密钥的频率是在安全性和确保更新所有 cookie 和令牌之间进行权衡的结果。考虑每三到六个月创建一个新键，并在创建新键后一到两个月删除旧键。如果用户在添加新密钥和删除旧密钥之间的期间处于非活动状态，则该用户必须重新进行身份验证。

Rotating keys also applies to offline tokens. To make sure they are updated, the applications need to refresh the tokens before the old keys are removed.

旋转键也适用于脱机令牌。为了确保它们得到更新，应用程序需要在删除旧键之前刷新标记。

Adding a generated keypair 添加生成的密钥对

Procedure 程序

1. Select the realm in the Admin Console.

在管理控制台中选择领域。

2. Click **Realm settings** in the menu.

单击菜单中的“域”设置。

3. Click the **Keys** tab.

单击“键”选项卡。

4. Click the **Providers** tab.

单击 Provider 选项卡。

5. Click **Add provider** and select **rsa-generated**.

单击“添加提供程序”并选择 rsa 生成的。

6. Enter a number in the **Priority** field. This number determines if the new key pair becomes the active key pair.

在“优先级”字段中输入一个数字。这个数字决定新密钥对是否成为活动密钥对。

7. Select a value for **AES Key size**.

为 AES 密钥大小选择一个值。

8. Click **Save**.

单击“保存”。

This action will generate a new keypair including a self-signed certificate.

此操作将生成一个包含自签名证书的新密钥对。

Changing the priority for a provider will not cause the keys to be re-generated, but if you want to change the keysize you can edit the provider and new keys will be generated.

更改提供程序的优先级不会导致重新生成密钥，但是如果希望更改密钥大小，则可以编辑提供程序并生成新密钥。

Adding an existing keypair and certificate 添加现有密钥对和证书

To add a keypair and certificate obtained elsewhere select **Providers** and choose **rsa** from the dropdown. You can change the priority to make sure the new keypair becomes the active keypair.

要添加密钥对和从其他地方获得的证书，请选择 Provider 并从下拉列表中选择 rsa。您可以更改优先级，以确保新密钥对成为活动密钥对。

Prerequisites 先决条件

- A private key file. The file must be PEM formatted.

私钥文件。该文件必须是 PEM 格式的。

Procedure 程序

1. Select the realm in the Admin Console.

在管理控制台中选择领域。

2. Click **Realm settings**.

单击“域设置”。

3. Click the **Keys** tab.

单击“键”选项卡。

4. Click the **Providers** tab.

单击 Provider 选项卡。

5. Click Add provider and select rsa.

单击“添加提供程序”并选择“rsa”。

6. Enter a number in the Priority field. This number determines if the new key pair becomes the active key pair.

在“优先级”字段中输入一个数字。这个数字决定新密钥对是否成为活动密钥对。

7. Click Browse... beside Private RSA Key to upload the private key file.

单击 Browse... 在 Private RSA Key 旁边上传私钥文件。

8. If you have a signed certificate for your private key, click Browse... beside X509 Certificate to upload the certificate file. Keycloak automatically generates a self-signed certificate if you do not upload a certificate.

如果您的私钥有已签名的证书，请单击 X509 证书旁边的 Browse... 以上载证书文件。如果您不上载证书，Keycloak 会自动生成一个自签名的证书。

9. Click Save.

单击“保存”。

Loading keys from a Java Keystore 从 Java 密钥存储加载密钥

To add a keypair and certificate stored in a Java Keystore file on the host select Providers and choose java-keystore from the dropdown. You can change the priority to make sure the new keypair becomes the active keypair.

要在主机上添加存储在 JavaKeystore 文件中的密钥对和证书，请选择 Provider 并从下拉列表中选择 Java-Keystore。您可以更改优先级，以确保新密钥对成为活动密钥对。

For the associated certificate chain to be loaded it must be imported to the Java Keystore file with the same Key Alias used to load the keypair.

要加载关联的证书链，必须将其导入到 JavaKeystore 文件中，并使用与加载密钥对相同的密钥别名。

Procedure 程序

1. Select the realm in the Admin Console.

在管理控制台中选择领域。

2. Click Realm settings in the menu.

单击菜单中的“域”设置。

3. Click the Keys tab.

单击“键”选项卡。

4. Click the Providers tab.

单击 Provider 选项卡。

5. Click Add provider and select java-keystore.

单击 Add Provider 并选择 java-keystore。

6. Enter a number in the **Priority** field. This number determines if the new key pair becomes the active key pair.

在“优先级”字段中输入一个数字。这个数字决定新密钥对是否成为活动密钥对。

7. Enter a value for **Keystore**.

输入 Keystore 的值。

8. Enter a value for **Keystore Password**.

输入 Keystore Password 的值。

9. Enter a value for **Key Alias**.

输入密钥别名的值。

10. Enter a value for **Key Password**.

输入密码键值。

11. Click **Save**.

单击“保存”。

Making keys passive 把钥匙变成被动的

Procedure 程序

1. Select the realm in the Admin Console.

在管理控制台中选择领域。

2. Click **Realm settings** in the menu.

单击菜单中的“域”设置。

3. Click the **Keys** tab.

单击“键”选项卡。

4. Click the **Providers** tab.

单击 Provider 选项卡。

5. Click the provider of the key you want to make passive.

单击要被动设置的密钥的提供程序。

6. Toggle Active to Off.

切换激活状态到关闭状态。

7. Click **Save**.

单击“保存”。

Disabling keys 关闭钥匙

Procedure 程序

1. Select the realm in the Admin Console.

在管理控制台中选择领域。

2. Click **Realm settings** in the menu.

单击菜单中的“域”设置。

3. Click the **Keys** tab.

单击“键”选项卡。

4. Click the **Providers** tab.

单击 Provider 选项卡。

5. Click the provider of the key you want to make passive.

单击要被动设置的密钥的提供程序。

6. Toggle **Enabled** to **Off**.

启用关闭切换。

7. Click **Save**.

单击“保存”。

Compromised keys 钥匙坏了

Keycloak has the signing keys stored just locally and they are never shared with the client applications, users or other entities. However, if you think that your realm signing key was compromised, you should first generate new keypair as described above and then immediately remove the compromised keypair.

钥匙斗篷拥有仅存储在本地的签名密钥，它们从不与客户端应用程序、用户或其他实体共享。但是，如果您认为您的领域签名密钥被破坏，您应该首先生成新的密钥对，如上所述，然后立即删除被破坏的密钥对。

Alternatively, you can delete the provider from the **Providers** table.

或者，可以从 Provider 表中删除提供程序。

Procedure 程序

1. Click **Clients** in the menu.

单击菜单中的 Clients。

2. Click **security-admin-console**.

单击 security-admin-sole。

3. Scroll down to the **Capability config** section.

向下滚动到“能力”配置部分。

4. Fill in the **Admin URL** field.

填写 AdminURL 字段。

5. Click the **Advanced** tab.

单击“高级”选项卡。

6. Click **Set to now** in the **Revocation** section.

在“撤销”部分中单击“现在设置”。

7. Click **Push**.

点击推。

Pushing the not-before policy ensures that client applications do not accept the existing tokens signed by the compromised key. The client application is forced to download new key pairs from Keycloak also so the tokens signed by the compromised key will be invalid.

推行 not-before 策略可以确保客户端应用程序不会接受由受损密钥签名的现有令牌。客户端应用程序还必须从 Keycloak 下载新的密钥对，因此由受损密钥签名的令牌将是无效的。

REST and confidential clients must set **Admin URL** so Keycloak can send clients the pushed not-before policy request.



REST 和机密客户端必须设置管理 URL，这样 Keycloak 就可以向客户端发送推送的 not-before 策略请求。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/user-federation.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/user-federation.adoc)

Organizations can have databases containing information, passwords, and other credentials. Typically, you cannot migrate existing data storage to a Keycloak deployment so Keycloak can federate existing external user databases. Keycloak supports LDAP and Active Directory, but you can also code extensions for any custom user database by using the Keycloak User Storage SPI.

组织可以拥有包含信息、密码和其他凭据的数据库。通常，您不能将现有数据存储迁移到 Keycloak 部署中，以便 Keycloak 可以联合现有的外部用户数据库。Keycloak 支持 LDAP 和 Active Directory，但是您也可以使用密钥斗篷用户存储 SPI 为任何自定义用户数据库编写扩展代码。

When a user attempts to log in, Keycloak examines that user's storage to find that user. If Keycloak does not find the user, Keycloak iterates over each User Storage provider for the realm until it finds a match. Data from the external data storage then maps into a standard user model the Keycloak runtime consumes. This user model then maps to OIDC token claims and SAML assertion attributes.

当一个用户试图登录时，“钥匙斗篷”检查该用户的存储空间以找到该用户。如果没有找到用户，则 Keycloak 环会遍历领域的每个用户存储提供程序，直到找到匹配项。然后，将来自外部数据存储的数据映射到 Keycloak 运行时使用的标准用户模型。然后，此用户模型映射到 OIDC 令牌声明和 SAML 断言属性。

External user databases rarely have the data necessary to support all the features of Keycloak, so the User Storage Provider can opt to store items locally in Keycloak user data storage. Providers can import users locally and sync periodically with external data storage. This approach depends on the capabilities of the provider and the configuration of the provider. For example, your external user data storage may not support OTP. The OTP can be handled and stored by Keycloak, depending on the provider.

外部用户数据库很少拥有支持“钥匙斗篷”所有特性所需的数据，因此用户存储提供程序可以选择将项存储在 Keycloak 用户数据存储本地。提供程序可以在本地导入用户，并定期与外部数据存储同步。此方法取决于提供程序的功能和提供程序的配置。例如，您的外部用户数据存储可能不支持 OTP。根据提供程序的不同，OTP 可以由钥匙斗篷处理和存储。

Adding a provider 添加提供程序

To add a storage provider, perform the following procedure:

若要添加存储提供程序，请执行以下过程：

Procedure 程序

1. Click **User Federation** in the menu.

在菜单中单击“用户联盟”。

User federation 用户联盟

The screenshot shows the Keycloak User Federation page. On the left is a sidebar with a dropdown set to "Master" and links for Manage, Clients, Client scopes, Realm roles, Users, Groups, Sessions, and Events. The main content area has a heading "User federation" with a sub-instruction: "User federation provides access to external databases and directories, such as LDAP and Active Directory. [Learn more](#)". Below this is a message: "To get started, select a provider from the list below." Underneath is a section titled "Add providers" with two cards: "Add Kerberos providers" and "Add Ldap providers".

2. Select the provider type card from the listed cards.

从列出的卡中选择提供程序类型卡。

Keycloak brings you to that provider's configuration page.

钥匙斗篷将您带到该提供程序的配置页面。

Dealing with provider failures 处理提供程序故障

If a User Storage Provider fails, you may not be able to log in and view users in the Admin Console. Keycloak does not detect failures when using a Storage Provider to look up a user, so it cancels the invocation. If you have a Storage Provider with a high priority that fails during user lookup, the login or user query fails with an exception and will not fail over to the next configured provider.

如果用户存储提供程序失败，则可能无法在管理控制台中登录和查看用户。在使用存储提供程序查找用户时，键隐形不会检测失败，因此它会取消调用。如果具有高优先级的存储提供程序在用户查找期间失败，则登录或用户查询将异常失败，并且不会故障转移到下一个配置的提供程序。

Keycloak searches the local Keycloak user database first to resolve users before any LDAP or custom User Storage Provider. Consider creating an administrator account stored in the local Keycloak user database in case of problems connecting to your LDAP and back ends.

在任何 LDAP 或自定义用户存储提供程序之前，键斗首先搜索本地键斗用户数据库以解析用户。如果连接到 LDAP 和后端时出现问题，可以考虑创建存储在本地 Keycon 用户数据库中的管理员帐户。

Each LDAP and custom User Storage Provider has an `enable` toggle on its Admin Console page. Disabling the User Storage Provider skips the provider when performing queries, so you can view and log in with user accounts in a different provider with lower priority. If your provider uses an `import` strategy and is disabled, imported users are still available for lookup in read-only mode.

每个 LDAP 和自定义用户存储提供程序在其管理控制台页上都有一个启用开关。禁用用户存储提供程序会在执行查询时跳过提供程序，因此您可以查看并使用优先级较低的不同提供程序中的用户帐户登录。如果提供程序使用导入策略并禁用该策略，则导入的用户仍然可以在只读模式下进行查找。

When a Storage Provider lookup fails, Keycloak does not fail over because user databases often have duplicate usernames or duplicate emails between them. Duplicate usernames and emails can cause problems because the user loads from one external data store when the admin expects them to load from another data store.

当存储提供程序查找失败时，“钥匙斗篷”不会故障转移，因为用户数据库之间通常有重复的用户名或重复的电子邮件。重复的用户名和电子邮件可能会导致问题，因为当管理员期望用户从另一个数据存储加载时，用户从一个外部数据存储加载。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/user-federation/ldap.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/user-federation/ldap.adoc)

Keycloak includes an LDAP/AD provider. You can federate multiple different LDAP servers in one Keycloak realm and map LDAP user attributes into the Keycloak common user model.

密钥斗篷包括 LDAP/AD 提供程序。您可以将多个不同的 LDAP 服务器联合到一个 Keycloak 领域中，并将 LDAP 用户属性映射到 Keycloak 公共用户模型中。

By default, Keycloak maps the username, email, first name, and last name of the user account, but you can also configure additional mappings. Keycloak's LDAP/AD provider supports password validation using LDAP/AD protocols and storage, edit, and synchronization modes.

默认情况下，Keycloak 映射用户帐户的用户名、电子邮件、名和姓，但是您也可以配置其他映射。Keycloak 的 LDAP/AD 提供程序支持使用 LDAP/AD 协议以及存储、编辑和同步模式进行密码验证。

Configuring federated LDAP storage 配置联合 LDAP 存储

Procedure 程序

1. Click User Federation in the menu.

在菜单中单击“用户联盟”。

User federation 用户联盟

The screenshot shows the Keycloak User federation configuration page. On the left, a sidebar lists various management options: Master, Manage, Clients, Client scopes, Realm roles, Users, Groups, Sessions, and Events. The 'Master' option is currently selected. The main content area is titled 'User federation' and contains the following text: 'User federation provides access to external databases and directories, such as LDAP and Active Directory.' Below this is a note: 'To get started, select a provider from the list below.' At the bottom of the main area are two buttons: 'Add Kerberos providers' and 'Add Ldap providers'. Both buttons feature a small icon of a server or database.

2. Click Add LDAP providers.

单击“添加 LDAP 提供程序”。

Keycloak brings you to the LDAP configuration page.

钥匙斗篷将您带到 LDAP 配置页面。

Storage mode 存储模式

Keycloak imports users from LDAP into the local Keycloak user database. This copy of the user database synchronizes on-demand or through a periodic background task. An exception exists for synchronizing passwords. Keycloak never imports passwords. Password validation always occurs on the LDAP server.

Keycon 将用户从 LDAP 导入到本地 Keycon 用户数据库中。此用户数据库副本可以根据需要或通过周期性的后台任务进行同步。存在同步密码的异常。密钥斗篷从不导入密码。密码验证始终在 LDAP 服务器上进行。

The advantage of synchronization is that all Keycloak features work efficiently because any required extra per-user data is stored locally. The disadvantage is that each time Keycloak queries a specific user for the first time, Keycloak performs a corresponding database insert.

同步的优点是所有 Keycloak 特性都能高效工作，因为任何所需的每个用户额外数据都存储在本地。缺点是，每次当 Keycon 第一次查询特定用户时，它都会执行相应的数据库插入操作。

You can synchronize the import with your LDAP server. Import synchronization is unnecessary when LDAP mappers always read particular attributes from the LDAP rather than the database.

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

You can use LDAP with Keycloak without importing users into the Keycloak user database. The LDAP server backs up the common user model that the Keycloak runtime uses. If LDAP does not support data that a Keycloak feature requires, that feature will not work. The advantage of this approach is that you do not have the resource usage of importing and synchronizing copies of LDAP users into the Keycloak user database.

您可以将 LDAP 与钥匙斗篷一起使用，而无需将用户导入到钥匙斗篷用户数据库中。LDAP 服务器将备份 Keycloak 运行时使用的公共用户模型。如果 LDAP 不支持钥匙斗篷特性所需的数据，那么该特性将无法工作。这种方法的优点是不需要将 LDAP 用户的副本导入并同步到 Keycloak 用户数据库中。

The **Import Users** switch on the LDAP configuration page controls this storage mode. To import users, toggle this switch to ON.

LDAP 配置页上的 Import Users 开关控制此存储模式。要导入用户，请将此开关切换到 ON。



If you disable **Import Users**, you cannot save user profile attributes into the Keycloak database. Also, you cannot save metadata except for user profile metadata mapped to the LDAP. This metadata can include role mappings, group mappings, and other metadata based on the LDAP mappers' configuration.

如果禁用“导入用户”，则无法将用户配置文件属性保存到“钥匙斗篷”数据库中。另外，除了映射到 LDAP 的用户配置文件元数据之外，您不能保存元数据。该元数据可以包括角色映射、组映射和其他基于 LDAP 映射器配置的元数据。

When you attempt to change the non-LDAP mapped user data, the user update is not possible. For example, you cannot disable the LDAP mapped user unless the user's `enabled` flag maps to an LDAP attribute.

尝试更改非 LDAP 映射的用户数据时，无法进行用户更新。例如，除非用户的启用标志映射到 LDAP 属性，否则不能禁用 LDAP 映射用户。

Edit mode 编辑模式

Users and admins can modify user metadata, users through the Account Console, and administrators through the Admin Console. The `Edit Mode` configuration on the LDAP configuration page defines the user's LDAP update privileges.

用户和管理员可以通过帐户控制台修改用户元数据，用户可以通过帐户控制台修改用户，管理员可以通过管理控制台修改管理员。LDAP 配置页上的 Edit Mode 配置定义了用户的 LDAP 更新权限。

READONLY 无理取闹

You cannot change the username, email, first name, last name, and other mapped attributes. Keycloak shows an error anytime a user attempts to update these fields. Password updates are not supported.

不能更改用户名、电子邮件、名、姓和其他映射属性。每当用户尝试更新这些字段时，键隐形就会显示一个错误。不支持密码更新。

WRITABLE 可写

You can change the username, email, first name, last name, and other mapped attributes and passwords and synchronize them automatically with the LDAP store.

您可以更改用户名、电子邮件、名、姓以及其他映射属性和密码，并将它们与 LDAP 存储自动同步。

UNSYNCED 不同步

Keycloak stores changes to the username, email, first name, last name, and passwords in Keycloak local storage, so the administrator must synchronize this data back to LDAP. In this mode, Keycloak deployments can update user metadata on read-only LDAP servers. This option also applies when importing users from LDAP into the local Keycloak user database.

“钥匙斗篷”将用户名、电子邮件、名字、姓氏和密码的更改存储在 Keycloak 本地存储器中，因此管理员必须将这些数据同步回 LDAP。在这种模式下，Keycloak 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 Keycloak 用户数据库时，此选项也适用。



When Keycloak creates the LDAP provider, Keycloak also creates a set of initial LDAP mappers. Keycloak configures these mappers based on a combination of the **Vendor**, **Edit Mode**, and **Import Users** switches. For example, when edit mode is UNSYNCED, Keycloak configures the mappers to read a particular user attribute from the database and not from the LDAP server. However, if you later change the edit mode, the mapper's configuration does not change because it is impossible to detect if the configuration changes changed in UNSYNCED mode. Decide the **Edit Mode** when creating the LDAP provider. This note applies to **Import Users** switch also.

在创建 LDAP 提供程序时，Keycloak 还创建一组初始 LDAP 映射程序。Keycloak 根据供应商、编辑模式和导入用户交换机的组合来配置这些映射程序。例如，当编辑模式为 UNSYNCED 时，Keycloak 把映射器配置为从数据库而不是从 LDAP 服务器读取特定的用户属性。但是，如果稍后更改编辑模式，则映射器的配置不会更改，因为无法检测配置是否在 UNSYNCED 模式下更改。在创建 LDAP 提供程序时确定编辑模式。本说明也适用于导入用户开关。

Other configuration options 其他配置选项

Console Display Name 控制台显示名称

The name of the provider to display in the admin console.

要在管理控制台中显示的提供程序的名称。

Priority 优先权

The priority of the provider when looking up users or adding a user.

查找用户或添加用户时提供程序的优先级。

Sync Registrations 同步注册

Toggle this switch to **ON** if you want new users created by Keycloak added to LDAP.

如果希望向 LDAP 中添加由 Keycloak 创建的新用户，请将此开关切换到 ON。

Allow Kerberos authentication 允许 Kerberos 身份验证

Enable Kerberos/SPNEGO authentication in the realm with user data provisioned from LDAP. For more information, see the Kerberos section.

使用 LDAP 提供的用户数据在领域中启用 Kerberos/SPNEGO 身份验证。

Other options 其他选择

Hover the mouse pointer over the tooltips in the Admin Console to see more details about these options.

将鼠标指针悬停在管理控制台中的工具提示上，以查看有关这些选项的详细信息。

Connecting to LDAP over SSL 通过 SSL 连接到 LDAP

When you configure a secure connection URL to your LDAP store (for example, `ldaps://myhost.com:636`), Keycloak uses SSL to communicate with the LDAP server. Configure a truststore on the Keycloak server side so that Keycloak can trust the SSL connection to LDAP.

当您配置到 LDAP 存储的安全连接 URL (例如 `ldaps://myhost.com:636`) 时，Keycloak 使用 SSL 与 LDAP 服务器通信。在钥匙斗篷服务器端配置一个信任存储库，以便钥匙斗篷可以信任到 LDAP 的 SSL 连接。

Configure the global truststore for Keycloak with the Truststore SPI. For more information about configuring the global truststore, see the [Configuring a Truststore] (<https://www.keycloak.org/server/keycloak-truststore>) guide. If you do not configure the Truststore SPI, the truststore falls back to the default mechanism provided by Java, which can be the file supplied by the `javax.net.ssl.trustStore` system property or the cacerts file from the JDK if the system property is unset.

使用 Truststore SPI 配置 Keycloak 的全局信任存储区。有关配置全局信任存储的详细信息，请参阅[配置信任存储] (<https://www.keycloak.org/server/keycloak-Truststore>) 指南。如果您没有配置 Truststore SPI，那么信任存储将回到 Java 提供的默认机制，如果没有设置系统属性，那么默认机制可以是 `javax.net.ssl.truststore` 系统属性提供的文件，或者是来自 JDK 的 `cacerts` 文件。

The `Use Truststore SPI` configuration property, in the LDAP federation provider configuration, controls the truststore SPI. By default, Keycloak sets the property to `Only for ldaps`, which is adequate for most deployments. Keycloak uses the Truststore SPI if the connection URL to LDAP starts with `ldaps` only.

LDAP 联合提供程序配置中的 `UseTruststoreSPI` 配置属性控制信任库 SPI。默认情况下，Keycloak 传送门将属性设置为 `Only for ldaps`，这对于大多数部署来说是足够的。如果到 LDAP 的连接 URL 仅以 `ldaps` 开始，则密钥斗篷使用 Truststore SPI。

Synchronizing LDAP users to Keycloak 将 LDAP 用户同步到钥匙斗篷

If you set the `Import Users` option, the LDAP Provider handles importing LDAP users into the Keycloak local database. The first time a user logs in, the LDAP provider imports the LDAP user into the Keycloak database and validates the LDAP password. This first time a user logs in is the only time Keycloak imports the user. If you click the `Users` menu in the Admin Console and click the `View all users` button, you only see the LDAP users authenticated at least once by Keycloak. Keycloak imports users this way, so this operation does not trigger an import of the entire LDAP user database.

如果设置 Import Users 选项，则 LDAP 提供程序将处理将 LDAP 用户导入 Keycover 本地数据库的工作。当用户第一次登录时，LDAP 提供程序将 LDAP 用户导入到 Keycover 数据库中，并验证 LDAP 密码。这是用户第一次登录，也是键斗篷唯一一次导入用户。如果单击管理控制台中的 Users 菜单并单击 View all Users 按钮，则只能看到 LDAP 用户至少经过一次 Keycon 的身份验证。键披风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。

If you want to sync all LDAP users into the Keycloak database, configure and enable the **Sync Settings** on the LDAP provider configuration page.

如果希望将所有 LDAP 用户同步到钥匙斗篷数据库中，请在 LDAP 提供程序配置页上配置并启用“同步设置”。

Two types of synchronization exist:

存在两种类型的同步：

Periodic Full sync 周期性全同步

This type synchronizes all LDAP users into the Keycloak database. The LDAP users already in Keycloak, but different in LDAP, directly update in the Keycloak database.

这种类型将所有 LDAP 用户同步到钥匙斗篷数据库中。LDAP 用户已经在 Keycloak，但在 LDAP 中不同，他们直接更新到 Keycover 数据库中。

Periodic Changed users sync 周期性更改用户同步

When synchronizing, Keycloak creates or updates users created or updated after the last sync only.

在进行同步时，仅在上次同步后创建或更新已创建或更新的用户。

The best way to synchronize is to click **Synchronize all users** when you first create the LDAP provider, then set up periodic synchronization of changed users.

同步的最佳方式是在首次创建 LDAP 提供程序时单击 Synchronize all users，然后设置已更改用户的定期同步。

LDAP mappers LDAP 映射器

LDAP mappers are `listeners` triggered by the LDAP Provider. They provide another extension point to LDAP integration. LDAP mappers are triggered when:

LDAP 映射器是由 LDAP 提供程序触发的侦听器。它们为 LDAP 集成提供了另一个扩展点。LDAP 映射器在以下情况下被触发：

- Users log in by using LDAP.

用户使用 LDAP 登录。

- Users initially register.

用户最初注册。

- The Admin Console queries a user.

管理控制台查询用户。

When you create an LDAP Federation provider, Keycloak automatically provides a set of `mappers` for this provider. This set is changeable by users, who can also develop mappers or update/delete existing ones.

当您创建 LDAP 联邦提供程序时，Keycon 会自动为此提供程序提供一组映射程序。该集合可由用户更改，用户还可以开发映射器或更新/删除现有映射器。

User Attribute Mapper **用户属性映射器**

This mapper specifies which LDAP attribute maps to the attribute of the Keycloak user. For example, you can configure the `mail` LDAP attribute to the `email` attribute in the Keycloak database. For this mapper implementation, a one-to-one mapping always exists.

这个映射器指定哪个 LDAP 属性映射到 Key 用户的属性。例如，您可以将邮件 LDAP 属性配置为 Keycloak 数据库中的 email 属性。对于此映射器实现，始终存在一对映射。

FullName Mapper **全名映射器**

This mapper specifies the full name of the user. Keycloak saves the name in an LDAP attribute (usually `cn`) and maps the name to the `firstName` and `lastName` attributes in the Keycloak database. Having `cn` to contain the full name of the user is common for LDAP deployments.

此映射器指定用户的全名。Keycon 将名称保存在 LDAP 属性(通常是 cn)中，并将该名称映射到 Keycloak 数据库中的 firstName 和 lastName 属性。对于 LDAP 部署来说，使用 cn 来包含用户的全名是很常见的。



When you register new users in Keycloak and `Sync Registrations` is ON for the LDAP provider, the `fullName` mapper permits falling back to the username. This fallback is useful when using Microsoft Active Directory (MSAD). The common setup for MSAD is to configure the `cn` LDAP attribute as `fullName` and, at the same time, use the `cn` LDAP attribute as the `RDN LDAP Attribute` in the LDAP provider configuration. With this setup, Keycloak falls back to the username. For example, if you create Keycloak user "john123" and leave `firstName` and `lastName` empty, then the `fullname` mapper saves "john123" as the value of the `cn` in LDAP. When you enter "John Doe" for `firstName` and `lastName` later, the `fullname` mapper updates LDAP `cn` to the "John Doe" value as falling back to the username is unnecessary.

当您在 Keycloak 注册新用户并且 LDAP 提供程序的同步注册为 ON 时, `fullName` 映射器允许回退到用户名。这种回退在使用 MicrosoftActiveDirectory (MSAD)时非常有用。MSAD 的常见设置是将 `cn` LDAP 属性配置为 `fullName`, 同时在 LDAP 提供程序配置中使用 `cn` LDAP 属性作为 RDN LDAP 属性。通过这种设置, Keycover 回到用户名。例如, 如果您创建 Key枕用户“john123”, 并将 `firstName` 和 `lastName` 保留为空, 那么 `fullname` 映射器将“john123”保存为 LDAP 中 `cn` 的值。当您稍后为 `firstName` 和 `lastName` 输入“John Doe”时, `fullname` 映射器将 LDAP `cn` 更新为“John Doe”值, 因为回退到用户名是不必要的。

Hardcoded Attribute Mapper 硬编码属性映射器

This mapper adds a hardcoded attribute value to each Keycloak user linked with LDAP. This mapper can also force values for the `enabled` or `emailVerified` user properties.

这个映射器将一个硬编码属性值添加到与 LDAP 链接的每个 Keycover 用户。此映射器还可以强制启用的或 `emailVerified` 用户属性的值。

Role Mapper 角色定位器

This mapper configures role mappings from LDAP into Keycloak role mappings. A single role mapper can map LDAP roles (usually groups from a particular branch of the LDAP tree) into roles corresponding to a specified client's realm roles or client roles. You can configure more Role mappers for the same LDAP provider. For example, you can specify that role mappings from groups under `ou=main,dc=example,dc=org` map to realm role mappings, and role mappings from groups under `ou=finance,dc=example,dc=org` map to client role mappings of client `finance`.

这个映射器配置从 LDAP 到 Keycloak 角色映射的角色映射。单个角色映射器可以将 LDAP 角色(通常来自 LDAP 树的特定分支的组)映射到与指定客户机的领域角色或客户机角色对应的角色。您可以为同一个 LDAP 提供程序配置更多的角色映射程序。例如, 可以指定 `ou = main`、`dc = example`、`dc = org` 映射到领域角色映射的组的角色映射, 以及 `ou = finance`、`dc = example`、`dc = org` 映射到客户金融的客户角色映射的组的角色映射。

Hardcoded Role Mapper 硬编码角色映射器

This mapper grants a specified Keycloak role to each Keycloak user from the LDAP provider.

这个映射器从 LDAP 提供程序将指定的 Keycon 角色授予每个 Keycon 用户。

Group Mapper 群组绘图软件

This mapper maps LDAP groups from a branch of an LDAP tree into groups within Keycloak. This mapper also propagates user-group mappings from LDAP into user-group mappings in Keycloak.

这个映射器将 LDAP 组从 LDAP 树的一个分支映射到 Keycon 中的组。该映射器还将用户组映射从 LDAP 传播到 Keycloak 的用户组映射。

MSAD User Account Mapper 用户帐户映射器

This mapper is specific to Microsoft Active Directory (MSAD). It can integrate the MSAD user account state into the Keycloak account state, such as enabled account or expired password. This mapper uses the `userAccountControl`, and `pwdLastSet` LDAP attributes, specific to MSAD and are not the LDAP standard. For example, if the value of `pwdLastSet` is `0`, the Keycloak user must update their password. The result is an `UPDATE_PASSWORD` required action added to the user. If the value of `userAccountControl` is `514` (disabled account), the Keycloak user is disabled.

此映射器特定于 MicrosoftActiveDirectory (MSAD)。它可以将 MSAD 用户帐户状态集成到 Keycover 帐户状态，例如启用帐户或过期密码。此映射器使用 MSAD 特有的 userAccountControl 和 pwdLastSet LDAP 属性，这些属性不是 LDAP 标准。例如，如果 pwdLastSet 的值为0，则 Keycover 用户必须更新其密码。结果是向用户添加了 UPDATE _ PASSWORD 所需的操作。如果 userAccountControl 的值为514(禁用的帐户)，则禁用 Keycover 用户。

Certificate Mapper 证书制作工具

This mapper maps X.509 certificates. Keycloak uses it in conjunction with X.509 authentication and `Full certificate in PEM format` as an identity source. This mapper behaves similarly to the `User Attribute Mapper`, but Keycloak can filter for an LDAP attribute storing a PEM or DER format certificate. Enable `Always Read Value From LDAP` with this mapper.

这个映射器映射 X.509证书。密钥斗篷将它与 X.509身份验证和 PEM 格式的完整证书一起作为标识源使用。这个映射器的行为类似于 User Attribute Mapper，但是 Keycon 可以筛选存储 PEM 或 DER 格式证书的 LDAP 属性。使用此映射器始终启用从 LDAP 读取值。

User Attribute mappers that map basic Keycloak user attributes, such as `username`, `firstname`, `lastname`, and `email`, to corresponding LDAP attributes. You can extend these and provide your own additional attribute mappings. The Admin Console provides tooltips to help with configuring the corresponding mappers.

User Attribute 映射器将基本的 Key 用户属性(如用户名、名、姓和电子邮件)映射到相应的 LDAP 属性。您可以扩展这些属性，并提供自己的附加属性映射。管理控制台提供了帮助配置相应映射器的工具提示。

Password hashing 密码哈希

When Keycloak updates a password, Keycloak sends the password in plain-text format. This action is different from updating the password in the built-in Keycloak database, where Keycloak hashes and salts the password before sending it to the database. For LDAP, Keycloak relies on the LDAP server to hash and salt the password.

当钥匙斗篷更新密码时，钥匙斗篷以纯文本格式发送密码。此操作不同于更新内置的密钥斗篷数据库中的密码，其中，在将密码发送到数据库之前，密钥斗篷对其进行哈希处理并对其进行盐化。对于 LDAP，Keycloak 依赖于 LDAP 服务器来散列和加盐密码。

By default, LDAP servers such as MSAD, RHDS, or FreeIPA hash and salt passwords. Other LDAP servers such as OpenLDAP or ApacheDS store the passwords in plain-text unless you use the *LDAPv3 Password Modify Extended Operation* as described in [RFC3062](#)

(<https://datatracker.ietf.org/doc/html/rfc5280#section-4.2.1.3>). Enable the LDAPv3 Password Modify Extended Operation in the LDAP configuration page. See the documentation of your LDAP server for more details.

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)以纯文本形式存储密码，除非使用 RFC3062 中描述的 LDAPv3 Password Amendment Extended Operation。在 LDAP 配置页中启用 LDAPv3 密码修改扩展操作。有关更多细节，请参见 LDAP 服务器的文档。



Always verify that user passwords are properly hashed and not stored as plaintext by inspecting a changed directory entry using 通过检查已更改的目录条目，始终验证用户密码是否正确散列，并且没有以明文形式存储 `ldapsearch` and `base64 decode` the 基地64 解码 `userPassword` attribute value. 属性值

Troubleshooting 故障排除

It is useful to increase the logging level to TRACE for the category `org.keycloak.storage.ldap`. With this setting, many logging messages are sent to the server log in the TRACE level, including the logging for all queries to the LDAP server and the parameters, which were used to send the queries. When you are creating any LDAP question on user forum or JIRA, consider attaching the server log with enabled TRACE logging. If it is too big, the good alternative is to include just the snippet from server log with the messages, which were added to the log during the operation, which causes the issues to you.

将类别 `org.keycloak.storage.ldap` 的日志记录级别增加到 TRACE 是有用的。通过这种设置，许多日志消息被发送到 TRACE 级别的服务器日志，包括对 LDAP 服务器的所有查询的日志记录和用于发送查询的参数。当您在用户论坛或 JIRA 上创建任何 LDAP 问题时，请考虑使用启用的 TRACE 日志记录附加服务器日志。如果太大，最好的替代方法是只包含来自服务器日志的代码片段和消息，这些消息是在操作期间添加到日志中的，这会给您带来问题。

- When you create an LDAP provider, a message appears in the server log in the INFO level starting with:

创建 LDAP 提供程序时，INFO 级别的服务器日志中将显示一条消息，开头为：

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

It shows the configuration of your LDAP provider. Before you are asking the questions or reporting bugs, it will be nice to include this message to show your LDAP configuration. Eventually feel free to replace some config changes, which you do not want to include, with some placeholder values. One example is `bindDn=some-placeholder`. For `connectionUrl`, feel free to replace it as well, but it is generally useful to include at least the protocol, which was used (`ldap` vs `ldaps`). Similarly it can be useful to include the details for configuration of your LDAP mappers, which are displayed with the message like this at the DEBUG level:

它显示了 LDAP 提供程序的配置。在您提出问题或报告错误之前，最好包含此消息以显示您的 LDAP 配置。最后，您可以使用一些占位符值替换一些不希望包含的配置更改。一个例子是 `bindDn = some-placeholder`。对于 `ConnectionUrl`，也可以自由地替换它，但是通常至少包含所使用的协议是有用的(`ldap` vs `ldaps`)。类似地，包含 LDAP 映射器配置的详细信息也是有用的，这些信息在 DEBUG 级显示如下：

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

Note those messages are displayed just with the enabled DEBUG logging.

请注意，这些消息仅在启用 DEBUG 日志记录时显示。

- For tracking the performance or connection pooling issues, consider setting the value of property `Connection Pool Debug Level` of the LDAP provider to value `all`. This will add lots of additional messages to server log with the included logging for the LDAP connection pooling. This can be used to track the issues related to connection pooling or performance.

为了跟踪性能或连接池问题，请考虑将 LDAP 提供程序的属性 `Connection Pool Debug Level` 的值设置为 `value all`。这将向服务器日志添加大量附加消息，包括用于 LDAP 连接池的日志记录。这可用于跟踪与连接池或性能相关的问题。



After changing the configuration of connection pooling, you may need to restart the Keycloak server to enforce re-initialization of the LDAP provider connection. 更改连接池的配置之后，可能需要重新启动 Keycloak 服务器，以强制重新初始化 LDAP 提供程序连接

If no more messages appear for connection pooling even after server restart, it can indicate that connection pooling does not work with your LDAP server.

如果在服务器重新启动后仍然没有出现更多的连接池消息，则可能表明连接池无法与 LDAP 服务器一起工作。

- For the case of reporting LDAP issue, you may consider to attach some part of your LDAP tree with the target data, which causes issues in your environment. For example if login of some user takes lot of time, you can consider attach his LDAP entry showing count of `member` attributes of various

"group" entries. In this case, it might be useful to add if those group entries are mapped to some Group LDAP mapper (or Role LDAP Mapper) in Keycloak etc.

对于报告 LDAP 问题的情况，您可以考虑将目标数据附加到 LDAP 树的某些部分，这会在您的环境中导致问题。例如，如果某个用户的登录需要很多时间，您可以考虑附加他的 LDAP 条目，显示各个“组”条目的成员属性的计数。在这种情况下，如果这些组条目被映射到 Keycloak 的某个组 LDAP 映射器(或角色 LDAP 映射器)，那么添加这些条目可能是有用的。

[Edit this section](#) 编辑这部分

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/user-federation/sssd.adoc)

[Report an issue](#) 报告一个问题

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/user-federation/sssd.adoc)

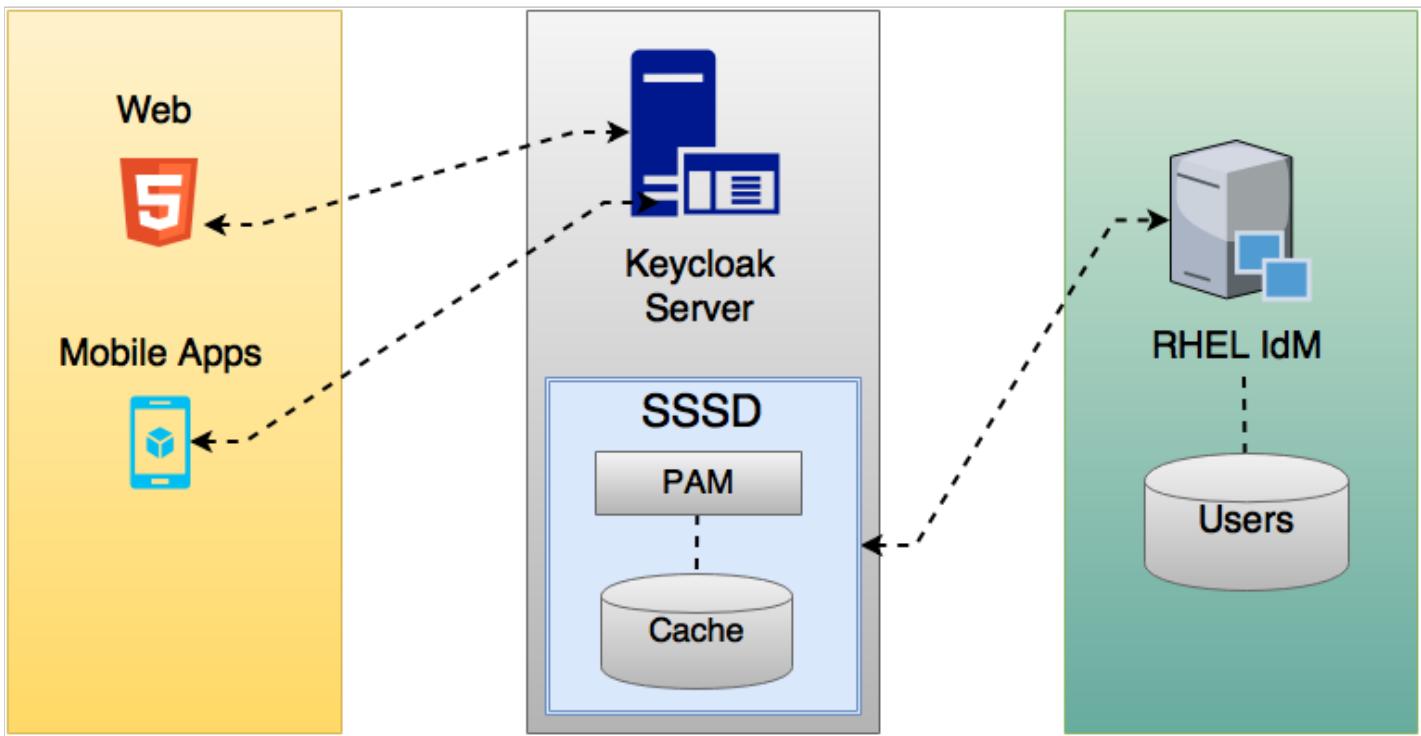
Keycloak includes the [System Security Services Daemon \(SSSD\)](#) (<https://fedoraproject.org/wiki/Features/SSSD>) plugin. SSSD is part of the Fedora and Red Hat Enterprise Linux (RHEL), and it provides access to multiple identities and authentication providers. SSSD also provides benefits such as failover and offline support. For more information, see [the Red Hat Enterprise Linux Identity Management documentation](#) (https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/system-level_authentication_guide/sssd)

密钥斗篷包括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，它提供对多个身份和身份验证提供程序的访问。SSSD 还提供诸如故障转移和脱机支持等好处。有关更多信息，请参见 Red Hat Enterprise Linux Identity Management 文档。

SSSD integrates with the FreeIPA identity management (IdM) server, providing authentication and access control. With this integration, Keycloak can authenticate against privileged access management (PAM) services and retrieve user data from SSSD. For more information about using Red Hat Identity Management in Linux environments, see [the Red Hat Enterprise Linux Identity Management documentation](#)

(https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/linux_domain_identity_authentication_and_policy_guide/index)

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，Keycon 可以针对特权访问管理(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 Red Hat Identity Management 的详细信息，请参阅 Red Hat Enterprise Linux Identity Management 文档。



Keycloak and SSSD communicate through read-only D-Bus interfaces. For this reason, the way to provision and update users is to use the FreeIPA/IdM administration interface. By default, the interface imports the username, email, first name, and last name.

密钥斗篷和SSSD通过只读D-Bus接口进行通信。因此，提供和更新用户的方法是使用FreeIPA/IDM管理界面。默认情况下，接口导入用户名、电子邮件、名和姓。

i Keycloak registers groups and roles automatically but does not synchronize them. Any changes made by the Keycloak administrator in Keycloak do not synchronize with SSSD.

键隐形自动注册组和角色，但不同步它们。在Keycloak的“钥匙斗篷”管理员所作的任何更改都不会与SSSD同步。

FreeIPA/IdM server FreeIPA/IDM 服务器

The [FreeIPA Docker image](https://hub.docker.com/r/freeipa/freeipa-server/) (<https://hub.docker.com/r/freeipa/freeipa-server/>) is available in Docker Hub. To set up the FreeIPA server, see the [FreeIPA documentation](https://www.freeipa.org/page/Quick_Start_Guide) (https://www.freeipa.org/page/Quick_Start_Guide).

在Docker Hub可以看到FreeIPA Docker图像，要设置FreeIPA服务器，请参阅FreeIPA文档。

Procedure 程序

1. Run your FreeIPA server using this command:

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
```

BASH

The parameter `-h` with `server.freeipa.local` represents the FreeIPA/IdM server hostname. Change `YOUR_PASSWORD` to a password of your own.

Local 参数-h 表示 FreeIPA/IDM 服务器主机名。

2. After the container starts, change the `/etc/hosts` file to include:

在容器启动后，将`/etc/hosts`文件更改为包括：

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

BASH

If you do not make this change, you must set up a DNS server.

如果不进行此更改，则必须设置 DNS 服务器。

3. Use the following command to enroll your Linux server in the IPA domain so that the SSSD federation provider starts and runs on Keycloak:

使用以下命令将您的 Linux 服务器注册到 IPA 域中，以便 SSSD 联合提供程序启动并运行在 Keycloak 上：

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

BASH

4. Run the following command on the client to verify the installation is working:

在客户机上运行以下命令以验证安装是否正常：

```
kinit admin
```

BASH

5. Enter your password.

输入密码。

6. Add users to the IPA server using this command:

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. Force set the user's password using kinit.

使用 kinit 强制设置用户的密码。

```
kinit <username>
```

BASH

8. Enter the following to restore normal IPA operation:

输入以下内容以恢复正常 IPA 操作：

```
kdestroy -A  
kinit admin
```

BASH

SSSD and D-Bus SSSD 及 D-Bus

The federation provider obtains the data from SSSD using D-BUS. It authenticates the data using PAM.

联合提供程序使用 D-BUS 从 SSSD 获取数据，并使用 PAM 对数据进行身份验证。

Procedure 程序

1. Install the sssd-dbus RPM.

安装 ssd-dbus RPM。

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

BASH

2. Run the following provisioning script:

运行以下配置脚本：

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

BASH

This script makes the following changes to `/etc/sssd/sssd.conf` :

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
```

BASH

3. Run `dbus-send` to ensure the setup is successful.

运行 `dbus-send` 以确保安装成功。

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

BASH

If the setup is successful, you see the user's group. If this command returns a timeout or an error, the federation provider running on Keycloak cannot retrieve any data. This error usually happens because the server is not enrolled in the FreeIPA IdM server, or does not have permission to access the SSSD service.

如果安装成功，您将看到用户的组。如果此命令返回超时或错误，则运行在 Keycloak 上的联合提供程序无法检索任何数据。发生此错误通常是因为服务器没有注册到 FreeIPA IDM 服务器，或者没有访问 SSSD 服务的权限。

If you do not have permission to access the SSSD service, ensure that the user running the Keycloak server is in the `/etc/sssd/sssd.conf` file in the following section:

如果您没有访问 SSSD 服务的权限，请确保运行 Keycloak 服务器的用户在以下部分的`/etc/SSSD/SSSD.conf`文件中：

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

BASH

Enabling the SSSD federation provider 启用 SSSD 联合提供程序

Keycloak uses DBus-Java to communicate at a low level with D-Bus. D-Bus depends on the [Unix Sockets Library](#) (<http://www.matthew.ath.cx/projects/java/>).

Key斗篷使用 DBus-Java 与 D-Bus 进行低层次的通信。

You can find an RPM for this library in [the keycloak repository](#)

(<https://github.com/keycloak/libunix-dbus-java/releases>). Before installing this RPM, check the RPM signature using this command:

您可以在密钥斗篷存储库中找到这个库的 RPM。在安装此 RPM 之前，请使用以下命令检查 RPM 签名：

```
$ rpm -K libunix-dbus-java-0.8.0-1.fc24.x86_64.rpm
libunix-dbus-java-0.8.0-1.fc24.x86_64.rpm:
Header V4 RSA/SHA256 Signature, key ID 84dc9914: OK
Header SHA1 digest: OK (d17bb7eba7a5304c1856ee4357c8ba4ec9c0b89)
V4 RSA/SHA256 Signature, key ID 84dc9914: OK
MD5 digest: OK (770c2e68d052cb4a4473e1e9fd8818cf)
```

BASH

Install the RPM using this command:

使用以下命令安装 RPM：

```
$ sudo yum install libunix-dbus-java-0.8.0-1.fc24.x86_64.rpm
```

BASH

Keycloak uses JNA to authenticate with PAM. Ensure you have the JNA package installed.

密钥斗篷使用 JNA 对 PAM 进行身份验证。确保您已经安装了 JNA 包。

```
$ sudo yum install jna
```

BASH

Use the `sssctl user-checks` command to validate your setup:

使用 sssctl user-check 命令验证设置:

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

Configuring a federated SSSD store 配置联合 SSSD 存储

After the installation, configure a federated SSSD store.

安装之后，配置一个联合 SSSD 存储。

Procedure 程序

1. Click **User Federation** in the menu.

在菜单中单击“用户联盟”。

2. From the **Add Provider** list select *sssd*. Keycloak brings you to the *sssd* configuration page.

从 AddProvider 列表中选择 *sssd*.Key 可以看到 *sssd* 配置页面。

3. Click **Save**.

单击“保存”。

You can now authenticate against Keycloak using FreeIPA/IdM credentials.

您现在可以使用 FreeIPA/IDM 凭据对 Keycloak 进行身份验证。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/user-federation/custom.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/user-federation/custom.adoc)

Keycloak does have a Service Provider Interface (SPI) for User Storage Federation to develop custom providers. You can find documentation on developing customer providers in the [Server Developer Guide](#) (https://www.keycloak.org/docs/latest/server_development/).

Keycloak确实有一个用于用户存储联盟的服务提供程序接口(SPI)，用于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提供程序的文档。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/assembly-managing-users.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/assembly-managing-users.adoc)

From the Admin Console, you have a wide range of actions you can perform to manage users.

从管理控制台，您可以执行大量操作来管理用户。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/users/proc-creating-user.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/users/proc-creating-user.adoc)

You create users in the realm where you intend to have applications needed by those users. Avoid creating users in the master realm, which is only intended for creating other realms.

您在打算拥有这些用户所需应用程序的领域中创建用户。避免在主域中创建用户，主域仅用于创建其他域。

Prerequisite 先决条件

- You are in a realm other than the master realm.

你所在的领域不是主人的领域。

Procedure 程序

1. Click **Users** in the menu.

单击菜单中的“用户”。

2. Click **Add User**.

单击“添加用户”。

3. Enter the details for the new user.

输入新用户的详细信息。



Username 用户名 is the only required field. 是唯一需要的字段

4. Click **Save**. After saving the details, the **Management** page for the new user is displayed.

单击 Save。保存详细信息后，将显示新用户的“管理”页。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/users/ref-user-credentials.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/users/ref-user-credentials.adoc)

You can manage credentials of a user in the **Credentials** tab.

可以在“凭据”选项卡中管理用户的凭据。

Credential management 证件管理

The screenshot shows the Keycloak administration interface. On the left, a sidebar menu includes 'Master', 'Manage', 'Clients', 'Client scopes', 'Realm roles', 'Users' (which is selected), 'Groups', 'Sessions', and 'Events'. The main content area is titled 'User details' for 'johndoe'. It has tabs for 'Details', 'Attributes', 'Credentials' (which is active and highlighted in blue), 'Role mapping', 'Groups', 'Consents', and 'Identity provider links'. Below the tabs, there's a large circular button with a plus sign. To its right, the text 'No credentials' is displayed. Underneath, a message states 'This user does not have any credentials. You can set password for this user.' At the bottom right of this section is a blue 'Set password' button.

You change the priority of credentials by dragging and dropping rows. The new order determines the priority of the credentials for that user. The topmost credential has the highest priority. The priority determines which credential is displayed first after a user logs in.

通过拖放行更改凭据的优先级。新顺序确定该用户凭据的优先级。最高的凭证具有最高的优先级。优先级确定用户登录后首先显示哪个凭据。

Type 类型

This column displays the type of credential, for example **password** or **OTP**.

此列显示凭据的类型，例如密码或 OTP。

User Label 用户标签

This is an assignable label to recognize the credential when presented as a selection option during login. It can be set to any value to describe the credential.

这是一个可分配的标签，用于识别在登录期间作为选择选项出现的凭据。可以将其设置为描述凭据的任何值。

Data 百科

This is the non-confidential technical information about the credential. It is hidden, by default. You can click **Show data...** to display the data for a credential.

这是关于证书的非机密技术信息。默认情况下，它是隐藏的。您可以单击 Show data... 以显示凭据的数据。

Actions 行动

Click **Reset password** to change the password for the user and **Delete** to remove the credential.

单击“重置密码”更改用户的密码，单击“删除”删除凭据。

You cannot configure other types of credentials for a specific user in the Admin Console; that task is the user's responsibility.

无法在管理控制台中为特定用户配置其他类型的凭据; 该任务由用户负责。

You can delete the credentials of a user in the event a user loses an OTP device or if credentials have been compromised. You can only delete credentials of a user in the **Credentials** tab.

您可以在用户丢失 OTP 设备或用户凭据受到损害的情况下删除用户的凭据。只能在“凭据”选项卡中删除用户的凭据。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/users/proc-setting-password-user.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/users/proc-setting-password-user.adoc)

If a user does not have a password, or if the password has been deleted, the **Set Password** section is displayed.

如果用户没有密码，或者密码已被删除，则显示“设置密码”部分。

If a user already has a password, it can be reset in the **Reset Password** section.

如果用户已经有密码，则可以在“重置密码”部分中重置密码。

Procedure 程序

1. Click **Users** in the menu. The **Users** page is displayed.

单击菜单中的“用户”。将显示“用户”页。

2. Select a user.

选择一个用户。

3. Click the **Credentials** tab.

单击“凭据”选项卡。

4. Type a new password in the **Set Password** section.

在“设置密码”部分中键入新密码。

5. Click **Set Password**.

单击“设置密码”。



If **Temporary 暂时的** is **ON 开始**, the user must change the password at the first login. To allow users to keep the password supplied, set **用户必须在第一次登录时更改密码** **Temporary 暂时的** to **到OFF. 关掉**. The user must click **用户必须单击Set Password 设定密码** to change the password. **更改密码**

6. Alternatively, you can send an email to the user that requests the user reset the password.

或者，您可以向请求用户重置密码的用户发送电子邮件。

a. Click **Credential Reset**.

单击凭据重置。

b. Select **Update Password** from the list.

从列表中选择“更新密码”。

c. Click **Send Email**. The sent email contains a link that directs the user to the **Update Password** window.

单击“发送电子邮件”。发送的电子邮件包含一个链接，该链接指示用户到“更新密码”窗口。

d. Optionally, you can set the validity of the email link. This is set to the default preset in the **Tokens** tab in **Realm Settings**.

您可以选择设置电子邮件链接的有效性。在“领域设置”的“令牌”选项卡中，此设置设置为默认预设。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/users/proc-creating-otp.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/users/proc-creating-otp.adoc)

If OTP is conditional in your realm, the user must navigate to Keycloak Account Console to reconfigure a new OTP generator. If OTP is required, then the user must reconfigure a new OTP generator when logging in.

如果 OTP 在您的领域中是有条件的，那么用户必须导航到钥匙斗篷帐户控制台来重新配置一个新的 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。

Alternatively, you can send an email to the user that requests the user reset the OTP generator. The following procedure also applies if the user already has an OTP credential.

或者，您可以向用户发送电子邮件，请求用户重置 OTP 生成器。如果用户已经具有 OTP 凭据，则还适用以下过程。

Prerequisite 先决条件

- You are logged in to the appropriate realm.

您已登录到适当的领域。

Procedure 程序

1. Click **Users** in the main menu. The **Users** page is displayed.

单击主菜单中的“用户”。将显示“用户”页。

2. Select a user.

选择一个用户。

3. Click the **Credentials** tab.

单击“凭据”选项卡。

4. Click **Credential Reset**.

单击凭据重置。

5. Select **Configure OTP**.

选择 ConfigureOTP。

6. Navigate to the **Reset Actions** list.

导航到“重置操作”列表。

7. Click **Configure OTP**.

单击“配置 OTP”。

8. Click **Send Email**. The sent email contains a link that directs the user to the **OTP setup page**.

单击“发送电子邮件”。发送的电子邮件包含一个链接，指示用户进入 OTP 设置页面。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/users/proc-configuring-user-attributes.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/users/proc-configuring-user-attributes.adoc)

User attributes provide a customized experience for each user. You can create a personalized identity for each user in the console by configuring user attributes.

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

Users 用户

The screenshot shows the Keycloak administration interface. On the left, a sidebar menu includes 'Master', 'Manage', 'Clients', 'Client scopes', 'Realm roles', 'Users' (which is selected), 'Groups', 'Sessions', 'Events', 'Configure', and 'Realm settings'. The main content area is titled 'User details' for 'johndoe'. It has tabs for 'Details', 'Attributes' (which is active), 'Credentials', 'Role mapping', 'Groups', 'Consents', and 'Identity provider links'. Under the 'Attributes' tab, there is a table with two rows. The first row has 'Key' as 'mobile' and 'Value' as '5555-5555-555'. The second row has 'Key' as 'Type a key' and 'Value' as 'Type a value'. Below the table is a button '+ Add an attribute'. At the bottom are 'Save' and 'Revert' buttons.

Prerequisite 先决条件

- You are in the realm where the user exists.

您处于用户存在的领域中。

Procedure 程序

1. Click **Users** in the menu.

单击菜单中的“用户”。

2. Select a user to manage.

选择要管理的用户。

3. Click the **Attributes** tab.

单击“属性”选项卡。

4. Enter the attribute name in the **Key** field.

在 Key 字段中输入属性名称。

5. Enter the attribute value in the **Value** field.

在 Value 字段中输入属性值。

6. Click **Save**.

单击“保存”。

Some read-only attributes are not supposed to be updated by the administrators. This includes attributes that are read-only by design like for example 管理员不应该更新某些只读属性。这包括设计为只读的属性，例如 `LDAP_ID`，which is filled automatically by the LDAP provider. Some other attributes should be read-only for typical user administrators due to security reasons. See the details in the , 由 LDAP 提供程序自动填充。由于安全原因，对于典型的用户管理员，其他一些属性应该是只读的。中查看详细信息Mitigating security threats 缓解安全威胁 chapter. 第二章



[Edit this section](#) 编辑这部分

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/users/con-user-registration.adoc)

[Report an issue](#) 报告一个问题

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/users/con-user-registration.adoc)

You can use Keycloak as a third-party authorization server to manage application users, including users who self-register. If you enable self-registration, the login page displays a registration link so that user can create an account.

您可以使用 Keycloak 作为第三方授权服务器来管理应用程序用户，包括自注册的用户。如果启用自注册，登录页面将显示一个注册链接，以便用户可以创建帐户。

Registration link 注册链接

The screenshot shows a login interface with the following elements:

- A title "Sign in to your account" centered at the top.
- A "Username or email" input field below the title.
- A "Password" input field below the username field.
- A large blue "Sign In" button centered below the password field.
- A "New user? Register" link located at the bottom of the form.

A user must add profile information to the registration form to complete registration. The registration form can be customized by removing or adding the fields that must be completed by a user.

用户必须向注册表单添加配置文件信息才能完成注册。可以通过删除或添加必须由用户完成的字段来自定义注册表单。

Additional resources 额外资源

- For more information on customizing user registration, see the [Server Developer Guide](#) (https://www.keycloak.org/docs/latest/server_development/).

有关自定义用户注册的详细信息，请参阅服务器开发人员指南。

[Edit this section](#) 编辑这部分

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/users/proc-enabling-user-registration.adoc)

[Report an issue](#) 报告一个问题

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/users/proc-enabling-user-registration.adoc)

Enable users to self-register.

允许用户自我注册。

Procedure 程序

1. Click **Realm Settings** in the main menu.

单击主菜单中的“领域设置”。

2. Click the **Login** tab.

单击 Login 选项卡。

3. Toggle **User Registration** to **ON**.

切换用户注册到 ON。

4. Click **Save**.

单击“保存”。

After you enable this setting, a **Register** link displays on the login page of the Admin Console.

启用此设置后，将在管理控制台的登录页面上显示一个 Register 链接。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/users/proc-registering-new-user.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/users/proc-registering-new-user.adoc)

As a new user, you must complete a registration form to log in for the first time. You add profile information and a password to register.

作为新用户，您必须填写注册表单才能首次登录。您可以添加要注册的配置文件信息和密码。

Registration form 登记表格

Register

First name

Last name

Email

Username

Password

Confirm password

[« Back to Login](#)

Register

Prerequisite 先决条件

- User registration is enabled.

已启用用户注册。

Procedure 程序

- Click the **Register** link on the login page. The registration page is displayed.

单击登录页面上的 Register 链接。将显示注册页面。

- Enter the user profile information.

输入用户配置文件信息。

- Enter the new password.

输入新密码。

- Click **Save**.

单击“保存”。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/users/con-required-actions.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/users/con-required-actions.adoc)

You can set the actions that a user must perform at the first login. These actions are required after the user provides credentials. After the first login, these actions are no longer required. You add required actions on the **Details** tab of that user.

您可以设置用户在第一次登录时必须执行的操作。在用户提供凭据之后，需要执行这些操作。在第一次登录之后，不再需要这些操作。您可以在该用户的 Details 选项卡上添加必需的操作。

The following are examples of required action types:

以下是所需操作类型的例子：

Update Password 更新密码

The user must change their password.

用户必须更改密码。

Configure OTP 配置 OTP

The user must configure a one-time password generator on their mobile device using either the Free OTP or Google Authenticator application.

用户必须使用 Free oTP 或 Google Authenticator 应用程序在其移动设备上配置一个一次性密码生成器。

Verify Email 确认电子邮件

The user must verify their email account. An email will be sent to the user with a validation link that they must click. Once this workflow is successfully completed, the user will be allowed to log in.

用户必须验证他们的电子邮件帐户。用户必须单击一个验证链接才能收到电子邮件。一旦此工作流成功完成，用户将被允许登录。

Update Profile 更新配置文件

The user must update profile information, such as name, address, email, and phone number.

用户必须更新配置文件信息，如姓名、地址、电子邮件和电话号码。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/users/proc-setting-required-actions.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/users/proc-setting-required-actions.adoc)

You can set the actions that are required for any user.

您可以设置任何用户所需的操作。

Procedure 程序

1. Click **Users** in the menu.

单击菜单中的“用户”。

2. Select a user from the list.

从列表中选择一个用户。

3. Navigate to the **Required User Actions** list.

导航到“必需的用户操作”列表。

The screenshot shows the Keycloak User details interface. On the left is a sidebar with navigation links: Master, Manage, Clients, Client scopes, Realm roles, Users (which is selected), Groups, Sessions, Events, Configure, Realm settings, Authentication, Identity providers, and User federation. The main content area is titled "User details" for the user "johndoe". It has tabs for Details, Attributes, Credentials, Role mapping, Groups, Consents, and Identity provider links. Under the Details tab, there are fields for ID (8611aac7-f28e-412e-8f6c-7dde3705ca6e), Created at (05/16/22 10:05:36 AM), Username (johndoe), Email (johndoe@example.com), Email verified (Off), First name (empty), Last name (empty), and Enabled (On). At the bottom, there is a "Required user actions" section with a "Select action" dropdown containing "Update Password" and a remove button (X). Below this are "Save" and "Revert" buttons.

4. Select all the actions you want to add to the account.

选择要添加到帐户的所有操作。

5. Click the X next to the action name to remove it.

单击操作名称旁边的 X 以删除它。

6. Click **Save** after you select which actions to add.

选择要添加的操作后，单击 Save。

[Report an issue](#) 报告一个问题

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/users/proc-setting-default-required-actions.adoc)

to a user created by the **Add User** button on the **Users** page or the **Register** link on the login page.

您可以在所有新用户首次登录之前指定需要执行哪些操作。这些要求适用于用户页面上的 Add User 按钮或登录页面上的 Register 链接创建的用户。

Procedure 程序

1. Click **Authentication** in the menu.

单击菜单中的“身份验证”。

2. Click the **Required Actions** tab.

单击“必要操作”选项卡。

3. Click the checkbox in the **Set as default action** column for one or more required actions. When a new user logs in for the first time, the selected actions must be executed.

单击 Set as default action 列中一个或多个必需操作的复选框。当新用户第一次登录时，必须执行选定的操作。

[Edit this section](#) 编辑这部分

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/users/proc-enabling-terms-conditions.adoc)

[Report an issue](#) 报告一个问题

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/users/proc-enabling-terms-conditions.adoc)

You can enable a required action that new users must accept the terms and conditions before logging in to Keycloak for the first time.

您可以启用一个所需的操作，新用户必须在首次登录到 Keycloak 之前接受条款和条件。

Procedure 程序

1. Click **Authentication** in the menu.

单击菜单中的“身份验证”。

2. Click the **Required Actions** tab.

单击“必要操作”选项卡。

3. Enable the **Terms and Conditions** action.

启用条款和条件操作。

4. Edit the `terms.ftl` file in the base login theme.

编辑基本登录主题中的 terms.ftl 文件。

Additional resources 额外资源

- For more information on extending and creating themes, see the [Server Developer Guide](#) (https://www.keycloak.org/docs/latest/server_development/).

有关扩展和创建主题的更多信息，请参见服务器开发人员指南。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/users/proc-searching-user.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/users/proc-searching-user.adoc)

Search for a user to view detailed information about the user, such as the user's groups and roles.

搜索用户以查看有关该用户的详细信息，如用户的组和角色。

Prerequisite 先决条件

- You are in the realm where the user exists.

您处于用户存在的领域中。

Procedure 程序

- Click **Users** in the main menu. This **Users** page is displayed.

单击主菜单中的“用户”。将显示此“用户”页。

- Type the full name, last name, first name, or email address of the user you want to search for in the search box. The search returns all users who match your criteria.

在搜索框中键入要搜索的用户的全名、姓、名或电子邮件地址。搜索返回符合条件的所有用户。

- Alternatively, you can click **View all users** to list every user in the system.

或者，您可以单击“查看所有用户”以列出系统中的每个用户。



This action searches only the local Keycloak database and not the federated database, such as LDAP. The backends for federated databases do not have a pagination mechanism that enables searching for users. 这个操作只搜索本地 Keycloak 数据库，而不搜索联邦数据库，例如 LDAP。联邦数据库的后端没有启用用户搜索的分页机制

- To search users from a federated backend, the user list must be synced into the Keycloak database. Adjust the search criteria to sync the backend users to the Keycloak database.

若要从联邦后端搜索用户，必须将用户列表同步到钥匙斗篷数据库中。调整搜索条件，以将后端用户同步到“钥匙斗篷”数据库。

b. Alternatively, click the **User Federation** in the left menu.

或者，单击左侧菜单中的“用户联盟”。

i. To apply changes to a selected user, click **Sync changed users** on the page with your federation provider.

若要将更改应用于选定的用户，请单击页上具有联合提供程序的已更改用户的“同步”。

ii. To apply changes to all users in the database, click **Sync all users** on the page with your federation provider.

若要将更改应用于数据库中的所有用户，请单击“使用联合提供程序同步页上的所有用户”。

Additional resources 额外资源

- For more information on user federation, see **User Federation**.

有关用户联合的更多信息，请参见用户联合。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/users/proc-deleting-user.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priorty=3&description=File:%20server_admin/topics/users/proc-deleting-user.adoc)

You can delete a user, who no longer needs access to applications. If a user is deleted, the user profile and data is also deleted.

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，也会删除用户配置文件和数据。

Procedure 程序

1. Click **Users** in the menu. The **Users** page is displayed.

单击菜单中的“用户”。将显示“用户”页。

2. Click **View all users** to find a user to delete.

单击“查看所有用户”以查找要删除的用户。

3. Click **Users** in the menu. The **Users** page is displayed.

单击菜单中的“用户”。将显示“用户”页。



Alternatively, you can use the search bar to find a user. 或者，您可以使用搜索栏查找用户

4. Click **Delete** from the action menu next to the user you want to remove and confirm deletion.

单击要删除的用户旁边的操作菜单中的“删除”并确认删除。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/users/proc-allow-user-to-delete-account.adoc)

Report an issue 报告一个问题
(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/users/proc-allow-user-to-delete-account.adoc)

End users and applications can delete their accounts in the Account Console if you enable this capability in the Admin Console. Once you enable this capability, you can give that capability to specific users.

如果您在管理控制台中启用此功能，最终用户和应用程序可以删除帐户控制台中的帐户。一旦启用了该功能，就可以将该功能提供给特定的用户。

Enabling the Delete Account Capability 启用删除帐户能力

You enable this capability on the **Required Actions** tab.

您可以在“必要操作”选项卡上启用此功能。

Procedure 程序

1. Click **Authentication** in the menu.

单击菜单中的“身份验证”。

2. Click the **Required Actions** tab.

单击“必要操作”选项卡。

3. Select **Enabled** on the **Delete Account** row.

在“删除帐户”行上选择“启用”。

Delete account on required actions tab 删除必需操作选项卡上的帐户

Authentication			
Required actions		Enabled	Set as default action
Configure OTP	<input checked="" type="checkbox"/>	On	<input type="checkbox"/> Off
Terms and Conditions	<input type="checkbox"/>	Off	<input checked="" type="checkbox"/> Disabled off
Update Password	<input checked="" type="checkbox"/>	On	<input type="checkbox"/> Off
Update Profile	<input checked="" type="checkbox"/>	On	<input type="checkbox"/> Off
Verify Email	<input checked="" type="checkbox"/>	On	<input type="checkbox"/> Off
Delete Account	<input checked="" type="checkbox"/>	On	<input type="checkbox"/> Off
Update User Locale	<input checked="" type="checkbox"/>	On	<input type="checkbox"/> Off
Webauthn Register Passwordless	<input type="checkbox"/>	Off	<input checked="" type="checkbox"/> Disabled off
Webauthn Register	<input type="checkbox"/>	Off	<input checked="" type="checkbox"/> Disabled off
Verify Profile	<input type="checkbox"/>	Off	<input checked="" type="checkbox"/> Disabled off

Giving a user the `delete-account` role

You can give specific users a role that allows account deletion.

您可以为特定用户提供允许删除帐户的角色。

Procedure 程序

1. Click **Users** in the menu.

单击菜单中的“用户”。

2. Select a user.

选择一个用户。

3. Click the **Role Mappings** tab.

单击“角色映射”选项卡。

4. Click the **Assign role** button.

单击“分配角色”按钮。

5. Click **account delete-account**.

点击帐户删除帐户。

6. Click **Assign**.

单击“分配”。

Delete-account role 删除帐户角色

The screenshot shows a dialog box titled "Assign roles to johndoe account". At the top, there is a search bar labeled "Search by role name" and a filter dropdown set to "account". Below the header, there is a table with columns "Name" and "Description". The table lists seven roles under the "account" origin. The "delete-account" role is checked, while the others are unchecked. The checked role has a blue checkmark icon next to it.

Name	Description
account view-profile	\${role_view-profile}
account view-applications	\${role_view-applications}
account view-consent	\${role_view-consent}
account manage-account-links	\${role_manage-account-links}
<input checked="" type="checkbox"/> account delete-account	\${role_delete-account}
account manage-account	\${role_manage-account}
account manage-consent	\${role_manage-consent}

At the bottom of the dialog, there are two buttons: "Assign" (highlighted in blue) and "Cancel".

Deleting your account 删除你的账户

Once you have the **delete-account** role, you can delete your own account.

一旦您拥有了删除帐户角色，您就可以删除自己的帐户。

1. Log into the Account Console.

登录帐户控制台。

2. At the bottom of the **Personal Info** page, click **Delete Account**.

在“个人信息”页的底部，单击“删除帐户”。

Delete account page 删除帐户页面

The screenshot shows a mobile application interface. On the left is a dark sidebar with a blue vertical bar at the top. The sidebar contains three items: "Personal info" (selected), "Account security" (with a right-pointing arrow), and "Applications". The main content area has a white background and a header "Personal info". Below the header is a sub-instruction "Manage your basic information." followed by a note "All fields are required.". There are four input fields: "Username" (text "johndoe"), "Email" (empty), "First name" (text "John"), and "Last name" (text "Doe"). At the bottom are two buttons: a blue "Save" button and a light blue "Cancel" button. Below these buttons is a section titled "Delete Account" with a dropdown icon. A warning message "This is irreversible. All your data will be permanently destroyed, and irretrievable." is displayed, along with a red "Delete" button.

Personal info

Account security >

Applications

Personal info

Manage your basic information.

All fields are required.

Username

johndoe

Email

First name

John

Last name

Doe

Save Cancel

▼ Delete Account

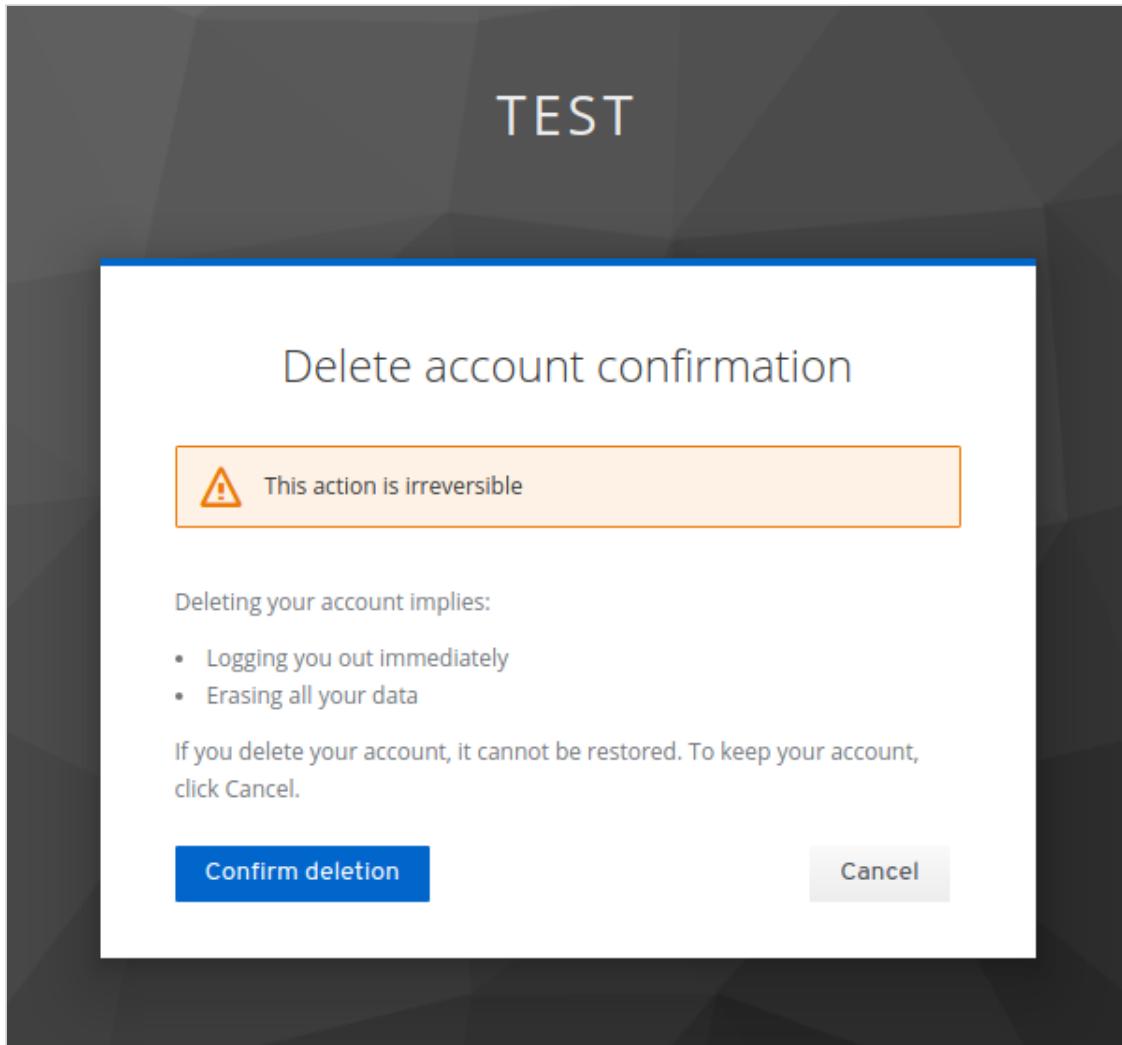
This is irreversible. All your data will be permanently destroyed, and irretrievable.

Delete

3. Enter your credentials and confirm the deletion.

输入您的凭据并确认删除。

Delete confirmation 删除确认



This action is irreversible. All your data in Keycloak will be removed.

这个行动是不可逆转的你在 Keycloak 的所有数据都会被删除。

[Edit this section](#) 编辑这部分

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/users/con-user-impersonation.adoc)

[Report an issue](#) 报告一个问题

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/users/con-user-impersonation.adoc)

An administrator with the appropriate permissions can impersonate a user. For example, if a user experiences a bug in an application, an administrator can impersonate the user to investigate or duplicate the issue.

具有适当权限的管理员可以模拟用户。例如，如果用户在应用程序中遇到 bug，管理员可以模拟用户调查或复制该问题。

Any user with the `impersonation` role in the realm can impersonate a user.

领域中具有模拟角色的任何用户都可以模拟用户。

Procedure 程序

1. Click **Users** in the menu.

单击菜单中的“用户”。

2. Click a user to impersonate.

单击要模拟的用户。

3. From the **Actions** list, select **Impersonate**.

从“操作”列表中，选择“模拟”。

The screenshot shows the Keycloak Admin Console interface. On the left, a sidebar menu is open with 'Master' selected. Under 'Manage', the 'Users' option is highlighted. The main content area is titled 'User details' for a user named 'johndoe'. The 'Details' tab is active, showing fields for 'ID' (8611aac7-f28e-412e-8f6c-7dde3705ca6e), 'Created at' (05/16/22 10:05:36 AM), 'Username' (johndoe), 'Email' (johndoe@example.com), 'Email verified' (Off), 'First name' (empty), 'Last name' (empty), and 'Enabled' (On). Below these fields is a 'Required user actions' section with a 'Select action' dropdown containing 'Update Password'. At the bottom of the page are 'Save' and 'Revert' buttons.

- If the administrator and the user are in the same realm, then the administrator will be logged out and automatically logged in as the user being impersonated.

如果管理员和用户在同一个领域，那么管理员将被注销，并作为被模拟的用户自动登录。

- If the administrator and user are in different realms, the administrator will remain logged in, and additionally will be logged in as the user in that user's realm.

如果管理员和用户位于不同的领域，则管理员将保持登录状态，并且将作为该用户领域中的用户另外登录。

In both instances, the **User Account Management** page of the impersonated user is displayed.

在这两个实例中，都将显示模拟用户的“用户帐户管理”页。

Additional resources 额外资源

- For more information on assigning administration permissions, see the Admin Console Access Control chapter.

有关分配管理权限的详细信息，请参阅管理控制台访问控制一章。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/users/proc-enabling-recaptcha.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/users/proc-enabling-recaptcha.adoc)

To safeguard registration against bots, Keycloak has integration with Google reCAPTCHA.

为了保护注册不受僵尸程序的攻击，钥匙斗篷与 Google reCAPTCHA 进行了集成。

Once reCAPTCHA is enabled, you can edit `register.ftl` in your login theme to configure the placement and styling of the reCAPTCHA button on the registration page.

启用 reCAPTCHA 后，可以在登录主题中编辑 `register.ftl`，以配置 reCAPTCHA 按钮在注册页面上的位置和样式。

Procedure 程序

1. Enter the following URL in a browser:

在浏览器中输入以下 URL:

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

BASH

2. Create an API key to get your reCAPTCHA site key and secret. Note the reCAPTCHA site key and secret for future use in this procedure.

创建一个 API 密钥来获取 reCAPTCHA 站点的密钥和机密。注意 reCAPTCHA 站点的密钥和机密，以备将来在此过程中使用。



The localhost works by default. You do not have to specify a domain. 本地主机默认工作。您不必指定域

3. Navigate to the Keycloak admin console.

导航到 Keycloak 管理控制台。

4. Click **Authentication** in the menu.

单击菜单中的“身份验证”。

5. Click the **Flows** tab.

单击 Flows 选项卡。

6. Select **Registration** from the list.

从列表中选择“注册”。

7. Set the reCAPTCHA requirement to Required. This enables reCAPTCHA.

将 reCAPTCHA 需求设置为“必需”，这将启用 reCAPTCHA。

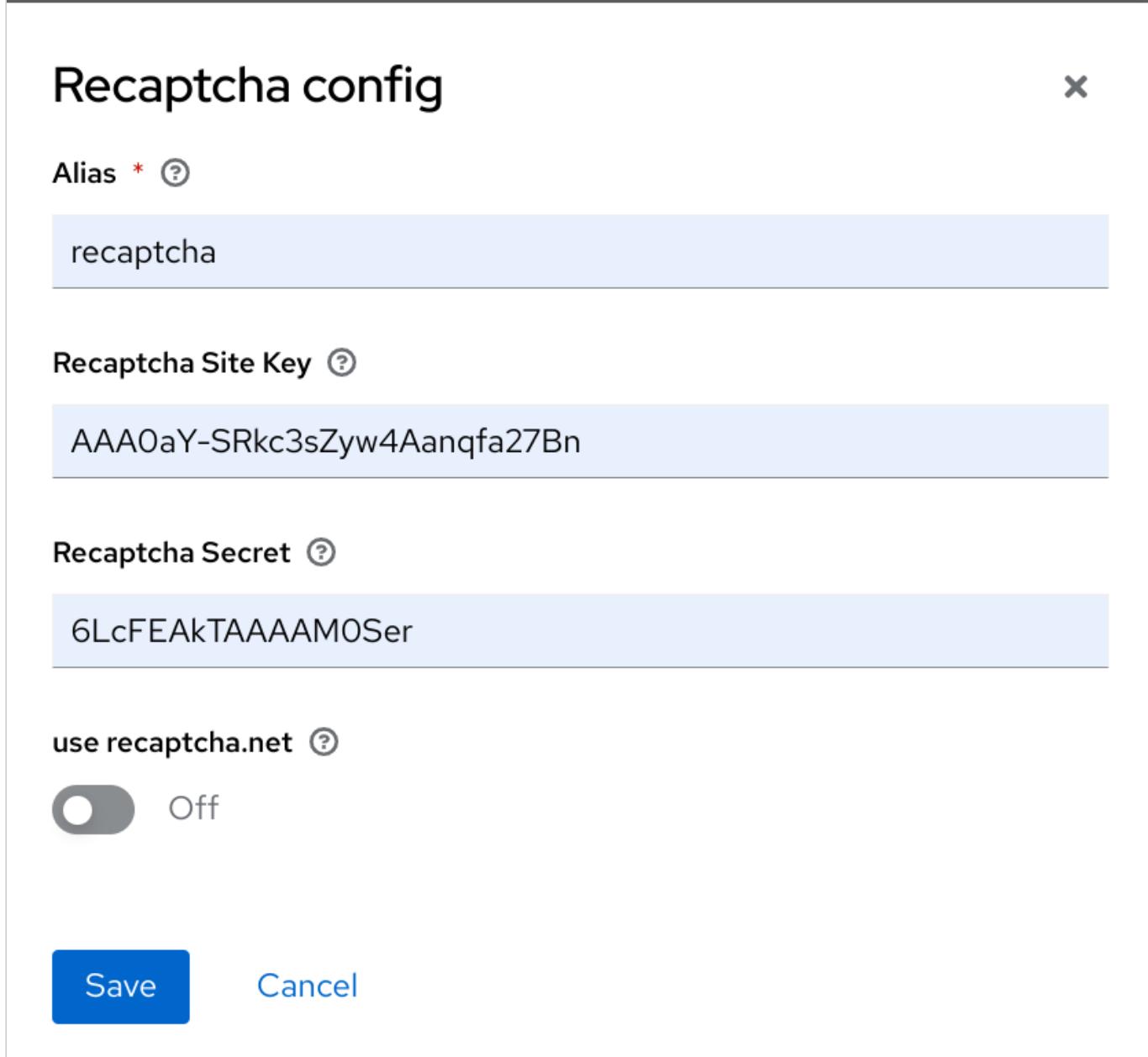
8. Click the gear icon  on the reCAPTCHA row.

单击 reCAPTCHA 行上的齿轮图标 something。

9. Click the Config link.

单击 Config 链接。

Recaptcha config page Recaptcha 配置页



The screenshot shows the 'Recaptcha config' page with the following fields:

- Alias ***: A text input field containing "recaptcha".
- Recaptcha Site Key**: A text input field containing "AAA0aY-SRkc3sZyw4Aanqfa27Bn".
- Recaptcha Secret**: A text input field containing "6LcFEAkTAAAAM0Ser".
- use recaptcha.net**: A toggle switch set to "Off".

At the bottom are two buttons: "Save" (blue) and "Cancel".

a. Enter the Recaptcha Site Key generated from the Google reCAPTCHA website.

输入从 Google reCAPTCHA 网站生成的 reCAPTCHA Site Key。

b. Enter the Recaptcha Secret generated from the Google reCAPTCHA website.

输入从 Google reCAPTCHA 网站生成的 reCAPTCHA Secret。

10. Authorize Google to use the registration page as an iframe.

授权 Google 使用注册页面作为 iframe。



In Keycloak, websites cannot include a login page dialog in an iframe. This restriction is to prevent clickjacking attacks. You need to change the default HTTP response headers that is set in Keycloak. 在 Keycloak，网站不能在 iframe 中包含登录页面对话框。此限制是为了防止点击劫持攻击。您需要更改在 Keycloak 设置的默认 HTTP 响应头

a. Click **Realm Settings** in the menu.

单击菜单中的“域设置”。

b. Click the **Security Defenses** tab.

单击“安全防御”选项卡。

c. Enter `https://www.google.com` in the field for the **X-Frame-Options** header.

在 x-Frame-Options 头部的字段中输入 https://www.google.com。

d. Enter `https://www.google.com` in the field for the **Content-Security-Policy** header.

在“内容-安全-策略”标题的字段中输入 https://www.google.com。

Additional resources 额外资源

- For more information on extending and creating themes, see the [Server Developer Guide](#) (https://www.keycloak.org/docs/latest/server_development/).

有关扩展和创建主题的更多信息，请参见服务器开发人员指南。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/users/user-profile.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/users/user-profile.adoc)

In Keycloak a user is associated with a set of attributes. These attributes are used to better describe and identify users within Keycloak as well as to pass over additional information about them to applications.

在 Keycloak，用户与一组属性相关联。这些属性可用于更好地描述和识别“钥匙斗篷”内的用户，以及向应用程序传递有关用户的其他信息。

A user profile defines a well-defined schema for representing user attributes and how they are managed within a realm. By providing a consistent view over user information, it allows administrators to control the different aspects on how attributes are managed as well as to make it much easier to extend Keycloak to support additional attributes.

用户配置文件定义了一个定义良好的模式，用于表示用户属性以及如何在领域中管理这些属性。通过提供一致的用户信息视图，它允许管理员控制属性管理的不同方面，并使得扩展 Keycloak 以支持其他属性变得更加容易。

Among other capabilities, user profile enables administrators to:

除其他功能外，用户配置文件使管理员能够：

- Define a schema for user attributes

为用户属性定义架构

- Define whether an attribute is required based on contextual information (e.g.: if required only for users, or admins, or both, or depending on the scope being requested.)

根据上下文信息定义是否需要属性(例如：如果仅对用户或管理员需要，或者两者都需要，或者取决于所请求的范围)

- Define specific permissions for viewing and editing user attributes, making possible to adhere to strong privacy requirements where some attributes can not be seen or be changed by third-parties (including administrators)

定义查看和编辑用户属性的特定权限，使得在某些属性无法被第三方(包括管理员)查看或更改的情况下，可以遵守严格的隐私要求

- Dynamically enforce user profile compliance so that user information is always updated and in compliance with the metadata and rules associated with attributes

动态强制用户配置文件遵从性，以便始终更新用户信息，并遵从与属性关联的元数据和规则

- Define validation rules on a per-attribute basis by leveraging the built-in validators or writing custom ones

通过利用内置的验证器或编写定制的验证器，在每个属性的基础上定义验证规则

- Dynamically render forms that users interact with like registration, update profile, brokering, and personal information in the account console, according to the attribute definitions and without any need to manually change themes.

根据属性定义，在帐户控制台中动态呈现用户交互的表单，如注册、更新配置文件、代理和个人信息，而不需要手动更改主题。

The User Profile capabilities are backed by the User Profile SPI. By default, these capabilities are disabled and realms are configured to use a default configuration that keeps backward compatibility with the legacy behavior.

用户配置文件功能由用户配置文件 SPI 支持。默认情况下，这些功能被禁用，领域被配置为使用与遗留行为保持向下兼容的默认配置。

The legacy behavior is about keeping the default constraints used by Keycloak when managing users root attributes such as username, email, first and last name, without any restriction on how custom attributes are managed. Regarding user flows such as registration, profile update, brokering, and managing accounts through the account console, users are restricted to use the attributes aforementioned with the possibility to change theme templates to support additional attributes.



遗留行为是关于在管理用户根属性(比如用户名、电子邮件、名和姓)时保留 Keycloak 使用的默认约束，对于如何管理自定义属性没有任何限制。关于用户流，如注册、配置文件更新、代理和通过帐户控制台管理帐户，用户受到限制，只能使用上述属性，并可能更改主题模板以支持其他属性。

If you are already using Keycloak, the legacy behavior is what you have been using so far.

如果您已经在使用钥匙斗篷，那么遗留行为就是您到目前为止一直在使用的行为。

Differently than the legacy behavior, the declarative provider gives you a lot more flexibility to define the user profile configuration to a realm through the administration console and a well-defined JSON schema.

与遗留行为不同的是，声明性提供程序通过管理控制台和定义良好的 JSON 模式为您提供了更大的灵活性，可以将用户配置文件配置定义到领域。

In the next sections, we'll be looking at how to use the declarative provider to define your own user profile configuration.

在下一节中，我们将研究如何使用声明性提供程序来定义自己的用户配置文件配置。



In the future, the legacy behavior will no longer be supported in Keycloak. Ideally, you should start looking at the new capabilities provided by the User Profile and migrate your realms accordingly.

在未来，这种遗留行为将不再在 Keycloak 得到支持。理想情况下，您应该开始查看用户配置文件提供的新功能，并相应地迁移您的领域。

Enabling the User Profile 启用用户配置文件

Declarative User Profile is **Technology Preview** and is not fully supported. This feature is disabled by default.

声明性用户配置文件是技术预览，不完全支持。默认情况下禁用此特性。



To enable start the server with `--features=preview` or `--features=declarative-user-profile`

要启用以下命令启动服务器：—— Features = 预览或—— Features = 声明性用户配置文件

In addition to enabling the `declarative_user_profile` feature, you should enable User Profile for a realm. To do that, click on the `Realm Settings` link on the left side menu and turn on the `User Profile Enabled` switch.

除了启用 Declaration _ User _ Profile 特性之外，还应该为领域启用用户配置文件。为此，单击左侧菜单上的“域设置”链接，然后打开“启用用户配置文件”开关。

The screenshot shows the 'General' tab of the 'Myrealm' realm settings. The 'User Profile Enabled' switch is turned on. Other settings shown include 'Realm ID' (myrealm), 'Display name' (empty), 'HTML Display name' (empty), 'Frontend URL' (empty), 'Require SSL' (External requests), 'User-managed access' (Off), and 'Endpoints' (OpenID Endpoint Configuration, SAML 2.0 Identity Provider Metadata).

Setting	Value
Realm ID *	myrealm
Display name	
HTML Display name	
Frontend URL	
Require SSL	External requests
User-managed access	Off
User Profile Enabled	On
Endpoints	OpenID Endpoint Configuration, SAML 2.0 Identity Provider Metadata

Once you enable it and click on the `Save` button, you can access the `User Profile` tab from where you can manage the configuration for user attributes.

一旦启用它并单击 Save 按钮，就可以访问 User Profile 选项卡，从中可以管理用户属性的配置。

By enabling the user profile for a realm, Keycloak is going to impose additional constraints on how attributes are managed based on the user profile configuration. In summary, here is the list of what you should expect when the feature is enabled:

通过为领域启用用户配置文件，Keycloak 将对基于用户配置文件配置管理属性的方式施加额外的约束。总而言之，下面列出了启用该特性时您应该期望看到的内容：

- From an administration point of view, the `Attributes` tab at the user details page will only show the attributes defined in the user profile configuration. The conditions defined on a per-attribute basis will also be taken into account when managing attributes.

从管理的角度来看，用户详细信息页面上的 `Attributes` 选项卡将只显示在用户配置文件配置中定义的属性。在管理属性时，还将考虑按属性定义的条件。

- User facing forms like registration, update profile, brokering, and personal info in the account console, are going to be rendered dynamically based on the user profile configuration. For that, Keycloak is going to rely on different templates to render these forms dynamically.

用户面对的表单，如注册、更新配置文件、代理和帐户控制台中的个人信息，将根据用户配置文件配置动态呈现。为了实现这一点，Keycloak 将依赖于不同的模板来动态呈现这些表单。

In the next topics, we'll be exploring how to manage the user profile configuration and how it affects your realm.

在下一个主题中，我们将探讨如何管理用户配置文件配置以及它如何影响您的领域。

Managing the User Profile 管理用户配置文件

The user profile configuration is managed on a per-realm basis. For that, click on the `Realm Settings` link on the left side menu and then click on the `User Profile` tab.

用户配置文件配置是在每个领域的基础上进行管理的。为此，单击左侧菜单上的“领域设置”链接，然后单击“用户配置文件”选项卡。

User Profile Tab 用户配置文件标签

Master

Enabled

Action ▾

Localization			Security defenses		Sessions	Tokens	Client policies	User profile
Attributes		Attributes group		JSON editor				
All groups	▼	Create attribute						
Name								
# username		Display name					Attribute group	
# email		\${username}					...	
# firstName		\${email}					...	
# lastName		\${firstName}					...	
# test		\${lastName}					...	
		test attribute					...	

In the **Attributes** sub-tab you have a list of the attributes currently associated with the user profile. By default, the configuration is created based on the user root attributes and each attribute is configured with some defaults in terms of validation and permissioning.

在 Attribute 子选项卡中，有一个当前与用户配置文件关联的属性列表。默认情况下，配置是基于用户根属性创建的，每个属性在验证和权限方面都配置了一些默认值。

In the **Attribute Groups** sub-tab you can manage attribute groups. An attribute group allows you to correlate attributes so that they are displayed together when rendering user facing forms.

在“属性组”子选项卡中，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窗体时将它们一起显示。

i For now, attribute groups are only used for rendering purposes but in the future they should also enable defining top-level configurations to the attributes they are linked to.

目前，属性组仅用于呈现目的，但在将来，它们还应该能够为链接到它们的属性定义顶级配置。

In the **JSON Editor** sub-tab you can view and edit the configuration using a well-defined JSON schema. Any change you make when at any other tab are reflected in the JSON configuration shown at this tab.

在 JSON Editor 子选项卡中，您可以使用定义良好的 JSON 模式查看和编辑配置。在任何其他选项卡上进行的任何更改都反映在该选项卡上显示的 JSON 配置中。

In the next section, you are going to learn how to manage the configuration from the **Attributes** sub-tab.

在下一节中，您将学习如何从 Attribute 子选项卡管理配置。

Managing Attributes 管理属性

At the **Attributes** sub-tab you can create, edit, and delete the attributes associated with the user profile.

在 Attributes subtab 中，可以创建、编辑和删除与用户配置文件关联的属性。

To define a new attribute and associate it with the user profile, click on the **Create attribute** button at the top the attribute listing.

要定义一个新属性并将其与用户配置文件关联，请单击属性列表顶部的 Create 属性按钮。

Attribute Configuration 属性配置

Realm settings > User profile > Create attribute

Create attribute

Create a new attribute

General settings

Name *	<input type="text" value="test"/>	Jump to section
Display name	<input type="text" value="test attribute"/>	General settings
Attribute group	<input type="text"/>	Permission
Enabled when	<input type="radio"/> Always <input checked="" type="radio"/> Scopes are requested	Validations
Required	<input checked="" type="checkbox"/> On	Annotations
Required for	<input checked="" type="radio"/> Both users and applications <input type="radio"/> Only users <input type="radio"/> Only applications	

Required when Always Scopes are requested

phone ✖ microprofile-jwt ✖
address ✖ Show 7

Permission

Who can edit? User Admin

Who can view? User Admin

Validations

[+ Add validator](#)

Validator name	Config

[Create](#) [Cancel](#)

When configuring the attribute you can define the following settings:

在配置属性时，可以定义以下设置：

Name 姓名

The name of the attribute.

属性的名称。

Display name 显示名称

A user-friendly name for the attribute, mainly used when rendering user-facing forms. It supports internationalization so that values can be loaded from message bundles.

属性的用户友好名称，主要用于呈现面向用户的表单。它支持国际化，因此可以从消息包装载值。

Attribute Group 属性组

The attribute group to which the attribute belongs to, if any.

属性所属的属性组(如果有的话)。

Enabled when scope 范围时启用

Allows you to define a list of scopes to dynamically enable an attribute. If not set, the attribute is always enabled and its constraints are always enforced when managing user profiles as well as when rendering user-facing forms. Otherwise, the same constraints only apply when any of the scopes in the list is requested by clients.

允许您定义范围列表以动态启用属性。如果未设置，则始终启用该属性，并且在管理用户配置文件以及呈现面向用户的表单时始终强制执行该属性的约束。否则，同样的约束只有在客户端请求列表中的任何作用域时才适用。

Required 需要

Set the attribute as required. If not enabled, the attribute is optional. Otherwise, the attribute must be provided by users and administrators with the possibility to also make the attribute required only for users or administrators as well as based on the scopes requested by clients.

根据需要设置属性。如果未启用，则属性是可选的。否则，该属性必须由用户和管理员提供，而且还可以根据客户端请求的范围为用户或管理员提供所需的属性。

Permission 请求许可

In this section, you can define read and write permissions for users and administrators.

在本节中，您可以为用户和管理员定义读写权限。

Validation 确认

In this section, you can define the validations that will be performed when managing the attribute value. Keycloak provides a set of built-in validators you can choose from with the possibility to add your own.

在本节中，您可以定义在管理属性值时执行的验证。Key斗篷提供了一组内置的验证器，您可以从中选择，也可以添加自己的验证器。

Annotation 注释

In this section, you can associate annotations to the attribute. Annotations are mainly useful to pass over additional metadata to frontends for rendering purposes.

在本节中，您可以将注释关联到属性。注释主要用于将额外的元数据传递到前端以便呈现。

Managing Permissions 管理权限

In the `Permission` section, you can define the level of access users and administrators have to read and write to an attribute.

在 Permission 部分，可以定义用户和管理员必须读写属性的访问级别。

Permission

Who can edit? ? User Admin

Who can view? ? User Admin

For that, you can use the following settings:

为此，您可以使用以下设置：

Can user view? 用户可以查看吗？

If enabled, users can view the attribute. Otherwise, users don't have access to the attribute.

如果启用，用户可以查看该属性。否则，用户无法访问该属性。

Can user edit? 用户可以编辑吗？

If enabled, users can view and edit the attribute. Otherwise, users don't have access to write to the attribute.

如果启用，用户可以查看和编辑属性。否则，用户无权写入该属性。

Can admin view? 可以管理视图吗？

If enabled, administrators can view the attribute. Otherwise, administrators don't have access to the attribute.

如果启用，管理员可以查看该属性。否则，管理员无法访问该属性。

Can admin edit? 管理员可以编辑吗？

If enabled, administrators can view and edit the attribute. Otherwise, administrators don't have access to write to the attribute.

如果启用，管理员可以查看和编辑该属性。否则，管理员无权写入该属性。



When you create an attribute, no permission is set to the attribute. Effectively, the attribute won't be accessible by either users or administrators. Once you create the attribute, make sure to set the permissions accordingly to that the attribute is only visible by the target audience.

创建属性时，不会对该属性设置权限。实际上，无论是用户还是管理员都无法访问该属性。创建属性之后，请确保相应地设置权限，使该属性只能被目标受众看到。

Permissioning has a direct impact on how and who can manage the attribute, as well as on how the attribute is rendered in user-facing forms.

权限直接影响到如何以及由谁来管理属性，以及如何以面向用户的表单呈现属性。

For instance, by marking an attribute as only viewable by users, the administrators won't have access to the attribute when managing users through the administration console (neither from the User API). Also, users won't be able to change the attribute when updating their profiles. An interesting configuration if user attributes are fetched from an existing identity store (federation) and you just want to make attributes visible to users without any possibility to update the attribute other than through the source identity store.

例如，通过将一个属性标记为只能被用户查看，管理员在通过管理控制台管理用户时将无法访问该属性(也不能从用户 API 中访问)。此外，用户在更新配置文件时不能更改属性。如果用户属性是从现有的标识存储(联合)中获取的，并且您只希望使属性对用户可见，而且除了通过源标识存储之外不可能更新属性，那么这是一个有趣的配置。

Similarly, you can also mark an attribute as writable only for administrators with read-only access for users. In this case, only administrators are going to be allowed to manage the attribute.

类似地，还可以将属性标记为只适用于具有用户只读访问权限的管理员的可写属性。在这种情况下，只允许管理员管理该属性。

Depending on your privacy requirements, you might also want attributes inaccessible to administrators but with read-write permissions for users.

根据您的隐私要求，您可能还希望管理员无法访问属性，但是用户具有读写权限。

Make sure to set the correct permissions whenever you add a new attribute to the user profile configuration.

在向用户配置文件配置中添加新属性时，请确保设置了正确的权限。

Managing validations 管理验证

In the `Validation` section, you can choose from different forms of validation to make sure the attribute value conforms to specific rules.

在验证部分，您可以从不同的验证形式中进行选择，以确保属性值符合特定的规则。

Attribute Validation 属性验证

Validations	
Validator name	Config
local-date	Delete

Keycloak provides different validators out of the box:

钥匙斗篷提供了不同的验证程序：

Name 姓名	Description 描述	Configuration 配置
length 长度	Check the length of a string value based on a minimum and maximum length. 根据最小和最大长度检查字符串值的长度。	min: an integer to define the minimum allowed length. Min: 定义最小允许长度的整数。 max: an integer to define the maximum allowed length. Max: 定义最大允许长度的整数。 trim-disabled: a boolean to define whether the value is trimmed prior to validation. 装饰-禁用: 一个布尔值，用于定义在验证之前是否对值进行装饰。

Name 姓名	Description 描述	Configuration 配置
integer 整数	<p>Check if the value is an integer and within a lower and/or upper range. If no range is defined, the validator only checks whether the value is a valid number.</p> <p>检查该值是否是一个整数，并且在一个较低和/或较高的范围内。如果没有定义范围，验证器只检查值是否为有效数字。</p>	<p>min: an integer to define the lower range.</p> <p>Min: 定义较低范围的整数。</p> <p>max: an integer to define the upper range.</p> <p>Max: 定义上限的整数。</p>
double 双倍	<p>Check if the value is a double and within a lower and/or upper range. If no range is defined, the validator only checks whether the value is a valid number.</p> <p>检查该值是否为 double，并且在较低和/或较高的范围内。如果没有定义范围，验证器只检查值是否为有效数字。</p>	<p>min: an integer to define the lower range.</p> <p>Min: 定义较低范围的整数。</p> <p>max: an integer to define the upper range.</p> <p>Max: 定义上限的整数。</p>
uri 尤里	<p>Check if the value is a valid URI.</p> <p>检查该值是否是有效的 URI。</p>	<p>None</p> <p>没有</p>
pattern 模式	<p>Check if the value matches a specific RegEx pattern.</p> <p>检查该值是否与特定的 RegEx 模式匹配。</p>	<p>pattern: the RegEx pattern to use when validating values.</p> <p>Pattern: 验证值时使用的 RegEx 模式。</p> <p>error-message: the key of the error message in i18n bundle. If not set a generic message is used.</p> <p>Error-message: i18n 包中错误消息的键。如果没有设置，则使用通用消息。</p>

Name 姓名	Description 描述	Configuration 配置
email 电子邮件	Check if the value has a valid e-mail format. 检查该值是否具有有效的电子邮件格式。	None 没有
local-date 本地约会	Check if the value has a valid format based on the realm and/or user locale. 检查该值是否具有基于领域和/或用户区域设置的有效格式。	None 没有
person-name-prohibited-characters 人名禁用字符	Check if the value is a valid person name as an additional barrier for attacks such as script injection. The validation is based on a default RegEx pattern that blocks characters not common in person names. 检查该值是否是一个有效的人名，作为攻击(如脚本注入)的额外屏障。验证基于缺省的 RegEx 模式，该模式阻止人名中不常见的字符。	error-message: the key of the error message in i18n bundle. If not set a generic message is used. Error-message: i18n 包中错误消息的键。如果没有设置，则使用通用消息。
username-prohibited-characters 禁止使用的用户名字符	Check if the value is a valid username as an additional barrier for attacks such as script injection. The validation is based on a default RegEx pattern that blocks characters not common in usernames. 检查该值是否是一个有效的用户名，作为攻击(如脚本注入)的额外屏障。验证基于缺省的 RegEx 模式，该模式阻止用户名中不常见的字符。	error-message: the key of the error message in i18n bundle. If not set a generic message is used. Error-message: i18n 包中错误消息的键。如果没有设置，则使用通用消息。

Name 姓名	Description 描述	Configuration 配置
options 选择	<p>Check if the value is from the defined set of allowed values. Useful to validate values entered through select and multiselect fields.</p> <p>检查该值是否来自已定义的允许值集。用于验证通过 select 和 multiselect 字段输入的值。</p>	<p>options: array of strings containing allowed values.</p> <p>选项: 包含允许值的字符串数组。</p>

Managing annotations 管理注释

In order to pass additional information to frontends, attributes can be decorated with annotations to dictate how attributes are rendered. This capability is mainly useful when extending Keycloak themes to render pages dynamically based on the annotations associated with attributes. This mechanism is used for example to configure Form input filed for attribute.

为了向前端传递附加信息，可以用注释修饰属性，以指示如何呈现属性。这个功能主要用于基于与属性相关联的注释动态地呈现页面的 Key 斗篷主题扩展时。例如，此机制用于为属性配置 Form 输入文件。

Attribute Annotation 属性注释

Validations

 Add validator

Validator name	Config	
local-date		Delete

Annotations

Annotations	Key	Value	
	type	date	
	Type a key	Type a value	
 Add an attribute			

Managing Attribute Groups 管理属性组

At the **Attribute Groups** sub-tab you can create, edit, and delete attribute groups. An attribute group allows you to define a container for correlated attributes so that they are rendered together when at the user-facing forms.

在“属性组”子选项卡中，可以创建、编辑和删除属性组。属性组允许您为相关属性定义一个容器，以便它们在面向用户的表单上一起呈现。

Attribute Group List 属性组列表

Attributes	Attributes group	JSON editor
	Create attributes group	1 - 2 ▾ < >
Name	Display name	Display description
personallInfo	Personal Information	⋮
addressInfo	Address Information	⋮



You can't delete attribute groups that are bound to attributes. For that, you should first update the attributes to remove the binding.

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。

To create a new group, click on the **Create attributes group** button on the top of the attribute groups listing.

要创建一个新组，单击属性组清单顶部的 Create Attribute group 按钮。

Attribute Group Configuration 属性组配置

Realm settings > User profile > Create attributes group

Create attributes group

Name *	<input type="text"/>
Display name	<input type="text"/>
Display description	<input type="text"/>

Annotations

Annotations	Key	Value
	<input type="text"/> Type a key	<input type="text"/> Type a value
	+ Add an attribute	

Save **Cancel**

When configuring the group you can define the following settings:

在配置组时，可以定义以下设置：

Name 姓名

The name of the group.

组织的名称。

Display name 显示名称

A user-friendly name for the group, mainly used when rendering user-facing forms. It supports internationalization so that values can be loaded from message bundles.

用户友好的组名，主要用于呈现面向用户的表单。它支持国际化，因此可以从消息包装载值。

Display description 显示描述

A user-friendly text that will be displayed as a tooltip when rendering user-facing forms.

一种用户友好的文本，在呈现面向用户的表单时将作为工具提示显示。

Annotation 注释

In this section, you can associate annotations to the attribute. Annotations are mainly useful to pass over additional metadata to frontends for rendering purposes.

在本节中，您可以将注释关联到属性。注释主要用于将额外的元数据传递到前端以便呈现。

Using the JSON configuration 使用JSON配置

The user profile configuration is stored using a well-defined JSON schema. You can choose from editing the user profile configuration directly by clicking on the `JSON Editor` sub-tab.

用户配置文件配置使用定义良好的JSON模式存储。您可以通过单击JSON Editor子选项卡直接编辑用户配置文件配置。

JSON Configuration JSON 配置

```
1  {
2    "attributes": [
3      {
4        "name": "username",
5        "displayName": "${username}",
6        "validations": {
7          "length": {
8            "min": 3,
9            "max": 255
10         },
11         "username-prohibited-characters": {}
12       }
13     },
14     {
15       "name": "email",
16       "displayName": "${email}",
17       "validations": {
18         "email": {},
19         "length": {
20           "max": 255
21         }
22       },
23     },
24     {
25       "name": "firstName",
26       "displayName": "First Name",
27       "validations": {
28         "length": {
29           "min": 1,
30           "max": 50
31         }
32       }
33     }
34   }
35 }
```

SaveRevert

The JSON schema is defined as follows:

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"  
    }  
  ]  
}
```

JSON

The schema supports as many attributes as you need.

该架构支持您所需的任意多个属性。

For each attribute you should define a `name` and, optionally, the `required`, `permission`, and the `annotations` settings.

对于每个属性，您应该定义一个名称，并可选地定义必需的、权限和注释设置。

Required property 需要的财产

The `required` setting defines whether an attribute is required. Keycloak allows you to set an attribute as required based on different conditions.

必需的设置定义是否需要属性。钥匙斗篷允许您根据不同的条件设置所需的属性。

When the `required` setting is defined as an empty object, the attribute is always required.

当所需的设置定义为空对象时，属性始终是必需的。

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

JSON

On the other hand, you can choose to make the attribute required only for users, or administrators, or both. As well as mark the attribute as required only in case a specific scope is requested when the user is authenticating in Keycloak.

另一方面，您可以选择只为用户、管理员或两者提供所需的属性。并且只有在用户在 Keycloak 进行身份验证时请求特定范围时才将该属性标记为必需。

To mark an attribute as required for a user and/or administrator, set the `roles` property as follows:

若要将属性标记为用户和/或管理员所需的，请按以下方式设置角色属性：

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

JSON

The `roles` property expects an array whose values can be either `user` or `admin`, depending on whether the attribute is required by the user or the administrator, respectively.

Role 属性需要一个数组，其值可以是 `user`，也可以是 `admin`，具体取决于用户或管理员分别需要该属性。

Similarly, you can choose to make the attribute required when a set of one or more scopes is requested by a client when authenticating a user. For that, you can use the `scopes` property as follows:

类似地，在对用户进行身份验证时，当客户端请求一组或多个范围时，您可以选择使该属性成为必需的。为此，可以使用 `scope` 属性如下：

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

JSON

The `scopes` property is an array whose values can be any string representing a client scope.

Scope 属性是一个数组，其值可以是表示客户端范围的任何字符串。

Permissions property 许可财产

The attribute-level `permissions` property can be used to define the read and write permissions to an attribute. The permissions are set based on whether these operations can be performed on the attribute by a user, or administrator, or both.

属性级权限属性可用于定义对属性的读写权限。权限的设置基于这些操作是否可以由用户、管理员或两者对属性执行。

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

JSON

Both `view` and `edit` properties expect an array whose values can be either `user` or `admin`, depending on whether the attribute is viewable or editable by the user or the administrator, respectively.

视图和编辑属性都需要一个数组，其值可以是用户或管理员，具体取决于属性分别由用户或管理员查看或编辑。

When the `edit` permission is granted, the `view` permission is implicitly granted.

授予编辑权限时，将隐式授予视图权限。

Annotations property 注释属性

The attribute-level `annotation` property can be used to associate additional metadata to attributes. Annotations are mainly useful for passing over additional information about attributes to frontends rendering user attributes based on the user profile configuration. Each annotation is a key/value pair.

属性级注释属性可用于将其他元数据关联到属性。注释主要用于传递关于属性的附加信息，以便根据用户配置文件配置预先呈现用户属性。每个注释都是一个键/值对。

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

Using dynamic forms 使用动态表单

One of the main capabilities of User Profile is the possibility to dynamically render user-facing forms based on attributes metadata. When you have the feature enabled to your realm, forms like registration and update profile are rendered using specific theme templates to dynamically render pages based on the user profile configuration.

User Profile 的主要功能之一是可以基于属性元数据动态呈现面向用户的表单。当您为您的领域启用了这个特性时，注册和更新配置文件之类的表单将使用特定的主题模板呈现，以根据用户配置文件配置动态呈现页面。

That said, you shouldn't need to customize templates at all if the default rendering mechanisms serve to your needs. In case you still need customizations to themes, here are the templates you should be looking at:

也就是说，如果默认呈现机制能够满足您的需要，那么您根本就不需要定制模板。如果您仍然需要对主题进行自定义，以下是您应该关注的模板：

Template 模板	Description 描述
base/login/update-user-profile.ftl	The template that renders the update profile page.
Base/login/update-user-profile.ftl	呈现更新配置文件页的模板。
base/login/register-user-profile.ftl	The template that renders the registration page.
Base/login/register-user-profile.ftl	呈现注册页的模板。

Template 模板	Description 描述
base/login/idp-review-user-profile.ftl Base/login/idp-review-user-profile. ftl	The template that renders the page to review/update the user profile when federating users through brokering. 在通过代理联合用户时，呈现页面以查看/更新用户配置文件的模板。
base/login/user-profile-commons.ftl Base/login/user-profile-commons. ftl	The template that renders input fields in forms based on attributes configuration. Used from all three page templates described above. New input types can be implemented here. 基于属性配置以表单形式呈现输入字段的模板。从上面描述的所有三个页面模板中使用。可以在这里实现新的输入类型。

The default rendering mechanism provides the following capabilities:

默认呈现机制提供以下功能:

- Dynamically display fields based on the permissions set to attributes.
根据属性设置的权限动态显示字段。
- Dynamically render markers for required fields based on the constraints set to the attributes.
根据属性设置的约束动态呈现所需字段的标记。
- Dynamically render field input type (text, date, number, select, multiselect) set to an attribute.
动态呈现字段输入类型(text、 date、 number、 select、 multiselect)设置为属性。
- Dynamically render read-only fields depending on the permissions set to an attribute.
根据属性设置的权限动态呈现只读字段。
- Dynamically order fields depending on the order set to the attributes.
根据属性设置的顺序动态排序字段。
- Dynamically group fields that belong to a same attribute group.
对属于同一属性组的字段进行动态分组。

Ordering attributes 排列属性

The attributes order is set by dragging and dropping the attribute rows on the attribute listing page.

通过拖放属性列表页上的属性行来设置属性顺序。

Ordering Attributes 排序属性

Attributes	Attributes group	JSON editor
All groups		Create attribute
Attribute group		
username	username	`\${username}`
email	email	`\${email}`
firstName	firstName	`\${firstName}`
lastName	lastName	`\${lastName}`

The order you set in this page is respected when fields are rendered in dynamic forms.

当字段以动态形式呈现时，将尊重在此页中设置的顺序。

Grouping attributes 属性分组

When dynamic forms are rendered, they will try to group together attributes that belong to a same attribute group.

呈现动态表单时，它们将尝试将属于同一属性组的属性组合在一起。

Dynamic Update Profile Form 动态更新配置文件表格

* Required fields

Update Account Information

Email *

alice@keycloak.org

Personal Information

First name *

John

Last name *

Doe

Date of Birth

|

Address Information

Address *



Please specify this field.

Postal Code *



Please specify this field.

When attributes are linked to an attribute group, the attribute order is also important to make sure attributes within the same group are close together, within a same group header. Otherwise, if attributes within a group do not have a sequential order you might have the same group header rendered multiple times in the dynamic form.



当属性链接到一个属性组时，属性顺序对于确保同一组中的属性在同一组标头内靠得很近也很重要。否则，如果组中的属性没有顺序顺序，则可能会以动态形式多次呈现相同的组标头。

Configuring Form input filed for Attributes 配置为属性归档的表单输入

Keycloak provides built-in annotations to configure which input type will be used for the attribute in dynamic forms and other aspects of its visualization.

Keycloak 提供了内置的注释，用于配置将用于动态表单中的属性的输入类型以及属性可视化的其他方面。

Available annotations are:

现有注释如下：

Name 姓名	Description 描述
inputType 输入类型	Type of the form input field. Available types are described in a table below. 表单输入字段的类型。可用类型在下表中描述。
inputHelperTextBefore	Helper text rendered before (above) the input field. Direct text or internationalization pattern (like \${i18n.key}) can be used here. Text is NOT html escaped when rendered into the page, so you can use html tags here to format the text, but you also have to correctly escape html control characters. 在输入字段之前(上面)呈现的助手文本。这里可以使用直接文本或国际化模式(如 \${i18n.key})。文本在呈现到页面时不会转义 html，所以您可以在使用 html 标记来格式化文本，但是您也必须正确地转义 html 控件字符。

Name 姓名	Description 描述
inputHelperTextAfter	<p>Helper text rendered after (under) the input field. Direct text or internationalization pattern (like \${i18n.key}) can be used here. Text is NOT html escaped when rendered into the page, so you can use html tags here to format the text, but you also have to correctly escape html control characters.</p> <p>在输入字段之后(之下)呈现的助手文本。这里可以使用直接文本或国际化模式(如 \${i18n.key})。文本在呈现到页面时不会转义 html，所以您可以在这里使用 html 标记来格式化文本，但是您也必须正确地转义 html 控件字符。</p>
inputOptionsFromValidation InputOptionsFrom 验证	<p>Annotation for select and multiselect types. Optional name of custom attribute validation to get input options from. See detailed description below.</p> <p>选择和多选类型的注释。获取输入选项的自定义属性验证的可选名称。请参阅下面的详细描述。</p>
inputOptionLabelsI18nPrefix InputOptionLabelsI18nPrefix 输入选项标签 I18nPrefix	<p>Annotation for select and multiselect types. Internationalization key prefix to render options in UI. See detailed description below.</p> <p>选择和多选类型的注释。用于在 UI 中呈现选项的国际化键前缀。请参阅下面的详细说明。</p>
inputOptionLabels InputOptionLabels 输入选项标签	<p>Annotation for select and multiselect types. Optional map to define UI labels for options (directly or using internationalization). See detailed description below.</p> <p>选择和多选类型的注释。为选项定义 UI 标签的可选映射(直接或使用国际化)。请参阅下面的详细描述。</p>

Name 姓名	Description 描述
inputTypePlaceholder 输入类型占位符	<p>HTML input <code>placeholder</code> attribute applied to the field - specifies a short hint that describes the expected value of an input field (e.g. a sample value or a short description of the expected format). The short hint is displayed in the input field before the user enters a value.</p> <p>应用于字段的 HTML 输入占位符属性-指定描述输入字段预期值的简短提示(例如样本值或预期格式的简短描述)。在用户输入值之前，会在输入字段中显示简短提示。</p>
inputTypeSize 输入类型大小	<p>HTML input <code>size</code> attribute applied to the field - specifies the width, in characters, of a single line input field. For fields based on HTML <code>select</code> type it specifies number of rows with options shown. May not work, depending on css in used theme!</p> <p>应用于字段的 HTML 输入大小属性-指定单行输入字段的宽度(以字符为单位)。对于基于 HTML 选择类型的字段，它指定显示选项的行数。可能不工作，取决于在使用的 CSS 主题！</p>
inputTypeCols InputTypeCols 输入类型	<p>HTML input <code>cols</code> attribute applied to the field - specifies the width, in characters, for <code>textarea</code> type. May not work, depending on css in used theme!</p> <p>HTML 输入 <code>protocol</code> 属性应用于字段-指定文本区类型的宽度(以字符为单位)。可能不工作，取决于在使用的 CSS 主题！</p>

Name 姓名	Description 描述
inputTypeRows	<p>HTML input rows attribute applied to the field - specifies the height, in characters, for textarea type. For select fields it specifies number of rows with options shown. May not work, depending on css in used theme!</p> <p>HTML 输入行属性应用到字段-指定文本区类型的高度(以字符为单位)。对于 select 字段，它指定显示选项的行数。可能不工作，取决于在使用的 CSS 主题！</p>
inputTypePattern 输入类型模式	<p>HTML input pattern attribute applied to the field providing client side validation - specifies a regular expression that an input field's value is checked against. Useful for single line inputs.</p> <p>HTML 输入模式属性应用于提供客户端验证的字段-指定一个正则表达式，用于检查输入字段的值。用于单线输入。</p>
inputTypeMaxLength 输入类型最大长度	<p>HTML input maxlength attribute applied to the field providing client side validation - maximal length of the text which can be entered into the input field. Useful for text fields.</p> <p>HTML input maxlength 属性应用于提供客户端验证的字段-可以输入到输入字段中的文本的最大长度。对文本字段有用。</p>
inputTypeMinLength 输入类型最小长度	<p>HTML input minlength attribute applied to the field providing client side validation - minimal length of the text which can be entered into the input field. Useful for text fields.</p> <p>HTML 输入 minlength 属性应用于提供客户端验证的字段-可以输入到输入字段的文本的最小长度。对文本字段有用。</p>

Name 姓名	Description 描述
inputTypeMax 输入类型最大化	HTML input <code>max</code> attribute applied to the field providing client side validation - maximal value which can be entered into the input field. Useful for numeric fields. HTML 输入 <code>max</code> 属性应用于字段提供客户端验证-最大值可以输入到输入字段。对数字字段有用。
inputTypeMin 输入输入法	HTML input <code>min</code> attribute applied to the field providing client side validation - minimal value which can be entered into the input field. Useful for numeric fields. HTML 输入 <code>min</code> 属性应用于提供客户端验证的字段-可以输入到输入字段的最小值。对数字字段有用。
inputTypeStep 输入类型步骤	HTML input <code>step</code> attribute applied to the field - Specifies the interval between legal numbers in an input field. Useful for numeric fields. 应用于字段的 HTML 输入 <code>step</code> 属性-指定输入字段中合法数字之间的间隔。对数字字段有用。

Field types use HTML form field tags and attributes applied to them - they behave based on the HTML specifications and browser support for them.



字段类型使用 HTML 表单字段标记和应用于它们的属性——它们的行为基于 HTML 规范和浏览器对它们的支持。

Visual rendering also depends on css styles applied in the used theme.

视觉呈现还取决于在所用主题中应用的 css 样式。

Available `inputType` annotation values:

可用的 `inputType` 注释值:

Name 姓名	Description 描述	HTML tag used 使用 HTML 标记

Name 姓名	Description 描述	HTML tag used 使用 HTML 标记
text 短信	Single line text input. 单行文本输入。	input 输入
textarea 文字区	Multiple line text input. 多行文本输入。	textarea 文字区
select 选择	Common single select input. See description how to configure options below. 常见的单选输入。请参阅下面的说明如何配置选项。	select 选择
select-radiobuttons 选择-单选按钮	Single select input through group of radio buttons. See description how to configure options below. 通过一组单选按钮进行单选输入。请参阅下面的说明如何配置选项。	group of input 输入组输入组
multiselect 多重选择	Common multiselect input. See description how to configure options below. 常见的多选输入。请参阅下面的说明如何配置选项。	select 选择
multiselect-checkboxes 多重选择复选框	Multiselect input through group of checkboxes. See description how to configure options below. 通过一组复选框多选输入。请参阅下面如何配置选项的说明。	group of input 输入组输入组

Name 姓名	Description 描述	HTML tag used 使用 HTML 标记
html5-email Html5-电子邮件	Single line text input for email address based on HTML 5 spec. 基于 HTML5 规范的电子邮件地址单行文本输入。	input 输入
html5-tel	Single line text input for phone number based on HTML 5 spec. 基于 HTML5 规范的电话号码单行文本输入。	input 输入
html5-url	Single line text input for URL based on HTML 5 spec. 基于 HTML5 规范的 URL 单行文本输入。	input 输入
html5-number Html5-数字	Single line input for number (integer or float depending on step) based on HTML 5 spec. 基于 HTML5 规范的单行数字输入 (整数或浮点数取决于步长)。	input 输入
html5-range	Slider for number entering based on HTML 5 spec. 基于 HTML5 规范的数字输入滑块。	input 输入
html5-datetime-local	Date Time input based on HTML 5 spec. 基于 HTML5 规范的日期时间输入。	input 输入
html5-date Html5-日期	Date input based on HTML 5 spec. 基于 HTML5 规范的日期输入。	input 输入

Name 姓名	Description 描述	HTML tag used 使用 HTML 标记
html5-month Htm5个月	Month input based on HTML 5 spec. 基于 HTML5 规范的月份输入。	input 输入
html5-week Htm5周	Week input based on HTML 5 spec. 基于 HTML5 规范的周输入。	input 输入
html5-time	Time input based on HTML 5 spec. 基于 HTML5 规范的时间输入。	input 输入

Defining options for select and multiselect fields 为选择和多选字段定义选项

Options for select and multiselect fields are taken from validation applied to the attribute to be sure validation and field options presented in UI are always consistent. By default, options are taken from built-in `options` validation.

选择和多选字段的选项取自应用于属性的验证，以确保验证和 UI 中提供的字段选项始终保持一致。默认情况下，选项取自内置选项验证。

You can use various ways to provide nice human-readable labels for select and multiselect options. The simplest case is when attribute values are same as UI labels. No extra configuration is necessary in this case.

您可以使用各种方法为选择和多选选项提供良好的人类可读的标签。最简单的情况是当属性值与 UI 标签相同时。在这种情况下不需要额外的配置。

Option values same as UI labels 选项值与 UI 标签相同

Validations

[+ Add validator](#)

Validator na...	Config	
options	{"options":["SW Engineer", "SW architect"]}	Delete

Annotations

Annotations	Key	Value	
	inputType	select	-

[+ Add an attribute](#)

When attribute value is kind of ID not suitable for UI, you can use simple internationalization support provided by `inputOptionLabelsI18nPrefix` annotation. It defines prefix for internationalization keys, option value is dot appended to this prefix.

当属性值的 ID 类型不适合 UI 时，可以使用 `inputOptionLabelsI18nPrefix` 注释提供的简单国际化支持。它定义了国际化键的前缀，选项值是点附加到这个前缀。

Simple internationalization for UI labels using i18n key prefix 使用 i18n 键前缀的 UI 标签的简单国际化

Validations

Validator name

 Add validator

Validator n... Config

options	{"options":["SW Engineer","SW architect"]}	
---------	--	---

Annotations

Annotations	Key	Value	
	inputType	select	
	inputOptionLabelsI18nPrefix	userprofile.jobtitle	
 Add an attribute			

Localized UI label texts for option value have to be provided by `userprofile.jobtitle.sweng` and `userprofile.jobtitle.swarch` keys then, using common localization mechanism.

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，使用公共本地化机制。

You can also use `inputOptionLabels` annotation to provide labels for individual options. It contains map of labels for option - key in the map is option value (defined in validation), and value in the map is UI label text itself or its internationalization pattern (like `${i18n.key}`) for that option.

还可以使用 `inputOptionLabels` 注释为各个选项提供标签。它包含选项的标签映射——映射中的选项键是选项值(在验证中定义)，映射中的值是该选项的 UI 标签文本本身或其国际化模式(如 `${i18n.key}`)。



You have to use User Profile [JSON Editor](#) to enter map as `inputOptionLabels` annotation value.

必须使用 UserProfileJSON 编辑器输入 map 作为 `inputOptionLabels` 注释值。

Example of directly entered labels for individual options without internationalization:

没有国际化的单个选项的直接输入标签示例:

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

Example of the internationalized labels for individual options:

单个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}"
      }
    }
  }
]
...
]

```

Localized texts have to be provided by `jobtitle.swengineer` and `jobtitle.swarchitect` keys then, using common localization mechanism.

然后，必须使用公共本地化机制，由 `jobtitle.swengineering` 和 `jobtitle.swarchitect` 键提供本地化文本。

Custom validator can be used to provide options thanks to `inputOptionsFromValidation` attribute annotation. This validation have to have `options` config providing array of options. Internationalization works the same way as for options provided by built-in `options` validation.

由于 `inputOptionsFromValidation` 属性注释，可以使用自定义验证器来提供选项。此验证必须具有提供选项数组的选项配置。国际化的工作方式与内置选项验证提供的选项相同。

Options provided by custom validator 自定义验证程序提供的选项

Attribute **jobTitle** configuration

Name ⓘ	<input type="text" value="jobTitle"/>
Display name ⓘ	<input type="text" value="Job Title"/>
Attribute Group ⓘ	<input type="button" value="▼"/>
Enabled when scope ⓘ	<input type="text" value="Select a scope..."/>
Required ⓘ	<input type="button" value="OFF"/>

> Permission

▼ Validation

Add Validator ⓘ	<input type="button" value="Add Validator..."/>	<input type="button" value="▼"/>
-----------------	---	----------------------------------

Name	Config	Actions
customOptionsValidator	{"options":["SW Engineer","SW architect"]}	Delete

▼ Annotation

Key	Value	Actions
inputOptionsFromValidation	customOptionsValidator	Delete
inputType	select	Delete
		Add

Forcing User Profile compliance 强制遵从用户配置文件

In order to make sure user profiles are in compliance with the configuration, administrators may use the `VerifyProfile` required action to eventually force users to update their profiles when authenticating to Keycloak.

为了确保用户配置文件与配置保持一致，管理员可以使用 VerifyProfile 所需的操作来最终强制用户在进行钥匙斗篷身份验证时更新他们的配置文件。



The `VerifyProfile` action is similar to the `UpdateProfile` action. However, it leverages all the capabilities provided by the user profile to automatically enforce compliance with the user profile configuration.

`VerifyProfile` 操作类似于 `UpdateProfile` 操作。但是，它利用用户配置文件提供的所有功能来自动强制遵从用户配置文件配置。

When enabled, the `VerifyProfile` action is going to perform the following steps when the user is authenticating:

启用后，当用户进行身份验证时，`VerifyProfile` 操作将执行以下步骤：

- Check whether the user profile is fully compliant with the user profile configuration set to the realm.
检查用户配置文件是否完全符合设置为领域的用户配置文件配置。
- If not, perform an additional step during the authentication so that the user can update any missing or invalid attribute.
如果没有，则在身份验证期间执行附加步骤，以便用户可以更新任何丢失或无效的属性。
- If the user profile is compliant with the configuration, no additional step is performed, and the user continues with the authentication process.
如果用户配置文件与配置兼容，则不执行其他步骤，用户继续执行身份验证过程。

By default, the `VerifyProfile` action is disabled. To enable it, click on the `Authentication` link on the left side menu and then click on the `Required Actions` tab. At this tab, select the `Enabled` switch of the `VerifyProfile` action.

默认情况下，`VerifyProfile` 操作是禁用的。要启用它，请单击左侧菜单上的“身份验证”链接，然后单击“必要操作”选项卡。在此选项卡上，选择 `VerifyProfile` 操作的 `Enable` 开关。

Registering the VerifyProfile Required Action 注册 `VerifyProfile` 所需的操作

Required actions	Enabled	Set as default action
Configure OTP	<input checked="" type="checkbox"/> On	<input type="checkbox"/> Off
Terms and Conditions	<input type="checkbox"/> Off	<input type="checkbox"/> Disabled off
Update Password	<input checked="" type="checkbox"/> On	<input type="checkbox"/> Off
Update Profile	<input checked="" type="checkbox"/> On	<input type="checkbox"/> Off
Verify Email	<input checked="" type="checkbox"/> On	<input type="checkbox"/> Off
Delete Account	<input type="checkbox"/> Off	<input type="checkbox"/> Disabled off
Update User Locale	<input checked="" type="checkbox"/> On	<input type="checkbox"/> Off
Verify Profile	<input checked="" type="checkbox"/> On	<input type="checkbox"/> Off
Webauthn Register Passwordless	<input type="checkbox"/> Off	<input type="checkbox"/> Disabled off
Webauthn Register	<input type="checkbox"/> Off	<input type="checkbox"/> Disabled off

Migrating to User Profile 迁移到用户配置文件

Before enabling the User Profile capabilities to a realm, there are some important considerations you should be aware of. By providing a single place to manage attribute metadata, the feature is very strict about the attributes that can be set to users and how they are managed.

在将用户配置文件功能启用到领域之前，有一些重要的注意事项您应该注意。通过提供一个单独的地方来管理属性元数据，该特性对可以设置给用户的属性以及如何管理这些属性非常严格。

In terms of user management, administrators are able to manage only the attributes defined in the user profile configuration. Any other attribute set to the user and not yet defined in the user profile configuration won't be accessible. It is recommended to update your user profile configuration with all the user attributes you want to expose either to users or administrators.

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。无法访问设置给用户但尚未在用户配置文件配置中定义的任何其他属性。建议使用要向用户或管理员公开的所有用户属性更新用户配置文件配置。

The same recommendation applies for those accessing the User REST API to query user information.

同样的建议也适用于那些访问用户 REST API 以查询用户信息的用户。

In regards to Keycloak internal user attributes such as `LDAP_ID`, `LDAP_ENTRY_DN`, or `KERBEROS_PRINCIPAL`, if you want to be able to access those attributes you should have them as attributes in your user profile configuration. The recommendation is to mark these attributes as viewable only to administrators so that you can look at them when managing the user attributes through the administration console or querying users via User API.

至于 Keycloak 的内部用户属性，如 `LDAP_ID`、`LDAP_ENTRY_DN` 或 `KERBEROS_PRINCIPAL`，如果你想访问这些属性，你应该在你的用户配置文件中将它们作为属性。建议将这些属性标记为仅对管理员可见，以便在通过管理控制台管理用户属性或通过 User API 查询用户时可以查看它们。

In regards to theming, if you already have customizations to the legacy templates (those hardcoded with user root attributes) your custom templates won't be used when rendering user-facing forms but the new templates that render these forms dynamically. Ideally, you should avoid having any customizations to templates and try to stick with the behavior provided by these new templates to dynamically render forms for you. If they are still not enough to address your requirements, you can either customize them or provide us with any feedback so that we discuss whether it makes sense to enhance the new templates.

关于主题化，如果您已经对遗留模板进行了自定义(那些用用户根属性硬编码的模板)，那么在呈现面向用户的表单时将不会使用自定义模板，而是使用动态呈现这些表单的新模板。理想情况下，您应该避免对模板进行任何自定义，并尝试坚持使用这些新模板提供的行为来为您动态呈现表单。如果它们仍然不足以满足您的需求，您可以自定义它们，或者向我们提供任何反馈，以便我们讨论增强新模板是否有意义。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/users/ref-personal-data-collected.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/users/ref-personal-data-collected.adoc)

By default, Keycloak collects the following data:

默认情况下，“钥匙斗篷”收集以下数据：

- Basic user profile data, such as the user email, first name, and last name.

基本用户配置文件数据，如用户电子邮件、名和姓。

- Basic user profile data used for social accounts and references to the social account when using a social login.

用于社交帐户的基本用户配置文件数据以及使用社交登录时对社交帐户的引用。

- Device information collected for audit and security purposes, such as the IP address, operating system name, and the browser name.

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。

The information collected in Keycloak is highly customizable. The following guidelines apply when making customizations:

在 Keycloak 收集的资料是高度可自订的，以下指引适用于自订资料：

- Registration and account forms can contain custom fields, such as birthday, gender, and nationality. An administrator can configure Keycloak to retrieve data from a social provider or a user storage provider such as LDAP.

注册和帐户表单可以包含自定义字段，如生日、性别和国籍。管理员可以配置 Keycloak 从社交提供者或用户存储提供者(如 LDAP)检索数据。

- Keycloak collects user credentials, such as password, OTP codes, and WebAuthn public keys. This information is encrypted and saved in a database, so it is not visible to Keycloak administrators. Each type of credential can include non-confidential metadata that is visible to administrators such as the algorithm that is used to hash the password and the number of hash iterations used to hash the password.

密钥斗篷收集用户凭据，如密码、OTP 代码和 WebAuthn 公钥。此信息被加密并保存在数据库中，因此对于钥匙斗篷管理员来说是不可见的。每种类型的凭据可以包括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次数。

- With authorization services and UMA support enabled, Keycloak can hold information about some objects for which a particular user is the owner.

在启用授权服务和 UMA 支持的情况下，Keycloak 可以保存某个特定用户是其所有者的某些对象的信息。

[Edit this section](#) 编辑这部分

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/sessions.adoc)

[Report an issue](#) 报告一个问题

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/sessions.adoc)

When users log into realms, Keycloak maintains a user session for each user and remembers each client visited by the user within the session. Realm administrators can perform multiple actions on each user session:

当用户登录到领域时，Keycloak 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。域管理员可以在每个用户会话上执行多个操作：

- View login statistics for the realm.
查看领域的登录统计信息。
- View active users and where they logged in.
查看活动用户及其登录位置。
- Log a user out of their session.

从用户的会话中注销用户。

- Revoke tokens.
撤销令牌。
- Set up token timeouts.
设置令牌超时。
- Set up session timeouts.
设置会话超时。

[Edit this section](#) 编辑这部分

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/sessions/administering.adoc)

[Report an issue](#) 报告一个问题

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/sessions/administering.adoc)

To see a top-level view of the active clients and sessions in Keycloak, click **Sessions** from the menu.

要查看 Keycloak 活动客户端和会话的顶层视图，请单击菜单中的“会话”。

Sessions 会议

User	Started	Last access
admin	5/24/2022, 11:30:12 AM	5/24/2022, 11:30:12 AM

Signing out all active sessions 退出所有活动会话

You can sign out all users in the realm. From the **Action** list, select **Sign out all active sessions**. All SSO cookies become invalid. Keycloak notifies clients by using the Keycloak OIDC client adapter of the logout event. Clients requesting authentication within active browser sessions must log in again. Client types such as SAML do not receive a back-channel logout request.

您可以注销领域中的所有用户。从“操作”列表中，选择“签出所有活动会话”。所有 SSO Cookie 都无效。键隐形通过使用注销事件的键隐形 OIDC 客户端适配器通知客户端。在活动浏览器会话中请求身份验证的客户端必须再次登录。诸如 SAML 之类的客户端类型不接收反向通道注销请求。

Clicking Sign out all active sessions does not revoke outstanding access tokens. Outstanding tokens must expire naturally. For clients using the Keycloak OIDC client adapter, you can push a revocation policy to revoke the token, but this does not work for other adapters.



单击“登出所有活动会话”不会撤消未执行的访问令牌。未使用的令牌必须自然过期。对于使用 Keycloak OIDC 客户端适配器的客户端，您可以推送一个撤销策略来撤销令牌，但是这对其它适配器不起作用。

Viewing client sessions 查看客户端会话

Procedure 程序

1. Click **Clients** in the menu.

单击菜单中的 Clients。

2. Click the **Sessions** tab.

单击“会话”选项卡。

3. Click a client to see that client's sessions.

单击一个客户端以查看该客户端的会话。

Client sessions 客户会话

The screenshot shows the Keycloak interface for managing clients. On the left, a sidebar menu includes 'Master', 'Manage', 'Clients' (which is selected), 'Client scopes', 'Realm roles', 'Users', 'Groups', and 'Sessions'. The main content area is titled 'Clients > Client details' for 'security-admin-console-v2' (OpenID Connect). It displays a table of sessions:

User	Started	Last access
admin	5/24/2022, 11:30:12 AM	5/24/2022, 11:36:24 AM

Viewing user sessions 查看用户会话

Procedure 程序

1. Click **Users** in the menu.

单击菜单中的“用户”。

2. Click the **Sessions** tab.

单击“会话”选项卡。

3. Click a user to see that user's sessions.

单击用户可查看该用户的会话。

User sessions 用户会话

The screenshot shows the Keycloak User details page for a user named 'admin'. The left sidebar has a 'Sessions' section selected. The main content area shows a single session entry:

Started	Last access	Clients
5/24/2022, 11:30:12 AM	5/24/2022, 11:51:25 AM	security-admin-console-v2

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/sessions/revocation.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/sessions/revocation.adoc)

If your system is compromised, you can revoke all active sessions and access tokens.

如果您的系统受到攻击，您可以撤消所有活动会话和访问令牌。

Procedure 程序

1. Click **Sessions** in the menu.

单击菜单中的 Sessions。

2. From the **Actions** list, select **Sign out all active sessions**.

从“操作”列表中，选择“签出所有活动会话”。

Revocation 撤销

The screenshot shows a 'Revocation' dialog box. The text inside the box reads:

This is a way to revoke all active sessions and access tokens. Not before means you can revoke any tokens issued before the date.

Not before

Tue May 24 2022 13:08:32 GMT+0200 (Central European Summer ...)

Buttons at the bottom of the dialog box include: Set to now, Clear, Push, and Cancel.

3. Specify a time and date where sessions or tokens issued before that time and date are invalid using this console.

使用此控制台指定在此时间和日期之前发出的会话或令牌无效的时间和日期。

- Click **Set to now** to set the policy to the current time and date.

单击 Set to now 将策略设置为当前时间和日期。

- Click **Push** to push this revocation policy to any registered OIDC client with the Keycloak OIDC client adapter.

单击 Push 将此撤销策略推送到任何具有 Keycloak OIDC 客户端适配器的注册 OIDC 客户端。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/sessions/timeouts.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/sessions/timeouts.adoc)

Keycloak includes control of the session, cookie, and token timeouts through the **Sessions** and **Tokens** tabs in the **Realm settings** menu.

“钥匙斗篷”包括通过“域设置”菜单中的“会话”和“令牌”选项卡控制会话、 cookie 和令牌超时。

Sessions tab 会议标签

Master

Enabled

Action ▾

Realm settings are settings that control the options for users, applications, roles, and groups in the current realm. [Learn more](#) ↗

◀ gin

Email

Themes

Keys

Events

Localization

Security defenses

Sessions

▶

SSO Session Settings

SSO Session Idle ⓘ 30 Minutes ▾

SSO Session Max ⓘ 10 Hours ▾

SSO Session Idle Remember Me ⓘ 0 Minutes ▾

SSO Session Max Remember Me ⓘ 0 Minutes ▾

Client session settings

Client Session Idle ⓘ 0 Minutes ▾

Client Session Max ⓘ 0 Minutes ▾

Offline session settings

Offline Session Idle ⓘ 30 Days ▾

Offline Session Max ⓘ Disabled
Limited ⓘ

Login settings

Login timeout ⓘ 30 Minutes ▾

Login action timeout ⓘ 5 Minutes ▾

Save

Revert

Configuration 配置	Description 描述
SSO Session Idle SSO 会话空闲	<p>This setting is for OIDC clients only. If a user is inactive for longer than this timeout, the user session is invalidated. This timeout value resets when clients request authentication or send a refresh token request. Keycloak adds a window of time to the idle timeout before the session invalidation takes effect. See the note later in this section.</p> <p>此设置仅适用于 OIDC 客户端。如果用户的非活动时间超过此超时时间，则该用户会话将失效。当客户端请求身份验证或发送刷新令牌请求时，此超时值将重置。在会话无效生效之前，Keycloak 会在空闲超时添加一个时间窗口。请参阅本节后面的说明。</p>
SSO Session Max 高级特别职务官员会议	<p>The maximum time before a user session expires.</p> <p>用户会话过期前的最长时间。</p>
SSO Session Idle Remember Me 空闲会话记住我	<p>This setting is similar to the standard SSO Session Idle configuration but specific to logins with Remember Me enabled. Users can specify longer session idle timeouts when they click Remember Me when logging in. This setting is an optional configuration and, if its value is not greater than zero, it uses the same idle timeout as the SSO Session Idle configuration.</p> <p>此设置类似于标准的 SSO 会话空闲配置，但特定于启用 RememberMe 的登录。当用户单击“登录时请记住我”时，可以指定更长的会话空闲超时。此设置是可选配置，如果其值不大于零，则使用与 SSO 会话空闲配置相同的空闲超时。</p>

Configuration 配置	Description 描述
SSO Session Max Remember Me 记住我	<p>This setting is similar to the standard SSO Session Max but specific to Remember Me logins. Users can specify longer sessions when they click Remember Me when logging in. This setting is an optional configuration and, if its value is not greater than zero, it uses the same session lifespan as the SSO Session Max configuration.</p> <p>此设置类似于标准 SSO 会话 Max，但特定于 RememberMe 登录。用户可以在登录时单击 RememberMe 时指定更长的会话。此设置是可选配置，如果其值不大于零，则使用与 SSO Session Max 配置相同的会话生命期。</p>
Client Session Idle 客户端会话空闲	<p>If the user is inactive for longer than this timeout, refresh token requests bump the idle timeout. This setting specifies a shorter idle timeout of refresh tokens than the session idle timeout, but users can override it for individual clients. This setting is an optional configuration and, when set to zero, uses the same idle timeout in the SSO Session Idle configuration.</p> <p>如果用户的非活动时间长于此超时时间，则刷新令牌请求将提高空闲超时时间。此设置指定的刷新令牌的空闲超时时间比会话空闲超时时间短，但用户可以为单个客户机重写它。此设置是可选配置，当设置为零时，在 SSO 会话空闲配置中使用相同的空闲超时。</p>

Configuration 配置	Description 描述
Client Session Max 客户端会话最大值	<p>The maximum time before a refresh token expires and invalidates. This setting specifies a shorter timeout of refresh tokens than the session timeout, but users can override it for individual clients. This setting is an optional configuration and, when set to zero, uses the same idle timeout in the SSO Session Max configuration.</p> <p>刷新标记过期和失效之前的最长时间。此设置指定的刷新令牌超时时间比会话超时时间短，但用户可以为单个客户机重写它。此设置是可选配置，当设置为零时，在SSO会话Max配置中使用相同的空闲超时。</p>
Offline Session Idle 脱机会话空闲	<p>This setting is for offline access. The amount of time the session remains idle before Keycloak revokes its offline token. Keycloak adds a window of time to the idle timeout before the session invalidation takes effect. See the note later in this section.</p> <p>此设置用于脱机访问。会话保持空闲的时间长度，直到“钥匙斗篷”撤消其脱机令牌。在会话无效生效之前，Keycloak会为空闲超时添加一个时间窗口。请参阅本节后面的说明。</p>
Offline Session Max Limited 离线时段 Max 有限公司	<p>This setting is for offline access. If this flag is ON, Offline Session Max can control the maximum time the offline token remains active, regardless of user activity. Client Offline Session Idle and Client Offline Session Max are enabled.</p> <p>此设置用于脱机访问。如果此标志为ON，则脱机会话最大值可以控制脱机令牌保持活动的最大时间，而不管用户活动如何。启用客户端脱机会话空闲和客户端脱机会话最大值。</p>

Configuration 配置	Description 描述
Offline Session Max 离线会话最大	This setting is for offline access, and it is the maximum time before Keycloak revokes the corresponding offline token. This option controls the maximum amount of time the offline token remains active, regardless of user activity. 此设置用于脱机访问，并且是 Keycloak 撤销相应的脱机令牌之前的最长时间。此选项控制脱机令牌保持活动的最长时间，而不管用户活动如何。
Login timeout 登录超时	The total time a logging in must take. If authentication takes longer than this time, the user must start the authentication process again. 登录所需的总时间。如果身份验证花费的时间超过这个时间，则用户必须再次启动身份验证过程。
Login action timeout 登录操作超时	The Maximum time users can spend on any one page during the authentication process. 在身份验证过程中，用户可以在任何一个页面上花费的最大时间。

Tokens tab 代币账单

Master Enabled Action ▾

Realm settings are settings that control the options for users, applications, roles, and groups in the current realm. [Learn more ↗](#)

Localization Security defenses Sessions **Tokens** Client policies User profile

General

Default Signature Algorithm RS256

Refresh tokens

Revoke Refresh Token Disabled ?

Access tokens

Access Token Lifespan	<input type="text" value="1"/>	Minutes <input type="button" value="▼"/>
②	It is recommended for this value to be shorter than the SSO session idle timeout: 30 minutes	
Access Token Lifespan For Implicit Flow ②	<input type="text" value="15"/>	Minutes <input type="button" value="▼"/>
Client Login Timeout	<input type="text" value="1"/>	Minutes <input type="button" value="▼"/>
②		

Action tokens

User-Initiated Action Lifespan ②	<input type="text" value="5"/>	Minutes <input type="button" value="▼"/>
Default Admin-Initiated Action Lifespan ②	<input type="text" value="12"/>	Hours <input type="button" value="▼"/>

Override Action Tokens

Email Verification	<input type="text"/>	Minutes <input type="button" value="▼"/>
IdP account email verification	<input type="text"/>	Minutes <input type="button" value="▼"/>
Forgot password	<input type="text"/>	Minutes <input type="button" value="▼"/>
Execute actions	<input type="text"/>	Minutes <input type="button" value="▼"/>

[Revert](#)

Configuration 配置	Description 描述
Default Signature Algorithm 默认签名算法	The default algorithm used to assign tokens for the realm. 用于为领域分配令牌的默认算法。

Configuration 配置	Description 描述
Revoke Refresh Token 撤销刷新令牌	When Enabled, Keycloak revokes refresh tokens and issues another token that the client must use. This action applies to OIDC clients performing the refresh token flow. 启用时, Keycloak 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客户机。
Access Token Lifespan 访问令牌生命周期	When Keycloak creates an OIDC access token, this value controls the lifetime of the token. 当Keycloak 创建OIDC访问令牌时, 此值控制令牌的生存期。
Access Token Lifespan For Implicit Flow 隐式流的访问令牌生命周期	With the Implicit Flow, Keycloak does not provide a refresh token. A separate timeout exists for access tokens created by the Implicit Flow. 对于隐式流, Keycloak 不提供刷新令牌。隐式流创建的访问令牌存在单独的超时。
Client login timeout 客户端登录超时	The maximum time before clients must finish the Authorization Code Flow in OIDC. 客户端必须完成OIDC中的授权代码流之前的最长时间。
User-Initiated Action Lifespan 用户发起的行动寿命	The maximum time before a user's action permission expires. Keep this value short because users generally react to self-created actions quickly. 用户操作权限过期前的最长时间。保持此值短, 因为用户通常对自创建的操作作出快速反应。

Configuration 配置	Description 描述
Default Admin-Initiated Action Lifespan 默认的管理启动操作生命周期	<p>The maximum time before an action permission sent to a user by an administrator expires. Keep this value long to allow administrators to send e-mails to offline users. An administrator can override the default timeout before issuing the token.</p> <p>管理员发送给用户的操作权限过期前的最长时间。保持此值较长，以便管理员能够向脱机用户发送电子邮件。管理员可以在发出令牌之前覆盖默认超时。</p>
Email Verification 电子邮件验证	<p>Specifies independent timeout for email verification.</p> <p>指定电子邮件验证的独立超时。</p>
IdP account email verification IDP 帐户电子邮件验证	<p>Specifies independent timeout for IdP account email verification.</p> <p>指定 IDP 帐户电子邮件验证的独立超时。</p>
Forgot password 忘了密码	<p>Specifies independent timeout for forgot password.</p> <p>指定忘记密码的独立超时。</p>
Execute actions 执行行动	<p>Specifies independent timeout for execute actions.</p> <p>指定执行操作的独立超时。</p>

For idle timeouts, a two-minute window of time exists that the session is active. For example, when you have the timeout set to 30 minutes, it will be 32 minutes before the session expires.

对于空闲超时，存在会话处于活动状态的两分钟时间窗口。例如，如果将超时时间设置为30分钟，则会话将在32分钟后到期。



This action is necessary for some scenarios in cluster and cross-data center environments where the token refreshes on one cluster node a short time before the expiration and the other cluster nodes incorrectly consider the session as expired because they have not yet received the message about a successful refresh from the refreshing node.

对于集群和跨数据中心环境中的某些场景，这个操作是必需的，在这些场景中，令牌在到期前很短的时间内刷新一个集群节点，而其他集群节点错误地认为会话已经过期，因为它们还没有收到从刷新节点成功刷新的消息。

[Edit this section](#) 编辑这部分

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/sessions/offline.adoc)

[Report an issue](#) 报告一个问题

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/sessions/offline.adoc)

During [offline access](#) (https://openid.net/specs/openid-connect-core-1_0.html#OfflineAccess) logins, the client application requests an offline token instead of a refresh token. The client application saves this offline token and can use it for future logins if the user logs out. This action is useful if your application needs to perform offline actions on behalf of the user even when the user is not online. For example, a regular data backup.

在脱机访问登录期间，客户端应用程序请求脱机令牌而不是刷新令牌。客户端应用程序保存此脱机令牌，并且在用户注销时可以将其用于以后的登录。如果应用程序需要代表用户执行脱机操作(即使用户不在线)，则此操作非常有用。例如，常规数据备份。

The client application is responsible for persisting the offline token in storage and then using it to retrieve new access tokens from the Keycloak server.

客户端应用程序负责在存储中持久化脱机令牌，然后使用它从 Keycloak 服务器检索新的访问令牌。

The difference between a refresh token and an offline token is that an offline token never expires and is not subject to the `SSO Session Idle` timeout and `SSO Session Max` lifespan. The offline token is valid after a user logout or server restart. You must use the offline token for a refresh token action at least once per thirty days or for the value of the Offline Session Idle.

刷新令牌和脱机令牌的区别在于脱机令牌永远不会过期，而且不受SSO会话空闲超时和SSO会话Max生命周期的影响。脱机令牌在用户注销或服务器重新启动后有效。对于刷新令牌操作，必须每30天至少使用一次脱机令牌，或者对于脱机会话空闲的值，必须使用脱机令牌。

If you enable Offline Session Max Limited, offline tokens expire after 60 days even if you use the offline token for a refresh token action. You can change this value, Offline Session Max, in the Admin Console.

如果启用脱机会话Max Limited，即使使用脱机令牌执行刷新令牌操作，脱机令牌也会在60天后过期。您可以在管理控制台中更改此值“脱机会话最大值”。

If you enable the Revoke Refresh Token option, you can use each offline token once only. After refresh, you must store the new offline token from the refresh response instead of the previous one.

如果启用“撤消刷新令牌”选项，则每个脱机令牌只能使用一次。刷新后，必须从刷新响应中存储新的脱机令牌，而不是前一个。

Users can view and revoke offline tokens that Keycloak grants them in the User Account Console. Administrators can revoke offline tokens for individual users in the Admin Console in the `Consents` tab. Administrators can view all offline tokens issued in the `Offline Access` tab of each client. Administrators can revoke offline tokens by setting a revocation policy.

用户可以在用户帐户控制台中查看和撤消Keycloak授予他们的脱机令牌。管理员可以在“同意”选项卡中的“管理控制台”中撤销售单个用户的脱机令牌。管理员可以查看每个客户端的“脱机访问”选项卡中发出的所有脱机令牌。管理员可以通过设置撤销策略来撤销脱机令牌。

To issue an offline token, users must have the role mapping for the realm-level `offline_access` role. Clients must also have that role in their scope. Clients must add an `offline_access` client scope as an `Optional client scope` to the role, which is done by default.

若要发出离线令牌，用户必须具有领域级离线访问角色的角色映射。客户机在其作用域中也必须具有该角色。客户端必须将离线访问客户端作用域作为可选客户端作用域添加到该角色，这在默认情况下是完成的。

Clients can request an offline token by adding the parameter `scope=offline_access` when sending their authorization request to Keycloak. The Keycloak OIDC client adapter automatically adds this parameter when you use it to access your application's secured URL (such as, http://localhost:8080/customer-portal/secured?scope=offline_access). The Direct Access Grant and Service Accounts support offline tokens if you include `scope=offline_access` in the authentication request body.

客户端可以通过添加参数`scope = off _ access`来请求离线令牌。当用户使用该参数访问应用程序的受保护URL(比如，http://localhost:8080/customer-portal/secured?scope=offline_access)时，Keycloak客户端适配器会自动添加该参数。直接访问授权和服务帐户支持脱机令牌，如果您在身份验证请求正文中包含`scope = off _ Access`。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/sessions/preloading.adoc)

[Report an issue 报告一个问题](#)

In addition to Infinispan caches, offline sessions are stored in a database which means they will be available even after server restart. By default, the offline sessions are not preloaded from the database into the Infinispan caches during the server startup, because this approach has a drawback if there are many offline sessions to be preloaded. It can significantly slow down the server startup time. Therefore, the offline sessions are lazily fetched from the database by default.

除了 Infinispan 缓存之外，脱机会话存储在数据库中，这意味着即使在服务器重新启动之后它们也是可用的。默认情况下，在服务器启动期间，脱机会话不会从数据库预加载到 Infinispan 缓存中，因为如果有许多脱机会话需要预加载，这种方法有一个缺点。它可以显著降低服务器启动时间。因此，默认情况下将从数据库延迟地获取脱机会话。

However, Keycloak can be configured to preload the offline sessions from the database into the Infinispan caches during the server startup. It can be achieved by setting `preloadOfflineSessionsFromDatabase` property in the `userSessions` SPI to `true`.

但是，可以配置 Keycloak 在服务器启动期间将脱机会话从数据库预加载到 Infinispan 缓存中。可以通过将 `userSessions` SPI 中的 `preloadOfflineSessionsFromDatabase` 属性设置为 `true` 来实现。

The following example shows how to configure offline sessions preloading.

下面的示例演示如何配置脱机会话预加载。

```
bin/kc.[sh|bat] start --spi-user-sessions-infinispan-preload-offline-sessions-from-database=true
```

BASH

[Edit this section](#) 编辑这部分

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/sessions/transient.adoc)

[Report an issue](#) 报告一个问题

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/sessions/transient.adoc)

You can conduct transient sessions in Keycloak. When using transient sessions, Keycloak does not create a user session after successful authentication. Keycloak creates a temporary, transient session for the scope of the current request that successfully authenticates the user. Keycloak can run protocol mappers using transient sessions after authentication.

你可以在 Keycloak 进行临时治疗。在使用瞬态会话时，在成功身份验证之后，Keycloak 不会创建用户会话。Keycloak 为当前请求的范围创建一个临时的临时会话，该会话成功地验证了用户。在身份验证之后，密钥斗篷可以使用瞬态会话运行协议映射程序。

During transient sessions, the client application cannot refresh tokens, introspect tokens, or validate a specific session. Sometimes these actions are unnecessary, so you can avoid the additional resource use of persisting user sessions. This session saves performance, memory, and network communication (in cluster and cross-data center environments) resources.

在瞬态会话期间，客户端应用程序无法刷新令牌、内省令牌或验证特定会话。有时这些操作是不必要的，因此您可以避免持久化用户会话的额外资源使用。此会话节省性能、内存和网络通信(在集群和跨数据中心环境中)资源。

[Edit this section](#) 编辑这部分

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/assembly-roles-groups.adoc)

[Report an issue](#) 报告一个问题

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/assembly-roles-groups.adoc)

Roles and groups have a similar purpose, which is to give users access and permissions to use applications. Groups are a collection of users to which you apply roles and attributes. Roles define specific applications permissions and access control.

角色和组具有类似的用途，即为用户提供访问权限和使用应用程序的权限。组是将角色和属性应用于其中的用户的集合。角色定义特定的应用程序权限和访问控制。

A role typically applies to one type of user. For example, an organization may include `admin`, `user`, `manager`, and `employee` roles. An application can assign access and permissions to a role and then assign multiple users to that role so the users have the same access and permissions. For example, the Admin Console has roles that give permission to users to access different parts of the Admin Console.

角色通常应用于一种类型的用户。例如，组织可能包括管理员、用户、经理和员工角色。应用程序可以为一个角色分配访问权限和权限，然后为该角色分配多个用户，使用户具有相同的访问权限和权限。例如，管理控制台的角色允许用户访问管理控制台的不同部分。

There is a global namespace for roles and each client also has its own dedicated namespace where roles can be defined.

角色有一个全局命名空间，每个客户端也有自己的专用命名空间，可以在其中定义角色。

[Edit this section](#) 编辑这部分

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/roles-groups/proc-creating-realm-roles.adoc)

[Report an issue](#) 报告一个问题

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/roles-groups/proc-creating-realm-roles.adoc)

Realm-level roles are a namespace for defining your roles. To see the list of roles, click **Realm Roles** in the menu.

域级别角色是用于定义角色的命名空间。要查看角色列表，请单击菜单中的“领域角色”。

The screenshot shows the Keycloak admin interface. On the left, there's a sidebar with a dropdown set to 'Master' and a list of management options: Manage, Clients, Client scopes, Realm roles (which is selected and highlighted in blue), Users, Groups, Sessions, and Events. The main content area is titled 'Realm roles' with a subtitle 'Realm roles are the roles that you define for use in the current realm.' and a 'Learn more' link. It features a search bar 'Search role by name' and a 'Create role' button. A table lists existing roles with columns for 'Role name', 'Composite', and 'Description'. The roles listed are: admin (Composite: True, Description: \${role_admin}), create-realm (Composite: False, Description: \${role_create-realm}), default-roles-master (Composite: True, Description: \${role_default-roles}), offline_access (Composite: False, Description: \${role_offline-access}), and uma_authorization (Composite: False, Description: \${role_uma_authorization}).

Procedure 程序

1. Click Create Role.

单击“创建角色”。

2. Enter a Role Name.

输入角色名。

3. Enter a Description.

输入描述。

4. Click Save.

单击“保存”。

Add role 增加角色

The screenshot shows the 'Role details' page for adding a new role. The sidebar on the left is identical to the previous screenshot, with 'Realm roles' selected. The main area shows a role named 'developer'. The 'Details' tab is active, showing the 'Role name' field containing 'developer' and a 'Description' field which is currently empty. At the bottom are 'Save' and 'Revert' buttons.

The **description** field can be localized by specifying a substitution variable with `${var-name}` strings. The localized value is configured to your theme within the themes property files. See the [Server Developer Guide](#) (https://www.keycloak.org/docs/latest/server_development/) for more details.

可以通过使用 \${ var-name } 字符串指定替换变量来本地化 description 字段。本地化值在主题属性文件中配置为您的主题。有关详细信息，请参阅服务器开发人员指南。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/roles-groups/con-client-roles.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/roles-groups/con-client-roles.adoc)

Client roles are namespaces dedicated to clients. Each client gets its own namespace. Client roles are managed under the **Roles** tab for each client. You interact with this UI the same way you do for realm-level roles.

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名称空间。客户端角色在每个客户端的 Roles 选项卡下进行管理。与此 UI 交互的方式与对领域级角色的方式相同。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/roles-groups/proc-converting-composite-roles.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/roles-groups/proc-converting-composite-roles.adoc)

Any realm or client level role can become a *composite role*. A *composite role* is a role that has one or more additional roles associated with it. When a composite role is mapped to a user, the user gains the roles associated with the composite role. This inheritance is recursive so users also inherit any composite of composites. However, we recommend that composite roles are not overused.

任何领域或客户端级别的角色都可以成为复合角色。复合角色是指具有一个或多个与之关联的其他角色的角色。将复合角色映射到用户时，用户将获得与复合角色关联的角色。这种继承是递归的，因此用户也可以继承任何组合。但是，我们建议不要过度使用复合角色。

Procedure 程序

1. Click **Realm Roles** in the menu.

单击菜单中的“领域角色”。

2. Click the role that you want to convert.

单击要转换的角色。

3. From the **Action** list, select **Add associated roles**.

从“操作”列表中，选择“添加关联角色”。

Composite role 复合角色

Add roles to developer

x

Filter by roles ▾ Search role by name ➔

<input type="checkbox"/> Role name	Description
<input type="checkbox"/> admin	\${role_admin}
<input type="checkbox"/> create-realm	\${role_create-realm}
<input type="checkbox"/> default-roles-master	\${role_default-roles}
<input checked="" type="checkbox"/> employee	
<input type="checkbox"/> offline_access	\${role_offline-access}
<input type="checkbox"/> uma_authorization	\${role_uma_authorization}

1 - 6 ▾ < >

Add Cancel

The role selection UI is displayed on the page and you can associate realm level and client level roles to the composite role you are creating.

角色选择 UI 显示在页面上，您可以将领域级和客户端级角色关联到正在创建的复合角色。

In this example, the **employee** realm-level role is associated with the **developer** composite role. Any user with the **developer** role also inherits the **employee** role.

在此示例中，雇员领域级角色与开发人员组合角色相关联。具有开发人员角色的任何用户也继承雇员角色。

i When creating tokens and SAML assertions, any composite also has its associated roles added to the claims and assertions of the authentication response sent back to the client.

在创建令牌和 SAML 断言时，任何组合还将其相关角色添加到发送回客户端的身份验证响应的断言和声明中。

[Edit this section 编辑这部分](#)
(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/roles-groups/proc-assigning-role-mappings.adoc)

[Report an issue 报告一个问题](#)
(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/roles-groups/proc-assigning-role-mappings.adoc)

You can assign role mappings to a user through the **Role Mappings** tab for that user.

可以通过用户的“角色映射”选项卡将角色映射分配给该用户。

Procedure 程序

1. Click **Users** in the menu.

单击菜单中的“用户”。

2. Click the user that you want to perform a role mapping on.

单击要对其执行角色映射的用户。

3. Click the **Role mappings** tab.

单击“角色映射”选项卡。

4. Click **Assign role**.

单击“分配角色”。

5. Select the role you want to assign to the user from the dialog.

从对话框中选择要分配给用户的角色。

6. Click **Assign**.

单击“分配”。

Role mappings 角色映射

Name	Inherited	Description
default-roles-master	False	\${role_default-roles}
uma_authorization	True	\${role_uma_authorization}
offline_access	True	\${role_offline-access}
account:view-profile	True	\${role_view-profile}
account:manage-account-links	True	\${role_manage-account-links}
account:manage-account	True	\${role_manage-account}

In the preceding example, we are assigning the composite role **developer** to a user. That role was created in the Composite Roles topic.

在上面的示例中，我们将组合角色开发人员分配给用户。该角色是在“复合角色”主题中创建的。

Effective role mappings 有效的角色映射

Name	Inherited	Description
employee	True	-
developer	False	-
default-roles-master	False	\${role_default-roles}
uma_authorization	True	\${role_uma_authorization}
offline_access	True	\${role_offline-access}
account view-profile	True	\${role_view-profile}
account manage-account-links	True	\${role_manage-account-links}
account manage-account	True	\${role_manage-account}

When the **developer** role is assigned, the **employee** role associated with the **developer** composite is displayed with **Inherited "True"**. Inherited roles are the roles explicitly assigned to users and roles that are inherited from composites.

当分配开发人员角色时，与开发人员组合关联的雇员角色显示为“True”。继承角色是显式分配给用户的角色和从组合继承的角色。

[Edit this section](#) 编辑这部分

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/roles-groups/proc-using-default-roles.adoc)

[Report an issue](#) 报告一个问题

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/roles-groups/proc-using-default-roles.adoc)

Use default roles to automatically assign user role mappings when a user is created or imported through Identity Brokering.

使用默认角色可以在通过身份代理创建或导入用户时自动分配用户角色映射。

Procedure 程序

1. Click **Realm settings** in the menu.

单击菜单中的“域”设置。

2. Click the **User registration** tab.

单击“用户注册”选项卡。

Default roles 默认角色

The screenshot shows the Keycloak Admin UI with the 'Master' realm selected. The left sidebar has 'Realm settings' highlighted. The main content area is titled 'Master' and shows 'Realm settings are settings that control the options for users, applications, roles, and groups in the current realm.' Below this is a navigation bar with tabs: General, Login, Email, Themes, Keys, Events, Localization, Security defenses, Sessions, Tokens, Client policies, and User registration. The 'User registration' tab is currently selected. Under 'Default roles', there is a search bar 'Search role by name' and a checkbox 'Hide inherited roles'. A blue 'Add role' button is visible. Below these are five rows of roles:

Role name	Inherited from	Description
account manage-account	-	\${role_manage-account}
offline_access	-	\${role_offline-access}
uma_authorization	-	\${role_uma_authorization}
account view-profile	-	\${role_view-profile}
account manage-account-links	manage-account	\${role_manage-account-links}

This screenshot shows that some *default roles* already exist.

这个屏幕截图显示一些默认角色已经存在。

[Edit this section](#) 编辑这部分

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/roles-groups/con-role-scope-mappings.adoc)

[Report an issue](#) 报告一个问题

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/roles-groups/con-role-scope-mappings.adoc)

On creation of an OIDC access token or SAML assertion, the user role mappings become claims within the token or assertion. Applications use these claims to make access decisions on the resources controlled by the application. Keycloak digitally signs access tokens and applications re-use them to invoke remotely secured REST services. However, these tokens have an associated risk. An attacker can obtain these tokens and use their permissions to compromise your networks. To prevent this situation, use *Role Scope Mappings*.

在创建 OIDC 访问令牌或 SAML 断言时，用户角色映射成为令牌或断言中的声明。应用程序使用这些声明对应用程序控制的资源进行访问决策。键掩码对访问令牌进行数字签名，应用程序重用它们来调用远程安全的 REST 服务。但是，这些令牌具有相关的风险。攻击者可以获得这些令牌并使用它们的权限来破坏您的网络。若要防止此情况，请使用角色范围映射。

Role Scope Mappings limit the roles declared inside an access token. When a client requests a user authentication, the access token they receive contains only the role mappings that are explicitly specified for the client's scope. The result is that you limit the permissions of each individual access token instead of giving the client access to all the user's permissions.

角色作用域映射限制访问令牌内部声明的角色。当客户端请求用户身份验证时，它们接收的访问令牌仅包含为客户端范围显式指定的角色映射。其结果是限制每个单独访问令牌的权限，而不是让客户端访问所有用户的权限。

By default, each client gets all the role mappings of the user. You can view the role mappings for a client.

默认情况下，每个客户端获取用户的所有角色映射。您可以查看客户端的角色映射。

Procedure 程序

1. Click Clients in the menu.

单击菜单中的 Clients。

2. Click the client to go to the details.

单击客户端查看详细信息。

3. Click the Client scopes tab.

单击 Client scope 选项卡。

4. Click the link in the row with *Dedicated scope and mappers for this client*

单击此客户端的具有专用作用域和映射器的行中的链接

5. Click the Scope tab.

单击 Scope 选项卡。

Full scope 全面观察

The screenshot shows the Keycloak management interface. On the left, there's a sidebar with a dropdown menu set to 'Master' and several navigation items: 'Manage', 'Clients' (which is highlighted with a blue bar), 'Client scopes', 'Realm roles', 'Users', and 'Groups'. On the right, the main content area is titled 'Clients > Client details > Dedicated scopes' and shows a client named 'myapp'. It says, 'This is a client scope which includes the dedicated mappers and scope'. Below this, there are two tabs: 'Mappers' and 'Scope', with 'Scope' being the active tab. Under the 'Scope' tab, there's a setting labeled 'Full scope allowed' with a help icon and a toggle switch that is turned 'On'. The background of the main content area has a light gray gradient.

By default, the effective roles of scopes are every declared role in the realm. To change this default behavior, toggle **Full Scope Allowed** to **ON** and declare the specific roles you want in each client. You can also use client scopes to define the same role scope mappings for a set of clients.

默认情况下，范围的有效角色是领域中的每个已声明角色。若要更改此默认行为，请切换“允许为 ON 的全范围”，并在每个客户端中声明所需的特定角色。还可以使用客户端作用域为一组客户端定义相同的角色作用域映射。

Partial scope 部分显微镜

Master

Clients

Client scopes

Realm roles

Users

Groups

Sessions

Events

Configure

Realm settings

Clients > Client details > Dedicated scopes

myapp

This is a client scope which includes the dedicated mappers and scope

Mappers Scope

Full scope allowed Off

No roles for this client

You haven't created any roles for this client. Create a role to get started.

Assign role

[Edit this section](#) 编辑这部分

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/roles-groups/proc-managing-groups.adoc)

[Report an issue](#) 报告一个问题

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/roles-groups/proc-managing-groups.adoc)

Groups in Keycloak manage a common set of attributes and role mappings for each user. Users can be members of any number of groups and inherit the attributes and role mappings assigned to each group.

Keycloak 的组为每个用户管理一组共同的属性和角色映射。用户可以是任意数量的组的成员，并继承分配给每个组的属性和角色映射。

To manage groups, click **Groups** in the menu.

要管理组，请单击菜单中的“组”。

Groups 团体

Master

Manage

Clients

Client scopes

Realm roles

Users

Groups

Groups

A group is a set of attributes and role mappings that can be applied to a user. You can create, edit, and delete groups and

Search for groups Create group

<input type="checkbox"/> Group name
<input type="checkbox"/> Sales

Groups are hierarchical. A group can have multiple subgroups but a group can have only one parent. Subgroups inherit the attributes and role mappings from their parent. Users inherit the attributes and role mappings from their parent as well.

团队是有等级的。一个组可以有多个子组，但是一个组只能有一个父组。子组从其父组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。

If you have a parent group and a child group, and a user that belongs only to the child group, the user in the child group inherits the attributes and role mappings of both the parent group and the child group.

如果您有一个父组和一个子组，以及一个只属于该子组的用户，则子组中的用户将继承父组和子组的属性和角色映射。

The following example includes a top-level **Sales** group and a child **North America** subgroup.

下面的示例包括顶级 Sales 组和子北美子组。

To add a group:

添加组:

1. Click the group.

点击群组。

2. Click **Create group**.

单击“创建组”。

3. Enter a group name.

输入组名。

4. Click **Create**.

单击 Create。

5. Click the group name.

单击组名。

The group management page is displayed.

将显示组管理页。

Group 小组

Groups > Sales > Group details

Europe

Child groups Members Attributes Role mapping Permissions

No groups in this sub group

You haven't created any groups in this sub group.

Create group

Attributes and role mappings you define are inherited by the groups and users that are members of the group.

您定义的属性和角色映射由作为组成员的组和用户继承。

To add a user to a group:

将用户添加到组中:

1. Click **Users** in the menu.

单击菜单中的“用户”。

2. Click the user that you want to perform a role mapping on. If the user is not displayed, click **View all users**.

单击要对其进行角色映射的用户。如果未显示用户，请单击“查看所有用户”。

3. Click **Groups**.

单击“组”。

User groups 用户组

Users > jimlincoln

jimlincoln

Details Attributes Credentials Role Mappings Groups Consents Sessions Identity Provider Links

Group Membership Search... View all groups Leave

Available Groups Search... View all groups Join

Sales Europe North America

4. Click **Join Group**.

单击“加入组”。

5. Select a group from the dialog.

从对话框中选择一个组。

6. Select a group from the Available Groups tree.

从“可用组”树中选择一个组。

7. Click Join.

单击加入。

To remove a group from a user:

从用户中删除组:

1. Click Users in the menu.

单击菜单中的“用户”。

2. Click the user to be removed from the group.

单击要从组中删除的用户。

3. Click Leave on the group table row.

单击组表行上的“保留”。

In this example, the user *jimlincoln* is in the *North America* group. You can see *jimlincoln* displayed under the **Members** tab for the group.

在这个示例中，用户 *jimlincoln* 位于北美组中。你可以看到吉姆林肯显示在会员标签下面。

Group membership 小组成员

The screenshot shows the Keycloak Groups interface. On the left, there's a sidebar with navigation links: Master, Manage, Clients, Client scopes, Realm roles, Users (which is selected), Groups (which is highlighted in dark grey), and Sessions. The main content area shows the 'Sales' group details. The 'Members' tab is selected. At the top of the member list, there's a 'Add member' button and a checkbox for 'Include sub-group users'. The member list table has columns: Name, Email, First name, Last name, and Membership. It contains two rows: one for 'jimlincoln' (Email: -, First name: -, Last name: -, Membership: /Sales/Nor...th America) and one for 'john Doe' (Email: -, First name: John, Last name: Doe, Membership: /Sales/Europe, /Sales/Nor...th America).

[Edit this section 编辑这部分](#)
(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/roles-groups/con-comparing-groups-roles.adoc)

[Report an issue 报告一个问题](#)
(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/roles-groups/con-comparing-groups-roles.adoc)

Groups and roles have some similarities and differences. In Keycloak, groups are a collection of users to which you apply roles and attributes. Roles define types of users and applications assign permissions and access control to roles.

群体和角色有一些相似之处和不同之处。在 Keycloak，组是将角色和属性应用于其中的用户的集合。角色定义用户类型，应用程序为角色分配权限和访问控制。

Composite Roles are similar to Groups as they provide the same functionality. The difference between them is conceptual. Composite roles apply the permission model to a set of services and applications. Use composite roles to manage applications and services.

组合角色与组类似，因为它们提供相同的功能。他们之间的区别是概念上的。复合角色将权限模型应用于一组服务和应用程序。使用组合角色来管理应用程序和服务。

Groups focus on collections of users and their roles in an organization. Use groups to manage users.

组主要关注用户的集合及其在组织中的角色。使用组来管理用户。

[Edit this section 编辑这部分](#)
(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/roles-groups/proc-specifying-default-groups.adoc)

[Report an issue 报告一个问题](#)
(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priorty=3&description=File:%20server_admin/topics/roles-groups/proc-specifying-default-groups.adoc)

To automatically assign group membership to any users who is created or who is imported through Identity Brokering, you use default groups.

若要自动将组成员资格分配给通过标识代理创建或导入的任何用户，请使用默认组。

1. Click **Realm settings** in the menu.

单击菜单中的“域”设置。

2. Click the **User registration** tab.

单击“用户注册”选项卡。

3. Click the **Default Groups** tab.

单击“默认组”选项卡。

Default groups 默认组

The screenshot shows the Keycloak Admin UI for the 'Master' realm. The left sidebar has a dark theme with options like Manage, Clients, Client scopes, Realm roles, Users, Groups, Sessions, Events, Configure, Realm settings (which is selected), and Authentication. The main area is titled 'Master' and says 'Realm settings are settings that control the options for users, applications, roles, and groups in the current realm.' A 'Learn more' link is available. At the top right are 'Enabled' (switched on) and 'Action' dropdown. Below are tabs: Events, Localization, Security defenses, Sessions, Tokens, Client policies, User profile, and User registration (which is selected). Under 'Default groups', there's a help link 'What is the function of default groups?' and a search bar 'Search group'. A blue 'Add groups' button is present. A table lists existing groups: 'Sales' with path '/Sales'. There are also '1-1' dropdowns and a three-dot menu.

This screenshot shows that some *default groups* already exist.

这个屏幕截图显示一些默认组已经存在。

[Edit this section](#) 编辑这部分

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/authentication.adoc)

[Report an issue](#) 报告一个问题

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/authentication.adoc)

This chapter covers several authentication topics. These topics include:

这一章涵盖了几个认证主题，这些主题包括：

- Enforcing strict password and One Time Password (OTP) policies.

执行严格的密码和一次性密码(OTP)策略。

- Managing different credential types.

管理不同的凭据类型。

- Logging in with Kerberos.

用 Kerberos 登录。

- Disabling and enabling built-in credential types.

禁用和启用内置凭据类型。

[Edit this section](#) 编辑这部分

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/authentication/password-policies.adoc)

[Report an issue](#) 报告一个问题

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/authentication/password-policies.adoc)

When Keycloak creates a realm, it does not associate password policies with the realm. You can set a simple password with no restrictions on its length, security, or complexity. Simple passwords are unacceptable in production environments. Keycloak has a set of password policies available through the Admin Console.

创建领域时，它不将密码策略与领域关联。您可以设置一个简单的密码，不限制其长度、安全性或复杂性。简单的密码在生产环境中是不可接受的。密钥斗篷有一组可通过管理控制台使用的密码策略。

Procedure 程序

1. Click **Authentication** in the menu.

单击菜单中的“身份验证”。

2. Click the **Policies** tab.

单击“策略”选项卡。

3. Select the policy to add in the **Add policy** drop-down box.

在“添加策略”下拉框中选择要添加的策略。

4. Enter a value that applies to the policy chosen.

输入应用于所选策略的值。

5. Click **Save**.

单击“保存”。

Password policy

The screenshot shows the Keycloak Admin Console interface. The left sidebar is dark-themed and includes a dropdown for 'Master' realm, followed by a list of management sections: Manage, Clients, Client scopes, Realm roles, Users, Groups, Sessions, Events, Configure, Realm settings, and Authentication. The 'Authentication' section is currently selected, indicated by a blue bar at the bottom of the sidebar. The main content area has a light background and is titled 'Authentication'. It contains a sub-header 'Authentication is the area where you can configure and manage different credential types.' with a 'Learn more' link. Below this, there are tabs for 'Flows', 'Required actions', and 'Policies', with 'Policies' being the active tab. Under 'Policies', there are four options: 'Password policy' (selected), 'OTP Policy', 'Webauthn Policy', and 'Webauthn Passwordless Policy'. A dropdown menu labeled 'Add policy' is open, showing 'Password policy' as the selected option. Below the dropdown are two configuration fields: 'Hashing Iterations *' with a value of '27500' and a range slider, and 'Minimum Length *' with a value of '8' and a range slider. At the bottom of the form are 'Save' and 'Reload' buttons. The overall layout is clean and modern, typical of a web-based administration tool.

密码政策

After saving the policy, Keycloak enforces the policy for new users and sets an Update Password action for existing users to ensure they change their password the next time they log in. For example:

在保存策略之后，Keycloak对新用户强制执行策略，并为现有用户设置更新密码操作，以确保他们在下次登录时更改密码。例如：

Failed password policy 密码策略失败

Update password

! Invalid password: minimum length 6.

New Password

Confirm password

Submit

Password policy types 密码策略类型

HashAlgorithm 哈希算法

Passwords are not stored in cleartext. Before storage or validation, Keycloak hashes passwords using standard hashing algorithms. PBKDF2 is the only built-in and default algorithm available. See the [Server Developer Guide](#) (https://www.keycloak.org/docs/latest/server_development/) on how to add your own hashing algorithm.

密码不存储在明文中。在存储或验证之前，密钥斗篷使用标准哈希算法对密码进行哈希处理。PBKDF2是唯一可用的内置和默认算法。有关如何添加自己的哈希算法，请参阅服务器开发人员指南。



If you change the hashing algorithm, password hashes in storage will not change until the user logs in.

如果更改哈希算法，则存储中的密码哈希在用户登录之前不会更改。

Hashing iterations 散列迭代

Specifies the number of times Keycloak hashes passwords before storage or verification. The default value is 27,500.

指定在存储或验证之前密码散列的次数。默认值为27,500。

Keycloak hashes passwords to ensure that hostile actors with access to the password database cannot read passwords through reverse engineering.

密码破解程式会对密码进行哈希处理，以确保可进入密码资料库的敌对逆向工程无法透过密码破解密码。



A high hashing iteration value can impact performance as it requires higher CPU power.

高散列迭代值会影响性能，因为它需要更高的 CPU 能力。

Digits 数字

The number of numerical digits required in the password string.

密码字符串中所需的数字位数。

Lowercase characters 小写字母

The number of lower case letters required in the password string.

密码字符串中所需的小写字母数。

Uppercase characters 大写字母

The number of upper case letters required in the password string.

密码字符串中所需的大写字母数。

Special characters 特殊人物

The number of special characters required in the password string.

密码字符串中所需的特殊字符数。

Not username 不是用户名

The password cannot be the same as the username.

密码不能与用户名相同。

Not email 不是电子邮件

The password cannot be the same as the email address of the user.

密码不能与用户的电子邮件地址相同。

Regular expression 正则表达式

Password must match one or more defined regular expression patterns.

密码必须与一个或多个已定义的正则表达式模式匹配。

Expire password 密码过期

The number of days the password is valid. When the number of days has expired, the user must change their password.

密码有效的天数。当天数过期时，用户必须更改密码。

Not recently used 最近没用过

Password cannot be already used by the user. Keycloak stores a history of used passwords. The number of old passwords stored is configurable in Keycloak.

用户无法使用密码。密钥斗篷存储使用过的密码的历史记录。存储的旧密码数量在 Keycloak 是可配置的。

Password blacklist 密码黑名单

Password must not be in a blacklist file.

密码不能在黑名单文件中。

- Blacklist files are UTF-8 plain-text files with Unix line endings. Every line represents a blacklisted password.

黑名单文件是以 Unix 行结尾的 UTF-8纯文本文件。每一行代表一个被列入黑名单的密码。

- Keycloak compares passwords in a case-insensitive manner. All passwords in the blacklist must be lowercase.

密钥隐形以不区分大小写的方式比较密码。黑名单中的所有密码必须是小写的。

- The value of the blacklist file must be the name of the blacklist file.

黑名单文件的值必须是黑名单文件的名称。

- Blacklist files resolve against `${jboss.server.data.dir}/password-blacklists/` by default. Customize this path using:

默认情况下，黑名单文件根据 `${jboss.server.data.dir}/password-Blacklist/`解析:

- The `keycloak.password.blacklists.path` property.
Keycloak.password.blacklists.path 属性。
- The `blacklistsPath` property of the `passwordBlacklist` policy SPI configuration.

PasswordBlacklist 策略 SPI 配置的 `blacklistsPath` 属性。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/authentication/otp-policies.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/authentication/otp-policies.adoc)

Keycloak has several policies for setting up a FreeOTP or Google Authenticator One-Time Password generator. Click the **Authentication** menu and click the **OTP Policy** tab.

对于设置 FreeOTP 或谷歌身份验证器的一次性密码生成器，“钥匙斗篷”有几种策略。单击“身份验证”菜单并单击“OTP 策略”选项卡。

Otp policy Otp 政策

The screenshot shows the Keycloak administrative interface. On the left, a sidebar menu is open under the 'Master' dropdown, showing options like Manage, Clients, Client scopes, Realm roles, Users, Groups, Sessions, Events, Configure, Realm settings, Authentication (which is selected and highlighted in blue), Identity providers, and User federation. The main content area is titled 'Authentication' and describes it as a place to configure and manage different credential types. It features tabs for Flows, Required actions, Policies (which is selected and highlighted in blue), Password policy, OTP Policy (selected), Webauthn Policy, and Webauthn Passwordless Policy. Under the 'OTP Policy' tab, several configuration options are displayed: 'OTP type' (Time based is selected), 'OTP hash algorithm' (SHA1), 'Number of digits' (6 is selected), 'Look ahead window' (set to 1), 'OTP Token period' (30 seconds), 'Supported actions' (FreeOTP, Google Authenticator), and 'Reusable token' (On). At the bottom are 'Save' and 'Reload' buttons.

Keycloak generates a QR code on the OTP set-up page, based on information configured in the **OTP Policy** tab. FreeOTP and Google Authenticator scan the QR code when configuring OTP.

基于在 OTP 策略选项卡中配置的信息，在 OTP 设置页上生成一个二维码。FreeOTP 和 Google Authenticator 在配置 OTP 时扫描二维码。

Time-based or counter-based one time passwords 基于时间的或基于计数器的一次性密码
The algorithms available in Keycloak for your OTP generators are time-based and counter-based.

在 Keycloak，你的一次性交易发生器的算法是基于时间和计数器的。

With Time-Based One Time Passwords (TOTP), the token generator will hash the current time and a shared secret. The server validates the OTP by comparing the hashes within a window of time to the submitted value. TOTPs are valid for a short window of time.

使用基于时间的一次密码(TOTP)，令牌生成器将散列当前时间和共享秘密。服务器通过比较一个时间窗口内的散列和提交的值来验证 OTP。TOTP 在短时间内有效。

With Counter-Based One Time Passwords (HOTP), Keycloak uses a shared counter rather than the current time. The Keycloak server increments the counter with each successful OTP login. Valid OTPs change after a successful login.

对于基于计数器的一次性密码(HOTP) , Key斗篷使用共享计数器而不是当前时间。每次成功登录 OTP 时, Key斗篷服务器都会递增计数器。有效的 OTP 在成功登录后更改。

TOTP is more secure than HOTP because the matchable OTP is valid for a short window of time, while the OTP for HOTP is valid for an indeterminate amount of time. HOTP is more user-friendly than TOTP because no time limit exists to enter the OTP.

TOTP 比 HOTP 更安全, 因为可匹配的 OTP 在短时间内有效, 而 HOTP 的 OTP 在不确定的时间段内有效。HOTP 比 TOTP 更方便用户, 因为不存在进入 OTP 的时间限制。

HOTP requires a database update every time the server increments the counter. This update is a performance drain on the authentication server during heavy load. To increase efficiency, TOTP does not remember passwords used, so there is no need to perform database updates. The drawback is that it is possible to re-use TOTPs in the valid time interval.

每次服务器递增计数器时, HOTP 都需要更新数据库。此更新会在重负载期间影响身份验证服务器的性能。为了提高效率, TOTP 不会记住所使用的密码, 因此不需要执行数据库更新。缺点是可以在有效的时间间隔内重用 TOTP。

TOTP configuration options TOTP 配置选项

OTP hash algorithm OTP 哈希算法

The default algorithm is SHA1. The other, more secure options are SHA256 and SHA512.

默认算法是 SHA1。其他更安全的选项是 SHA256 和 SHA512。

Number of digits 位数

The length of the OTP. Short OTP's are user-friendly, easier to type, and easier to remember. Longer OTP's are more secure than shorter OTP's.

OTP 的长度。简短的 OTP 是用户友好的, 更容易输入, 更容易记住。较长的 OTP 比较短的 OTP 更安全。

Look around window 看看窗户周围

The number of intervals the server attempts to match the hash. This option is present in Keycloak if the clock of the TOTP generator or authentication server becomes out-of-sync. The default value of 1 is adequate. For example, if the time interval for a token is 30 seconds, the default value of 1 means it will accept valid tokens in the 90-second window (time interval 30 seconds + look ahead 30 seconds + look behind 30 seconds). Every increment of this value increases the valid window by 60 seconds (look ahead 30 seconds + look behind 30 seconds).

服务器试图匹配哈希的间隔数。在 Keycloak，如果 TOTP 生成器或认证服务器的时钟失去同步，就会出现这个选项。默认值1是足够的。例如，如果令牌的时间间隔为30秒，默认值1意味着它将在90秒的窗口中接受有效的令牌(时间间隔30秒 + 向前看30秒 + 向后看30秒)。此值的每次增加都会使有效窗口增加60秒(向前看30秒 + 向后看30秒)。

OTP token period OTP 代币期

The time interval in seconds the server matches a hash. Each time the interval passes, the token generator generates a TOTP.

服务器匹配哈希的时间间隔(秒)。每次间隔过去时，令牌生成器都会生成一个 TOTP。

Reusable code 可重用代码

Determine whether OTP tokens can be reused in the authentication process or user needs to wait for the next token. Users cannot reuse those tokens by default, and the administrator needs to explicitly specify that those tokens can be reused.

确定 OTP 令牌是否可以在身份验证过程中重用，或者用户需要等待下一个令牌。默认情况下，用户不能重用这些令牌，管理员需要明确指定可以重用这些令牌。

HOTP configuration options HOTP 配置选项

OTP hash algorithm OTP 哈希算法

The default algorithm is SHA1. The other, more secure options are SHA256 and SHA512.

默认算法是 SHA1。其他更安全的选项是 SHA256 和 SHA512。

Number of digits 位数

The length of the OTP. Short OTPs are user-friendly, easier to type, and easier to remember. Longer OTPs are more secure than shorter OTPs.

OTP 的长度。简短的 OTP 是用户友好的，更容易输入，更容易记住。较长的 OTP 比较短的 OTP 更安全。

Look ahead window 看前面的窗户

The number of intervals the server attempts to match the hash. This option is present in Keycloak if the clock of the TOTP generator or authentication server become out-of-sync. The default value of 1 is adequate. This option is present in Keycloak to cover when the user's counter gets ahead of the server.

服务器试图匹配哈希的间隔数。在 Keycloak，如果 TOTP 生成器或认证服务器的时钟失去同步，就会出现这个选项。默认值1是足够的。在 Keycloak，当用户的计数器超过服务器时，可以使用这个选项。

Initial counter 初始计数器

The value of the initial counter.

初始计数器的值。

An *authentication flow* is a container of authentications, screens, and actions, during log in, registration, and other Keycloak workflows. To view all the flows, actions, and checks, each flow requires:

身份验证流是在登录、注册和其他 Keycloak 工作流期间的身份验证、屏幕和操作的容器。要查看所有流、操作和检查，每个流都需要：

Built-in flows 内置流程

Keycloak has several built-in flows. You cannot modify these flows, but you can alter the flow's requirements to suit your needs.

钥匙斗篷有几个内置的流程。您不能修改这些流程，但是您可以修改流程的需求以满足您的需求。

Procedure 程序

1. Click **Authentication** in the menu.

单击菜单中的“身份验证”。

2. Click on the *Browser* item in the list to see the details.

单击列表中的 Browser 项以查看详细信息。

Browser flow 浏览器流程

The screenshot shows the Keycloak Administration interface under the 'Authentication' section. It displays the 'Flow details' for a 'Browser' flow. The left sidebar lists various management sections, and the right panel shows the configuration of authentication steps in a tree-like structure. The 'Cookie' step is marked as 'Alternative'. The 'Kerberos' step is 'Disabled'. The 'Identity Provider Redirector' and 'forms' (which includes 'Username Password Form') are also marked as 'Alternative'. The 'Browser - Conditional OTP' step is marked as 'Conditional'. The 'Condition - user configured' and 'OTP Form' steps are marked as 'Required'. At the bottom, there are buttons for '+ Add step' and '+ Add sub-flow'.

Auth type 授权类型

The name of the authentication or the action to execute. If an authentication is indented, it is in a sub-flow. It may or may not be executed, depending on the behavior of its parent.

身份验证或要执行的操作的名称。如果身份验证是缩进的，则该身份验证位于子流中。它可以执行，也可以不执行，这取决于它的父级的行为。

1. Cookie

饼干

The first time a user logs in successfully, Keycloak sets a session cookie. If the cookie is already set, this authentication type is successful. Since the cookie provider returned success and each execution at this level of the flow is *alternative*, Keycloak does not perform any other execution. This results in a successful login.

当用户第一次成功登录时，Keycloak设置一个会话 cookie。如果已经设置了 Cookie，则此身份验证类型成功。由于 Cookie 提供程序返回成功，并且流的这个级别的每次执行都是可选的，因此 Keycloak 不执行任何其他执行。这将导致成功登录。

2. Kerberos

This authenticator is disabled by default and is skipped during the Browser Flow.

默认情况下禁用此身份验证程序，并在浏览器流期间跳过该身份验证程序。

3. Identity Provider Redirector

身份提供程序重定向器

This action is configured through the **Actions > Config** link. It redirects to another IdP for identity brokering.

此操作通过 **Actions > Config** 链接进行配置。

4. Forms

表格

Since this sub-flow is marked as *alternative*, it will not be executed if the **Cookie** authentication type passed. This sub-flow contains an additional authentication type that needs to be executed. Keycloak loads the executions for this sub-flow and processes them.

由于此子流被标记为可选，因此如果 Cookie 身份验证类型通过，则不会执行该子流。此子流包含需要执行的其他身份验证类型。键隐藏加载此子流的执行并处理它们。

The first execution is the **Username Password Form**, an authentication type that renders the username and password page. It is marked as *required*, so the user must enter a valid username and password.

第一个执行是 Username Password Form，它是呈现用户名和密码页的身份验证类型。它被标记为必需的，因此用户必须输入有效的用户名和密码。

The second execution is the **Browser - Conditional OTP** sub-flow. This sub-flow is *conditional* and executes depending on the result of the **Condition - User Configured** execution. If the result is true, Keycloak loads the executions for this sub-flow and processes them.

第二个执行是 Browser-ConditionalOTP 子流。此子流是条件的，并根据条件用户配置执行的结果执行。如果结果为真，Keycloak 加载此子流的执行并处理它们。

The next execution is the **Condition - User Configured** authentication. This authentication checks if Keycloak has configured other executions in the flow for the user. The **Browser - Conditional OTP** sub-flow executes only when the user has a configured OTP credential.

下一个执行是条件用户配置的身份验证。这个身份验证检查用户是否在流中配置了其他执行操作。Browser-ConditionalOTP 子流只有在用户拥有配置好的 OTP 凭据时才会执行。

The final execution is the **OTP Form**. Keycloak marks this execution as *required* but it runs only when the user has an OTP credential set up because of the setup in the *conditional* sub-flow. If not, the user does not see an OTP form.

最后执行的是 OTP 表单。键隐形将此执行标记为必需的，但它仅在用户由于条件子流中的设置而设置了 OTP 凭据时才运行。如果没有，用户就看不到 OTP 表单。

Requirement 规定

A set of radio buttons that control the execution of an action executes.

控制操作执行的一组单选按钮。

Required 需要

All *Required* elements in the flow must be successfully sequentially executed. The flow terminates if a required element fails.

流中的所有必需元素必须顺序执行成功。如果必需元素失败，流将终止。

Alternative 另一种选择

Only a single element must successfully execute for the flow to evaluate as successful. Because the *Required* flow elements are sufficient to mark a flow as successful, any *Alternative* flow element within a flow containing *Required* flow elements will not execute.

只有单个元素必须成功执行才能使流成功进行评估。因为“必需流”元素足以将流标记为成功，所以包含“必需流”元素的流中的任何可选流元素都不会执行。

Disabled 残疾人

The element does not count to mark a flow as successful.

元素不会将流标记为成功。

Conditional 有条件的

This requirement type is only set on sub-flows.

这个需求类型只在子流上设置。

- A *Conditional* sub-flow contains executions. These executions must evaluate to logical statements.

条件子流包含执行。这些执行必须计算为逻辑语句。

- If all executions evaluate as *true*, the *Conditional* sub-flow acts as *Required*.

如果所有的执行都被评估为 *true*，那么条件子流就充当了“必需”。

- If all executions evaluate as *false*, the *Conditional* sub-flow acts as *Disabled*.

如果所有执行都被评估为 *false*，则条件子流充当 *Disable*。

- If you do not set an execution, the *Conditional* sub-flow acts as *Disabled*.

如果不设置执行，则“条件”子流充当“禁用”。

- If a flow contains executions and the flow is not set to *Conditional*, Keycloak does not evaluate the executions, and the executions are considered functionally *Disabled*.

如果一个流包含执行，并且该流没有设置为“条件”，那么 Keycloak 不会计算执行，并且执行在功能上被认为 是禁用的。

Creating flows 创建流程

Important functionality and security considerations apply when you design a flow.

在设计流时应用重要的功能和安全注意事项。

To create a flow, perform the following:

若要创建流，请执行以下操作：

Procedure 程序

1. Click **Authentication** in the menu.

单击菜单中的“身份验证”。

2. Click **Create flow**.

单击 Create flow。

You can copy and then modify an existing flow. Click the "Action list" (the three dots at the end of the row), click **Duplicate**, and enter a name for the new flow.



可以复制并修改现有流。单击“ Action list”(行末尾的三个点)，单击 Duplicate，然后输入新流的名称。

When creating a new flow, you must create a top-level flow first with the following options:

创建新流时，必须首先创建一个顶级流，其中包含以下选项：

Alias 化名

The name of the flow.

流的名称。

Description 描述

The description you can set to the flow.

可以设置到流的说明。

Top-Level Flow Type 顶层流动类型

The type of flow. The type **client** is used only for the authentication of clients (applications). For all other cases, choose **generic**.

流动的类型。类型客户端仅用于客户端(应用程序)的身份验证。对于所有其他情况，选择通用。

Create a top-level flow 创建一个顶级流程

Master

Authentication > Create flow

Create flow

You can create a top level flow within this from

Name * ⓘ

Description ⓘ

Flow type ⓘ Basic flow

Create Cancel

When Keycloak has created the flow, Keycloak displays the **Add step**, and **Add flow** buttons.

创建了流之后，显示“添加”步骤和“添加流”按钮。

An empty new flow 一条空的新河流

Master

Authentication > Flow details

New flow Not in use

No steps

You can start defining this flow by adding a sub-flow or an execution

Add an execution
Execution can have a wide range of actions, from sending a reset email to validating an OTP Add execution

Add a sub-flow
Sub-Flows can be either generic or form. The form type is used to construct a sub-flow that generates a single flow for the user. Sub-flows are a special type of execution that evaluate as successful depending on how the executions they contain evaluate. Add sub-flow

Three factors determine the behavior of flows and sub-flows.

有三个因素决定了流和子流的行为。

- The structure of the flow and sub-flows.

流和子流的结构。

- The executions within the flows

在流中执行

- The requirements set within the sub-flows and the executions.

子流中设置的需求和执行。

Executions have a wide variety of actions, from sending a reset email to validating an OTP. Add executions with the **Add step** button. Hover over the question mark next to **Provider**, to see a description of the execution.

执行有各种各样的操作，从发送重置电子邮件到验证 OTP。使用 Add step 按钮添加执行。将鼠标悬停在 Provider 旁边的问号上，查看执行的说明。

Adding an authentication execution 添加身份验证执行

Add step to New flow

X

1 - 10 ▾ < >

Browser Redirect for Cookie free authentication

Perform a 302 redirect to get user agent's current URI on authenticate path with an auth_session_id query parameter. This is for client's that do not support cookies.

Cookie

Validates the SSO cookie set by the auth server.

Username Password Challenge

Proprietary challenge protocol for CLI clients that queries for username password

Choose User

Choose a user to reset credentials for

Password

Validates the password supplied as a 'password' form parameter in direct grant request

WebAuthn Authenticator

Authenticator for WebAuthn. Usually used for WebAuthn two-factor authentication

Kerberos

Initiates the SPNEGO protocol. Most often used with Kerberos.

Reset Password

Sets the Update Password required action if execution is REQUIRED. Will also set it if execution is OPTIONAL and the password is currently configured for it.

X509/Validate Username

Validates username and password from X509 client certificate received as a part of mutual SSL handshake.

Password Form

Validates a password from login form.

Docker Authenticator

Uses HTTP Basic authentication to validate docker users, returning a docker error token on auth failure

1 - 10 ▾ < >

Add

Cancel

Two types of executions exist, *automatic executions* and *interactive executions*. *Automatic executions* are similar to the **Cookie** execution and will automatically perform their action in the flow. *Interactive executions* halt the flow to get input. Executions executing successfully set their status to *success*. For a flow to complete, it needs at least one execution with a status of *success*.

有两种类型的执行，自动执行和交互执行。自动执行类似于 Cookie 执行，并将自动在流中执行它们的操作。交互式执行会停止获取输入的流。执行成功将其状态设置为成功。为了完成一个流，它至少需要一次执行并且处于成功状态。

You can add sub-flows to top-level flows with the **Add flow** button. The **Add flow** button displays the **Create Execution Flow** page. This page is similar to the **Create Top Level Form** page. The difference is that the **Flow Type** can be **generic** (default) or **form**. The **form** type constructs a sub-flow that generates a form for the user, similar to the built-in **Registration** flow. Sub-flows success depends on how their executions evaluate, including their contained sub-flows. See the execution requirements section for an in-depth explanation of how sub-flows work.

您可以使用 Add flow 按钮将子流添加到顶级流。“添加流”按钮显示“创建执行流”页。此页类似于“创建顶级表单”页。区别在于流类型可以是通用的(默认)或形式。表单类型构造一个子流，该子流为用户生成一个表单，类似于内置的 Registry 流。子流的成功取决于它们的执行如何评估，包括它们所包含的子流。有关子流如何工作的深入解释，请参阅执行需求部分。



After adding an execution, check the requirement has the correct value.

添加执行之后，检查需求的值是否正确。

All elements in a flow have a **Delete** option in the **Actions** menu. This action removes the element from the flow. Executions have a menu item (the gear icon) to configure the execution. It is also possible to add executions and sub-flows to sub-flows with the **Add step** and **Add flow** links.

流中的所有元素在“操作”菜单中都有一个“删除”选项。此操作将元素从流中移除。执行有一个 something 菜单项(齿轮图标)来配置执行。还可以使用 Add 步骤和 Add flow 链接将执行和子流添加到子流。

Since the order of execution is important, you can move executions and sub-flows up and down by dragging their names.

由于执行顺序很重要，所以可以通过拖动它们的名称来上下移动执行和子流。

Make sure to properly test your configuration when you configure the authentication flow to confirm that no security holes exist in your setup. We recommend that you test various corner cases. For example, consider testing the authentication behavior for a user when you remove various credentials from the user's account before authentication.

配置身份验证流以确认安装中不存在安全漏洞时，请确保正确测试配置。我们建议您测试各种角落案例。例如，考虑在身份验证之前从用户帐户中删除各种凭据时测试用户的身份验证行为。



As an example, when 2nd-factor authenticators, such as OTP Form or WebAuthn Authenticator, are configured in the flow as REQUIRED and the user does not have credential of particular type, the user will be able to set up the particular credential during authentication itself. This situation means that the user does not authenticate with this credential as he set up it right during the authentication. So for browser authentication, make sure to configure your authentication flow with some 1st-factor credentials such as Password or WebAuthn Passwordless Authenticator.

例如，当第二因素身份验证器(如 OTP Form 或 WebAuthn Authenticator)在流中配置为 REQUIRED 且用户没有特定类型的凭证时，用户将能够在身份验证过程中设置特定的凭证。这种情况意味着当用户在身份验证期间正确设置此凭据时，不使用此凭据进行身份验证。因此，对于浏览器身份验证，请确保使用一些第一要素凭据(如 Password 或 WebAuthn Passwordless Authenticator)配置身份验证流。

Creating a password-less browser login flow 创建无密码的浏览器登录流

To illustrate the creation of flows, this section describes creating an advanced browser login flow. The purpose of this flow is to allow a user a choice between logging in using a password-less manner with WebAuthn, or two-factor authentication with a password and OTP.

为了说明流的创建，本节介绍如何创建高级浏览器登录流。此流的目的是允许用户在使用 WebAuthn 的无密码方式登录或使用密码和 OTP 的双因素身份验证之间进行选择。

Procedure 程序

1. Click **Authentication** in the menu.

单击菜单中的“身份验证”。

2. Click the **Flows** tab.

单击 Flows 选项卡。

3. Click **Create flow**.

单击 Create flow。

4. Enter **Browser Password-less** as a name.

输入 Browser Password-less 作为名称。

5. Click **Create**.

单击 Create。

6. Click **Add execution**.

单击“添加执行”。

7. Select **Cookie** from the list.

从列表中选择 Cookie。

8. Click **Add**.

单击“添加”。

9. Select **Alternative** for the **Cookie** authentication type to set its requirement to alternative.

为 Cookie 身份验证类型选择 Alternative 以将其需求设置为 Alternative。

10. Click **Add step**.

单击“添加步骤”。

11. Select **Kerberos** from the list.

从列表中选择 Kerberos。

12. Click **Add**.

单击“添加”。

13. Click **Add step**.

单击“添加步骤”。

14. Select **Identity Provider Redirector** from the list.

从列表中选择标识提供程序重定向器。

15. Click **Add**.

单击“添加”。

16. Select **Alternative** for the **Identity Provider Redirector** authentication type to set its requirement to alternative.

为身份提供程序重定向身份验证类型选择 Alternative，以将其要求设置为 Alternative。

17. Click **Add sub-flow**.

单击 Add subflow。

18. Enter **Forms** as a name.

输入 Forms 作为名称。

19. Click **Add**.

单击“添加”。

20. Select **Alternative** for the **Forms** authentication type to set its requirement to alternative.

为 Forms 身份验证类型选择 Alternative 以将其要求设置为 Alternative。

The common part with the browser flow 浏览器流的公共部分

The screenshot shows the 'Flow details' page for a 'Browser Password-less' flow. It lists four authentication steps:

Steps	Requirement
Cookie	Alternative
Kerberos	Disabled
Identity Provider Redirector	Alternative
Forms	Alternative

At the bottom, there are buttons for '+ Add step' and '+ Add sub-flow'. There are also icons for adding, deleting, and editing steps.

21. Click + menu of the **Forms** execution.

表单执行的单击 + 菜单。

22. Select **Add step**.

选择 Add step。

23. Select **Username Form** from the list.

从列表中选择 Username Form。

24. Click **Add**.

单击“添加”。

At this stage, the form requires a username but no password. We must enable password authentication to avoid security risks.

在这个阶段，表单需要用户名，但不需要密码。我们必须启用密码身份验证，以避免安全风险。

1. Click + menu of the **Forms** sub-flow.

单击 Forms 子流程的 + 菜单。

2. Click **Add sub-flow**.

单击 Add subflow。

3. Enter **Authentication** as name.

输入 Authentication 作为 name。

4. Click **Add**.

单击“添加”。

5. Select **Required** for the **Authentication** authentication type to set its requirement to required.

选择“身份验证身份验证类型的必要条件”以将其要求设置为“必要条件”。

6. Click + menu of the **Authentication** sub-flow.

“身份验证”子流的“单击 +”菜单。

7. Click **Add step**.

单击“添加步骤”。

8. Select **Webauthn Passwordless Authenticator** from the list.

从列表中选择 Webauthn Passwordless Authenticator。

9. Click **Add**.

单击“添加”。

10. Select **Alternative** for the **Webauthn Passwordless Authenticator** authentication type to set its requirement to alternative.

为 Webauthn Passwordless Authenticator 身份验证类型选择 Alternative，以将其要求设置为 Alternative。

11. Click + menu of the **Authentication** sub-flow.

“身份验证”子流的“单击 +”菜单。

12. Click **Add sub-flow**.

单击 Add subflow。

13. Enter **Password with OTP** as name.

输入以 OTP 作为名称的密码。

14. Click **Add**.

单击“添加”。

15. Select **Alternative** for the **Password with OTP** authentication type to set its requirement to alternative.

为具有 OTP 身份验证类型的 Password 选择 Alternative，以将其要求设置为 Alternative。

16. Click + menu of the **Password with OTP** sub-flow.

点击 + 菜单的密码与 OTP 子流程。

17. Click **Add step**.

单击“添加步骤”。

18. Select **Password Form** from the list.

从列表中选择“密码表单”。

19. Click **Add**.

单击“添加”。

20. Select **Required** for the **Password Form** authentication type to set its requirement to required.

选择 Password Form 身份验证类型的“必需”以将其要求设置为“必需”。

21. Click + menu of the **Password with OTP** sub-flow.

点击 + 菜单的密码与 OTP 子流程。

22. Click **Add step**.

单击“添加步骤”。

23. Select **OTP Form** from the list.

从列表中选择 OTP Form。

24. Click **Add**.

单击“添加”。

25. Click **Required** for the **OTP Form** authentication type to set its requirement to required.

单击 OTP Form 身份验证类型的“必需”以将其要求设置为“必需”。

Finally, change the bindings.

最后，更改绑定。

1. Click the **Action** menu at the top of the screen.

单击屏幕顶部的 Action 菜单。

2. Select **Bind flow** from the menu.

从菜单中选择 Bind flow。

3. Click the **Browser Flow** drop-down list.

单击“浏览器流”下拉列表。

4. Click **Save**.

单击“保存”。

A password-less browser login 无密码的浏览器登录

Steps	Requirement
Cookie	Alternative
Kerberos	Disabled
Identity Provider Redirector	Alternative
Forms	Alternative
Username Form	Required
Authentication	Required
WebAuthn Passwordless Authenticator	Alternative
Password Form	Disabled
OTP Form	Required
Password with OTP	Disabled
Password Form	Disabled
OTP Form	Required

[+ Add step](#) [+ Add sub-flow](#)

After entering the username, the flow works as follows:

输入用户名后，流程如下：

If users have WebAuthn passwordless credentials recorded, they can use these credentials to log in directly. This is the password-less login. The user can also select **Password with OTP** because the **WebAuthn Passwordless** execution and the **Password with OTP** flow are set to **Alternative**. If they are set to **Required**, the user has to enter WebAuthn, password, and OTP.

如果用户记录了 WebAuthn 无密码凭据，他们可以使用这些凭据直接登录。这是无密码登录。用户还可以选择 Password with OTP，因为 WebAuthn Passwordless 执行和 Password with OTP flow 被设置为 Alternative。如果它们被设置为“必需”，则用户必须输入 WebAuthn、密码和 OTP。

If the user selects the **Try another way** link with `WebAuthn passwordless` authentication, the user can choose between `Password` and `Security Key` (WebAuthn passwordless). When selecting the password, the user will need to continue and log in with the assigned OTP. If the user has no WebAuthn credentials, the user must enter the password and then the OTP. If the user has no OTP credential, they will be asked to record one.

如果用户选择使用 WebAuthn 无密码身份验证的“尝试另一种方式”链接，则用户可以在 Password 和 Security Key (WebAuthn 无密码)之间进行选择。当选择密码时，用户将需要继续并使用指定的 OTP 登录。如果用户没有 WebAuthn 凭据，则必须输入密码，然后输入 OTP。如果用户没有 OTP 凭证，他们将被要求记录一个。

Since the WebAuthn Passwordless execution is set to **Alternative** rather than **Required**, this flow will never ask the user to register a WebAuthn credential. For a user to have a Webauthn credential, an administrator must add a required action to the user. Do this by:

由于 WebAuthn Password 执行被设置为 Alternative 而不是 Required，因此此流将永远不会要求用户注册 WebAuthn 凭据。为了让用户拥有 Webauthn 凭证，管理员必须向用户添加必需的操作。这样做：

1. Enabling the **Webauthn Register Passwordless** required action in the realm (see the WebAuthn documentation).

在领域中启用 WebAuthn Register Password 所需的操作(请参见 WebAuthn 文档)。

2. Setting the required action using the **Credential Reset** part of a user's Credentials management menu.

使用用户的“凭据”管理菜单的“凭据重置”部分设置所需的操作。

Creating an advanced flow such as this can have side effects. For example, if you enable the ability to reset the password for users, this would be accessible from the password form. In the default **Reset Credentials** flow, users must enter their username. Since the user has already entered a username earlier in the **Browser Password-less** flow, this action is unnecessary for Keycloak and suboptimal for user experience. To correct this problem, you can:

创建这样的高级流可能会产生副作用。例如，如果您启用了为用户重置密码的能力，那么可以从密码表单访问这个功能。在默认的“重置凭据”流中，用户必须输入用户名。由于用户已经在 Browser Password-less 流的前面输入了用户名，因此此操作对于 Keycloak 是不必要的，对于用户体验则是次优的。为了纠正这个问题，你可以：

- Duplicate the **Reset Credentials** flow. Set its name to **Reset Credentials for password-less**, for example.
复制“重置凭据”流。例如，将其名称设置为“无密码重置凭据”。
- Click **Delete** (trash icon) of the **Choose user** step.
单击“选择用户”步骤的“删除”(回收图标)。
- In the **Action** menu, select **Bind flow** and select **Reset credentials flow** from the dropdown and click **Save**
在“操作”菜单中，选择“绑定流”并从下拉列表中选择“重置凭据流”，然后单击“保存”

This section describes how to create advanced browser login flow using the step-up mechanism. The purpose of step-up authentication is to allow access to clients or resources based on a specific authentication level of a user.

本节介绍如何使用升级机制创建高级浏览器登录流。升级身份验证的目的是允许基于用户的特定身份验证级别访问客户端或资源。

Procedure 程序

1. Click **Authentication** in the menu.

单击菜单中的“身份验证”。

2. Click the **Flows** tab.

单击 Flows 选项卡。

3. Click **Create flow**.

单击 Create flow。

4. Enter **Browser Incl Step up Mechanism** as a name.

以名称输入 BrowserIncStep up Mechanism。

5. Click **Save**.

单击“保存”。

6. Click **Add execution**.

单击“添加执行”。

7. Select **Cookie** from the list.

从列表中选择 Cookie。

8. Click **Add**.

单击“添加”。

9. Select **Alternative** for the **Cookie** authentication type to set its requirement to alternative.

为 Cookie 身份验证类型选择 Alternative 以将其需求设置为 Alternative。

10. Click **Add sub-flow**.

单击 Add subflow。

11. Enter **Auth Flow** as a name.

输入 Auth Flow 作为名称。

12. Click **Add**.

单击“添加”。

13. Click **Alternative** for the **Auth Flow** authentication type to set its requirement to alternative.

单击 Auth Flow 身份验证类型的 Alternative 将其需求设置为 Alternative。

Now you configure the flow for the first authentication level.

现在为第一个身份验证级别配置流。

1. Click + menu of the **Auth Flow**.

Auth Flow 的 Click + 菜单。

2. Click **Add sub-flow**.

单击 Add subflow。

3. Enter **1st Condition Flow** as a name.

输入“第一条件流”作为名称。

4. Click **Add**.

单击“添加”。

5. Click **Conditional** for the **1st Condition Flow** authentication type to set its requirement to conditional.

单击“第1条件流”身份验证类型的“条件”以将其要求设置为“条件”。

6. Click + menu of the **1st Condition Flow**.

点击 + 菜单的第一个条件流。

7. Click **Add condition**.

单击“添加条件”。

8. Select **Conditional - Level Of Authentication** from the list.

从列表中选择“条件级身份验证”。

9. Click **Add**.

单击“添加”。

10. Click **Required** for the **Conditional - Level Of Authentication** authentication type to set its requirement to required.

单击“条件级身份验证的必要条件”身份验证类型，将其要求设置为“必要条件”。

11. Click + menu of the **Conditional - Level Of Authentication**.

单击“条件级身份验证”的“+”菜单。

12. Click  (gear icon).

点击 something (齿轮图标)。

13. Enter **Level 1** as an alias.

输入级别1作为别名。

14. Enter 1 for the Level of Authentication (LoA).

为身份验证级别(LoA)输入1。

15. Set Max Age to 36000. This value is in seconds and it is equivalent to 10 hours, which is the default SSO Session Max timeout set in the realm. As a result, when a user authenticates with this level, subsequent SSO logins can re-use this level and the user does not need to authenticate with this level until the end of the user session, which is 10 hours by default.

将最大年龄设置为36000。该值以秒为单位，等效于10小时，这是域中设置的默认SSO会话最大超时值。因此，当用户使用此级别进行身份验证时，随后的SSO登录可以重用此级别，用户直到用户会话结束(默认为10小时)才需要使用此级别进行身份验证。

16. Click Save

单击“保存”

Configure the condition for the first authentication level 配置第一个身份验证级别的条件

The screenshot shows a configuration dialog titled "Condition - Level of Authentication config". It contains three main sections: "Alias", "Level of Authentication (LoA)", and "Max Age". The "Alias" field is filled with "Level 1". The "Level of Authentication (LoA)" field is filled with "1". The "Max Age" field is filled with "36000". At the bottom, there are "Save" and "Cancel" buttons.

Condition - Level of Authentication config

Alias * ⓘ

Level 1

Level of Authentication (LoA) ⓘ

1

Max Age ⓘ

36000

Save Cancel

17. Click + menu of the 1st Condition Flow.

点击 + 菜单的第一个条件流。

18. Click Add step.

单击“添加步骤”。

19. Select **Username Password Form** from the list.

从列表中选择“用户名密码表单”。

20. Click **Add**.

单击“添加”。

Now you configure the flow for the second authentication level.

现在为第二个身份验证级别配置流。

1. Click + menu of the **Auth Flow**.

Auth Flow 的 Click + 菜单。

2. Click **Add sub-flow**.

单击 Add subflow。

3. Enter **2nd Condition Flow** as an alias.

输入第二条件流作为别名。

4. Click **Add**.

单击“添加”。

5. Click **Conditional** for the **2nd Condition Flow** authentication type to set its requirement to conditional.

单击“第2条件流”身份验证类型的“条件”以将其需求设置为“条件”。

6. Click + menu of the **2nd Condition Flow**.

点击 + 菜单的第二个条件流。

7. Click **Add condition**.

单击“添加条件”。

8. Select **Conditional - Level Of Authentication** from the item list.

从项列表中选择“条件级身份验证”。

9. Click **Add**.

单击“添加”。

10. Click **Required** for the **Conditional - Level Of Authentication** authentication type to set its requirement to required.

单击“条件级身份验证的必要条件”身份验证类型，将其要求设置为“必要条件”。

11. Click  (gear icon).

点击 something (齿轮图标)。

12. Enter **Level 2** as an alias.

输入 Level 2作为别名。

13. Enter **2** for the Level of Authentication (LoA).

为身份验证级别(LoA)输入2。

14. Set Max Age to **0**. As a result, when a user authenticates, this level is valid just for the current authentication, but not any subsequent SSO authentications. So the user will always need to authenticate again with this level when this level is requested.

设置最大年龄为0。因此，当用户进行身份验证时，此级别仅对当前身份验证有效，而对后续的SSO身份验证无效。因此，当请求此级别时，用户总是需要再次使用此级别进行身份验证。

15. Click Save

单击“保存”

Configure the condition for the second authentication level 配置第二个身份验证级别的条件

Condition - Level of Authentication config ×

Alias * ?

Level of Authentication (LoA) ?

Max Age ?

Save

Cancel

16. Click + menu of the 2nd Condition Flow.

点击 + 菜单的第二个条件流。

17. Click **Add step**.

单击“添加步骤”。

18. Select **OTP Form** from the list.

从列表中选择 OTP Form。

19. Click **Add**.

单击“添加”。

20. Click **Required** for the **OTP Form** authentication type to set its requirement to required.

单击 OTP Form 身份验证类型的“必需”以将其要求设置为“必需”。

Finally, change the bindings.

最后，更改绑定。

1. Click the **Action** menu at the top of the screen.

单击屏幕顶部的 Action 菜单。

2. Select **Bind flow** from the list.

从列表中选择 Bind flow。

3. Select **Browser Flow** in the dropdown.

在下拉列表中选择 Browser Flow。

4. Click **Save**.

单击“保存”。

Browser login with step-up mechanism 使用升级机制登录浏览器

Browser Incl Step up Mechanism Not in use

Steps	Requirement
Cookie	Alternative
Auth Flow	Alternative
1st Condition Flow	Conditional
Condition - Level of Authentication	Required
Username Password Form	Required
2nd Condition Flow	Conditional
Condition - Level of Authentication	Required
OTP Form	Disabled

Request a certain authentication level **请求一定的认证级别**

To use the step-up mechanism, you specify a requested level of authentication (LoA) in your authentication request. The `claims` parameter is used for this purpose:

若要使用升级机制，请在身份验证请求中指定所请求的身份验证级别(LoA)。索赔参数用于此目的:

```
https://{{DOMAIN}}/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
```

The `claims` parameter is specified in a JSON representation:

债权参数是在 JSON 表示法中指定的:

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

The Keycloak javascript adapter has support for easy construct of this JSON and sending it in the login request. See [Javascript adapter documentation](#) (https://www.keycloak.org/docs/latest/securing_apps/#_javascript_adapter) for more details.

Key斗篷 javascript 适配器支持轻松构造这个 JSON 并在登录请求中发送它。有关详细信息，请参阅 [Javascript 适配器文档](#)。

You can also use simpler parameter `acr_values` instead of `claims` parameter to request particular levels as non-essential. This is mentioned in the OIDC specification.

您还可以使用更简单的参数 `acr_value` 而不是声明参数来请求非必需的特定级别。OIDC 规范中提到了这一点。

You can also configure the default level for the particular client, which is used when the parameter `acr_values` or the parameter `claims` with the `acr` claim is not present. For further details, see [Client ACR configuration](#).

您还可以为特定客户机配置默认级别，当不存在 `acr_value` 参数或 `acr` 声明的参数声明时，将使用该级别。有关详细信息，请参阅 [客户端 ACR 配置](#)。

To request the `acr_values` as text (such as `gold`) instead of a numeric value, you configure the mapping between the ACR and the LoA. It is possible to configure it at the realm level (recommended) or at the client level. For configuration see [ACR to LoA Mapping](#) 而不是数值，您可以配置 ACR 和 LoA 之间的映射。可以在领域级别(推荐)或客户端级别配置它。有关配置，请参见 [ACR to LoA Mapping](#) 从 ACR 到 LoA 映射。

For more details see the [official OIDC specification](#) (https://openid.net/specs/openid-connect-core-1_0.html#acrSemantics).

有关更多细节，请参见官方 OIDC 规范。

Flow logic

流程逻辑

The logic for the previous configured authentication flow is as follows:

If a client request a high authentication level, meaning Level of Authentication 2 (LoA 2), a user has to perform full 2-factor authentication: Username/Password + OTP. However, if a user already has a session in Keycloak, that was logged in with username and password (LoA 1), the user is only asked for the second authentication factor (OTP).

前面配置的身份验证流程的逻辑如下: 如果客户端请求高级身份验证级别, 即身份验证级别2(LoA2) , 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 已经有一个使用用户名和密码(loa1)登录的会话, 该用户只会被要求使用第二个身份验证因子(OTP)。

The option **Max Age** in the condition determines how long (how much seconds) the subsequent authentication level is valid. This setting helps to decide whether the user will be asked to present the authentication factor again during a subsequent authentication. If the particular level X is requested by the `claims` or `acr_values` parameter and user already authenticated with level X, but it is expired (for example max age is configured to 300 and user authenticated before 310 seconds) then the user will be asked to re-authenticate again with the particular level. However if the level is not yet expired, the user will be automatically considered as authenticated with that level.

条件中的 Max Age 选项确定后续身份验证级别的有效时间(多少秒)。此设置有助于确定是否要求用户在随后的身份验证过程中再次显示身份验证因子。如果特定级别 X 是通过声明或 acr_value 参数请求的, 并且用户已经通过了级别 X 的身份验证, 但是它已经过期了(例如, 最大年龄被配置为300, 用户在310秒之前通过了身份验证) , 那么用户将被要求再次通过特定级别进行身份验证。但是, 如果该级别尚未过期, 则将自动认为该用户已通过该级别的身份验证。

Using **Max Age** with the value 0 means, that particular level is valid just for this single authentication. Hence every re-authentication requesting that level will need to authenticate again with that level. This is useful for operations that require higher security in the application (e.g. send payment) and always require authentication with the specific level.

使用值为0的最大年龄意味着, 该特定级别仅对于此单个身份验证有效。因此, 每次请求该级别的重新身份验证都需要再次使用该级别进行身份验证。这对于在应用程序中需要更高安全性的操作(例如发送付款)很有用, 并且总是需要具有特定级别的身份验证。

Note that parameters such as `claims` or `acr_values` might be changed by the user in the URL when the login request is sent from the client to the Keycloak via the user's browser. This situation can be mitigated if client uses PAR (Pushed authorization request), a request object, or other mechanisms that prevents the user from rewrite the parameters in the URL. Hence after the authentication, clients are encouraged to check the ID Token to double-check that `acr` in the token corresponds to the expected level. 在令牌中对应于预期的级别



If no explicit level is requested by parameters, the Keycloak will require the authentication with the first LoA condition found in the authentication flow, such as the Username/Password in the preceding example. When a user was already authenticated with that level and that level expired, the user is not required to re-authenticate, but `acr` in the token will have the value 0. This result is considered as authentication based solely on `long-lived browser cookie` as mentioned in the section 2 of OIDC Core 1.0 specification.

如果参数没有请求显式级别，Keycloak 将需要使用身份验证流中的第一个 LoA 条件(如前面示例中的 Username/Password)进行身份验证。当用户已经通过该级别的身份验证且该级别已过期时，用户不需要重新进行身份验证，但令牌中的 acr 值为0。正如 OIDC Core 1.0 规范第2部分所提到的，这个结果被认为是仅基于长寿命浏览器 cookie 的身份验证。

A conflict situation may arise when an admin specifies several flows, sets different LoA levels to each, and assigns the flows to different clients. However, the rule is always the same: if a user has a certain level, it needs only have that level to connect to a client. It's up to the admin to make sure that the LoA is coherent. 当管理员指定几个流、为每个流设置不同的 LoA 级别并将流分配给不同的客户端时，可能会出现冲突情况。但是，规则总是相同的：如果用户具有某个级别，那么它只需要具有该级别就可以连接到客户端。这取决于管理员确保 LoA 是连贯的

Example scenario

示例场景

1. Max Age is configured as 300 seconds for level 1 condition.

最大年龄配置为300秒为1级条件。

2. Login request is sent without requesting any acr. Level 1 will be used and the user needs to authenticate with username and password. The token will have `acr=1`.

发送登录请求时不请求任何 acr。将使用级别1，用户需要使用用户名和密码进行身份验证。该标记将具有 `acr = 1`。

3. Another login request is sent after 100 seconds. The user is automatically authenticated due to the SSO and the token will return `acr=1`.

另一个登录请求在100秒后发送。由于 SSO，用户将被自动验证，令牌将返回 `acr = 1`。

4. Another login request is sent after another 201 seconds (301 seconds since authentication in point 2). The user is automatically authenticated due to the SSO, but the token will return `acr=0` due the level 1 is considered expired.

另一个登录请求在另外201秒后发送(从第2点的身份验证算起，为301秒)。由于 SSO，用户将自动进行身份验证，但是由于级别1被认为过期，令牌将返回 `acr = 0`。

5. Another login request is sent, but now it will explicitly request ACR of level 1 in the `claims` parameter. User will be asked to re-authenticate with username/password and then `acr=1` will be returned in the token.

将发送另一个登录请求，但是现在它将显式请求债权参数中级别为1的 ACR。用户将被要求使用用户名/密码重新进行身份验证，然后在令牌中返回 `acr = 1`。

ACR claim in the token

标记中的 ACR 索赔

ACR claim is added to the token by the `acr loa level` protocol mapper defined in the `acr` client scope. This client scope is the realm default client scope and hence will be added to all newly created clients in the realm.

ACR 声明由 ACR 客户机范围内定义的 ACR loa 级协议映射器添加到令牌中。这个客户端作用域是领域默认的客户端作用域，因此将被添加到领域中所有新创建的客户端。

In case you do not want `acr` claim inside tokens or you need some custom logic for adding it, you can remove the client scope from your client.

如果您不想在令牌中使用 `acr` 声明，或者需要一些自定义逻辑来添加它，那么可以从客户机中删除客户机作用域。

Note when the login request initiates a request with the `claims` parameter requesting `acr` as `essential` claim, then Keycloak will always return one of the specified levels. If it is not able to return one of the specified levels (For example if the requested level is unknown or bigger than configured conditions in the authentication flow), then Keycloak will throw an error.

注意，当登录请求发起一个请求时，声明参数请求 `acr` 作为基本声明，那么 Keycloak 将始终返回指定级别之一。如果它不能返回指定的级别之一（例如，如果请求的级别未知或大于身份验证流中的配置条件），那么 Keycloak 将抛出一个错误。

User session limits 用户会话限制

Limits on the number of sessions that a user can have can be configured. Sessions can be limited per realm or per client.

限制用户可以配置的会话数。每个领域或每个客户端都可以限制会话。

To add session limits to a flow, perform the following steps.

若要向流添加会话限制，请执行以下步骤。

1. Click Add step for the flow.

单击流的 Add 步骤。

2. Select **User session count limiter** from the item list.

从项目列表中选择“用户会话计数限制器”。

3. Click Add.

单击“添加”。

4. Click **Required** for the **User Session Count Limiter** authentication type to set its requirement to required.

单击“用户会话计数限制器身份验证类型的必要条件”以将其要求设置为“必要条件”。

5. Click  (gear icon) for the **User Session Count Limiter**.

用户会话计数限制器的单击 something (齿轮图标)。

6. Enter an alias for this config.

输入此配置的别名。

7. Enter the required maximum number of sessions that a user can have in this realm. For example, if 2 is the value, 2 SSO sessions is the maximum that each user can have in this realm. If 0 is the value, this check is disabled.

输入用户在此领域中可以拥有的所需的最大会话数。例如，如果值为2，则2个SSO会话是每个用户在此领域中可以拥有的最大值。如果值为0，则禁用此检查。

8. Enter the required maximum number of sessions a user can have for the client. For example, if 2 is the value, then 2 SSO sessions is the maximum in this realm for each client. So when a user is trying to authenticate to client `foo`, but that user has already authenticated in 2 SSO sessions to client `foo`, either the authentication will be denied or an existing sessions will be killed based on the behavior configured. If a value of 0 is used, this check is disabled. If both session limits and client session limits are enabled, it makes sense to have client session limits to be always lower than session limits. The limit per client can never exceed the limit of all SSO sessions of this user.

输入用户可以为客户端拥有的所需的最大会话数。例如，如果值为2，那么对于每个客户机，此领域中的最大值是2个SSO会话。因此，当用户试图对客户机 `foo` 进行身份验证，但该用户已经在2个SSO会话中对客户机 `foo` 进行了身份验证时，身份验证将被拒绝，或者根据配置的行为杀死现有会话。如果使用的值为0，则禁用此检查。如果同时启用了会话限制和客户端会话限制，那么客户端会话限制总是低于会话限制是有意义的。每个客户端的限制永远不能超过此用户的所有SSO会话的限制。

9. Select the behavior that is required when the user tries to create a session after the limit is reached.

Available behaviors are:

在达到限制后，选择用户尝试创建会话时所需的行为。可用的行为包括：

- **Deny new session** - when a new session is requested and the session limit is reached, no new sessions can be created.

拒绝新会话——当请求一个新会话并且达到会话限制时，不能创建新会话。

- **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.
- 终止最老的会话-当一个新的会话被请求并且会话限制已经达到时，最老的会话将被删除并创建新的会话。

10. Optionally, add a custom error message to be displayed when the limit is reached.

或者，添加一个自定义错误消息，以便在达到限制时显示。

Note that the user session limits should be added to your bound **Browser flow**, **Direct grant flow**, **Reset credentials** and also to any **Post broker login flow**. The authenticator should be added at the point when the user is already known during authentication (usually at the end of the authentication flow) and should be typically REQUIRED. Note that it is not possible to have ALTERNATIVE and REQUIRED executions at the same level. For example for the default browser flow, it may be necessary to wrap the existing flow as a REQUIRED level-1 subflow and add **User Session Count Limiter** to the same level as this new subflow. Example of such flow is below.

请注意，用户会话限制应该添加到您的绑定浏览器流、直接授权流、重置凭据以及任何 Post Broker 登录流中。身份验证器应该在身份验证期间(通常在身份验证流的末尾)用户已知的时候添加，并且通常应该是 REQUIRED。请注意，不可能在同一级别上执行替代和必需的执行。例如，对于默认浏览器流，可能需要将现有流包装为 REQUIRED level-1子流，并将 User Session Count Limiter 添加到与此新子流相同的级别。这种流程的例子如下。

Steps	Requirement
browser-authentication-subflow	Required
Cookie	Alternative
copy of browser forms	Alternative
Username Password Form	Required
User session count limiter	Required

Currently, the administrator is responsible for maintaining consistency between the different configurations.

目前，管理员负责维护不同配置之间的一致性。

Note also that the user session limit feature is not available for CIBA.

还要注意，用户会话限制特性不适用于 CIBA。

Script Authenticator 脚本验证器

Ability to upload scripts through the Admin Console and REST endpoints is deprecated.

不建议通过管理控制台和 REST 端点上载脚本。

For more details see [JavaScript Providers](#)

(https://www.keycloak.org/docs/latest/server_development/#_script_providers).

有关详细信息，请参阅 [JavaScript 提供程序](#)。

[Edit this section 编辑这部分](#)

https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/authentication/kerberos.adoc

[Report an issue 报告一个问题](#)

https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?componentId=12313920&componentId=12323375&issueType=1&priority=3&description=File:%20server_admin/topics/authentication/kerberos.adoc

Keycloak supports login with a Kerberos ticket through the Simple and Protected GSSAPI Negotiation Mechanism (SPNEGO) protocol. SPNEGO authenticates transparently through the web browser after the user authenticates the session. For non-web cases, or when a ticket is not available during login, Keycloak supports login with Kerberos username and password.

通过简单和受保护的 GSSAPI 协商机制(SPNEGO)协议，Key斗篷支持使用 Kerberos 票证登录。在用户对会话进行身份验证后，SPNEGO 通过 Web 浏览器进行透明的身份验证。对于非 web 情况，或者在登录期间票据不可用时，Key斗篷支持使用 Kerberos 用户名和密码登录。

A typical use case for web authentication is the following:

Web 认证的典型用例如下：

1. The user logs into the desktop.

用户登录到桌面。

2. The user accesses a web application secured by Keycloak using a browser.

用户使用浏览器访问一个由 Keycover 保护的 web 应用程序。

3. The application redirects to Keycloak login.

应用程序将重定向到“钥匙斗篷”登录。

4. Keycloak renders the HTML login screen with status 401 and HTTP header `WWW-Authenticate: Negotiate`

钥匙斗篷呈现具有状态401和 HTTP 头 WWW-Authenticate: 协商的 HTML 登录屏幕

5. If the browser has a Kerberos ticket from desktop login, the browser transfers the desktop sign-on information to Keycloak in header `Authorization: Negotiate 'spnego-token'`. Otherwise, it displays the standard login screen, and the user enters the login credentials.

如果浏览器有一个桌面登录的 Kerberos 票据，浏览器会将桌面登录信息传输到 Keycloak 的头部授权：协商“spnego-token”。否则，它将显示标准登录屏幕，用户将输入登录凭据。

6. Keycloak validates the token from the browser and authenticates the user.

“钥匙斗篷”从浏览器验证令牌并对用户进行身份验证。

7. If using LDAPFederationProvider with Kerberos authentication support, Keycloak provisions user data from LDAP. If using KerberosFederationProvider, Keycloak lets the user update the profile and pre-fill login data.

如果使用具有 Kerberos 身份验证支持的 LDAPFederationProvider，则 Keycloak 将提供来自 LDAP 的用户数据。如果使用 KerberosFederationProvider，则 Keycloak 允许用户更新配置文件并预先填充登录数据。

8. Keycloak returns to the application. Keycloak and the application communicate through OpenID Connect or SAML messages. Keycloak acts as a broker to Kerberos/SPNEGO login. Therefore Keycloak authenticating through Kerberos is hidden from the application.

钥匙斗篷返回到应用程序。密钥斗篷和应用程序通过 OpenID 连接或 SAML 消息进行通信。密钥斗篷充当 Kerberos/SPNEGO 登录的代理。因此，通过 Kerberos 进行身份验证对应用程序是隐藏的。

Perform the following steps to set up Kerberos authentication:

执行以下步骤来设置 Kerberos 身份验证：

1. The setup and configuration of the Kerberos server (KDC).

Kerberos 服务器(KDC)的设置和配置。

2. The setup and configuration of the Keycloak server.

密钥斗篷服务器的设置和配置。

3. The setup and configuration of the client machines.

客户端机器的设置和配置。

Setup of Kerberos server Kerberos 服务器的设置

The steps to set up a Kerberos server depends on the operating system (OS) and the Kerberos vendor. Consult Windows Active Directory, MIT Kerberos, and your OS documentation for instructions on setting up and configuring a Kerberos server.

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 WindowsActiveDirectory、MIT Kerberos 和操作系统文档。

During setup, perform these steps:

在安装过程中，执行以下步骤：

1. Add some user principals to your Kerberos database. You can also integrate your Kerberos with LDAP, so user accounts provision from the LDAP server.

向 Kerberos 数据库添加一些用户主体。您还可以将 Kerberos 与 LDAP 集成，这样就可以从 LDAP 服务器提供用户帐户。

2. Add service principal for "HTTP" service. For example, if the Keycloak server runs on `www.mydomain.org`, add the service principal `HTTP/www.mydomain.org@<kerberos realm>`.

为“HTTP”服务添加服务主体。例如，如果 Keycloak 服务器在 `www.mydomain.org` 上运行，那么添加服务主体 `HTTP/www.mydomain.org@<kerberos domain>`。

On MIT Kerberos, you run a "kadmin" session. On a machine with MIT Kerberos, you can use the command:

在 MIT Kerberos 上运行“kadmin”会话：

```
sudo kadmin.local
```

Then, add HTTP principal and export its key to a keytab file with commands such as:

然后，添加 HTTP 主体并将其密钥导出到 keytab 文件中，命令如下：

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

Ensure the keytab file `/tmp/http.keytab` is accessible on the host where Keycloak is running.

确保 keytab 文件 `/tmp/http.keytab` 在运行 Keycloak 的主机上是可访问的。

Setup and configuration of Keycloak server 密钥斗篷服务器的设置和配置

Install a Kerberos client on your machine.

在计算机上安装 Kerberos 客户端。

Procedure 程序

1. Install a Kerberos client. If your machine runs Fedora, Ubuntu, or RHEL, install the [freeipa-client](https://www.freeipa.org/page/Downloads) (<https://www.freeipa.org/page/Downloads>) package, containing a Kerberos client and other utilities.

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

2. Configure the Kerberos client (on Linux, the configuration settings are in the [/etc/krb5.conf](https://web.mit.edu/kerberos/krb5-1.12/doc/admin/conf_files/krb5_conf.html) (https://web.mit.edu/kerberos/krb5-1.12/doc/admin/conf_files/krb5_conf.html) file).

配置 Kerberos 客户机(在 Linux 上，配置设置在 `/etc/krb5.conf` 文件中)。

Add your Kerberos realm to the configuration and configure the HTTP domains your server runs on.

将 Kerberos 领域添加到配置中，并配置服务器运行所在的 HTTP 域。

For example, for the MYDOMAIN.ORG realm, you can configure the `domain_realm` section like this:

例如，对于 MYDOMAIN.ORG 领域，您可以像下面这样配置 `domain_realm` 部分：

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

3. Export the keytab file with the HTTP principal and ensure the file is accessible to the process running the Keycloak server. For production, ensure that the file is readable by this process only.

导出带有 HTTP 主体的 keytab 文件，并确保运行 Keycloak 服务器的进程可以访问该文件。对于生产，请确保该文件只能由此过程读取。

For the MIT Kerberos example above, we exported keytab to the `/tmp/http.keytab` file. If your *Key Distribution Centre (KDC)* and Keycloak run on the same host, the file is already available.

对于上面的 MIT Kerberos 示例，我们将 keytab 导出到 `/tmp/http.keytab` 文件。如果密钥分发中心(KDC)和密钥披风在同一台主机上运行，则该文件已可用。

Enabling SPNEGO processing 启用 SPNEGO 处理

By default, Keycloak disables SPNEGO protocol support. To enable it, go to the browser flow and enable Kerberos.

默认情况下，Keycloak 禁用 SPNEGO 协议支持。要启用它，请转到浏览器流并启用 Kerberos。

Browser flow 浏览器流程

The screenshot shows the Keycloak Administration interface under the 'Authentication' section. On the left sidebar, 'Authentication' is selected. The main content area is titled 'Authentication > Flow details' for 'Browser' (Default, Built-in). It displays a list of authentication steps and their requirements:

Steps	Requirement
Cookie	Alternative
Kerberos	Disabled
Identity Provider Redirector	Alternative
forms Username, password, otp and other auth forms.	Alternative
Username Password Form	Required
Browser - Conditional OTP Flow to determine if the OTP is required for the authentication	Conditional
Condition - user configured	Required
OTP Form	Required

Add buttons: + Add step, + Add sub-flow

Set the **Kerberos** requirement from *disabled* to *alternative* (Kerberos is optional) or *required* (browser must have Kerberos enabled). If you have not configured the browser to work with SPNEGO or Kerberos, Keycloak falls back to the regular login screen.

将 Kerberos 需求从禁用设置为可选(Kerberos 是可选的)或必需设置(浏览器必须启用 Kerberos)。如果您没有将浏览器配置为与 SPNEGO 或 Kerberos 一起使用，则“钥匙斗篷”将回到常规登录屏幕。

Configure Kerberos user storage federation providers 配置 Kerberos 用户存储联合提供程序
 You must now use User Storage Federation to configure how Keycloak interprets Kerberos tickets. Two different federation providers exist with Kerberos authentication support.

您现在必须使用用户存储联盟来配置 Keycloak 如何解释 Kerberos 票据。有两个不同的联合提供程序支持 Kerberos 身份验证。

To authenticate with Kerberos backed by an LDAP server, configure the LDAP Federation Provider.

要使用 LDAP 服务器支持的 Kerberos 进行身份验证，请配置 LDAP 联邦提供程序。

Procedure 程序

1. Go to the configuration page for your LDAP provider.

转到 LDAP 提供程序的配置页面。

Ldap kerberos integration Ldap Kerberos 集成

The screenshot shows the 'Synchronization settings' and 'Kerberos integration' sections of the Keycloak configuration. In the 'Synchronization settings' section, 'Import users' is set to 'On'. In the 'Kerberos integration' section, 'Allow Kerberos authentication' is set to 'Off'. A sidebar on the right lists various configuration sections: General options, Connection and authentication settings, LDAP searching and updating, Synchronization settings (which is currently selected), Kerberos integration, Cache settings, and Advanced settings.

Synchronization settings

Import users On

Batch size

Periodic full sync Off

Periodic changed users sync Off

Kerberos integration

Allow Kerberos authentication Off

Use Kerberos for password authentication Off

Jump to section

General options

Connection and authentication settings

LDAP searching and updating

Synchronization settings

Kerberos integration

Cache settings

Advanced settings

2. Toggle Allow Kerberos authentication to ON

切换允许 Kerberos 身份验证为 ON

Allow Kerberos authentication makes Keycloak use the Kerberos principal access user information so information can import into the Keycloak environment.

允许 Kerberos 身份验证可以使 Keycloak 使用 Kerberos 主体访问用户信息，这样信息就可以导入到钥匙斗篷环境中。

If an LDAP server is not backing up your Kerberos solution, use the **Kerberos User Storage Federation Provider**.

如果 LDAP 服务器没有备份 Kerberos 解决方案，请使用 Kerberos 用户存储联邦提供程序。

Procedure 程序

1. Click User Federation in the menu.

在菜单中单击“用户联盟”。

2. Select Kerberos from the Add provider select box.

从“添加提供程序”选择框中选择 Kerberos。

Kerberos user storage provider Kerberos 用户存储提供程序

The screenshot shows the 'Add Kerberos provider' configuration page in the Keycloak admin UI. The left sidebar is dark and lists various management options like Clients, Client scopes, Realm roles, etc. The 'User federation' option is highlighted. The main panel has two main sections: 'Required Settings' and 'Cache settings'. In 'Required Settings', fields include 'Console display name' (with a required asterisk), 'Kerberos realm' (with a required asterisk), 'Server principal' (with a required asterisk), and 'Key tab' (with a required asterisk). Below these are toggle switches for 'Debug' (Off), 'Allow password authentication' (Off), and 'Update first login' (Off). In the 'Cache settings' section, the 'Cache policy' dropdown is set to 'DEFAULT'. At the bottom are 'Save' and 'Cancel' buttons.

The **Kerberos** provider parses the Kerberos ticket for simple principal information and imports the information into the local Keycloak database. User profile information, such as first name, last name, and email, are not provisioned.

Kerberos 提供程序解析 Kerberos 票据以获得简单的主体信息，并将这些信息导入本地 Key 数据库。没有提供用户配置文件信息，例如名字、姓氏和电子邮件。

Setup and configuration of client machines 客户端机器的设置和配置

Client machines must have a Kerberos client and set up the `krb5.conf` as described above. The client machines must also enable SPNEGO login support in their browser. See [configuring Firefox for Kerberos](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/system-level_authentication_guide/configuring_applications_for_sso) (https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/system-level_authentication_guide/configuring_applications_for_sso) if you are using the Firefox browser.

客户机机器必须有一个 Kerberos 客户机，并按照上面的描述设置 krb5.conf。客户端计算机还必须在其浏览器中启用 SPNEGO 登录支持。如果您正在使用 Firefox 浏览器，请参阅为 Kerberos 配置 Firefox。

The `.mydomain.org` URI must be in the `network.negotiate-auth.trusted-uris` configuration option.

`. mydomain.org` URI 必须位于 `network.treaty-auth. trust-uris` 配置选项中。

In Windows domains, clients do not need to adjust their configuration. Internet Explorer and Edge can already participate in SPNEGO authentication.

在 Windows 域中，客户端不需要调整其配置。Internet Explorer 和 Edge 已经可以参与 SPNEGO 认证。

Example setups 示例设置

Keycloak and FreeIPA docker image 钥匙斗篷和 FreeIPA 码头图像

When you install [docker](https://www.docker.com/) (<https://www.docker.com/>), run a docker image with the FreeIPA server installed. FreeIPA provides an integrated security solution with MIT Kerberos and 389 LDAP server. The image also contains a Keycloak server configured with an LDAP Federation provider and enabled SPNEGO/Kerberos authentication against the FreeIPA server. See details [here](#) (<https://github.com/mposolda/keycloak-freeipa-docker/blob/master/README.md>).

安装 docker 时，运行安装了 FreeIPA 服务器的 docker 映像。FreeIPA 通过 MIT Kerberos 和 389LDAP 服务器提供了一个集成的安全解决方案。这个映像还包含一个配置了 LDAP 联邦提供程序并启用了针对 FreeIPA 服务器的 SPNEGO/Kerberos 身份验证的 Keycon 服务器。点击这里查看详情。

ApacheDS testing Kerberos server ApacheDS 测试 Kerberos 服务器

For quick testing and unit tests, use a simple [ApacheDS](https://directory.apache.org/apacheds/) (<https://directory.apache.org/apacheds/>) Kerberos server. You must build Keycloak from the source and then run the Kerberos server with the maven-exec-plugin from our test suite. See details [here](#) (<https://github.com/keycloak/keycloak/blob/main/docs/tests.md#kerberos-server>).

为了进行快速测试和单元测试，请使用一个简单的 ApacheDSKerberos 服务器。您必须从源代码构建 Keycon，然后使用我们的测试套件中的 maven-exec-plugin 运行 Kerberos 服务器。点击这里查看详情。

Credential delegation 全权委托

Kerberos supports the credential delegation. Applications may need access to the Kerberos ticket so they can re-use it to interact with other services secured by Kerberos. Because the Keycloak server processed the SPNEGO protocol, you must propagate the GSS credential to your application within the OpenID Connect token claim or a SAML assertion attribute. Keycloak transmits this to your application from the Keycloak server. To insert this claim into the token or assertion, each application must enable the built-in protocol mapper `gss delegation credential`. This mapper is available in the **Mappers** tab of the application's client page. See Protocol Mappers chapter for more details.

Kerberos 支持凭据委托。应用程序可能需要访问 Kerberos 票据，以便可以重用它来与 Kerberos 保护的其他服务进行交互。因为 Keycon 服务器处理了 SPNEGO 协议，所以必须在 OpenID Connect 令牌声明或 SAML 断言属性中将 GSS 凭据传播到应用程序。钥匙斗篷将此信息从钥匙斗篷服务器传输到您的应用程序。要将此声明插入到令牌或断言中，每个应用程序必须启用内置的协议映射器 `gss delegation credential`。这个映射器可以在应用程序客户端页面的 Mappers 选项卡中找到。有关详细信息，请参阅协议映射程序章节。

Applications must deserialize the claim it receives from Keycloak before using it to make GSS calls against other services. When you deserialize the credential from the access token to the `GSSCredential` object, create the `GSSContext` with this credential passed to the `GSSManager.createContext` method. For example:

应用程序必须反序列化它从 Keycloak 收到的索赔，然后才能使用它对其他服务进行 GSS 呼叫。当您将凭据从访问令牌反序列化到 `GSSCredential` 对象时，创建 `GSSContext`，并将此凭据传递给 `GSSManager.createContext` 方法。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, krb5id,  
    deserializedGssCredential, GSSContext.DEFAULT_LIFETIME);
```

JAVA

Examples of this code exist in `examples/kerberos` in the Keycloak example distribution or demo distribution download. You can also check the example sources directly [here](#) (<https://github.com/keycloak/keycloak/tree/main/examples/kerberos>).

这段代码的示例存在于 Keycloak 示例发行版或演示发行版下载的示例/kerberos 中。您还可以直接在这里检查示例源代码。



Configure `forwardable` Kerberos tickets in `krb5.conf` file and add support for delegated credentials to your browser.

在 `krb5.conf` 文件中配置可转发 Kerberos 票证，并在浏览器中添加对委托凭据的支持。



Credential delegation has security implications, so use it only if necessary and only with HTTPS. See [this article](#) (https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/system-level_authentication_guide/configuring_applications_for_sso) for more details and an example.

凭据委托具有安全隐患，因此只有在必要时才使用它，并且只能与 HTTPS 一起使用。有关更多细节和示例，请参见本文。

Cross-realm trust 跨界信任

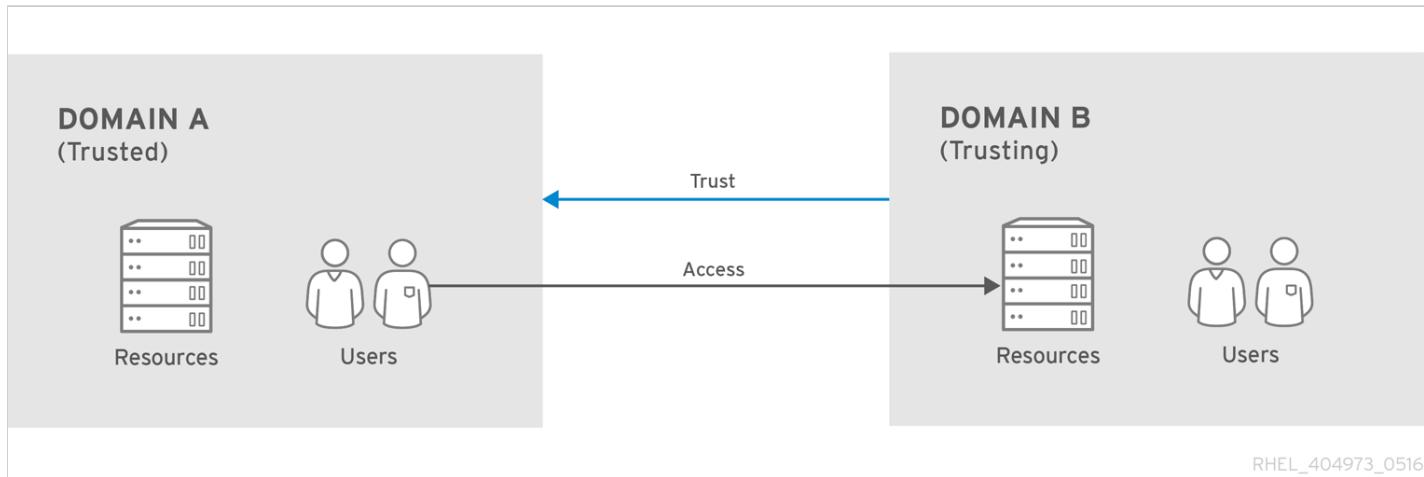
In the Kerberos protocol, the `realm` is a set of Kerberos principals. The definition of these principals exists in the Kerberos database, which is typically an LDAP server.

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

The Kerberos protocol allows cross-realm trust. For example, if 2 Kerberos realms, A and B, exist, then cross-realm trust will allow the users from realm A to access realm B's resources. Realm B trusts realm A.

Kerberos 协议允许跨领域信任。例如，如果存在2个 Kerberos 领域 A 和 B，那么跨领域信任将允许领域 A 的用户访问领域 B 的资源。领域 B 信任领域 A。

Kerberos cross-realm trust Kerberos 跨领域信任



The Keycloak server supports cross-realm trust. To implement this, perform the following:

钥匙斗篷服务器支持跨领域信任。要实现这一点，请执行以下操作：

- Configure the Kerberos servers for the cross-realm trust. Implementing this step depends on the Kerberos server implementations. This step is necessary to add the Kerberos principal `krbtgt/B@A` to the Kerberos databases of realm A and B. This principal must have the same keys on both Kerberos realms. The principals must have the same password, key version numbers, and ciphers in both realms. Consult the Kerberos server documentation for more details.

为跨领域信任配置 Kerberos 服务器。实现此步骤取决于 Kerberos 服务器实现。这个步骤是将 Kerberos 主体 `krbtgt/B@A` 添加到领域 A 和 B 的 Kerberos 数据库所必需的。此主体在两个 Kerberos 领域上必须具有相同的密钥。两个领域中的主体必须具有相同的密码、密钥版本号和密码。有关详细信息，请参阅 Kerberos 服务器文档。



The cross-realm trust is unidirectional by default. You must add the principal `krbtgt/A@B` to both Kerberos databases for bidirectional trust between realm A and realm B. However, trust is transitive by default. If realm B trusts realm A and realm C trusts realm B, then realm C trusts realm A without the principal, `krbtgt/C@A`, available. Additional configuration (for example, `capaths`) may be necessary on the Kerberos client-side so clients can find the trust path. Consult the Kerberos documentation for more details.

缺省情况下，跨领域信任是单向的。您必须将主体 `krbtgt/A@B` 添加到两个 Kerberos 数据库中，以便在领域 A 和领域 B 之间进行双向信任。如果领域 B 信任领域 A，而领域 C 信任领域 B，那么领域 C 信任没有主体 `krbtgt/C@A` 可用的领域 A。Kerberos 客户端可能需要额外的配置(例如，大小写)，以便客户端可以找到信任路径。有关详细信息，请参阅 Kerberos 文档。

- Configure Keycloak server

配置钥匙斗篷服务器

- When using an LDAP storage provider with Kerberos support, configure the server principal for realm B, as in this example: `HTTP/mydomain.com@B`. The LDAP server must find the users from realm A if users from realm A are to successfully authenticate to Keycloak, because Keycloak must perform the SPNEGO flow and then find the users.

在使用支持 Kerberos 的 LDAP 存储提供程序时，为领域 B 配置服务器主体，如本例所示：`HTTP/mydomain.com@B`。如果领域 a 的用户要成功地向 Keycloak 验证身份，那么 LDAP 服务器必须找到领域 a 的用户，因为 Keycon 必须执行 SPNEGO 流，然后找到用户。

For example, Kerberos principal user `john@A` must be available in the LDAP under an LDAP DN such as `uid=john,ou=People,dc=example,dc=com`. If you want users from realm A and B to authenticate, ensure that LDAP can find users from both realms A and B.

例如，Kerberos 主用户 `john@A` 必须在 LDAP DN 下可用，比如 `uid = john, ou = People, dc = example, dc = com`。如果希望领域 A 和 B 的用户进行身份验证，请确保 LDAP 可以找到领域 A 和 B 的用户。

- When using a Kerberos user storage provider (typically, Kerberos without LDAP integration), configure the server principal as `HTTP/mydomain.com@B`, and users from Kerberos realms A and B must be able to authenticate.

当使用 Kerberos 用户存储提供程序(通常是没有 LDAP 集成的 Kerberos)时，将服务器主体配置为 `HTTP/mydomain.com@B`，来自 Kerberos 领域 A 和 B 的用户必须能够进行身份验证。



When using the Kerberos user storage provider, there cannot be conflicting users among Kerberos realms. If conflicting users exist, Keycloak maps them to the same user.

在使用 Kerberos 用户存储提供程序时，Kerberos 领域之间不能有冲突用户。如果存在冲突用户，Keycloak 会将它们映射到同一个用户。

Troubleshooting 故障排除

If you have issues, enable additional logging to debug the problem:

如果有问题，启用附加日志来调试问题：

- Enable `Debug` flag in the Admin Console for Kerberos or LDAP federation providers
在管理控制台中为 Kerberos 或 LDAP 联合提供程序启用 Debug 标志
- Enable TRACE logging for category `org.keycloak` to receive more information in server logs
启用 org.keycloak 类别的 TRACE 日志记录，以便在服务器日志中接收更多信息
- Add system properties `-Dsun.security.krb5.debug=true` and `-Dsun.security.spnego.debug=true`
添加系统属性 `-Dsun.security.krb5.debug = true` 和 `-Dsun.security.spnego.debug = true`

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/authentication/x509.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/authentication/x509.adoc)

Keycloak supports logging in with an X.509 client certificate if you have configured the server to use mutual SSL authentication.

如果您已经将服务器配置为使用相互 SSL 身份验证，Keycloak 支持使用 X.509 客户端证书登录。

A typical workflow:

一个典型的工作流程：

- A client sends an authentication request over SSL/TLS channel.
客户机通过 SSL/TLS 通道发送身份验证请求。
- During the SSL/TLS handshake, the server and the client exchange their x.509/v3 certificates.
在 SSL/TLS 握手过程中，服务器和客户机交换它们的 x.509/v3 证书。
- The container (WildFly) validates the certificate PKIX path and the certificate expiration date.

容器(WildFly)验证证书 PKIX 路径和证书过期日期。

- The x.509 client certificate authenticator validates the client certificate by using the following methods:

X. 509客户端证书身份验证器使用以下方法验证客户端证书:

- Checks the certificate revocation status by using CRL or CRL Distribution Points.
使用 CRL 或 CRL 分发点检查证书吊销状态。
- Checks the Certificate revocation status by using OCSP (Online Certificate Status Protocol).
使用 OCSP (联机证书状态协议)检查证书撤销状态。
- Validates whether the key in the certificate matches the expected key.
验证证书中的密钥是否与预期的密钥匹配。
- Validates whether the extended key in the certificate matches the expected extended key.
验证证书中的扩展密钥是否与预期的扩展密钥匹配。

- If any of the these checks fail, the x.509 authentication fails. Otherwise, the authenticator extracts the certificate identity and maps it to an existing user.

如果这些检查中的任何一个失败，则 x.509身份验证失败。否则，身份验证器将提取证书标识并将其映射到现有用户。

When the certificate maps to an existing user, the behavior diverges depending on the authentication flow:

当证书映射到现有用户时，行为会根据身份验证流而发生变化:

- In the Browser Flow, the server prompts users to confirm their identity or sign in with a username and password.

在 Browser Flow 中，服务器提示用户确认其身份或使用用户名和密码登录。

- In the Direct Grant Flow, the server signs in the user.

在 Direct Grant Flow 中，服务器在用户中进行签名。



Note that it is the responsibility of the web container to validate certificate PKIX path. X.509 authenticator on the Keycloak side provides just the additional support for check the certificate expiration, certificate revocation status and key usage. If you are using Keycloak deployed behind reverse proxy, make sure that your reverse proxy is configured to validate PKIX path. If you do not use reverse proxy and users directly access the WildFly, you should be fine as WildFly makes sure that PKIX path is validated as long as it is configured as described below. 请注意，Web 容器负责验证证书 PKIX 路径。Key斗篷端的 X.509 身份验证器提供了检查证书过期、证书吊销状态和密钥使用的额外支持。如果您正在使用部署在反向代理之后的 Keycloak，请确保将反向代理配置为验证 PKIX 路径。如果不使用反向代理并且用户直接访问 WildFly，那么应该没有问题，因为 WildFly 确保只要按照下面所述配置 PKIX 路径，就可以验证 PKIX 路径

Features 特征

Supported Certificate Identity Sources:

资料来源:

- Match SubjectDN by using regular expressions

使用正则表达式匹配 SubjectDN

- X500 Subject's email attribute

X500主题的电子邮件属性

- X500 Subject's email from Subject Alternative Name Extension (RFC822Name General Name)

X500主题的电子邮件从主题替代名称扩展(RFC822Name 一般名称)

- X500 Subject's other name from Subject Alternative Name Extension. This other name is the User Principal Name (UPN), typically.

X500主题的其他名称来自主题替代名称扩展。这个其他名称是用户主体名称(UPN)，通常。

- X500 Subject's Common Name attribute

X500主题的公共名称属性

- Match IssuerDN by using regular expressions

使用正则表达式匹配 IssuerDN

- Certificate Serial Number

证书编号

- Certificate Serial Number and IssuerDN

证书编号和发证机构

- SHA-256 Certificate thumbprint

SHA-256证书拇指指纹

- Full certificate in PEM format

PEM 格式的完整证书

Regular expressions 正则表达式

Keycloak extracts the certificate identity from Subject DN or Issuer DN by using a regular expression as a filter. For example, this regular expression matches the email attribute:

Key斗篷使用正则表达式作为筛选器，从主题 DN 或发行者 DN 中提取证书标识。例如，这个正则表达式匹配 email 属性：

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

The regular expression filtering applies if the `Identity Source` is set to either `Match SubjectDN using regular expression` or `Match IssuerDN using regular expression`.

如果将 `Identity Source` 设置为使用正则表达式的 `Match SubjectDN` 或使用正则表达式的 `Match IssuerDN`, 则应用正则表达式筛选。

Mapping certificate identity to an existing user 将证书标识映射到现有用户

The certificate identity mapping can map the extracted user identity to an existing user's username, email, or a custom attribute whose value matches the certificate identity. For example, setting `Identity source` to `Subject's email` or `User mapping method` to `Username` or `email` makes the X.509 client certificate authenticator use the email attribute in the certificate's Subject DN as the search criteria when searching for an existing user by username or by email.

证书标识映射可以将提取的用户标识映射到现有用户的用户名、电子邮件或其值与证书标识匹配的自定义属性。例如，将 `Identity source` 设置为 `Subject` 的 `email`, 或将 `User mapping method` 设置为 `Username` 或 `email`, 使得 X.509 客户端证书认证器在通过用户名或电子邮件搜索现有用户时，使用证书的 Subject DN 中的 `email` 属性作为搜索条件。

- If you disable **Login with email** at realm settings, the same rules apply to certificate authentication. Users are unable to log in by using the email attribute.
如果在领域设置中禁用电子邮件登录，则同样的规则也适用于证书身份验证。用户无法使用 email 属性登录。
- Using **Certificate Serial Number and IssuerDN** as an identity source requires two custom attributes for the serial number and the IssuerDN.
使用证书序列号和 IssuerDN 作为标识源需要序列号和 IssuerDN 的两个自定义属性。
- **SHA-256 Certificate thumbprint** is the lowercase hexadecimal representation of SHA-256 certificate thumbprint.



- SHA-256证书拇指指纹是 SHA-256证书拇指指纹的小写十六进制表示。
- Using **Full certificate in PEM format** as an identity source is limited to the custom attributes mapped to external federation sources, such as LDAP. Keycloak cannot store certificates in its database due to length limitations, so in the case of LDAP, you must enable **Always Read Value From LDAP**.
- 使用 PEM 格式的 Full 证书作为标识源仅限于映射到外部联合源(如 LDAP)的自定义属性。由于长度限制，Keycon 无法在其数据库中存储证书，因此在 LDAP 的情况下，必须启用 Always Read Value From LDAP。

Extended certificate validation 扩展证书验证

- Revocation status checking using CRL.
使用 CRL 进行撤销状态检查。
- Revocation status checking using CRL/Distribution Point.
使用 CRL/分发点进行撤销状态检查。
- Revocation status checking using OCSP/Responder URI.
使用 OCSP/Responder URI 进行撤销状态检查。
- Certificate KeyUsage validation.
证书键使用验证。
- Certificate ExtendedKeyUsage validation.
证书扩展键使用验证。

Adding X.509 client certificate authentication to browser flows 向浏览器流添加 X.509客户端证书身份验证

1. Click **Authentication** in the menu.

单击菜单中的“身份验证”。

2. Click the **Browser** flow.

单击浏览器流。

3. From the Action list, select **Duplicate**.

从“操作”列表中，选择“复制”。

4. Enter a name for the copy.

输入副本的名称。

5. Click **Duplicate**.

单击“复制”。

6. Click **Add step**.

单击“添加步骤”。

7. Click "X509/Validate Username Form".

点击“X509/验证用户名表单”。

8. Click **Add**.

单击“添加”。

X509 execution X509处决

Add step to Copy of browser

x

1 - 10 ▾ < >

Browser Redirect for Cookie free authentication

Perform a 302 redirect to get user agent's current URI on authenticate path with an auth_session_id query parameter. This is for client's that do not support cookies.

Cookie

Validates the SSO cookie set by the auth server.

Username Password Challenge

Proprietary challenge protocol for CLI clients that queries for username password

Choose User

Choose a user to reset credentials for

Password

Validates the password supplied as a 'password' form parameter in direct grant request

WebAuthn Authenticator

Authenticator for WebAuthn. Usually used for WebAuthn two-factor authentication

Kerberos

Initiates the SPNEGO protocol. Most often used with Kerberos.

Reset Password

Sets the Update Password required action if execution is REQUIRED. Will also set it if execution is OPTIONAL and the password is currently configured for it.

X509/Validate Username

Validates username and password from X509 client certificate received as a part of mutual SSL handshake.

Password Form

Validates a password from login form.

Docker Authenticator

Uses HTTP Basic authentication to validate docker users, returning a docker error token on auth failure

1 - 10 ▾ < >

Add

Cancel

9. Click and drag the "X509/Validate Username Form" over the "Browser Forms" execution.

在“Browser Forms”执行上单击并拖动“X509/ValidateUsername Form”。

10. Set the requirement to "ALTERNATIVE".

设定要求为「另类」。

X509 browser flow X509浏览器流程

Steps	Requirement
Cookie	Alternative
Kerberos	Alternative
Identity Provider Redirector	Alternative
X509/Validate Username Form	Alternative
X.509 Browser forms Username, password, otp and other auth forms.	Alternative

11. Click the Action menu.

单击“操作”菜单。

12. Click the Bind flow.

单击 Bind 流。

13. Click the Browser flow from the drop-down list.

单击下拉列表中的浏览器流。

14. Click Save.

单击“保存”。

X509 browser flow bindings X509浏览器流绑定

Docker auth	Built-in	<input checked="" type="checkbox"/> Docker auth	Used by Docker clients to authenticate against the IDP
X.509 Browser		<input checked="" type="checkbox"/> Browser flow	browser based authentication
Client-flow		<input checked="" type="checkbox"/> Specific clients	

Configuring X.509 client certificate authentication 配置 X.509客户端证书身份验证

X509 configuration X509配置

X509/Validate Username Form config

x

Alias *



User Identity Source [?](#)

Match SubjectDN using regular expression



Canonical DN representation enabled [?](#)

Off

Enable Serial Number hexadecimal representation [?](#)

Off

A regular expression to extract user identity [?](#)

(.*?)(?:\$)

User mapping method [?](#)

Custom Attribute Mapper



A name of user attribute [?](#)



+ Add a name of user attribute

Check certificate validity [?](#)

On

CRL Checking Enabled [?](#)

Off

Enable CRL Distribution Point to check certificate revocation status [?](#)



Off

CRL Path

Add crt path

OCSP Checking Enabled



Off

OCSP Fail-Open Behavior



Off

OCSP Responder Uri

User Identity Source **用户身份来源**

Defines the method for extracting the user identity from a client certificate.

定义从客户端证书提取用户标识的方法。

Canonical DN representation enabled **启用了规范 DN 表示**

Defines whether to use canonical format to determine a distinguished name. The official Java API documentation

(<https://docs.oracle.com/javase/8/docs/api/javax/security/auth/x500/X500Principal.html#getName-java.lang.String->) describes the format. This option affects the two User Identity Sources *Match SubjectDN using regular expression* and *Match IssuerDN using regular expression* only. Enable this option when you set up a new Keycloak instance. Disable this option to retain backward compatibility with existing Keycloak instances.

定义是否使用规范格式来确定专有名称。官方 JavaAPI 文档描述了这种格式。此选项影响使用正则表达式的两个用户标识源“匹配 SubjectDN”和仅使用正则表达式的“匹配 IssuerDN”。在设置新的密钥斗篷实例时启用此选项。禁用此选项可以保留现有的 Key 向下兼容实例。

Enable Serial Number hexadecimal representation **启用序列号十六进制表示法**

Represent the serial number (<https://datatracker.ietf.org/doc/html/rfc5280#section-4.1.2.2>) as hexadecimal. The serial number with the sign bit set to 1 must be left padded with 00 octet. For example, a serial number with decimal value 161, or a1 in hexadecimal representation is encoded as 00a1, according to

RFC5280. See [RFC5280, appendix-B](https://datatracker.ietf.org/doc/html/rfc5280#appendix-B) (<https://datatracker.ietf.org/doc/html/rfc5280#appendix-B>) for more details.

将序列号表示为十六进制。将符号位设置为1的序列号必须用00个八位组填充。例如，根据 RFC5280，具有十进制值161或十六进制表示形式的 a1 的序列号被编码为 00a1。有关详细信息，请参阅 RFC5280，附录-B。

A regular expression 正则表达式

A regular expression to use as a filter for extracting the certificate identity. The expression must contain a single group.

用作提取证书标识的筛选器的正则表达式。该表达式必须包含单个组。

User Mapping Method 用户映射方法

Defines the method to match the certificate identity with an existing user. *Username or email* searches for existing users by username or email. *Custom Attribute Mapper* searches for existing users with a custom attribute that matches the certificate identity. The name of the custom attribute is configurable.

定义将证书标识与现有用户匹配的方法。通过用户名或电子邮件搜索现有用户的用户名或电子邮件。自定义属性映射器搜索具有与证书标识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。

A name of user attribute 用户属性的名称

A custom attribute whose value matches against the certificate identity. Use multiple custom attributes when attribute mapping is related to multiple values, For example, 'Certificate Serial Number and IssuerDN'.

其值与证书标识匹配的自定义属性。当属性映射与多个值相关时，使用多个自定义属性，例如“证书序列号和 IssuerDN”。

CRL Checking Enabled 启用 CRL 检查

Check the revocation status of the certificate by using the Certificate Revocation List. The location of the list is defined in the **CRL file path** attribute.

使用证书吊销列表检查证书的撤销状态。列表的位置在 CRL file path 属性中定义。

Enable CRL Distribution Point to check certificate revocation status 启用 CRL 分发点来检查证书撤销状态

Use CDP to check the certificate revocation status. Most PKI authorities include CDP in their certificates.

使用 CDP 检查证书撤销状态。大多数 PKI 权限在他们的证书中包含 CDP。

CRL file path CRL 文件路径

The path to a file containing a CRL list. The value must be a path to a valid file if the **CRL Checking Enabled** option is enabled.

包含 CRL 列表的文件的路径。如果启用了 CRL 检查启用选项，则该值必须是有效文件的路径。

OCSP Checking Enabled 启用 OCSP 检查

Checks the certificate revocation status by using Online Certificate Status Protocol.

使用联机证书状态协议检查证书吊销状态。

OCSP Fail-Open Behavior OCSP 失败-开放行为

By default the OCSP check must return a positive response in order to continue with a successful authentication. Sometimes however this check can be inconclusive: for example, the OCSP server could be unreachable, overloaded, or the client certificate may not contain an OCSP responder URI. When this setting is turned ON, authentication will be denied only if an explicit negative response is received by the OCSP responder and the certificate is definitely revoked. If a valid OCSP response is not available the authentication attempt will be accepted.

默认情况下，OCSP 检查必须返回一个积极的响应，以便继续进行成功的身份验证。然而，有时这种检查可能是不确定的：例如，OCSP 服务器可能是不可访问的、重载的，或者客户机证书可能不包含 OCSP 响应者 URI。当这个设置被打开时，只有当 OCSP 响应者接收到一个显式的否定响应并且证书被明确撤销时，身份验证才会被拒绝。如果无法获得有效的 OCSP 响应，则将接受身份验证尝试。

OCSP Responder URI OCSP 响应者 URI

Override the value of the OCSP responder URI in the certificate.

重写证书中 OCSP 响应程序 URI 的值。

Validate Key Usage 验证密钥使用情况

Verifies the certificate's KeyUsage extension bits are set. For example, "digitalSignature,KeyEncipherment" verifies if bits 0 and 2 in the KeyUsage extension are set. Leave this parameter empty to disable the Key Usage validation. See [RFC5280, Section-4.2.1.3](#) (<https://datatracker.ietf.org/doc/html/rfc5280#section-4.2.1.3>) for more information. Keycloak raises an error when a key usage mismatch occurs.

验证是否设置了证书的 KeyUse 扩展位。例如，“Digital alSignature, KeyEncipherment”验证是否设置了 KeyUse 扩展插件中的位0和位2。将此参数保留为空以禁用密钥使用验证。有关更多信息，请参见 RFC5280, Section-4.2.1.3。键使用不匹配时，键隐形会引发错误。

Validate Extended Key Usage 验证扩展密钥的使用

Verifies one or more purposes defined in the Extended Key Usage extension. See [RFC5280, Section-4.2.1.12](#) (<https://datatracker.ietf.org/doc/html/rfc5280#section-4.2.1.12>) for more information. Leave this parameter empty to disable the Extended Key Usage validation. Keycloak raises an error when flagged as critical by the issuing CA and a key usage extension mismatch occurs.

验证扩展键使用扩展中定义的一个或多个目的。有关更多信息，请参见 RFC5280, Section-4.2.1.12。将此参数保留为空以禁用“扩展密钥使用”验证。当发出 CA 标记为关键时，键隐形会引发错误，并且发生键使用扩展不匹配。

Validate Certificate Policy 验证证书策略

Verifies one or more policy OIDs as defined in the Certificate Policy extension. See [RFC5280, Section-4.2.1.4](https://datatracker.ietf.org/doc/html/rfc5280#section-4.2.1.4) (<https://datatracker.ietf.org/doc/html/rfc5280#section-4.2.1.4>). Leave the parameter empty to disable the Certificate Policy validation. Multiple policies should be separated using a comma.

验证在证书策略扩展中定义的一个或多个策略 OID。参见 RFC5280, 第4.2.1.4节。保留参数为空，以禁用证书策略验证。多个策略应使用逗号分隔。

Certificate Policy Validation Mode 证书策略验证模式

When more than one policy is specified in the `Validate Certificate Policy` setting, it decides whether the matching should check for all requested policies to be present, or one match is enough for a successful authentication. Default value is `All`, meaning that all requested policies should be present in the client certificate.

当在验证证书策略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是否存在，或者一个匹配足以成功进行身份验证。默认值为 `All`，这意味着所有请求的策略都应该出现在客户端证书中。

Bypass identity confirmation 绕过身份确认

If enabled, X.509 client certificate authentication does not prompt the user to confirm the certificate identity. Keycloak signs in the user upon successful authentication.

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标识。成功身份验证后用户中的密钥隐形签名。

Revalidate client certificate 重新验证客户端证书

If set, the client certificate trust chain will be always verified at the application level using the certificates present in the configured trust store. This can be useful if the underlying web server does not enforce client certificate chain validation, for example because it is behind a non-validating load balancer or reverse proxy, or when the number of allowed CAs is too large for the mutual SSL negotiation (most browsers cap the maximum SSL negotiation packet size at 32767 bytes, which corresponds to about 200 advertised CAs). By default this option is off.

如果设置了，将始终使用配置的信任存储区中存在的证书在应用程序级别验证客户端证书信任链。如果底层 Web 服务器不强制执行客户端证书链验证，例如因为它位于非验证负载平衡器或反向代理之后，或者当允许的 CA 数量太大而无法进行相互 SSL 协商时(大多数浏览器将最大 SSL 协商数据包大小限制在32767字节，相当于大约200个广告的 CA)，那么这可能是有用的。默认情况下，此选项是关闭的。

Adding X.509 Client Certificate Authentication to a Direct Grant Flow 将 X.509客户端证书身份验证添加到直接授予流

1. Click **Authentication** in the menu.

单击菜单中的“身份验证”。

2. Select **Duplicate** from the "Action list" to make a copy of the built-in "Direct grant" flow.

从“操作列表”中选择“复制”以复制内置的“直接授予”流。

3. Enter a name for the copy.

输入副本的名称。

4. Click **Duplicate**.

单击“复制”。

5. Click the created flow.

单击创建的流。

6. Click the trash can icon  of the "Username Validation" and click **Delete**.

单击“用户名验证”的垃圾桶图标，然后单击“删除”。

7. Click the trash can icon  of the "Password" and click **Delete**.

单击“密码”的垃圾桶图标，然后单击“删除”。

8. Click **Add step**.

单击“添加步骤”。

9. Click "X509/Validate Username".

单击“X509/ValidUsername”。

10. Click **Add**.

单击“添加”。

X509 direct grant execution X509直接授予的执行

Add step to Copy of direct grant

×

11 - 20 ▾ < >

Docker Authenticator

Uses HTTP Basic authentication to validate docker users, returning a docker error token on auth failure

Username Password Form for identity provider reauthentication

Validates a password from login form. Username may be already known from identity provider authentication

Allow access

Authenticator will always successfully authenticate. Useful for example in the conditional flows to be used after satisfying the previous conditions

Verify existing account by Email

Email verification of existing Keycloak user, that wants to link his user account with identity provider

Automatically set existing user

Automatically set existing user to authentication context without any verification

X509/Validate Username Form

Validates username and password from X509 client certificate received as a part of mutual SSL handshake.

Basic Auth Challenge

Challenge-response authentication using HTTP BASIC scheme.

Deny access

Access will be always denied. Useful for example in the conditional flows to be used after satisfying the previous conditions

Identity Provider Redirector

Redirects to default Identity Provider or Identity Provider specified with kc_idp_hint query parameter

Username Validation

Validates the username supplied as a 'username' form parameter in direct grant request

Reset OTP

Sets the Configure OTP required action.

11 - 20 ▾ < >

Add

Cancel

11. Set up the x509 authentication configuration by following the steps described in the x509 Browser Flow section.

按照 x509 Browser Flow 部分中描述的步骤设置 x509 身份验证配置。

12. Click the **Bindings** tab.

单击 Bindings 选项卡。

13. Click the **Direct Grant Flow** drop-down list.

单击“直接授予流”下拉列表。

14. Click the newly created "x509 Direct Grant" flow.

单击新创建的“x509 Direct Grant”流。

15. Click **Save**.

单击“保存”。

X509 direct grant flow bindings X509直接授权流绑定

[X509 Direct grant](#)

Direct grant flow

OpenID Connect Resource Owner Grant

[Edit this section](#) 编辑这部分

https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/authentication/webauthn.adoc

[Report an issue](#) 报告一个问题

https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?jqlString=component%3D12313920&issuetype%3D1&priority%3D3&description%3DFile%20server_admin/topics/authentication/webauthn.adoc

Keycloak provides support for [W3C Web Authentication \(WebAuthn\)](https://www.w3.org/TR/webauthn/) (<https://www.w3.org/TR/webauthn/>).

Keycloak works as a WebAuthn's [Relying Party \(RP\)](https://www.w3.org/TR/webauthn/#webauthn-relying-party) (<https://www.w3.org/TR/webauthn/#webauthn-relying-party>)

钥匙斗篷提供对 W3C Web 身份验证(WebAuthn)的支持，作为 WebAuthn 的依赖方(RP)工作。

 WebAuthn's operations success depends on the user's WebAuthn supporting authenticator, browser, and platform. Make sure your authenticator, browser, and platform support the WebAuthn specification.

WebAuthn 的操作成功与否取决于用户支持身份验证器、浏览器和平台的 WebAuthn。确保您的身份验证器、浏览器和平台支持 WebAuthn 规范。

Setup 设置

The setup procedure of WebAuthn support for 2FA is the following:

WebAuthn 对2FA 的支持的设置过程如下:

Enable WebAuthn authenticator registration 启用 WebAuthn 身份验证器注册

1. Click **Authentication** in the menu.

单击菜单中的“身份验证”。

2. Click the **Required Actions** tab.

单击“必要操作”选项卡。

3. Toggle the Webauthn Register switch to ON.

将 Webauthn Register 切换到 ON。

Toggle the **Default Action** switch to **ON** if you want all new users to be required to register their WebAuthn credentials.

如果希望要求所有新用户注册其 WebAuthn 凭据，请将 Default Action 切换到 ON。

Adding WebAuthn authentication to a browser flow 向浏览器流添加 WebAuthn 身份验证

1. Click **Authentication** in the menu.

单击菜单中的“身份验证”。

2. Click the **Browser** flow.

单击浏览器流。

3. Select **Duplicate** from the "Action list" to make a copy of the built-in **Browser** flow.

从“操作列表”中选择“复制”以复制内置的浏览器流。

4. Enter "WebAuthn Browser" as the name of the copy.

输入“WebAuthn Browser”作为副本的名称。

5. Click **Duplicate**.

单击“复制”。

6. Click the name to go to the details

单击名称查看详细信息

7. Click the trash can icon of the "WebAuthn Browser Browser - Conditional OTP" and click **Delete**.

点击“WebAuthn 浏览器浏览器-条件 OTP”的垃圾桶图标，然后点击删除。

If you require WebAuthn for all users:

如果您要求所有用户使用 WebAuthn:

1. Click + menu of the **WebAuthn Browser Forms**.

WebAuthn 浏览器窗体的单击 + 菜单。

2. Click **Add step**.

单击“添加步骤”。

3. Click **WebAuthn Authenticator**.

单击 WebAuthn Authenticator。

4. Click Add.

单击“添加”。

5. Select Required for the WebAuthn Authenticator authentication type to set its requirement to required.

选择“WebAuthn Authenticator 身份验证类型的必要条件”以将其要求设置为“必要条件”。

The screenshot shows a configuration interface for a 'Webauthn browser' flow. It lists several authentication methods with their requirements:

Steps	Requirement
Cookie	Alternative
Kerberos	Alternative
Identity Provider Redirector	Alternative
Webauthn browser forms Username, password, otp and other auth forms.	Alternative
Username Password Form	Required
WebAuthn Authenticator	Required

At the bottom left, there are buttons for '+ Add step' and '+ Add sub-flow'. On the right side of each row, there are icons for edit, delete, and settings.

6. Click the Action menu at the top of the screen.

单击屏幕顶部的 Action 菜单。

7. Select Bind flow from the drop-down list.

从下拉列表中选择 Bind flow。

8. Select Browser from the drop-down list.

从下拉列表中选择 Browser。

9. Click Save.

单击“保存”。



If a user does not have WebAuthn credentials, the user must register WebAuthn credentials.

如果用户没有 WebAuthn 凭据，则必须注册 WebAuthn 凭据。

Users can log in with WebAuthn if they have a WebAuthn credential registered only. So instead of adding the **WebAuthn Authenticator** execution, you can:

如果用户只注册了 WebAuthn 凭证，则可以使用 WebAuthn 登录。因此，不需要添加 WebAuthn Authenticator 执行，您可以：

Procedure 程序

1. Click + menu of the **WebAuthn Browser Forms** row.

“WebAuthn 浏览器窗体”行的“单击 +”菜单。

2. Click **Add sub-flow**.

单击 Add subflow。

3. Enter "Conditional 2FA" for the *name* field.

在 name 字段中输入“Conditional2FA”。

4. On the **WebAuthn Browser Forms** row, click the plus sign + and select **Add step**.

在 WebAuthn Browser Forms 行上，单击加号 + 并选择 Add step。

5. Select **Conditional** for the **Conditional 2FA** to set its requirement to conditional.

选择“条件2FA”的“条件”以将其需求设置为“条件”。

6. On the **Conditional 2FA** row, click the plus sign + and select **Add condition**.

在“条件2FA”行上，单击加号 + 并选择“添加条件”。

7. Click **Add condition**.

单击“添加条件”。

8. Select **Condition - User Configured**.

选择条件-用户配置。

9. Click **Add**.

单击“添加”。

10. Select **Required** for the **Condition - User Configured** to set its requirement to required.

选择“条件-用户配置的必要条件”以将其需求设置为“必要条件”。

11. Drag and drop **WebAuthn Authenticator** into the **Conditional 2FA** flow

将 WebAuthn Authenticator 拖放到条件2FA 流中

12. Select **Alternative** for the **WebAuthn Authenticator** to set its requirement to alternative.

选择 Alternative for the WebAuthn Authenticator 将其要求设置为 Alternative。

Steps	Requirement
Cookie	Alternative
Kerberos	Alternative
Identity Provider Redirector	Alternative
Webauthn browser forms Username, password, otp and other auth forms.	Alternative
Username Password Form	Required
Conditional 2FA	Conditional
Condition - user configured	Required
WebAuthn Authenticator	Alternative

[+ Add step](#) [+ Add sub-flow](#)

The user can choose between using WebAuthn and OTP for the second factor:

用户可以选择使用 WebAuthn 和 OTP 作为第二个因素:

Procedure 程序

1. On the **Conditional 2FA** row, click the plus sign + and select **Add step**.

在“条件2FA”行上，单击加号 + 并选择“添加步骤”。

2. Click **Add step**.

单击“添加步骤”。

3. Select **OTP Form** from the list.

从列表中选择 OTP Form。

4. Click **Add**.

单击“添加”。

5. Select **Alternative** for the **OTP Form** to set its requirement to alternative.

为 OTP 表单选择 Alternative 以将其要求设置为 Alternative。

Steps	Requirement
Cookie	Alternative
Kerberos	Alternative
Identity Provider Redirector	Alternative
Webauthn browser forms Username, password, otp and other auth forms.	Alternative
Username Password Form	Required
Conditional 2FA	Conditional
Condition - user configured	Required
WebAuthn Authenticator	Alternative
OTP Form	Alternative

[+ Add step](#) [+ Add sub-flow](#)

Authenticate with WebAuthn authenticator 使用 WebAuthn 身份验证器进行身份验证

After registering a WebAuthn authenticator, the user carries out the following operations:

注册 WebAuthn 身份验证器后，用户执行以下操作：

- Open the login form. The user must authenticate with a username and password.
打开登录表单。用户必须使用用户名和密码进行身份验证。
- The user's browser asks the user to authenticate by using their WebAuthn authenticator.
用户的浏览器要求用户使用其 WebAuthn 身份验证器进行身份验证。

Managing WebAuthn as an administrator 作为管理员管理 WebAuthn

Managing credentials 管理证件

Keycloak manages WebAuthn credentials similarly to other credentials from User credential management:

Keycon 管理 WebAuthn 凭据的方式类似于来自用户凭据管理的其他凭据：

- Keycloak assigns users a required action to create a WebAuthn credential from the **Reset Actions** list and select **Webauthn Register**.

Keycon 为用户分配一个必需的操作，以便从 ResetActions 列表中创建 WebAuthn 凭据，并选择 WebAuthn Register。

- Administrators can delete a WebAuthn credential by clicking **Delete**.

管理员可以通过单击 Delete 删除 WebAuthn 凭据。

- Administrators can view the credential's data, such as the AAGUID, by selecting **Show data....**

管理员可以通过选择 Show data... 查看凭据的数据，比如 AAGUID。

- Administrators can set a label for the credential by setting a value in the **User Label** field and saving the data.

管理员可以通过在 UserLabel 字段中设置值并保存数据来为凭据设置标签。

Managing policy 管理政策

Administrators can configure WebAuthn related operations as **WebAuthn Policy** per realm.

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

Procedure 程序

1. Click **Authentication** in the menu.

单击菜单中的“身份验证”。

2. Click the **Policy** tab.

单击 Policy 选项卡。

3. Click the **WebAuthn Policy** tab.

单击 WebAuthn Policy 选项卡。

4. Configure the items within the policy (see description below).

配置策略中的项(参见下面的说明)。

5. Click **Save**.

单击“保存”。

The configurable items and their description are as follows:

可配置项及其说明如下:

Configuration 配置	Description 描述

Configuration 配置	Description 描述
<p>Relying Party Entity Name 依赖方实体名称</p>	<p>The readable server name as a WebAuthn Relying Party. This item is mandatory and applies to the registration of the WebAuthn authenticator. The default setting is "keycloak". For more details, see WebAuthn Specification (https://www.w3.org/TR/webauthn/#dictionary-pkcredentialentity)</p> <p>.</p> <p>作为 WebAuthn 依赖方的可读服务器名称。此项是强制性的，适用于 WebAuthn 身份验证器的注册。默认的设置是“密钥斗篷”。有关详细信息，请参见 WebAuthn 规范。</p>
<p>Signature Algorithms 签名算法</p>	<p>The algorithms telling the WebAuthn authenticator which signature algorithms to use for the Public Key Credential (https://www.w3.org/TR/webauthn/#iface-pkcredential)</p> <ul style="list-style-type: none"> . Keycloak uses the Public Key Credential to sign and verify Authentication Assertions (https://www.w3.org/TR/webauthn/#authentication-assertion) . If no algorithms exist, the default ES256 (https://datatracker.ietf.org/doc/html/rfc8152#section-8.1) is adapted. ES256 is an optional configuration item applying to the registration of WebAuthn authenticators. For more details, see WebAuthn Specification (https://www.w3.org/TR/webauthn/#dictdef-publickeycredentialparameters) . <p>告诉 WebAuthn 身份验证器使用哪些签名算法来获得公钥凭证的算法。Key斗篷使用公钥凭据来签名和验证身份验证断言。如果没有算法存在，则采用默认的 ES256。ES256是一个可选的形态项目，适用于 WebAuthn 验证器的注册。有关详细信息，请参见 WebAuthn 规范。</p>

Configuration 配置	Description 描述
Relying Party ID 依赖方身份证	<p>The ID of a WebAuthn Relying Party that determines the scope of Public Key Credentials (https://www.w3.org/TR/webauthn/#public-key-credential)</p> <ul style="list-style-type: none"> . The ID must be the origin's effective domain. <p>This ID is an optional configuration item applied to the registration of WebAuthn authenticators. If this entry is blank, Keycloak adapts the host part of Keycloak's base URL. For more details, see WebAuthn Specification (https://www.w3.org/TR/webauthn/).</p> <p>WebAuthn 依赖方的 ID，该 ID 确定公钥凭据的范围。ID 必须是源的有效域。此 ID 是一个可选的形态项目，用于注册 WebAuthn 身份验证器。如果此条目为空，则“钥匙斗篷”将调整“钥匙斗篷”基 URL 的主机部分。有关详细信息，请参见 WebAuthn 规范。</p>
Attestation Conveyance Preference 认证转让优先权	<p>The WebAuthn API implementation on the browser (WebAuthn Client (https://www.w3.org/TR/webauthn/#webauthn-client)) is the preferential method to generate Attestation statements. This preference is an optional configuration item applying to the registration of the WebAuthn authenticator. If no option exists, its behavior is the same as selecting "none". For more details, see WebAuthn Specification (https://www.w3.org/TR/webauthn/).</p> <p>浏览器上的 WebAuthn API 实现(WebAuthn Client)是生成证明语句的优先方法。此选项是一个可选的形态项目，适用于 WebAuthn 身份验证器的注册。如果不存在选项，则其行为与选择“none”相同。有关详细信息，请参见 WebAuthn 规范。</p>

Configuration 配置	Description 描述
Authenticator Attachment 认证机构附件	<p>The acceptable attachment pattern of a WebAuthn authenticator for the WebAuthn Client. This pattern is an optional configuration item applying to the registration of the WebAuthn authenticator. For more details, see <u>WebAuthn Specification</u> (https://www.w3.org/TR/webauthn/#enumdef-authenticatorattachment)</p> <p>.</p> <p>WebAuthn 客户端的 WebAuthn 身份验证器可接受的附件模式。这个模式是一个可选的形态项目，适用于 WebAuthn 身份验证器的注册。有关详细信息，请参见 WebAuthn 规范。</p>
Require Resident Key 需要驻留密钥	<p>The option requiring that the WebAuthn authenticator generates the Public Key Credential as <u>Client-side-resident Public Key Credential Source</u> (https://www.w3.org/TR/webauthn/#Client-side-resident). This option applies to the registration of the WebAuthn authenticator. If left blank, its behavior is the same as selecting "No". For more details, see <u>WebAuthn Specification</u> (https://www.w3.org/TR/webauthn/#enumdef-authenticatorselectioncriteria-requireresidentkey)</p> <p>.</p> <p>要求 WebAuthn 身份验证器生成作为客户端驻留公钥凭据源的公钥凭据的选项。此选项适用于 WebAuthn 身份验证器的注册。如果留空，则其行为与选择“否”相同。有关详细信息，请参见 WebAuthn 规范。</p>

Configuration 配置	Description 描述
<p>User Verification Requirement 用户验证要求</p>	<p>The option requiring that the WebAuthn authenticator confirms the verification of a user. This is an optional configuration item applying to the registration of a WebAuthn authenticator and the authentication of a user by a WebAuthn authenticator. If no option exists, its behavior is the same as selecting "preferred". For more details, see WebAuthn Specification for registering a WebAuthn authenticator (https://www.w3.org/TR/webauthn/#dom-authenticatorselectioncriteria-userverification) and WebAuthn Specification for authenticating the user by a WebAuthn authenticator (https://www.w3.org/TR/webauthn/#dom-publickeycredentialrequestoptions-userverification) .</p> <p>要求 WebAuthn 身份验证器确认用户验证的选项。这是一个可选的形态项目，适用于 WebAuthn 身份验证器的注册和 WebAuthn 身份验证器对用户的身份验证。如果不存在选项，则其行为与选择“首选”相同。有关更多细节，请参见 WebAuthn 规范(用于注册 WebAuthn 身份验证器)和 WebAuthn 规范(用于通过 WebAuthn 身份验证器对用户进行身份验证)。</p>

Configuration 配置	Description 描述
<p>Timeout 暂停</p>	<p>The timeout value, in seconds, for registering a WebAuthn authenticator and authenticating the user by using a WebAuthn authenticator. If set to zero, its behavior depends on the WebAuthn authenticator's implementation. The default value is 0. For more details, see WebAuthn Specification for registering a WebAuthn authenticator (https://www.w3.org/TR/webauthn/#dom-publickeycredentialcreationoptions-timeout) and WebAuthn Specification for authenticating the user by a WebAuthn authenticator (https://www.w3.org/TR/webauthn/#dom-publickeycredentialrequestoptions-timeout) .</p> <p>用于注册 WebAuthn 身份验证器和使用 WebAuthn 身份验证器对用户进行身份验证的超时值(秒)。如果设置为零，则其行为取决于 WebAuthn 身份验证器的实现。默认值为0。有关更多细节，请参见 WebAuthn 规范(用于注册 WebAuthn 身份验证器) 和 WebAuthn 规范(用于通过 WebAuthn 身份验证器对用户进行身份验证)。</p>
<p>Avoid Same Authenticator Registration 避免使用相同的认证程序注册</p>	<p>If enabled, Keycloak cannot re-register an already registered WebAuthn authenticator.</p> <p>如启用，Keycloak 不能重新注册已注册的 WebAuthn 身份验证器。</p>
<p>Acceptable AAGUIDs 可接受的 AAGUID</p>	<p>The white list of AAGUIDs which a WebAuthn authenticator must register against.</p> <p>WebAuthn 身份验证器必须注册的 AAGUID 的白色列表。</p>

Attestation statement verification 证明声明确认

When registering a WebAuthn authenticator, Keycloak verifies the trustworthiness of the attestation statement generated by the WebAuthn authenticator. Keycloak requires the trust anchor's certificates imported into the [truststore] (<https://www.keycloak.org/server/keycloak-truststore>).

在注册 WebAuthn 身份验证器时，Keycon 验证由 WebAuthn 身份验证器生成的证明语句的可信性。“钥匙斗篷”要求将信任锚的证书导入[信任存储库](<https://www.Keycloak.org/server/Keycloak-truststore>)。

To omit this validation, disable this truststore or set the WebAuthn policy's configuration item "Attestation Conveyance Preference" to "none".

若要省略此验证，请禁用此信任存储库或将 WebAuthn 策略的形态项目“证明传输首选项”设置为“无”。

Managing WebAuthn credentials as a user 以用户身份管理 WebAuthn 凭据

Register WebAuthn authenticator 注册 WebAuthn 身份验证器

The appropriate method to register a WebAuthn authenticator depends on whether the user has already registered an account on Keycloak.

注册 WebAuthn 身份验证器的适当方法取决于用户是否已经注册了 Keycloak 上的帐户。

New user 新用户

If the **WebAuthn Register** required action is **Default Action** in a realm, new users must set up the WebAuthn security key after their first login.

如果 WebAuthn Register 所需的操作是领域中的 Default Action，则新用户必须在第一次登录后设置 WebAuthn 安全密钥。

Procedure 程序

1. Open the login form.

打开登录表单。

2. Click **Register**.

点击注册。

3. Fill in the items on the form.

填写表格上的项目。

4. Click **Register**.

点击注册。

After successfully registering, the browser asks the user to enter the text of their WebAuthn authenticator's label.

成功注册后，浏览器要求用户输入其 WebAuthn 身份验证器标签的文本。

Existing user 现有用户

If **WebAuthn Authenticator** is set up as required as shown in the first example, then when existing users try to log in, they are required to register their WebAuthn authenticator automatically:

如果按照第一个示例中的要求设置 WebAuthn Authenticator，那么当现有用户尝试登录时，需要自动注册其 WebAuthn Authenticator：

Procedure 程序

1. Open the login form.

打开登录表单。

2. Enter the items on the form.

在表单中输入项目。

3. Click Save.

单击“保存”。

4. Click Login.

点击登录。

After successful registration, the user's browser asks the user to enter the text of their WebAuthn authenticator's label.

注册成功后，用户的浏览器要求用户输入其 WebAuthn 身份验证器标签的文本。

Passwordless WebAuthn together with Two-Factor 双因素无密码网络认证

Keycloak uses WebAuthn for two-factor authentication, but you can use WebAuthn as the first-factor authentication. In this case, users with `passwordless` WebAuthn credentials can authenticate to Keycloak without a password. Keycloak can use WebAuthn as both the passwordless and two-factor authentication mechanism in the context of a realm and a single authentication flow.

Keycloak 使用 WebAuthn 进行双因素身份验证，但是可以使用 WebAuthn 作为第一因素身份验证。在这种情况下，使用无密码的 WebAuthn 凭证的用户可以在没有密码的情况下向 Keycloak 进行身份验证。在领域和单个身份验证流的上下文中，Keycloak 可以使用 WebAuthn 作为无密码身份验证机制和双因素身份验证机制。

An administrator typically requires that Security Keys registered by users for the WebAuthn passwordless authentication meet different requirements. For example, the security keys may require users to authenticate to the security key using a PIN, or the security key attests with a stronger certificate authority.

管理员通常要求用户为 WebAuthn 无密码身份验证注册的安全密钥满足不同的要求。例如，安全密钥可能要求用户使用 PIN 对安全密钥进行身份验证，或者安全密钥使用更强的证书颁发机构进行验证。

Because of this, Keycloak permits administrators to configure a separate `WebAuthn Passwordless Policy`. There is a required `Webauthn Register Passwordless` action of type and separate authenticator of type `WebAuthn Passwordless Authenticator`.

正因为如此，Keycover 允许管理员配置单独的 WebAuthn 无密码策略。有一个类型为 WebAuthn Register Password 的必需操作和类型为 WebAuthn Password Authenticator 的独立验证器。

Setup 设置

Set up WebAuthn passwordless support as follows:

设置 WebAuthn 无密码支持如下：

1. (if not already present) Register a new required action for WebAuthn passwordless support. Use the steps described in Enable WebAuthn Authenticator Registration. Register the **Webauthn Register Passwordless** action.

(如果还没有出现)为 WebAuthn 无密码支持注册一个新的必需操作。使用“启用 WebAuthn 身份验证器注册”中描述的步骤。注册 Webauthn Register 无密码操作。

2. Configure the policy. You can use the steps and configuration options described in Managing Policy. Perform the configuration in the Admin Console in the tab **WebAuthn Passwordless Policy**. Typically the requirements for the security key will be stronger than for the two-factor policy. For example, you can set the **User Verification Requirement** to **Required** when you configure the passwordless policy.

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 密码无关策略”选项卡中的“管理控制台”中执行配置。通常，对安全密钥的要求要强于双因素策略。例如，可以在配置无密码策略时将“用户验证要求”设置为“必需”。

3. Configure the authentication flow. Use the **WebAuthn Browser** flow described in Adding WebAuthn Authentication to a Browser Flow. Configure the flow as follows:

配置身份验证流。使用在将 WebAuthn 身份验证添加到浏览器流中描述的 WebAuthn 浏览器流。按以下方式配置流程：

- The **WebAuthn Browser Forms** subflow contains **Username Form** as the first authenticator. Delete the default **Username Password Form** authenticator and add the **Username Form** authenticator. This action requires the user to provide a username as the first step.
WebAuthn Browser Forms 子流包含作为第一个身份验证器的 Username Form。删除默认的用户名密码表单身份验证器，并添加用户名表单身份验证器。此操作要求用户在第一步中提供用户名。
- There will be a required subflow, which can be named **Passwordless Or Two-factor**, for example. This subflow indicates the user can authenticate with Passwordless WebAuthn credential or with Two-factor authentication.
将有一个必需的子流，它可以命名为 Passwordless 或 Two-factor，例如。此子流表明用户可以使用无密码的 WebAuthn 凭据或双因素身份验证进行身份验证。

- The flow contains **WebAuthn Passwordless Authenticator** as the first alternative.
该流包含 WebAuthn Passwordless Authenticator 作为第一个替代方案。

- The second alternative will be a subflow named **Password And Two-factor Webauthn**, for example. This subflow contains a **Password Form** and a **WebAuthn Authenticator**.
 例如，第二个替代方案是名为 Password And Two-factor Webauthn 的子流。此子流包含密码窗体和 WebAuthn 身份验证器。

The final configuration of the flow looks similar to this:

流程的最终配置看起来与下面类似：

PasswordLess flow 无密码流

The screenshot shows the 'Flow details' page for a 'Webauthn browser' flow. The flow consists of several steps and requirements:

Steps	Requirement
Cookie	Alternative
Kerberos	Alternative
Identity Provider Redirector	Alternative
Webauthn browser forms Username, password, otp and other auth forms.	Alternative
Username Password Form	Required
Passwordless Or Two-factor	Required
WebAuthn Passwordless Authenticator	Alternative
Password And Two-factor Webauthn	Alternative
Password Form	Required
WebAuthn Authenticator	Required

At the bottom, there are buttons for '+ Add step' and '+ Add sub-flow'.

You can now add **WebAuthn Register Passwordless** as the required action to a user, already known to Keycloak, to test this. During the first authentication, the user must use the password and second-factor WebAuthn credential. The user does not need to provide the password and second-factor WebAuthn credential if they use the WebAuthn Passwordless credential.

现在，您可以添加 WebAuthn Register Passwordless 作为必需的操作，以便用户(Keycloak 已经知道)来测试这一点。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 无密码凭据，则不需要提供密码和第二因素 WebAuthn 凭据。

LoginLess WebAuthn 无登录网络认证

Keycloak uses WebAuthn for two-factor authentication, but you can use WebAuthn as the first-factor authentication. In this case, users with `passwordless` WebAuthn credentials can authenticate to Keycloak without submitting a login or a password. Keycloak can use WebAuthn as both the loginless/passwordless and two-factor authentication mechanism in the context of a realm.

Keycloak 使用 WebAuthn 进行双因素身份验证，但是可以使用 WebAuthn 作为第一因素身份验证。在这种情况下，使用无密码的 WebAuthn 凭证的用户可以向 Keycloak 验证身份，而无需提交登录名或密码。在一个领域的上下文中，Keycloak 可以使用 WebAuthn 作为无登录/无密码身份验证机制和双因素身份验证机制。

An administrator typically requires that Security Keys registered by users for the WebAuthn loginless authentication meet different requirements. Loginless authentication requires users to authenticate to the security key (for example by using a PIN code or a fingerprint) and that the cryptographic keys associated with the loginless credential are stored physically on the security key. Not all security keys meet that kind of requirements. Check with your security key vendor if your device supports 'user verification' and 'resident key'. See Supported Security Keys.

管理员通常要求用户为 WebAuthn 无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)，并且与无登录凭据关联的加密密钥实际存储在安全密钥上。并非所有的安全密钥都符合这种要求。如果您的设备支持“用户验证”和“常驻密钥”，请与您的安全密钥供应商联系。参见支持的安全密钥。

Keycloak permits administrators to configure the `WebAuthn Passwordless Policy` in a way that allows loginless authentication. Note that loginless authentication can only be configured with `WebAuthn Passwordless Policy` and with `WebAuthn Passwordless` credentials. WebAuthn loginless authentication and WebAuthn passwordless authentication can be configured on the same realm but will share the same policy `WebAuthn Passwordless Policy`.

Keycloak 允许管理员以允许无登录身份验证的方式配置 WebAuthn 无密码策略。请注意，无登录身份验证只能使用 WebAuthn 无密码策略和 WebAuthn 无密码凭证进行配置。WebAuthn 无登录身份验证和 WebAuthn 无密码身份验证可以在同一领域配置，但将共享相同的策略 WebAuthn 无密码策略。

Setup 设置

Procedure 程序

Set up WebAuthn Loginless support as follows:

设置 WebAuthn Loginless 支持如下：

1. (if not already present) Register a new required action for WebAuthn passwordless support. Use the steps described in Enable WebAuthn Authenticator Registration. Register the **Webauthn Register Passwordless** action.

(如果还没有出现)为 WebAuthn 无密码支持注册一个新的必需操作。使用“启用 WebAuthn 身份验证器注册”中描述的步骤。注册 Webauthn Register 无密码操作。

2. Configure the **WebAuthn Passwordless Policy**. Perform the configuration in the Admin Console, **Authentication** section, in the tab **Policies → WebAuthn Passwordless Policy**. You have to set **User Verification Requirement** to **required** and **Require Resident Key** to **Yes** when you configure the policy for loginless scenario. Note that since there isn't a dedicated Loginless policy it won't be possible to mix authentication scenarios with user verification=no/resident key=no and loginless scenarios (user verification=yes/resident key=yes). Storage capacity is usually very limited on security keys meaning that you won't be able to store many resident keys on your security key.

配置 WebAuthn 无密码策略。在“管理控制台，身份验证”部分的“策略→ WebAuthn 无密码策略”选项卡中执行配置。在为无登录场景配置策略时，必须将用户验证需求设置为必需，并将常驻密钥设置为 Yes。请注意，由于没有专用的 Loginless 策略，因此不可能将身份验证场景与用户验证 = no/驻留密钥 = no 和无登录场景混合(用户验证 = yes/驻留密钥 = yes)。安全密钥的存储容量通常非常有限，这意味着您无法在安全密钥上存储许多常驻密钥。

3. Configure the authentication flow. Create a new authentication flow, add the "WebAuthn Passwordless" execution and set the Requirement setting of the execution to **Required**

配置身份验证流。创建一个新的身份验证流，添加“ WebAuthn Password”执行，并将执行的需求设置设置为“必需”

The final configuration of the flow looks similar to this:

流程的最终配置看起来与下面类似:

LoginLess flow 无登录流

Steps	Requirement
WebAuthn Passwordless Authenticator	Required

+ Add step + Add sub-flow

You can now add the required action **WebAuthn Register Passwordless** to a user, already known to Keycloak, to test this. The user with the required action configured will have to authenticate (with a username/password for example) and will then be prompted to register a security key to be used for loginless authentication.

您现在可以向 Keycloak 已经知道的用户添加所需的操作 WebAuthn Register Password 来测试这一点。配置了所需操作的用户将必须进行身份验证(例如，使用用户名/密码)，然后将被提示注册一个用于无登录身份验证的安全密钥。

Vendor specific remarks 供应商的具体说明

Compatibility check list 兼容性检查表

Loginless authentication with Keycloak requires the security key to meet the following features

使用钥匙斗篷的无登录身份验证需要安全密钥来满足以下特性

- FIDO2 compliance: not to be confused with FIDO/U2F

遵守 FIDO2: 不要与 FIDO/U2F 混淆

- User verification: the ability for the security key to authenticate the user (prevents someone finding your security key to be able to authenticate loginless and passwordless)

用户验证: 安全密钥对用户进行身份验证的能力(防止有人找到您的安全密钥以便能够对无登录和无密码的用户进行身份验证)

- Resident key: the ability for the security key to store the login and the cryptographic keys associated with the client application

驻留密钥: 安全密钥存储与客户端应用程序关联的登录名和加密密钥的能力

Windows Hello

To use Windows Hello based credentials to authenticate against Keycloak, configure the **Signature Algorithms** setting of the **WebAuthn Passwordless Policy** to include the RS256 value. Note that some browsers don't allow access to platform security key (like Windows Hello) inside private windows.

若要使用基于 WindowsHello 的凭据来针对 Keycloak 进行身份验证，请配置 WebAuthn Passwordless Policy 的签名算法设置以包含 RS256值。注意，有些浏览器不允许访问私有窗口中的平台安全密钥(如 WindowsHello)。

Supported security keys 支持的安全密钥

The following security keys have been successfully tested for loginless authentication with Keycloak:

下列安全密钥已经成功地测试了使用钥匙斗篷进行无登录身份验证:

- Windows Hello (Windows 10 21H1/21H2)

Windows Hello (Windows 1021H1/21H2)

- Yubico Yubikey 5 NFC

- Feitian ePass FIDO-NFC

飞天 ePass FIDO-NFC

[Edit this section](#) 编辑这部分

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/authentication/recovery-codes.adoc)

[Report an issue](#) 报告一个问题

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/authentication/recovery-codes.adoc)

You can configure Recovery codes for two-factor authentication by adding 'Recovery Authentication Code Form' as a two-factor authenticator to your authentication flow. For an example of configuring this authenticator, see WebAuthn.

通过在身份验证流中添加“恢复身份验证代码表单”作为双因素身份验证器，可以为双因素身份验证配置恢复代码。有关配置此身份验证器的示例，请参见 WebAuthn。

 Please note that Recovery Codes support is in development. Use this feature experimentally.

请注意，恢复代码支持正在开发中。

[Edit this section](#) 编辑这部分

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/authentication/conditions.adoc)

[Report an issue](#) 报告一个问题

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/authentication/conditions.adoc)

As was mentioned in Execution requirements, *Condition* executions can be only contained in *Conditional* subflow. If all *Condition* executions evaluate as true, then the *Conditional* sub-flow acts as *Required*. You can process the next execution in the *Conditional* sub-flow. If some executions included in the *Conditional* sub-flow evaluate as false, then the whole sub-flow is considered as *Disabled*.

正如在执行需求中提到的，条件执行只能包含在条件子流中。如果所有的条件执行都被评估为 true，那么条件子流就充当了“必需”。您可以处理条件子流中的下一个执行。如果条件子流中包含的一些执行被评估为 false，那么整个子流就被认为是禁用的。

Available conditions 现有条件

Condition - User Role

This execution has the ability to determine if the user has a role defined by *User role* field. If the user has the required role, the execution is considered as true and other executions are evaluated. The administrator has to define the following fields:

此执行能够确定用户是否具有由 *User role* 字段定义的角色。如果用户具有所需的角色，则认为执行为 true，并评估其他执行。管理员必须定义以下字段：

Alias 化名

Describes a name of the execution, which will be shown in the authentication flow.

描述执行的名称，该名称将显示在身份验证流中。

User role 用户角色

Role the user should have to execute this flow. To specify an application role the syntax is `appname.aprole` (for example `myapp.myrole`).

用户必须执行此流的角色。要指定应用程序角色，语法是 `appname.apole` (例如 `myapp.myrole`)。

Condition - User Configured

This checks if the other executions in the flow are configured for the user. The Execution requirements section includes an example of the OTP form.

这将检查流中的其他执行是否为用户配置。执行需求部分包含 OTP 表单的一个示例。

Condition - User Attribute

This checks if the user has set up the required attribute. There is a possibility to negate output, which means the user should not have the attribute. The User Attributes section shows how to add a custom attribute. You can provide these fields:

这将检查用户是否设置了所需的属性。有一种可能性是否定输出，这意味着用户不应该拥有该属性。“用户属性”部分显示如何添加自定义属性。您可以提供以下字段：

Alias 化名

Describes a name of the execution, which will be shown in the authentication flow.

描述执行的名称，该名称将显示在身份验证流中。

Attribute name 属性名

Name of the attribute to check.

要检查的属性的名称。

Expected attribute value 期望的属性值

Expected value in the attribute.

属性中的期望值。

Negate output 输出失败

You can negate the output. In other words, the attribute should not be present.

您可以否定输出。换句话说，该属性不应该存在。

Explicitly deny/allow access in conditional flows 在条件流中显式拒绝/允许访问

You can allow or deny access to resources in a conditional flow. The two authenticators `Deny Access` and `Allow Access` control access to the resources by conditions.

可以允许或拒绝访问条件流中的资源。两个身份验证器“拒绝访问”和“允许访问”按条件控制对资源的访问。

Allow Access

Authenticator will always successfully authenticate. This authenticator is not configurable.

身份验证器将始终成功地进行身份验证。此身份验证器不可配置。

Deny Access

Access will always be denied. You can define an error message, which will be shown to the user. You can provide these fields:

访问将始终被拒绝。您可以定义一条错误消息，该消息将显示给用户。您可以提供以下字段：

Alias 化名

Describes a name of the execution, which will be shown in the authentication flow.

描述执行的名称，该名称将显示在身份验证流中。

Error message 错误信息

Error message which will be shown to the user. The error message could be provided as a particular message or as a property in order to use it with localization. (i.e. "You do not have the role 'admin'.", `my-property-deny` in messages properties) Leave blank for the default message defined as property `access-denied`.

将显示给用户的错误消息。可以将错误消息作为特定消息或属性提供，以便将其用于本地化。(例如，“你没有‘管理’的角色。”在消息属性中保留空白，默认消息定义为属性访问拒绝。

Here is an example how to deny access to all users who do not have the role `role1` and show an error message defined by a property `deny-role1`. This example includes `Condition - User Role` and `Deny Access` executions.

下面是一个示例，说明如何拒绝访问所有不具有角色 `role1` 的用户，并显示由属性 `deny-role1` 定义的错误消息。此示例包括条件用户角色和拒绝访问执行。

Browser flow 浏览器流程

Conditions Form	Alternative	+ -
Username Form	Required	-
Access by Role	Conditional	+ -
Condition - user role	Required	-
Deny access	Required	-
Password Form	Required	-

Condition - user role configuration 条件-用户角色配置

Condition - user role config ×

Alias * ?

Must not have role1

User role ?

master ×

role1 ×

Negate output ?

On

Save
Cancel

Configuration of the 配置 Deny Access is really easy. You can specify an arbitrary Alias and required message like this: 您可以指定一个任意的别名和必需的消息, 如下所示:

Deny access config

X

Alias * [?](#)

role1-alias

Error message [?](#)

deny-role1

Save

Cancel

The last thing is defining the property with an error message in the login theme `messages_en.properties` (for English):

最后一件事是在登录主题 `message_en.properties` (用于英语)中定义带有错误消息的属性:

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

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/identity-broker.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/identity-broker.adoc)

An Identity Broker is an intermediary service connecting service providers with identity providers. The identity broker creates a relationship with an external identity provider to use the provider's identities to access the internal services the service provider exposes.

身份代理是连接服务提供者和身份提供者的中介服务。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程序公开的内部服务。

From a user perspective, identity brokers provide a user-centric, centralized way to manage identities for security domains and realms. You can link an account with one or more identities from identity providers or create an account based on the identity information from them.

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法来管理安全域和领域的身份。您可以将帐户与来自身份提供程序的一个或多个身份链接，或者根据来自身份提供程序的身份信息创建帐户。

An identity provider derives from a specific protocol used to authenticate and send authentication and authorization information to users. It can be:

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

- A social provider such as Facebook, Google, or Twitter.
社交服务提供商，如 Facebook、Google 或 Twitter。
- A business partner whose users need to access your services.
用户需要访问您的服务的业务合作伙伴。
- A cloud-based identity service you want to integrate.
您希望集成的基于云的身份识别服务。

Typically, Keycloak bases identity providers on the following protocols:

通常情况下，“钥匙斗篷”将身份提供者基于以下协议：

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

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/identity-broker/overview.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/identity-broker/overview.adoc)

When using Keycloak as an identity broker, Keycloak does not force users to provide their credentials to authenticate in a specific realm. Keycloak displays a list of identity providers from which they can authenticate.

当使用钥匙斗篷作为身份代理时，钥匙斗篷不强制用户提供其凭据以在特定领域进行身份验证。Key斗篷显示它们可以从中进行身份验证的标识提供程序列表。

If you configure a default identity provider, Keycloak redirects users to the default provider.

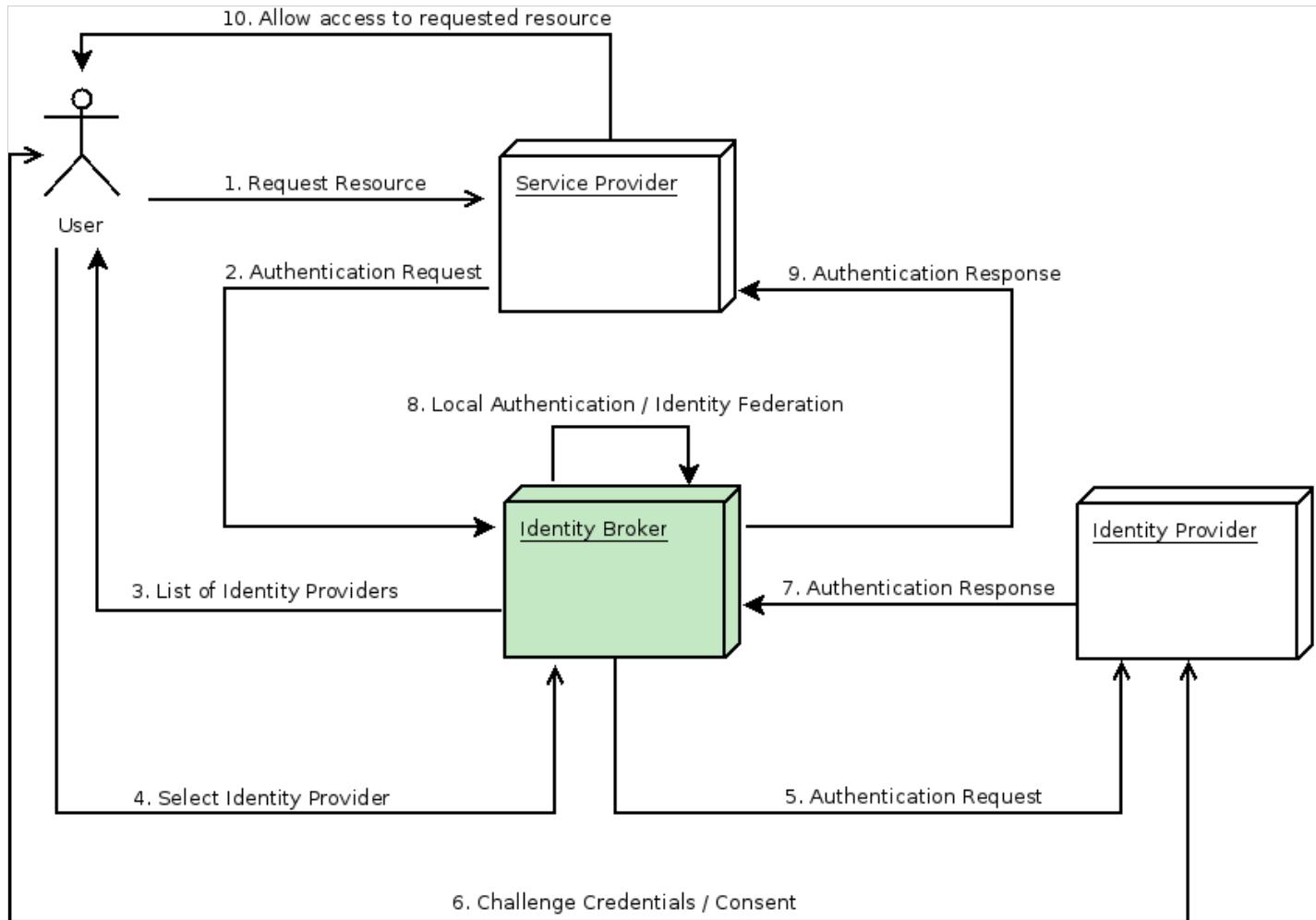
如果配置默认的标识提供程序，则 Keycloak 将用户重定向到默认的提供程序。



Different protocols may require different authentication flows. All the identity providers supported by Keycloak use the following flow.

不同的协议可能需要不同的身份验证流。

Identity broker flow 身份代理流



1. The unauthenticated user requests a protected resource in a client application.

未经身份验证的用户请求客户端应用程序中受保护的资源。

2. The client application redirects the user to Keycloak to authenticate.

客户端应用程序将用户重定向到 Keycloak 进行身份验证。

3. Keycloak displays the login page with a list of identity providers configured in a realm.

Key斗篷显示登录页面，其中包含在领域中配置的标识提供程序列表。

4. The user selects one of the identity providers by clicking its button or link.

用户通过单击标识提供程序的按钮或链接选择其中一个标识提供程序。

5. Keycloak issues an authentication request to the target identity provider requesting authentication and redirects the user to the identity provider's login page. The administrator has already set the connection properties and other configuration options for the Admin Console's identity provider.

Keycover 向请求身份验证的目标标识提供程序发出身份验证请求，并将用户重定向到标识提供程序的登录页。管理员已经为管理控制台的标识提供程序设置了连接属性和其他配置选项。

6. The user provides credentials or consents to authenticate with the identity provider.

用户提供凭据或同意与标识提供程序进行身份验证。

7. Upon successful authentication by the identity provider, the user redirects back to Keycloak with an authentication response. Usually, the response contains a security token used by Keycloak to trust the identity provider's authentication and retrieve user information.

当身份验证服务提供者成功验证身份后，用户会重定向回 Keycloak，并作出验证回应。通常，响应中包含一个安全令牌，由 Keycon 用于信任身份验证提供程序的身份验证和检索用户信息。

8. Keycloak checks if the response from the identity provider is valid. If valid, Keycloak imports and creates a user if the user does not already exist. Keycloak may ask the identity provider for further user information if the token does not contain that information. This behavior is *identity federation*. If the user already exists, Keycloak may ask the user to link the identity returned from the identity provider with the existing account. This behavior is *account linking*. With Keycloak, you can configure *Account linking* and specify it in the First Login Flow. At this step, Keycloak authenticates the user and issues its token to access the requested resource in the service provider.

键隐形检查来自标识提供程序的响应是否有效。如果有效，如果用户不存在，则 Key 导入并创建用户。如果令牌不包含进一步的用户信息，则密钥斗篷可能会要求身份提供程序提供进一步的用户信息。这种行为是身份联合。如果用户已经存在，则 Key枕可以要求用户将从标识提供程序返回的标识与现有帐户链接起来。此行为是帐户链接。使用 Keycon，您可以配置 Account 链接并在 First Login Flow 中指定它。在此步骤中，Keycon 验证用户并发出其令牌以访问服务提供程序中请求的资源。

9. When the user authenticates, Keycloak redirects the user to the service provider by sending the token previously issued during the local authentication.

当用户进行身份验证时，Keycover 通过发送在本地身份验证期间以前发出的令牌将用户重定向到服务提供者。

10. The service provider receives the token from Keycloak and permits access to the protected resource.

服务提供者从 Keycloak 接收令牌并允许访问受保护的资源。

Variations of this flow are possible. For example, the client application can request a specific identity provider rather than displaying a list of them, or you can set Keycloak to force users to provide additional information before federating their identity.

这种流动的变化是可能的。例如，客户端应用程序可以请求特定的标识提供程序，而不是显示它们的列表，或者可以设置 Key 来强制用户在联合标识之前提供其他信息。

At the end of the authentication process, Keycloak issues its token to client applications. Client applications are separate from the external identity providers, so they cannot see the client application's protocol or how they validate the user's identity. The provider only needs to know about Keycloak.

在身份验证过程结束时，Keycloak 将其令牌发送给客户端应用程序。客户端应用程序独立于外部标识提供程序，因此它们无法看到客户端应用程序的协议或它们如何验证用户的标识。提供者只需要知道钥匙斗篷。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/identity-broker/default-provider.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/identity-broker/default-provider.adoc)

Keycloak can redirect to an identity provider rather than displaying the login form. To enable this redirection:

键隐形可以重定向到标识提供程序，而不是显示登录表单：

Procedure 程序

1. Click **Authentication** in the menu.

单击菜单中的“身份验证”。

2. Click the **Browser flow**.

单击浏览器流。

3. Click the gear icon  on the **Identity Provider Redirector** row.

单击“身份提供程序重定向器”行上的齿轮图标 something。

4. Set **Default Identity Provider** to the identity provider you want to redirect users to.

将默认标识提供程序设置为要将用户重定向到的标识提供程序。

If Keycloak does not find the configured default identity provider, the login form is displayed.

如果没有找到已配置的默认标识提供程序，则显示登录窗体。

This authenticator is responsible for processing the `kc_idp_hint` query parameter. See the client suggested identity provider section for more information.

这个身份验证器负责处理 `kc_idp_tip` 查询参数。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/identity-broker/configuration.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/identity-broker/configuration.adoc)

The foundations of the identity broker configuration are identity providers (IDPs). Keycloak creates identity providers for each realm and enables them for every application by default. Users from a realm can use any of the registered identity providers when signing in to an application.

身份代理配置的基础是身份提供程序(IDP)。密钥斗篷为每个领域创建标识提供程序，并在默认情况下为每个应用程序启用它们。领域中的用户在登录到应用程序时可以使用任何已注册的标识提供程序。

Procedure 程序

1. Click **Identity Providers** in the menu.

单击菜单中的“标识提供程序”。

Identity Providers 身份证明文件提供者

The screenshot shows the 'Identity providers' configuration page in Keycloak. On the left, a sidebar lists navigation items like 'Master', 'Manage', 'Clients', etc., with 'Identity providers' currently selected. The main area is titled 'Identity providers' and contains a sub-header: 'Identity providers are social networks or identity brokers that allow users to authenticate to Keycloak.' Below this is a note: 'To get started, select a provider from the list below.' The page is divided into two sections: 'User-defined:' and 'Social:'. Under 'User-defined:', there are three cards: 'Keycloak OpenID Connect', 'OpenID Connect v1.0', and 'SAML v2.0'. Under 'Social:', there are several cards representing different providers: BitBucket, Facebook, GitHub, GitLab, Google, Instagram, LinkedIn, Microsoft, Openshift v3, Openshift v4, PayPal, StackOverflow, and Twitter. Each card has its logo and name.

2. Select an identity provider. Keycloak displays the configuration page for the identity provider you selected.

选择一个标识提供程序。Keycloak 显示所选标识提供程序的配置页。

Add Facebook identity Provider 添加 Facebook 身份提供商

The screenshot shows the 'Add Facebook provider' configuration page. The left sidebar is identical to the previous one. The main form is titled 'Add Facebook provider'. It contains four input fields: 'Redirect URI' with the value 'http://localhost:8180/realms/master/broker/facebook/endpoint', 'Client ID' (empty), 'Client Secret' (empty), and 'Display order' (empty). At the bottom are 'Add' and 'Cancel' buttons.

When you configure an identity provider, the identity provider appears on the Keycloak login page as an option. You can place custom icons on the login screen for each identity provider. See [custom icons](https://www.keycloak.org/docs/latest/server_development/#custom-identity-providers-icons) (https://www.keycloak.org/docs/latest/server_development/#custom-identity-providers-icons) for more information.

配置标识提供程序时，标识提供程序将作为一个选项出现在“钥匙斗篷”登录页上。您可以在每个标识提供程序的登录屏幕上放置自定义图标。有关更多信息，请参见自定义图标。

IDP login page IDP 登录页面

Sign in to your account

Username or email

Password

Remember me

Sign In

Or sign in with

Facebook

New user? [Register](#)

Social 社交

Social providers enable social authentication in your realm. With Keycloak, users can log in to your application using a social network account. Supported providers include Twitter, Facebook, Google, LinkedIn, Instagram, Microsoft, PayPal, Openshift v3, GitHub, GitLab, Bitbucket, and Stack Overflow.

社交提供程序在您的领域中启用社交身份验证。有了 Keycloak，用户可以使用社交网络帐户登录到您的应用程序。支持的提供商包括 Twitter、Facebook、Google、LinkedIn、Instagram、微软、PayPal、Openshift v3、GitHub、GitLab、Bitbucket 和 Stack Overflow。

Protocol-based 基于协议

Protocol-based providers rely on specific protocols to authenticate and authorize users. Using these providers, you can connect to any identity provider compliant with a specific protocol. Keycloak provides support for SAML v2.0 and OpenID Connect v1.0 protocols. You can configure and broker any identity provider based on these open standards.

基于协议的提供程序依赖于特定的协议来对用户进行身份验证和授权。使用这些提供程序，您可以连接到符合特定协议的任何标识提供程序。Keycloak 提供对 SAML v2.0 和 OpenID Connect v1.0 协议的支持。您可以根据这些开放标准配置和代理任何标识提供程序。

Although each type of identity provider has its configuration options, all share a common configuration. The following configuration options available:

尽管每种类型的标识提供程序都有自己的配置选项，但是它们都共享一个公共配置：

Table 1. Common Configuration 表1. 通用配置

Configuration 配置	Description 描述
Alias 别名	<p>The alias is a unique identifier for an identity provider and references an internal identity provider. Keycloak uses the alias to build redirect URIs for OpenID Connect protocols that require a redirect URI or callback URL to communicate with an identity provider. All identity providers must have an alias. Alias examples include <code>facebook</code>, <code>google</code>, and <code>idp.acme.com</code>.</p> <p>别名是身份提供者的唯一标识符，并引用内部身份提供者。密钥斗篷使用别名为 OpenID 连接协议构建重定向 URI，这些协议需要重定向 URI 或回调 URL 来与标识提供程序通信。所有身份提供程序都必须有一个别名。别名的例子包括 facebook、google 和 idp.acme.com。</p>
Enabled 启动	<p>Toggles the provider ON or OFF.</p> <p>切换提供程序的“开”或“关”。</p>
Hide on Login Page 隐藏在登录页面	<p>When ON, Keycloak does not display this provider as a login option on the login page. Clients can request this provider by using the 'kc_idp_hint' parameter in the URL to request a login.</p> <p>当 ON 时，在登录页面上不显示此提供程序作为登录选项。客户端可以通过使用 URL 中的“kc_idp_tip”参数请求此提供程序来请求登录。</p>
Account Linking Only 只连结帐户	<p>When ON, Keycloak links existing accounts with this provider. This provider cannot log users in, and Keycloak does not display this provider as an option on the login page.</p> <p>当开机时，“钥匙斗篷”将现有帐户链接到此提供程序。此提供程序无法让用户登录，并且在登录页面上不会将此提供程序显示为选项。</p>

Configuration 配置	Description 描述
Store Tokens 存储代币	When ON , Keycloak stores tokens from the identity provider. 当 ON 时, Key斗篷存储来自标识提供程序的令牌。
Stored Tokens Readable 存储令牌可读	When ON , users can retrieve the stored identity provider token. This action also applies to the <i>broker</i> client-level role <i>read token</i> . 在 ON 时, 用户可以检索存储的标识提供程序令牌。此操作也适用于代理客户端级别的角色读取令牌。
Trust Email 相信电子邮件	When ON , Keycloak trusts email addresses from the identity provider. If the realm requires email validation, users that log in from this identity provider do not need to perform the email verification process. 当 ON 时, Key斗篷信任来自身份提供程序的电子邮件地址。如果领域需要电子邮件验证, 那么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 Order 图形用户界面	The sort order of the available identity providers on the login page. 登录页上可用标识提供程序的排序顺序。
First Login Flow 首次登入流程	The authentication flow Keycloak triggers when users use this identity provider to log into Keycloak for the first time. 当用户首次使用此标识提供程序登录到钥匙斗篷时, 身份验证流将触发。

Configuration 配置	Description 描述
Post Login Flow 登入后流程	The authentication flow Keycloak triggers when a user finishes logging in with the external identity provider. 当用户完成与外部标识提供程序的登录时，身份验证流 Keycloak 触发。
Sync Mode 同步模式	Strategy to update user information from the identity provider through mappers. When choosing legacy , Keycloak used the current behavior. Import does not update user data and force updates user data when possible. See Identity Provider Mappers for more information. 通过映射器更新来自标识提供程序的用户信息的策略。选择“遗留”时，“钥匙斗篷”使用当前行为。导入不更新用户数据，并在可能时强制更新用户数据。 有关更多信息，请参见身份提供程序映射程序。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/identity-broker/social-login.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/identity-broker/social-login.adoc)

A social identity provider can delegate authentication to a trusted, respected social media account. Keycloak includes support for social networks such as Google, Facebook, Twitter, GitHub, LinkedIn, Microsoft, and Stack Overflow.

社交身份提供者可以将身份验证委托给受信任、受尊重的社交媒体帐户。Keycloak 包括对社交网络的支持，如谷歌、Facebook、Twitter、GitHub、LinkedIn、微软和 Stack Overflow。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/identity-broker/social/bitbucket.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/identity-broker/social/bitbucket.adoc)

To log in with Bitbucket, perform the following procedure.

若要使用 Bitbucket 登录，请执行以下过程。

Procedure 程序

1. Click **Identity Providers** in the menu.

单击菜单中的“标识提供程序”。

2. From the **Add provider** list, select **Bitbucket**.

从“添加提供程序”列表中，选择“Bitbucket”。

Add identity provider 添加标识提供程序

The screenshot shows the 'Add Bitbucket provider' configuration page in Keycloak. On the left, there's a sidebar with 'Master' selected and options like 'Manage', 'Clients', 'Client scopes', 'Realm roles', 'Users', 'Groups', 'Sessions', and 'Events'. The main area has a breadcrumb 'Identity providers > Add provider' and the title 'Add Bitbucket provider'. It contains fields for 'Redirect URI' (with a value of 'http://localhost:8180/realm/master/broker/bitbucket/endpoint'), 'Client ID' (empty), 'Client Secret' (empty), and 'Display order' (empty). At the bottom are 'Add' and 'Cancel' buttons.

3. Copy the value of **Redirect URI** to your clipboard.

将重定向 URI 的值复制到剪贴板。

4. In a separate browser tab, perform the [OAuth on Bitbucket Cloud](#)

(<https://support.atlassian.com/bitbucket-cloud/docs/use-oauth-on-bitbucket-cloud/>) process. When you click **Add Consumer**:

在单独的浏览器选项卡中，执行 Bitbucket 云进程的 OAuth:

a. Paste the value of **Redirect URI** into the **Callback URL** field.

将 Redirect URI 的值粘贴到 Callback URL 字段中。

b. Ensure you select **Email** and **Read** in the **Account** section to permit your application to read email.

确保您选择电子邮件和阅读帐户部分，以允许您的申请阅读电子邮件。

5. Note the **Key** and **Secret** values Bitbucket displays when you create your consumer.

注意创建使用者时 Bitbucket 显示的 Key 和 Secret 值。

6. In Keycloak, paste the value of the **Key** into the **Consumer Key** field.

在 Keycloak，将 Key 的值粘贴到 Consumer Key 字段中。

7. In Keycloak, paste the value of the **Secret** into the **Consumer Secret** field.

在 Keycloak，将“秘密”的价值粘贴到“消费者秘密”字段中。

8. Click **Add**.

单击“添加”。

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/identity-broker/social/facebook.adoc)

Procedure 程序

1. Click **Identity Providers** in the menu.

单击菜单中的“标识提供程序”。

2. From the **Add provider** list, select **Facebook**. Keycloak displays the configuration page for the Facebook identity provider.

从“添加提供程序”列表中，选择“Facebook.Key 显示 Facebook 身份提供程序的配置页面”。

Add identity provider 添加标识提供程序

The screenshot shows the 'Add Facebook provider' configuration page in Keycloak. On the left, there is a sidebar with a 'Master' dropdown and a 'Manage' section containing links for Clients, Client scopes, Realm roles, Users, Groups, Sessions, and Events. The main area has a title 'Add Facebook provider'. It contains four input fields: 'Redirect URI' with the value 'http://localhost:8180/realm/master/broker/facebook/endpoint', 'Client ID' (empty), 'Client Secret' (empty), and 'Display order' (empty). At the bottom are 'Add' and 'Cancel' buttons.

3. Copy the value of **Redirect URI** to your clipboard.

将重定向 URI 的值复制到剪贴板。

4. In a separate browser tab, follow the [Facebook Developer Guide's](#) (<https://developers.facebook.com/docs/development/>) instructions to create a project and client in Facebook.

在单独的浏览器选项卡中，按照 Facebook 开发者指南的说明在 Facebook 中创建项目和客户端。

- a. Ensure your app is a website-type app.

确保你的应用程序是一个网站类型的应用程序。

- b. Enter the **Redirect URI**'s value into the **Site URL** of the Facebook **Website** settings block.

在 Facebook 网站设置块的网站 URL 中输入重定向 URI 的值。

- c. Ensure the app is public.

确保应用程序是公开的。

5. Enter the **Client ID** and **Client Secret**

(<https://developers.facebook.com/docs/facebook-login/guides/access-tokens>) values from your Facebook app into the **Client ID** and **Client Secret** fields in Keycloak.

将 Facebook 应用程序中的客户 ID 和客户机密值输入到 Keycloak 的客户 ID 和客户机密字段中。

6. Click Add

单击“添加”

7. Enter the required scopes into the **Default Scopes** field. By default, Keycloak uses the `email` scope. See [Graph API](#) (<https://developers.facebook.com/docs/graph-api>) for more information about Facebook scopes.

在“默认作用域”字段中输入所需的作用域。默认情况下，Keycloak 使用电子邮件作用域。有关 Facebook 范围的更多信息，请参见 GraphAPI。

Keycloak sends profile requests to `graph.facebook.com/me?`

`fields=id,name,email,first_name,last_name` by default. The response contains the id, name, email, first_name, and last_name fields only. To fetch additional fields from the Facebook profile, add a corresponding scope and add the field name in the **Additional user's profile fields** configuration option field.

默认情况下，“钥匙斗篷”会向 `graph.facebook.com/me?fields=id,name,email,first_name,last_name` 发送配置文件请求。响应只包含 id、name、email、first_name 和 last_name 字段。要从 Facebook ID 中获取其他字段，请添加相应的作用域，并在“附加用户配置文件字段配置选项”字段中添加字段名。

[Edit this section](#) 编辑这部分

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/identity-broker/social/github.adoc)

[Report an issue](#) 报告一个问题

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/identity-broker/social/github.adoc)

To log in with GitHub, perform the following procedure.

若要使用 GitHub 登录，请执行以下过程。

Procedure 程序

1. Click **Identity Providers** in the menu.

单击菜单中的“身份提供程序”。

2. From the **Add provider** list, select **Github**.

从“添加提供程序”列表中，选择 Github。

Add identity provider 添加标识提供程序

Master

Manage

Clients

Client scopes

Realm roles

Users

Groups

Sessions

Events

Identity providers > Add provider

Add Github provider

Redirect URI ⓘ http://localhost:8180/realm/master/broker/github/endpoint

Client ID ⓘ

Client Secret ⓘ

Display order ⓘ

Add Cancel

3. Copy the value of **Redirect URI** to your clipboard.

将重定向 URI 的值复制到剪贴板。

4. In a separate browser tab, [create an OAUTH app](#)

(<https://docs.github.com/en/developers/apps/building-oauth-apps/creating-an-oauth-app>).

在单独的浏览器选项卡中，创建一个 OAUTH 应用程序。

a. Enter the value of **Redirect URI** into the **Authorization callback URL** field when creating the app.

在创建应用程序时，在 Authorization 回调 URL 字段中输入重定向 URI 的值。

b. Note the Client ID and Client secret on the management page of your OAUTH app.

请注意 OAUTH 应用程序管理页面上的客户端 ID 和客户端机密。

5. In Keycloak, paste the value of the **Client ID** into the **Client ID** field.

在 Keycloak，将客户 ID 的值粘贴到客户 ID 字段中。

6. In Keycloak, paste the value of the **Client Secret** into the **Client Secret** field.

在 Keycloak，将 Client Secret 的值粘贴到 Client Secret 字段中。

7. Click **Add**.

单击“添加”。

[Edit this section 编辑这部分](#)
(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/identity-broker/social/gitlab.adoc)

[Report an issue 报告一个问题](#)
(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/identity-broker/social/gitlab.adoc)

Procedure 程序

1. Click **Identity Providers** in the menu.

单击菜单中的“标识提供程序”。

2. From the **Add provider** list, select **GitLab**.

从“添加提供程序”列表中，选择“GitLab”。

Add identity provider 添加标识提供程序

Identity providers > Add provider

Add Gitlab provider

Redirect URI ⓘ http://localhost:8180/realms/master/broker/gitlab/endpoint

Client ID * ⓘ

Client Secret * ⓘ

Display order ⓘ

Add Cancel

3. Copy the value of **Redirect URI** to your clipboard.

将重定向 URI 的值复制到剪贴板。

4. In a separate browser tab, [add a new GitLab application](#)

(https://docs.gitlab.com/ee/integration/oauth_provider.html).

在单独的浏览器选项卡中，添加一个新的 GitLab 应用程序。

a. Use the **Redirect URI** in your clipboard as the **Redirect URI**.

使用剪贴板中的重定向 URI 作为重定向 URI。

b. Note the **Client ID** and **Client Secret** when you save the application.

保存应用程序时请注意客户端 ID 和客户端机密。

5. In Keycloak, paste the value of the **Client ID** into the **Client ID** field.

在 Keycloak，将客户 ID 的值粘贴到客户 ID 字段中。

6. In Keycloak, paste the value of the **Client Secret** into the **Client Secret** field.

在 Keycloak，将 Client Secret 的值粘贴到 Client Secret 字段中。

7. Click **Add**.

单击“添加”。

Edit this section 编辑这部分

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/identity-broker/social/google.adoc)

Report an issue 报告一个问题

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/identity-broker/social/google.adoc)

Procedure 程序

1. Click **Identity Providers** in the menu.

单击菜单中的“标识提供程序”。

2. From the Add provider list, select Google .

从“添加提供程序”列表中，选择“ Google ”。

Add identity provider 添加标识提供程序

The screenshot shows the 'Add Google provider' configuration page. On the left is a sidebar with 'Master' selected and options: Manage, Clients, Client scopes, Realm roles, Users, Groups, Sessions, and Events. The main area has 'Identity providers > Add provider' at the top. It contains fields for 'Redirect URI' (http://localhost:8180/realm/master/broker/google/endpoint), 'Client ID' (empty), 'Client Secret' (empty), and 'Display order' (empty). At the bottom are 'Add' and 'Cancel' buttons.

3. In a separate browser tab open the Google Cloud Platform console (<https://console.cloud.google.com/>).

在一个单独的浏览器标签打开谷歌云平台控制台。

4. In the Google dashboard for your Google app, click the OAuth consent screen menu. Create a consent screen, ensuring that the user type of the consent screen is external.

在您的 Google 应用程序的 Google 仪表板中，单击 OAuth 同意屏幕菜单。创建一个同意屏幕，确保同意屏幕的用户类型是外部的。

5. In the Google dashboard:

在谷歌的仪表盘上：

a. Click the Credentials menu.

单击“凭据”菜单。

b. Click CREATE CREDENTIALS - OAuth Client ID.

单击 CREATE CREDENTIALS-OAuth 客户端 ID。

c. From the Application type list, select Web application.

从“应用程序类型”列表中，选择“ Web 应用程序”。

d. Click Create.

单击 Create。

e. Note Your Client ID and Your Client Secret.

请注意您的客户 ID 和您的客户秘密。

6. In Keycloak, paste the value of the Your Client ID into the Client ID field.

在 Keycloak，将“您的客户 ID”的值粘贴到“客户 ID”字段中。

7. In Keycloak, paste the value of the Your Client Secret into the Client Secret field.

在 Keycloak，将“你的客户机密”的值粘贴到“客户机密”字段中。

8. Click Add

单击“添加”

9. Enter the required scopes into the **Default Scopes** field. By default, Keycloak uses the following scopes: `openid profile email`. See the [OAuth Playground](https://developers.google.com/oauthplayground) (<https://developers.google.com/oauthplayground/>) for a list of Google scopes.

在“默认作用域”字段中输入所需的作用域。默认情况下，Keycloak 使用以下作用域: `openid` 配置文件电子邮件。有关 Google 作用域的列表，请参见 OAuth Playground。

10. To restrict access to your GSuite organization's members only, enter the G Suite domain into the **Hosted Domain** field.

要限制只能访问 GSuite 组织的成员，请在“托管域”字段中输入 G Suite 域。

11. Click Save.

单击“保存”。

[Edit this section](#) 编辑这部分

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/identity-broker/social/linked-in.adoc)

[Report an issue](#) 报告一个问题

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/identity-broker/social/linked-in.adoc)

Procedure 程序

1. Click **Identity Providers** in the menu.

单击菜单中的“身份提供程序”。

2. From the **Add provider** list, select **LinkedIn**.

从“添加提供程序”列表中，选择 LinkedIn。

Add identity provider 添加标识提供程序

The screenshot shows the 'Add identity provider' dialog for LinkedIn. On the left is a sidebar with navigation links: Master, Manage, Clients, Client scopes, Realm roles, Users, Groups, Sessions, and Events. The 'Master' link is currently selected. The main area has a title 'Add LinkedIn provider'. It contains several input fields: 'Redirect URI' with the value 'http://localhost:8180/realm/master/broker/linkedin/endpoint'; 'Client ID' (marked with a red asterisk) which is empty; 'Client Secret' (marked with a red asterisk) which is also empty; and 'Display order' which is empty. At the bottom are 'Add' and 'Cancel' buttons.

3. Copy the value of **Redirect URI** to your clipboard.

将重定向 URI 的值复制到剪贴板。

4. In a separate browser tab, [create an app](https://www.linkedin.com/developer/apps) (<https://www.linkedin.com/developer/apps>).

在单独的浏览器选项卡中，创建一个应用程序。

a. After you create the app, click the **Auth** tab.

创建应用程序后，单击 Auth 选项卡。

b. Enter the value of **Redirect URI** into the **Authorized redirect URLs for your app** field.

在应用程序字段的 Authorized redirect URL 中输入重定向 URI 的值。

c. Note **Your Client ID** and **Your Client Secret**.

请注意您的客户 ID 和您的客户机密。

5. In Keycloak, paste the value of the **Client ID** into the **Client ID** field.

在 Keycloak，将客户 ID 的值粘贴到客户 ID 字段中。

6. In Keycloak, paste the value of the **Client Secret** into the **Client Secret** field.

在 Keycloak，将 Client Secret 的值粘贴到 Client Secret 字段中。

7. Click **Add**.

单击“添加”。

Configuration 配置

- With **Profile Projection** you can configure the **projection** parameter for profile requests.

使用概要文件投影，您可以为概要文件请求配置投影参数。

- For example, `(id,firstName,lastName,profilePicture(displayImage~:playableStreams))` produces the following profile request URL:

例如，(id、firstName、lastName、profilePicture (displayImage ~ : playableStreams))生成以下配置文件请求 URL:

```
https://api.linkedin.com/v2/me?projection=
(id,firstName,lastName,profilePicture(displayImage~:playableStreams))
```

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/identity-broker/social/microsoft.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/identity-broker/social/microsoft.adoc)

Procedure 程序

1. Click **Identity Providers** in the menu.

单击菜单中的“身份提供程序”。

2. From the **Add provider** list, select **Microsoft**.

从“添加提供程序”列表中，选择 Microsoft。

Add identity provider 添加标识提供程序

Identity providers > Add provider

Add Microsoft provider

Redirect URI

Client ID

Client Secret

Display order

Add Cancel

3. Copy the value of **Redirect URI** to your clipboard.

将重定向 URI 的值复制到剪贴板。

4. In a separate browser tab, create a Microsoft app (<https://account.live.com/developers/applications/create>).

在单独的浏览器选项卡中，创建一个 Microsoft 应用程序。

a. Click **Add URL** to add the redirect URL to the Microsoft app.

单击 AddURL 将重定向 URL 添加到 Microsoft 应用程序。

b. Note the **Application ID** and **Application Secret**.

注意申请 ID 和申请机密。

5. In Keycloak, paste the value of the **Application ID** into the **Client ID** field.

在 Keycloak，将 ApplicationID 的值粘贴到 ClientID 字段中。

6. In Keycloak, paste the value of the **Application Secret** into the **Client Secret** field.

在 Keycloak，将“应用秘密”的值粘贴到“客户秘密”字段中。

7. Click **Add**.

单击“添加”。

[Edit this section 编辑这部分](#)
(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/identity-broker/social/openshift.adoc)

[Report an issue 报告一个问题](#)
(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/identity-broker/social/openshift.adoc)

Procedure 程序

1. Click **Identity Providers** in the menu.

单击菜单中的“身份提供程序”。

2. From the **Add provider** list, select **Openshift**.

从“添加提供程序”列表中，选择“Openshift”。

Add identity provider 添加标识提供程序

Identity providers > Add provider

Add Openshift-v3 provider

Redirect URI ⓘ

Client ID * ⓘ

Client Secret * ⓘ

Display order ⓘ

3. Copy the value of Redirect URI to your clipboard.

将重定向 URI 的值复制到剪贴板。

4. Register your client using the `oc` command-line tool.

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  
' )
```

The 那个 `name` of your OAuth client. Passed as 您的 OAuth 客户端。通过为 `client_id`

① request parameter when making requests to 请求时，请求参数 `<openshift_master>/oauth/authorize` 和 `<openshift_master>/oauth/token`。

② The 那个 `secret` Keycloak uses for the 密钥斗篷用于 `client_secret` request parameter. 请求参数

The 那个 `redirect_uri` parameter specified in requests to 请求中指定的参数

③ `<openshift_master>/oauth/authorize` 和 `<openshift_master>/oauth/token` must be equal to (or prefixed by) one of the URIs in 中的一个 URI 相等(或前缀) `redirectURIs`。您可以从 `Redirect URI` 重定向 URI field in the Identity Provider screen “身份提供程序”屏幕中的

- The `grantMethod` Keycloak uses to determine the action when this client requests tokens but has not been granted access by the user. 当此客户端请求令牌但用户尚未授予访问权限时，键斗篷用于确定操作

OpenShift 4

Prerequisites 先决条件

1. Installation of jq (<https://stedolan.github.io/jq/>).

安装 jq。

2. X509_CA_BUNDLE configured in the container and set to
`/var/run/secrets/kubernetes.io/serviceaccount/ca.crt`.

在容器中配置并设置为`/var/run/secret/kubernetes.io/serviceaccount/ca.crt`的 X509 _ CA _ BUNDLE。

Procedure 程序

1. Run the following command on the command line and note the OpenShift 4 API URL output.

在命令行上运行以下命令，并注意 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"
```

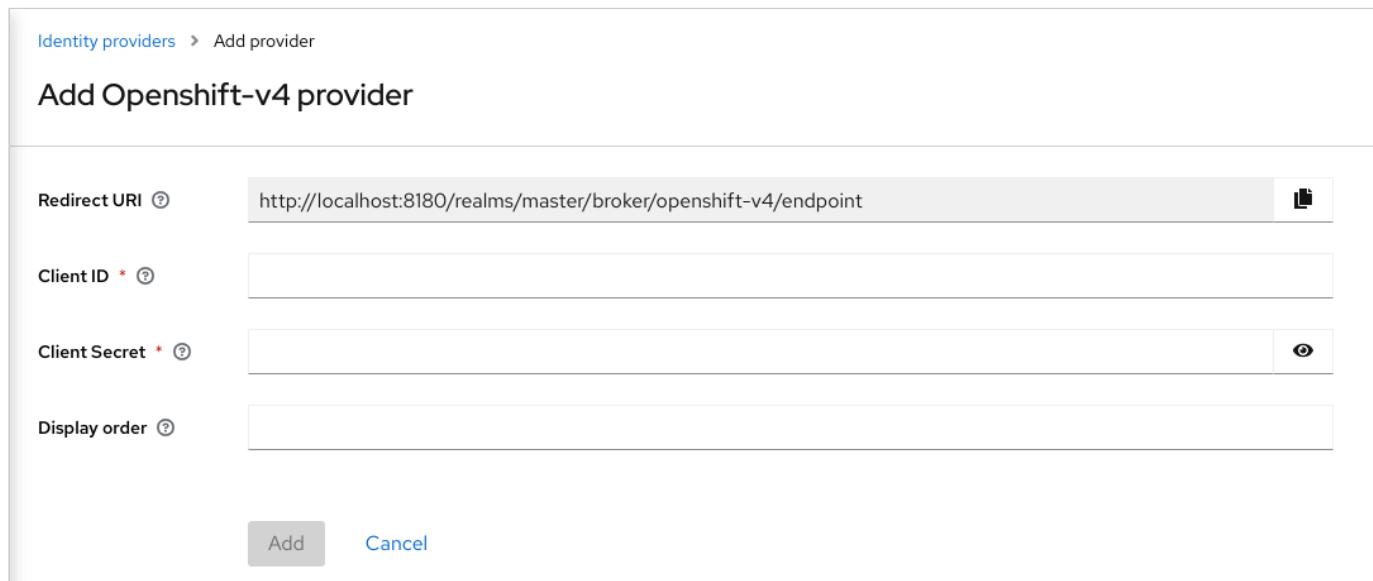
2. Click Identity Providers in the Keycloak menu.

单击 Keycloak 菜单中的“身份提供程序”。

3. From the Add provider list, select Openshift .

从“添加提供程序”列表中，选择“Openshift”。

Add identity provider 添加标识提供程序



The screenshot shows the 'Add Openshift-v4 provider' dialog. It includes fields for Redirect URI (http://localhost:8180/realms/master/broker/openshift-v4/endpoint), Client ID (empty), Client Secret (empty), and Display order (empty). At the bottom are 'Add' and 'Cancel' buttons.

Identity providers > Add provider

Add Openshift-v4 provider

Redirect URI

Client ID *

Client Secret *

Display order

4. Copy the value of Redirect URI to your clipboard.

将重定向 URI 的值复制到剪贴板。

5. Register your client using the `oc` command-line tool.

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 ❶  
  secret: "..." ❷  
  redirectURIs:  
    - "<copy pasted Redirect URI from OpenShift 4 Identity Providers page>" ❸  
  grantMethod: prompt ❹  
' )
```

The 那个 `name` of your OAuth client. Passed as 您的 OAuth 客户端。通过为 `client_id` request parameter when making requests to 请求时, 请求参数

❶ `<openshift_master>/oauth/authorize` and 还有 `<openshift_master>/oauth/token`。The 的 `name` parameter must be the same in the 参数必须相同 `OAuthClient` object and the Keycloak configuration. 对象和钥匙斗篷配置

❷ The 那个 `secret` Keycloak uses as the 密钥斗篷用作 `client_secret` request parameter. 请求参数

The 那个 `redirect_uri` parameter specified in requests to 请求中指定的参数

❸ `<openshift_master>/oauth/authorize` and 还有 `<openshift_master>/oauth/token` must be equal to (or prefixed by) one of the URIs in 中的一个 URI 相等(或前缀) `redirectURIs`。The easiest way to configure it correctly is to copy-paste it from Keycloak OpenShift 4 Identity Provider configuration page (.要正确配置它, 最简单的方法是将其复制粘贴到 KeycloakOpenShift 4 Identity Provider 配置页(`Redirect URI` field)). 字段)

❹ The 那个 `grantMethod` Keycloak uses to determine the action when this client requests tokens but has not been granted access by the user. 当此客户端请求令牌但用户尚未授予访问权限时, 键斗篷用于确定操作

See [official OpenShift documentation](#)

(https://docs.okd.io/latest/authentication/configuring-oauth-clients.html#oauth-register-additional-client_configuring-oauth-clients)

for more information.

有关更多信息, 请参见 OpenShift 官方文档。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/identity-broker/social/paypal.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/identity-broker/social/paypal.adoc)

1. Click **Identity Providers** in the menu.

单击菜单中的“身份提供程序”。

2. From the **Add provider** list, select **PayPal**.

从“添加提供程序”列表中，选择 PayPal。

Add identity provider 添加标识提供程序

The screenshot shows the 'Add Paypal provider' configuration page in Keycloak. On the left, a sidebar lists 'Master' and 'Manage' sections, followed by 'Clients', 'Client scopes', 'Realm roles', 'Users', 'Groups', 'Sessions', and 'Events'. The main area is titled 'Add Paypal provider' and contains the following fields:

- Redirect URI**: http://localhost:8180/realms/master/broker/paypal/endpoint
- Client ID**: (empty input field)
- Client Secret**: (empty input field)
- Display order**: (empty input field)

At the bottom are 'Add' and 'Cancel' buttons.

3. Copy the value of **Redirect URI** to your clipboard.

将重定向 URI 的值复制到剪贴板。

4. In a separate browser tab, open the [PayPal Developer applications area](https://developer.paypal.com/developer/applications)

(<https://developer.paypal.com/developer/applications>).

在单独的浏览器选项卡中，打开 PayPal Developer 应用程序区域。

a. Click **Create App** to create a PayPal app.

单击 CreateApp 创建 PayPal 应用程序。

b. Note the Client ID and Client Secret. Click the **Show** link to view the secret.

注意客户端 ID 和客户端机密。单击 Show 链接查看机密。

c. Ensure **Connect with PayPal** is checked.

确保勾选了“与 PayPal 连接”。

d. Set the value of the **Return URL** field to the value of **Redirect URI** from Keycloak.

将 Return URL 字段的值设置为 Redirect URI from Keycloak 的值。

5. In Keycloak, paste the value of the **Client ID** into the **Client ID** field.

在 Keycloak，将客户 ID 的值粘贴到客户 ID 字段中。

6. In Keycloak, paste the value of the **Client Secret** into the **Client Secret** field.

在 Keycloak，将 Client Secret 的值粘贴到 Client Secret 字段中。

7. Click **Add**.

单击“添加”。

[Edit this section 编辑此部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/identity-broker/social/stack-overflow.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/identity-broker/social/stack-overflow.adoc)

Procedure 程序

1. Click **Identity Providers** in the menu.

单击菜单中的“身份提供程序”。

2. From the **Add provider** list, select **Stack Overflow**.

从“添加提供程序”列表中，选择“堆栈溢出”。

Add identity provider 添加标识提供程序

The screenshot shows the 'Add Stackoverflow provider' configuration page. On the left, there is a sidebar with a dropdown menu set to 'Master' and a list of management options: Manage, Clients, Client scopes, Realm roles, Users, Groups, Sessions, and Events. The main area has a breadcrumb navigation path: 'Identity providers > Add provider'. The form fields are as follows:

- Redirect URI**: http://localhost:8180/realm/master/broker/stackoverflow/endpoint
- Client ID**: (empty input field)
- Client Secret**: (empty input field with a visibility icon)
- Display order**: (empty input field)

At the bottom right are 'Add' and 'Cancel' buttons.

3. In a separate browser tab, log into [registering your application on Stack Apps](#)

(<https://stackapps.com/apps/oauth/register>).

在单独的浏览器选项卡中，登录到 StackApps 上注册应用程序。

Register application 登记申请

StackExchange ▾ 1 help ▾ Search Q&A

stackapps

Questions Tags Users Badges Unanswered Ask Question

Register Your V2.0 Application

Application Name
Be Unique! Avoid implying an official Stack Exchange relationship.

Description

OAuth Domain
example.com, subdomains will be automatically whitelisted

Application Website
Where users can go to read about your application

Application Icon (optional)
Must be hosted by the Stack Exchange Imgur account Enable Client Side OAuth Flow

Why Register?

Because it's the neighborly thing to do. We like to know who is using our API, and how, so we can have the metrics we need to support your application and improve the API together.

Once it's ready for public consumption, we'll [help you promote your registered application](#) here on Stack Apps.

Upon registering, you'll be provided an API key which grants your app a **much** larger per-day [request quota](#) than using the API anonymously.

You'll also receive parameters for [authenticating users via OAuth 2.0](#).

Register Your Application

a. Enter your application name into the **Application Name** field.

在 ApplicationName 字段中输入应用程序名称。

b. Enter the OAuth domain into the **OAuth Domain** field.

在 OAuth Domain 字段中输入 OAuth 域。

c. Click **Register Your Application**.

单击“注册您的应用程序”。

Settings 设定



Questions Tags Users Badges Unanswered Ask Question

Keycloak

Client Id

7209

This Id identifies your application to the Stack Exchange API. Your application client id is **not** secret, and may be safely embeded in distributed binaries.

Pass this as `client_id` in our OAuth 2.0 flow.

Client Secret ([reset](#))

A8M5pezJvqp9G)Nfx6aw9A((

Pass this as `client_secret` in our OAuth 2.0 flow if your app uses the explicit path.

This **must be** kept secret. Do not embed it in client side code or binaries you intend to distribute. If you need client side authentication, use the implicit OAuth 2.0 flow.

Key

sZA2lCUcqAr6ZkBkpss4w((

Pass this as `key` when making requests against the Stack Exchange API to receive a [higher request quota](#).

This is not considered a secret, and may be safely embed in client side code or distributed binaries.

Description

Keycloak

This **text-only** blurb will be shown to users during authentication.

OAuth Domain

More Info

- [Authentication Statistics](#)
- [API Documentation](#)

4. Note the Client ID and Client Secret.

注意客户 ID 和客户机密。

5. In Keycloak, paste the value of the Client ID into the Client ID field.

在 Keycloak，将客户 ID 的值粘贴到客户 ID 字段中。

6. In Keycloak, paste the value of the Client Secret into the Client Secret field.

在 Keycloak，将 Client Secret 的值粘贴到 Client Secret 字段中。

7. Click Add.

单击“添加”。

[Edit this section 编辑这部分](#)
(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/identity-broker/social/twitter.adoc)

[Report an issue 报告一个问题](#)
(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/identity-broker/social/twitter.adoc)

Prerequisites 先决条件

1. A Twitter developer account.

一个 Twitter 开发者账号。

Procedure 程序

1. Click Identity Providers in the menu.

单击菜单中的“标识提供程序”。

2. From the **Add provider** list, select **Twitter**.

从“添加提供程序”列表中，选择“Twitter”。

Add identity provider 添加标识提供程序

The screenshot shows the 'Add Twitter provider' configuration page in Keycloak. On the left, there's a sidebar with 'Master' selected and a 'Manage' section containing links for Clients, Client scopes, Realm roles, Users, Groups, Sessions, and Events. The main area has a title 'Add Twitter provider'. It contains fields for 'Redirect URI' (set to 'http://localhost:8180/realm/master/broker/twitter/endpoint'), 'Client ID' (empty), 'Client Secret' (empty), and 'Display order' (empty). At the bottom are 'Add' and 'Cancel' buttons.

3. Copy the value of **Redirect URI** to your clipboard.

将重定向 URI 的值复制到剪贴板。

4. In a separate browser tab, create an app in [Twitter Application Management](https://developer.twitter.com/apps/) (<https://developer.twitter.com/apps/>).

在单独的浏览器选项卡中，在 Twitter 应用程序管理中创建一个应用程序。

- a. Enter any value for name and description.

为名称和描述输入任何值。

- b. The value for **Website** can be any valid URL except **localhost**.

网站的值可以是除 localhost 之外的任何有效 URL。

- c. Paste the value of the **Redirect URL** into the **Callback URL** field.

将重定向 URL 的值粘贴到 CallbackURL 字段中。

- d. When you create your Twitter app, note the value of **Consumer Key** and **Consumer Secret** in the **Keys and Access Tokens** section.

在创建 Twitter 应用程序时，请注意“密钥和访问令牌”部分中的“消费者密钥”和“消费者秘密”的值。

5. In Keycloak, paste the value of the **Consumer Key** into the **Client ID** field.

在 Keycloak，将 Consumer Key 的值粘贴到 Client ID 字段中。

6. In Keycloak, paste the value of the **Consumer Secret** into the **Client Secret** field.

在 Keycloak，将“消费者秘密”的值粘贴到“客户秘密”字段中。

7. Click **Add**.

单击“添加”。

[Edit this section](#) [Report a problem](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/identity-broker/social/instagram.adoc)

[Report an issue](#) [报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/identity-broker/social/instagram.adoc)

Procedure 程序

1. Click **Identity Providers** in the menu.

单击菜单中的“标识提供程序”。

2. From the **Add provider** list, select **Instagram**. Keycloak displays the configuration page for the Instagram identity provider.

从“添加提供程序”列表中，选择“Instagram. Key 显示 Instagram 标识提供程序的配置页”。

Add identity provider 添加标识提供程序

The screenshot shows the 'Add Instagram provider' configuration page. On the left is a sidebar with navigation links: Master, Manage, Clients, Client scopes, Realm roles, Users, Groups, Sessions, and Events. The 'Master' link is highlighted. The main area has a title 'Add Instagram provider'. It contains several input fields: 'Redirect URI' with the value 'http://localhost:8180/realm/master/broker/instagram/endpoint', 'Client ID' (empty), 'Client Secret' (empty), and 'Display order' (empty). At the bottom are 'Add' and 'Cancel' buttons.

3. Copy the value of **Redirect URI** to your clipboard.

将重定向 URI 的值复制到剪贴板。

4. In a separate browser tab, open the [Facebook Developer Console](#) (<https://developers.facebook.com/>).

在单独的浏览器选项卡中，打开 Facebook 开发者控制台。

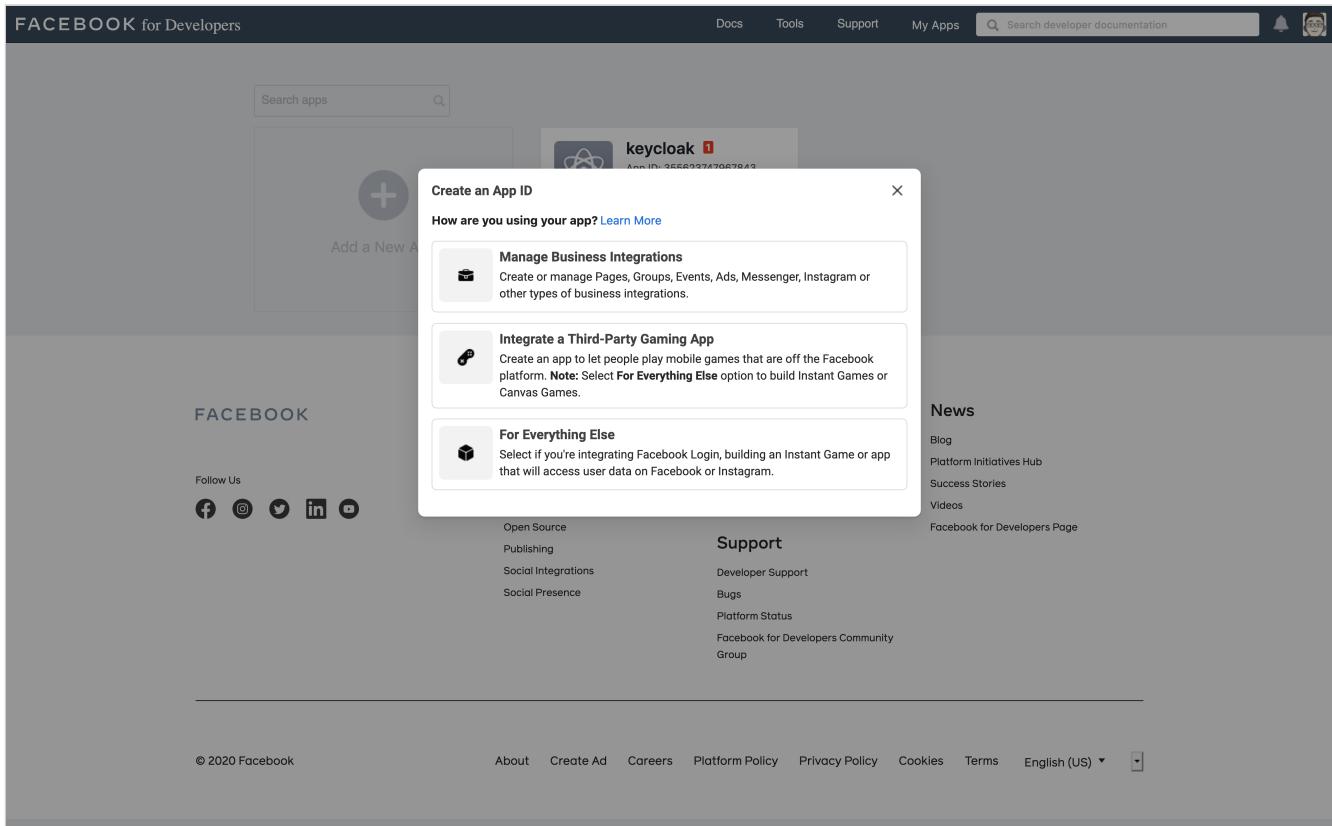
- a. Click **My Apps**.

点击我的应用程序。

- b. Select **Add a New App**.

选择“添加新应用程序”。

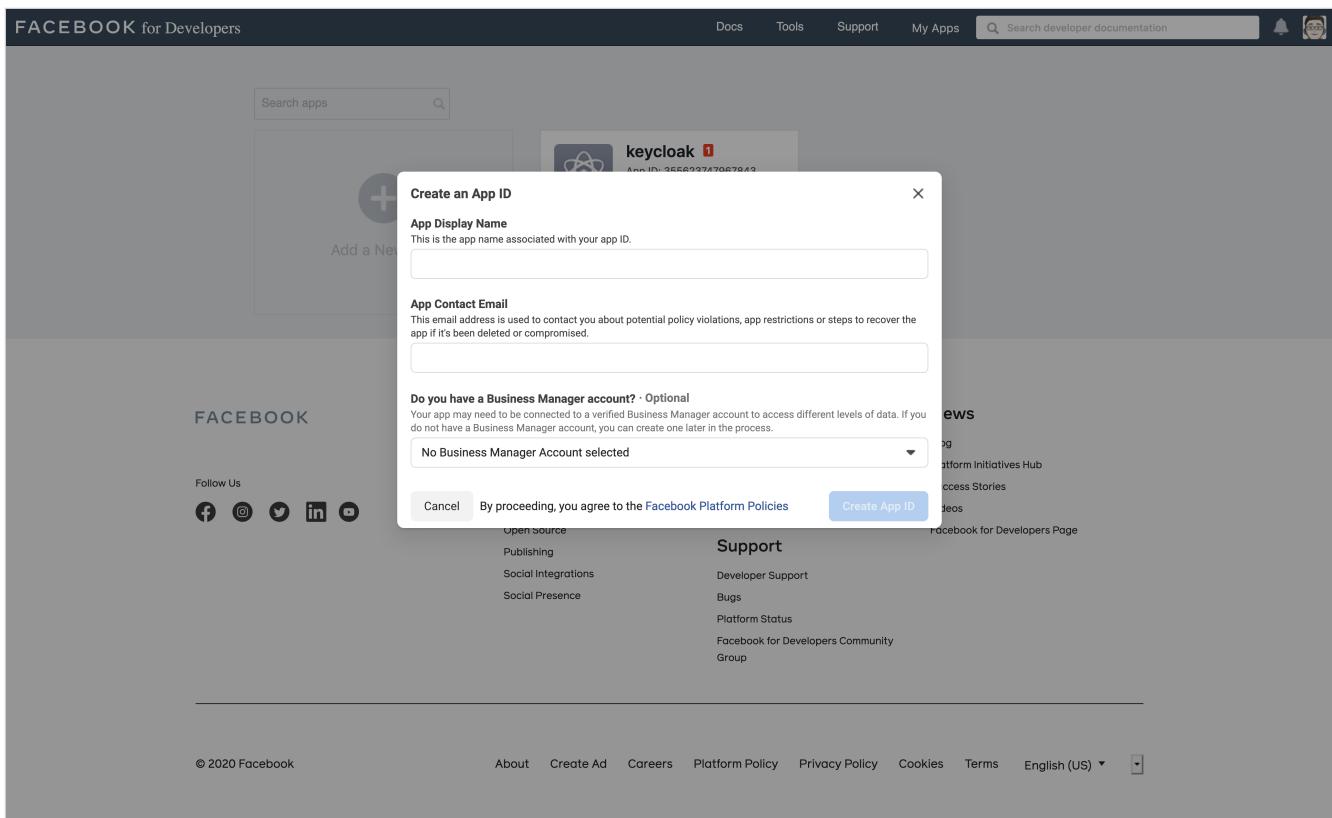
Add a new app 添加一个新的应用程序



c. Select For Everything Else.

选择其他所有选项。

Create a new app ID 创建一个新的应用程序 ID



d. Fill in all required fields.

填写所有必填字段。

e. Click Create App. Facebook then brings you to the dashboard.

单击 CreateApp.Facebook，然后将您带到仪表板。

f. In the navigation panel, select **Settings - Basic**.

在导航面板中，选择 Settings-Basic。

Add platform 添加平台

The screenshot shows the Facebook for Developers dashboard with the 'Basic' settings selected. On the left, there's a sidebar with options like Dashboard, Settings (selected), Basic, Advanced, Roles, Alerts, App Review, Products (Instagram), and Activity Log. The main content area is titled 'Data Protection Officer Contact Information'. It includes a note about GDPR requirements for companies in the EU to designate a Data Protection Officer. There are fields for Name (optional), Email, Street Address, Apt/Suite/Other (Optional), City/District, State/Province/Region, ZIP/Postal Code, and Country (set to United States). At the bottom right are 'Discard' and 'Save Changes' buttons.

g. Select + Add Platform.

选择 + 添加平台。

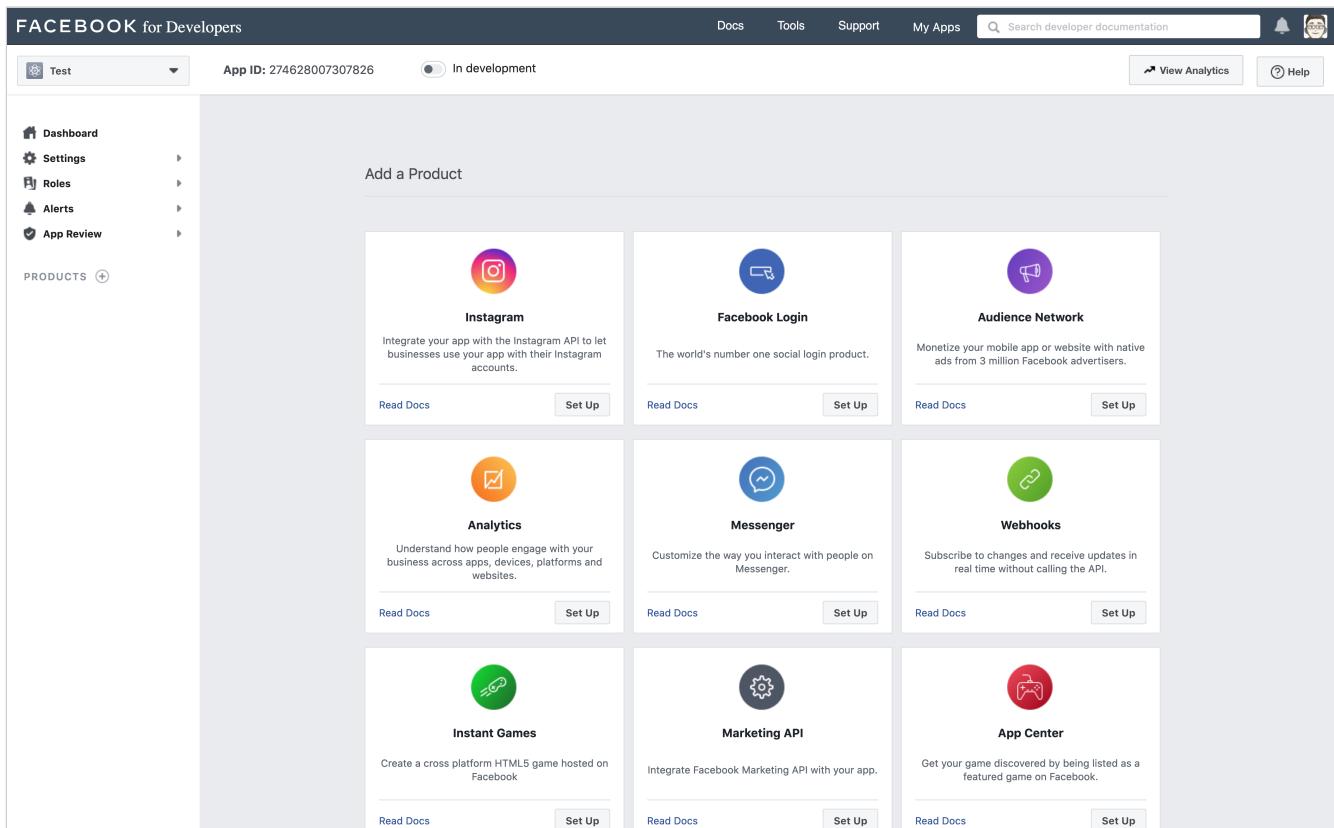
h. Click [Website].

按 [网址]。

i. Enter a URL for your site.

输入网站的 URL。

Add a product 添加产品



j. Select **Dashboard** from the menu.

从菜单中选择 Dashboard。

k. Click **Set Up** in the Instagram box.

单击 Instagram 框中的“设置”。

l. Select **Instagram - Basic Display** from the menu.

从菜单中选择 Instagram-Basic Display。

m. Click **Create New App**.

单击“创建新应用程序”。

Create a new Instagram app ID 创建一个新的 Instagram 应用程序 ID

The screenshot shows the Facebook for Developers dashboard with the Instagram Basic Display API selected. A modal window is open for creating a new Instagram app ID. The 'Display Name' field is set to 'Test'. The 'Instagram Platform Policies' section contains a note about retrieving a person's Instagram username and account type, mentioning the 'instagram_graph_user_profile' permission. A 'Create App' button is visible at the bottom of the modal.

n. Enter a value into **Display Name**.

在“显示名称”中输入一个值。

Set up the app 设置应用程序

The screenshot shows the Instagram Basic Display API configuration page. It displays the Instagram App ID (270474830705500) and Instagram App Secret (redacted). The configuration section includes fields for Instagram Display Name (set to 'Test'), Client OAuth Settings (Valid OAuth Redirect URIs), Deauthorize (Deauthorize Callback URL), Data Deletion Requests (Data Deletion Request URL), and User Token Generator. A 'Save Changes' button is located at the bottom right of the configuration area.

o. Paste the **Redirect URL** from Keycloak into the **Valid OAuth Redirect URIs** field.

将来自 Keycloak 的重定向 URL 粘贴到有效的 OAuth 重定向 URI 字段中。

p. Paste the **Redirect URL** from Keycloak into the **Deauthorize Callback URL** field.

将来自 Keycloak 的重定向 URL 粘贴到 Deauthorize Callback URL 字段中。

q. Paste the **Redirect URL** from Keycloak into the **Data Deletion Request URL** field.

将来自 Keycloak 的重定向 URL 粘贴到“数据删除请求 URL”字段中。

r. Click **Show** in the **Instagram App Secret** field.

单击 Instagram App Secret 字段中的 Show。

s. Note the **Instagram App ID** and the **Instagram App Secret**.

注意 Instagram 应用程序 ID 和 Instagram 应用程序的秘密。

t. Click **App Review - Requests**.

点击应用程序审查-请求。

u. Follow the instructions on the screen.

按照屏幕上的说明操作。

5. In Keycloak, paste the value of the **Instagram App ID** into the **Client ID** field.

在 Keycloak, 将 Instagram 应用程序 ID 的值粘贴到客户端 ID 字段中。

6. In Keycloak, paste the value of the **Instagram App Secret** into the **Client Secret** field.

在 Keycloak, 将 Instagram 应用程序秘密的值粘贴到客户端秘密字段中。

7. Click **Add**.

单击“添加”。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/identity-broker/oidc.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/identity-broker/oidc.adoc)

Keycloak brokers identity providers based on the OpenID Connect protocol. These identity providers (IDPs) must support the Authorization Code Flow defined in the specification to authenticate users and authorize access.

基于 OpenID 连接协议的密钥斗篷代理身份提供程序。这些标识提供程序(IDP)必须支持规范中定义的授权代码流，以验证用户和授权访问。

Procedure 程序

1. Click **Identity Providers** in the menu.

单击菜单中的“标识提供程序”。

2. From the **Add provider** list, select **OpenID Connect v1.0**.

从 Add Provider 列表中，选择 OpenID Connect v1.0。

Add identity provider 添加标识提供程序

Master

Manage

Clients

Client scopes

Realm roles

Users

Groups

Sessions

Events

Configure

Realm settings

Authentication

Identity providers

User federation

Identity providers

Add OpenID Connect provider

Redirect URI ⓘ http://localhost:8180/realm/master/broker/oidc/endpoint

Alias * ⓘ oidc

Display name ⓘ

Display order ⓘ

OpenID Connect settings

Use discovery endpoint ⓘ On

Discovery endpoint * ⓘ https://hostname/auth/realms/master/.well-known/openid-configuration

Show metadata

Client authentication ⓘ Client secret sent as post

Client ID * ⓘ

Client Secret * ⓘ

3. Enter your initial configuration options. See General IDP Configuration for more information about configuration options.

输入初始配置选项。有关配置选项的详细信息，请参阅常规 IDP 配置。

Table 2. OpenID connect config 表2. OpenID 连接配置

Configuration 配置	Description 描述
Authorization URL 授权网址	The authorization URL endpoint the OIDC protocol requires. OIDC 协议需要的授权 URL 端点。
Token URL 令牌网址	The token URL endpoint the OIDC protocol requires. OIDC 协议需要的令牌 URL 端点。
Logout URL 注销网址	The logout URL endpoint in the OIDC protocol. This value is optional. OIDC 协议中的注销 URL 端点。此值是可选的。

Configuration 配置	Description 描述
Backchannel Logout 反向通道注销	<p>A background, out-of-band, REST request to the IDP to log out the user. Some IDPs perform logout through browser redirects only, as they may identify sessions using a browser cookie.</p> <p>向 IDP 发送后台、带外、REST 请求以注销用户。有些 IDP 仅通过浏览器重定向执行注销，因为它们可能使用浏览器 cookie 标识会话。</p>
User Info URL 用户资料网址	<p>An endpoint the OIDC protocol defines. This endpoint points to user profile information.</p> <p>OIDC 协议定义的端点。该端点指向用户配置文件信息。</p>
Client Authentication 客户端认证	<p>Defines the Client Authentication method Keycloak uses with the Authorization Code Flow. In the case of JWT signed with a private key, Keycloak uses the realm private key. In the other cases, define a client secret. See the Client Authentication specifications (https://openid.net/specs/openid-connect-core-1_0.html#ClientAuthentication) for more information.</p> <p>在“授权代码流”中定义 Keycloak 使用的客户端身份验证方法。对于使用私钥签名的 JWT，Keycloak 使用领域私钥。在其他情况下，定义客户机机密。有关更多信息，请参见客户端身份验证规范。</p>
Client ID 客户名	<p>A realm acting as an OIDC client to the external IDP. The realm must have an OIDC client ID if you use the Authorization Code Flow to interact with the external IDP.</p> <p>充当外部 IDP 的 OIDC 客户端的领域。如果使用授权代码流与外部 IDP 交互，则领域必须具有 OIDC 客户端 ID。</p>

Configuration 配置	Description 描述
Client Secret 客户端机密	Client secret from an external vault. This secret is necessary if you are using the Authorization Code Flow. 来自外部保险库的客户端机密。如果您正在使用授权代码流，则此机密是必需的。
Client Assertion Signature Algorithm 客户端断言签名算法	Signature algorithm to create JWT assertion as client authentication. In the case of JWT signed with private key or Client secret as jwt, it is required. If no algorithm is specified, the following algorithm is adapted. RS256 is adapted in the case of JWT signed with private key. HS256 is adapted in the case of Client secret as jwt. 创建 JWT 断言作为客户端身份验证的签名算法。对于使用私钥或 Client secret 作为 JWT 签名的 JWT，它是必需的。如果未指定算法，则采用下列算法。RS256适用于使用私钥签名的JWT。HS256在客户端保密的情况下被改编为 jwt。
Issuer 发行人	Keycloak validates issuer claims, in responses from the IDP, against this value. 在来自 IDP 的响应中，Keycloak 根据此值验证发行者声明。
Default Scopes 默认作用域	A list of OIDC scopes Keycloak sends with the authentication request. The default value is openid. A space separates each scope. 随身份验证请求一起发送的 OIDC 范围的列表。默认值为 openid。一个空格将每个作用域分隔开。
Prompt 及时	The prompt parameter in the OIDC specification. Through this parameter, you can force re-authentication and other options. See the specification for more details. OIDC 规范中的提示参数。通过此参数，可以强制重新身份验证和其他选项。有关更多细节，请参见规范。

Configuration 配置	Description 描述
<p>Accepts prompt=none forward from client 从客户端接受命令行 = no 转发</p>	<p>Specifies if the IDP accepts forwarded authentication requests containing the <code>prompt=none</code> query parameter. If a realm receives an auth request with <code>prompt=none</code>, the realm checks if the user is currently authenticated and returns a <code>login_required</code> error if the user has not logged in. When Keycloak determines a default IDP for the auth request (using the <code>kc_idp_hint</code> query parameter or having a default IDP for the realm), you can forward the auth request with <code>prompt=none</code> to the default IDP. The default IDP checks the authentication of the user there. Because not all IDPs support requests with <code>prompt=none</code>, Keycloak uses this switch to indicate that the default IDP supports the parameter before redirecting the authentication request.</p> <p>指定 IDP 是否接受转发的身份验证请求，这些请求包含命令提示 = 无查询参数。如果一个领域接收到一个 auth 请求，并且提示 = none，那么该领域将检查用户当前是否已经通过身份验证，并且如果用户尚未登录，则返回一个 <code>login_need</code> 错误。当 Keycloak 为认证请求确定一个默认的 IDP (使用 <code>kc_idp_hint</code> 查询参数或者在领域中有一个默认的 IDP)时，您可以将认证请求转发到默认的 IDP，提示符 = none。默认的 IDP 检查那里的用户的身份验证。因为并非所有的 IDP 都支持执行命令行提示符 = 无的请求，所以 Keycloak 在重定向身份验证请求之前使用这个开关来指示默认的 IDP 支持该参数。</p> <p>If the user is unauthenticated in the IDP, the client still receives a <code>login_required</code> error. If the user is authentic in the IDP, the client can still receive an <code>interaction_required</code> error if Keycloak must display authentication pages that require user interaction. This authentication includes required actions (for example, password change), consent screens,</p>

Configuration 配置	Description 描述
	<p>and screens set to display by the <code>first broker login</code> flow or <code>post broker login</code> flow.</p> <p>如果用户在 IDP 中未经身份验证，客户机仍然会收到 <code>login_need</code> 错误。如果用户在 IDP 中是真实的，那么如果 Keycover 必须显示需要用户交互的身份验证页面，那么客户机仍然可以接收交互所需的错误。这种身份验证包括所需的操作(例如，更改密码)、同意屏幕以及设置为由第一个代理登录流或后代理登录流显示的屏幕。</p>
Validate Signatures 验证签名	<p>Specifies if Keycloak verifies signatures on the external ID Token signed by this IDP. If ON, Keycloak must know the public key of the external OIDC IDP. For performance purposes, Keycloak caches the public key of the external OIDC identity provider.</p> <p>指定 Key 是否在此 IDP 所签署的外部 ID 令牌上验證簽章。如果为 ON，则密钥斗篷必须知道外部 OIDC IDP 的公钥。出于性能目的，Key斗篷缓存外部 OIDC 标识提供程序的公钥。</p>
Use JWKS URL 使用 JWKS URL	<p>This switch is applicable if <code>Validate Signatures</code> is ON. If <code>Use JWKS URL</code> is ON, Keycloak downloads the IDP's public keys from the JWKS URL. New keys download when the identity provider generates a new keypair. If OFF, Keycloak uses the public key (or certificate) from its database, so when the IDP keypair changes, import the new key to the Keycloak database as well.</p> <p>如果验证签名为 ON，则此开关适用。如果 Use JWKS URL 是 ON，Keycon 将从 JWKS URL 下载 IDP 的公钥。标识提供程序生成新密钥对时下载新密钥。如果是 OFF，Key枕使用来自其数据库的公钥(或证书)，因此当 IDP 密钥对更改时，也将新密钥导入 Key枕数据库。</p>

Configuration 配置	Description 描述
<p>JWKS URL JWKS 网址</p>	<p>The URL pointing to the location of the IDP JWK keys. For more information, see the JWK specification (https://self-issued.info/docs/draft-ietf-jose-json-web-key.html)</p> <p>If you use an external Keycloak as an IDP, you can use a URL such as <code>http://broker-keycloak:8180/realms/test/protocol/openid-connect/certs</code> if your brokered Keycloak is running on <code>http://broker-keycloak:8180</code> and its realm is <code>test</code>.</p> <p>指向 IDP JWK 密钥位置的 URL。有关更多信息，请参见 JWK 规范。如果你使用一个外部的钥匙斗篷作为一个 IDP，你可以使用一个像 <code>http://broker-Keycloak:8180/realms/test/protocol/openid-connect/certs</code> 这样的 URL，如果你的代理的钥匙斗篷在 <code>http://broker-Keycloak:8180</code> 上运行并且它的领域是测试的。</p>
<p>Validating Public Key 验证公钥</p>	<p>The public key in PEM format that Keycloak uses to verify external IDP signatures. This key applies if <code>Use JWKS URL</code> is OFF.</p> <p>Keycover 用于验证外部 IDP 签名的 PEM 格式的公钥。如果 <code>UseJWKS URL</code> 为 OFF，则应用此键。</p>

Configuration 配置	Description 描述
Validating Public Key Id 验证公钥 ID	<p>This setting applies if Use JWKS URL is OFF. This setting specifies the ID of the public key in PEM format. Because there is no standard way for computing key ID from the key, external identity providers can use different algorithms from what Keycloak uses. If this field's value is not specified, Keycloak uses the validating public key for all requests, regardless of the key ID sent by the external IDP. When ON, this field's value is the key ID used by Keycloak for validating signatures from providers and must match the key ID specified by the IDP.</p> <p>如果 UseJWKS URL 为 OFF，则应用此设置。此设置以 PEM 格式指定公钥的 ID。因为没有从密钥计算密钥 ID 的标准方法，所以外部身份提供程序可以使用不同于钥匙斗篷所使用的算法。如果未指定此字段的值，则 Keycloak 将对所有请求使用验证公钥，而不管外部 IDP 发送的密钥 ID 如何。当 ON 时，此字段的值是 Keycloak 用于验证提供程序签名的密钥 ID，并且必须与 IDP 指定的密钥 ID 匹配。</p>

You can import all this configuration data by providing a URL or file that points to OpenID Provider Metadata. If you connect to a Keycloak external IDP, you can import the IDP settings from `<root>/realms/{realm-name}/.well-known/openid-configuration`. This link is a JSON document describing metadata about the IDP.

您可以通过提供指向 OpenID 提供程序元数据的 URL 或文件来导入所有这些配置数据。如果您连接到一个 Keycloak 外部 IDP，您可以从 `<root>/fields/{ domain-name }/` 导入 IDP 设置。著名的/openid 配置。这个链接是一个描述关于 IDP 的元数据的 JSON 文档。

[Edit this section 编辑这部分](https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/identity-broker/saml.adoc)
 (https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/identity-broker/saml.adoc)

[Report an issue 报告一个问题](https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/identity-broker/saml.adoc)
 (https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/identity-broker/saml.adoc)

Keycloak 可以通过 SAML v2.0 协议代理身份提供程序。

Keycloak 可以基于 SAML v2.0 协议代理身份提供程序。

Procedure 程序

1. Click Identity Providers in the menu.

单击菜单中的“标识提供程序”。

2. From the Add provider list, select SAML v2.0 .

从 Add Provider 列表中，选择 SAML v2.0。

Add identity provider 添加标识提供程序

The screenshot shows the 'Add SAML provider' configuration page in the Keycloak interface. The left sidebar is titled 'Master' and contains the following navigation items: Manage, Clients, Client scopes, Realm roles, Users, Groups, Sessions, Events, Configure, Realm settings, Authentication, Identity providers (which is selected and highlighted in blue), and User federation. The main content area has a header 'Identity providers > Add SAML provider'. It includes fields for 'Redirect URI' (http://localhost:8180/realm/master/broker/saml/endpoint), 'Alias' (saml), 'Display name' (empty), 'Display order' (empty), and an 'Endpoints' section with a link to 'SAML 2.0 Service Provider Metadata'. Below this is a 'SAML settings' section with a 'Service provider entity ID' (http://localhost:8180/realm/master) and a 'Use entity descriptor' toggle switch set to 'On'. There is also a field for 'SAML entity descriptor' (empty) and a link to 'Show metadata'. At the bottom are 'Add' and 'Cancel' buttons.

3. Enter your initial configuration options. See General IDP Configuration for more information about configuration options.

输入初始配置选项。有关配置选项的详细信息，请参阅常规 IDP 配置。

Table 3. SAML Config 表3. SAML 配置

Configuration 配置	Description 描述
Service Provider Entity ID 服务提供商实体 ID	The SAML Entity ID that the remote Identity Provider uses to identify requests from this Service Provider. By default, this setting is set to the realms base URL <root>/realms/{realm-name}. 远程标识提供程序用于标识来自此服务提供程序的请求的 SAML 实体 ID。默认情况下，此设置设置为领域基 URL <root>/fields/{ domain-name }。

Configuration 配置	Description 描述
Single Sign-On Service URL 单点登录服务 URL	<p>The SAML endpoint that starts the authentication process. If your SAML IDP publishes an IDP entity descriptor, the value of this field is specified there.</p> <p>启动身份验证过程的 SAML 端点。如果您的 SAMLIDP 发布了一个 IDP 实体描述符，则在那里指定该字段的值。</p>
Single Logout Service URL 单一注销服务 URL	<p>The SAML logout endpoint. If your SAML IDP publishes an IDP entity descriptor, the value of this field is specified there.</p> <p>SAML 注销端点。如果您的 SAMLIDP 发布了一个 IDP 实体描述符，则在那里指定该字段的值。</p>
Backchannel Logout 反向通道注销	<p>Toggle this switch to ON if your SAML IDP supports back channel logout.</p> <p>如果您的 SAML IDP 支持后台通道注销，请将此开关切换到 ON。</p>
NameID Policy Format NameID 策略格式	<p>The URI reference corresponding to a name identifier format. By default, Keycloak sets it to <code>urn:oasis:names:tc:SAML:2.0:nameid-format:persistent</code>.</p> <p>对于名称标识符格式的 URI 引用。默认情况下，Keycloak 将其设置为 <code>urn: oasis: names.tc: SAML: 2.0: nameid-format: always</code>。</p>

Configuration 配置	Description 描述
Principal Type 主要类型	<p>Specifies which part of the SAML assertion will be used to identify and track external user identities. Can be either Subject NameID or SAML attribute (either by name or by friendly name). Subject NameID value can not be set together with 'urn:oasis:names:tc:SAML:2.0:nameid-format:transient' NameID Policy Format value.</p> <p>指定 SAML 断言的哪一部分将用于标识和跟踪外部用户标识。可以是 Subject NameID 或 SAML 属性(通过名称或友好名称)。不能将主题 NameID 值与“urn: oasis: Names: tc: SAML: 2.0: NamelD-Format: 瞬态”NameID Policy Format 值一起设置。</p>
Principal Attribute 主体属性	<p>If a Principal type is non-blank, this field specifies the name ("Attribute [Name]") or the friendly name ("Attribute [Friendly Name]") of the identifying attribute.</p> <p>如果主体类型是非空的，则此字段指定标识属性的名称(“ Attribute [Name]”)或友好名称(“ Attribute [Friendly Name] ”)。</p>
Allow create 允许创建	<p>Allow the external identity provider to create a new identifier to represent the principal.</p> <p>允许外部标识提供程序创建一个新标识符来表示主体。</p>
HTTP-POST Binding Response HTTP-POST 绑定响应	<p>Controls the SAML binding in response to any SAML requests sent by an external IDP. When OFF, Keycloak uses Redirect Binding.</p> <p>控制 SAML 绑定以响应外部 IDP 发送的任何 SAML 请求。当关闭时，密钥斗篷使用重定向绑定。</p>

Configuration 配置	Description 描述
HTTP-POST Binding for AuthnRequest AuthnRequest 的 HTTP-POST 绑定	Controls the SAML binding when requesting authentication from an external IDP. When OFF, Keycloak uses Redirect Binding. 当从外部 IDP 请求身份验证时，控制 SAML 绑定。
Want AuthnRequests Signed 希望认证请求被签署	When ON, Keycloak uses the realm's keypair to sign requests sent to the external SAML IDP. 当 ON 时，Keycloak 使用领域的密钥对对发送到外部 SAML IDP 的请求进行签名。
Signature Algorithm 签名算法	If Want AuthnRequests Signed is ON, the signature algorithm to use. 如果“要签名的 AuthnRequestsSigned”为 ON，则使用签名算法。
SAML Signature Key Name 签名密钥名称	Signed SAML documents sent using POST binding contain the identification of signing key in <code>KeyName</code> element, which, by default, contains the Keycloak key ID. External SAML IDPs can expect a different key name. This switch controls whether <code>KeyName</code> contains: <ul style="list-style-type: none"> * <code>KEY_ID</code> - Key ID. * <code>CERT SUBJECT</code> - the subject from the certificate corresponding to the realm key. Microsoft Active Directory Federation Services expect <code>CERT SUBJECT</code> . <ul style="list-style-type: none"> * <code>NONE</code> - Keycloak omits the key name hint from the SAML message. 使用 POST 绑定发送的签名 SAML 文档包含 <code>KeyName</code> 元素中签名密钥的标识，默认情况下， <code>KeyName</code> 元素包含钥匙密钥 ID。外部 SAML IDP 可能需要不同的密钥名。此开关控制 <code>KeyName</code> 是否包含：* KEY_ID-KEY ID。CERT_SUBJECT-证书中与领域密钥对应的主题。微软活动目录联合服务期待 CERT_SUBJECT。* NONE-Key枕省略了 SAML 消息中的键名提示。

Configuration 配置	Description 描述
Force Authentication 武力认证	<p>The user must enter their credentials at the external IDP even when the user is already logged in.</p> <p>即使用户已经登录，用户也必须在外部 IDP 上输入其凭据。</p>
Validate Signature 确认签署	<p>When ON, the realm expects SAML requests and responses from the external IDP to be digitally signed.</p> <p>当 ON 时，领域期望来自外部 IDP 的 SAML 请求和响应进行数字签名。</p>
Validating X509 Certificate 验证 X509证书	<p>The public certificate Keycloak uses to validate the signatures of SAML requests and responses from the external IDP.</p> <p>公共证书 Keycloak 用于验证来自外部 IDP 的 SAML 请求和响应的签名。</p>
Sign Service Provider Metadata 签名服务提供者元数据	<p>When ON, Keycloak uses the realm's key pair to sign the SAML Service Provider Metadata descriptor.</p> <p>当 ON 时，Keycloak 使用领域的密钥对对 SAML 服务提供者元数据描述符进行签名。</p>
Pass subject 通过测试	<p>Controls if Keycloak forwards a login_hint query parameter to the IDP. Keycloak adds this field's value to the login_hint parameter in the AuthnRequest's Subject so destination providers can pre-fill their login form.</p> <p>控制 Keycloak 是否向 IDP 转发 login_tip 查询参数。Keycloak 将此字段的值添加到 AuthnRequest 的 Subject 中的 login_tip 参数，以便目标提供程序可以预先填写其登录表单。</p>

Configuration 配置	Description 描述
Attribute Consuming Service Index 属性消费服务索引	Identifies the attribute set to request to the remote IDP. Keycloak automatically adds the attributes mapped in the identity provider configuration to the autogenerated SP metadata document. 标识设置为向远程 IDP 请求的属性。Keycloak 自动将标识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。
Attribute Consuming Service Name 属性消费服务名称	A descriptive name for the set of attributes that are advertised in the autogenerated SP metadata document. 在自动生成的 SP 元数据文档中公布的一组属性的描述性名称。

You can import all configuration data by providing a URL or a file pointing to the SAML IDP entity descriptor of the external IDP. If you are connecting to a Keycloak external IDP, you can import the IDP settings from the URL `<root>/realms/{realm-name}/protocol/saml/descriptor`. This link is an XML document describing metadata about the IDP. You can also import all this configuration data by providing a URL or XML file pointing to the external SAML IDP's entity descriptor to connect to.

通过提供指向外部 IDP 的 SAML IDP 实体描述符的 URL 或文件，可以导入所有配置数据。如果您正在连接到一个 Keycloak 外部 IDP，那么您可以从 URL `<root>/fields/{domain-name}/protocol/saml/description` 中导入 IDP 设置。此链接是一个描述关于 IDP 的元数据的 XML 文档。您还可以通过提供指向要连接到的外部 SAML IDP 的实体描述符的 URL 或 XML 文件来导入所有这些配置数据。

Requesting specific AuthnContexts 请求特定的 AuthnContext

Identity Providers facilitate clients specifying constraints on the authentication method verifying the user identity. For example, asking for MFA, Kerberos authentication, or security requirements. These constraints use particular AuthnContext criteria. A client can ask for one or more criteria and specify how the Identity Provider must match the requested AuthnContext, exactly, or by satisfying other equivalents.

标识提供程序便于客户端指定验证用户标识的身份验证方法的约束。例如，请求 MFA、Kerberos 身份验证或安全需求。这些约束使用特定的 AuthnContext 条件。客户端可以请求一个或多个条件，并指定标识提供程序必须如何精确地匹配所请求的 AuthnContext，或者通过满足其他等效条件。

You can list the criteria your Service Provider requires by adding ClassRefs or DeclRefs in the Requested AuthnContext Constraints section. Usually, you need to provide either ClassRefs or DeclRefs, so check with your Identity Provider documentation which values are supported. If no ClassRefs or DeclRefs are

present, the Identity Provider does not enforce additional constraints.

通过在“请求的 AuthnContext 约束”部分中添加 ClassRefs 或 DeclRefs，可以列出服务提供程序所需的条件。通常，您需要提供 ClassRefs 或 DeclRefs，因此请查看您的 Identity Provider 文档，看看支持哪些值。如果不存在 ClassReff 或 DeclReff，则标识提供程序不强制执行其他约束。

Table 4. Requested AuthnContext Constraints 表4. 请求的 AuthnContext 约束

Configuration 配置	Description 描述
Comparison 比较	The method the Identity Provider uses to evaluate the context requirements. The available values are <code>Exact</code> , <code>Minimum</code> , <code>Maximum</code> , or <code>Better</code> . The default value is <code>Exact</code> . Identity Provider 用于评估上下文需求的方法。可用的值是精确值、最小值、最大值或更好值。默认值为 Exact。
AuthnContext ClassRefs	The AuthnContext ClassRefs describing the required criteria. 描述所需条件的 AuthnContext ClassReff。
AuthnContext DeclRefs	The AuthnContext DeclRefs describing the required criteria. 描述所需条件的 AuthnContext DeclRefs。

SP Descriptor SP 描述符

When you access the provider's SAML SP metadata, look for the `Endpoints` item in the identity provider configuration settings. It contains a `SAML 2.0 Service Provider Metadata` link which generates the SAML entity descriptor for the Service Provider. You can download the descriptor or copy its URL and then import it into the remote Identity Provider.

访问提供程序的 SAMLSP 元数据时，请在标识提供程序配置设置中查找 Endpoints 项。它包含一个 SAML 2.0 Service Provider Metadata 链接，该链接为 Service Provider 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标识提供程序中。

This metadata is also available publicly by going to the following URL:

该元数据也可通过访问以下网址公开获得：

```
http[s]://{{host:port}}/realms/{{realm-name}}/broker/{{broker-alias}}/endpointdescriptor
```

Ensure you save any configuration changes before accessing the descriptor.

确保在访问描述符之前保存任何配置更改。

Send subject in SAML requests 在 SAML 请求中发送主题

By default, a social button pointing to a SAML Identity Provider redirects the user to the following login URL:

默认情况下，指向 SAML 标识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}}/realms/${realm-name}/broker/{broker-alias}/login
```

Adding a query parameter named `login_hint` to this URL adds the parameter's value to SAML request as a Subject attribute. If this query parameter is empty, Keycloak does not add a subject to the request.

向此 URL 添加名为 `login_tip` 的查询参数将该参数的值作为 Subject 属性添加到 SAML 请求中。如果此查询参数为空，则 Keycloak 不会向请求添加主题。

Enable the "Pass subject" option to send the subject in SAML requests.

启用“ Pass subject”选项在 SAML 请求中发送主题。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/identity-broker/suggested.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/identity-broker/suggested.adoc)

OIDC applications can bypass the Keycloak login page by hinting at the identity provider they want to use. You can enable this by setting the `kc_idp_hint` query parameter in the Authorization Code Flow authorization endpoint.

OIDC 应用程序可以通过提示要使用的标识提供程序来绕过 Keycloak 登录页面。可以通过在 AuthorizationCodeFlow 授权端点中设置 `kc_idp_tip` 查询参数来启用此选项。

With Keycloak OIDC client adapters, you can specify this query parameter when you access a secured resource in the application.

使用 Keycloak OIDC 客户端适配器，可以在访问应用程序中的安全资源时指定此查询参数。

For example:

例如:

```
GET /myapplication.com?kc_idp_hint=facebook HTTP/1.1  
Host: localhost:8080
```

BASH

In this case, your realm must have an identity provider with a `facebook` alias. If this provider does not exist, the login form is displayed.

在这种情况下，您的领域必须有一个具有 `facebook` 别名的标识提供程序。如果此提供程序不存在，则显示登录窗体。

If you are using the `keycloak.js` adapter, you can also achieve the same behavior as follows:

如果使用 keycloak.js 适配器，还可以实现以下相同的行为：

```
const keycloak = new Keycloak('keycloak.json');  
  
keycloak.createLoginUrl({  
    idpHint: 'facebook'  
});
```

JAVASCRIPT

With the `kc_idp_hint` query parameter, the client can override the default identity provider if you configure one for the `Identity Provider Redirector` authenticator. The client can disable the automatic redirecting by setting the `kc_idp_hint` query parameter to an empty value.

如果为 Identity Provider ReDirector 身份验证器配置了默认的标识提供程序，那么使用 `kc_idp_tip` 查询参数，客户机可以覆盖默认的标识提供程序。客户端可以通过将 `kc_idp_tip` 查询参数设置为空值来禁用自动重定向。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/identity-broker/mappers.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/identity-broker/mappers.adoc)

You can import the SAML and OpenID Connect metadata, provided by the external IDP you are authenticating with, into the realm. After importing, you can extract user profile metadata and other information, so you can make it available to your applications.

您可以将正在进行身份验证的外部 IDP 提供的 SAML 和 OpenIDConnect 元数据导入到领域中。导入之后，可以提取用户配置文件元数据和其他信息，这样就可以使应用程序可以使用它。

Each user logging into your realm using an external identity provider has an entry in the local Keycloak database, based on the metadata from the SAML or OIDC assertions and claims.

每个使用外部身份提供程序登录到您的领域的用户，都会在本地 Keycloak 数据库中基于来自 SAML 或 OIDC 断言和声明的元数据有一个条目。

Procedure 程序

1. Click Identity Providers in the menu.

单击菜单中的“标识提供程序”。

2. Select one of the identity providers in the list.

在列表中选择一个标识提供程序。

3. Click the Mappers tab.

单击 Mapper 选项卡。

Identity provider mappers 身份提供者映射器

The screenshot shows the 'Identity provider mappers' page. On the left is a sidebar with options like Master, Manage, Clients, Client scopes, Realm roles, Users, Groups, Sessions, and Events. The 'Identity providers' section is highlighted. In the main area, it says 'Saml'. Below that are two tabs: 'Settings' and 'Mappers', with 'Mappers' being the active one. There is a large blue plus icon with a white '+' sign. Below it, the text 'No Mappers' is displayed, followed by the message 'There are currently no mappers for this identity provider.' At the bottom right is a blue 'Add mapper' button.

4. Click Add mapper.

单击 Add mapper。

Identity provider mapper 身份提供程序映射器

The screenshot shows the 'Add Identity Provider Mapper' page. The sidebar on the left has 'Identity providers' selected. The main form has the title 'Add Identity Provider Mapper'. It includes fields for 'Name' (with a required asterisk), 'Sync mode override' (set to 'Inherit'), 'Mapper type' (set to 'Advanced Attribute to Role'), 'Attributes' (with a 'Type a key' input and a 'Add an attribute' button), 'Regex Attribute' (switch set to 'Off'), 'Values' (with a dropdown showing 'master'), and a 'Role' dropdown. At the bottom are 'Save' and 'Cancel' buttons.

5. Select a value for Sync Mode Override. The mapper updates user information when users log in repeatedly according to this setting.

为同步模式覆盖选择一个值。当用户根据此设置重复登录时，映射器会更新用户信息。

a. Select **legacy** to use the behavior of the previous Keycloak version.

选择“**遗留**”以使用以前的 Keycloak 版本的行为。

b. Select **import** to import data from when the user was first created in Keycloak during the first login to Keycloak with a particular identity provider.

选择 **import**，以便在第一次使用特定的身份提供程序登录到“钥匙斗篷”时，从用户首次在 Keycloak 创建时导入数据。

c. Select **force** to update user data at each user login.

选择 **强制** 在每次用户登录时更新用户数据。

d. Select **inherit** to use the sync mode configured in the identity provider. All other options will override this sync mode.

选择“**继承**”以使用在标识提供程序中配置的同步模式。所有其他选项将重写此同步模式。

6. Select a mapper from the **Mapper Type** list. Hover over the **Mapper Type** for a description of the mapper and configuration to enter for the mapper.

从 Mapper Type 列表中选择映射器。将鼠标悬停在 Mapper Type 上，查看要为映射器输入的映射器和配置的说明。

7. Click **Save**.

单击“**保存**”。

For JSON-based claims, you can use dot notation for nesting and square brackets to access array fields by index. For example, `contact.address[0].country`.

对于基于 JSON 的声明，可以使用点符号表示嵌套，使用方括号按索引访问数组字段。例如，`contact.address [0]`。国家。

To investigate the structure of user profile JSON data provided by social providers, you can enable the DEBUG level logger `org.keycloak.social.user_profile_dump` when starting the server.

要研究社交服务提供商提供的用户配置文件 JSON 数据的结构，可以在启动服务器时启用 DEBUG 级别的日志器 `org.keycloak.social.user_profile_dump`。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/identity-broker/session-data.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/identity-broker/session-data.adoc)

After a user login from an external IDP, Keycloak stores user session note data that you can access. This data can be propagated to the client requesting log in using the token or SAML assertion passed back to the client using an appropriate client mapper.

在用户从外部 IDP 登录之后，Key枕存储您可以访问的用户会话说明数据。使用令牌或 SAML 断言，使用适当的客户端映射器传递回客户端，可以将此数据传播到客户端请求日志。

identity_provider 身份提供程序

The IDP alias of the broker used to perform the login.

用于执行登录的代理的 IDP 别名。

identity_provider_identity Identity _ Provider _ Identity

The IDP username of the currently authenticated user. Often, but not always, the same as the Keycloak username. For example, Keycloak can link a user john` to a Facebook user `john123@gmail.com`. In that case, the value of the user session note is `john123@gmail.com`.

当前经过身份验证的用户的 IDP 用户名。通常，但不总是，与 Keycloak 用户名相同。例如，Key枕可以将用户 john`链接到 Facebook 用户 john123@gmail.com。在这种情况下，用户会话注释的值是 john123@gmail.com。

You can use a Protocol Mapper of type `User Session Note` to propagate this information to your clients.

您可以使用 `UserSessionNote` 类型的 Protocol Mapper 将此信息传播到客户端。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/identity-broker/first-login-flow.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/identity-broker/first-login-flow.adoc)

When users log in through identity brokering, Keycloak imports and links aspects of the user within the realm's local database. When Keycloak successfully authenticates users through an external identity provider, two situations can exist:

当用户通过身份代理登录时，Key枕导入并链接领域的本地数据库中用户的各个方面。当 Key枕通过外部身份提供程序成功地对用户进行身份验证时，可能存在两种情况：

- Keycloak has already imported and linked a user account with the authenticated identity provider account. In this case, Keycloak authenticates as the existing user and redirects back to the application.

“钥匙斗篷”已导入用户帐户并将其与经过身份验证的身份提供程序帐户链接。在这种情况下，Key枕验证为现有用户并重定向回应用程序。

- No account exists for this user in Keycloak. Usually, you register and import a new account into the Keycloak database, but there may be an existing Keycloak account with the same email address. Automatically linking the existing local account to the external identity provider is a potential security hole. You cannot always trust the information you get from the external identity provider.

该用户在 Keycloak 没有帐户。通常，您可以注册一个新帐户并将其导入到钥匙斗篷数据库中，但是可能存在一个具有相同电子邮件地址的现有钥匙斗篷帐户。将现有本地帐户自动链接到外部标识提供程序是一个潜在的安全漏洞。您不能总是信任从外部标识提供程序获得的信息。

Different organizations have different requirements when dealing with some of these situations. With Keycloak, you can use the `First Login Flow` option in the IDP settings to choose a workflow for a user logging in from an external IDP for the first time. By default, the `First Login Flow` option points to the `first broker login` flow, but you can use your flow or different flows for different identity providers.

不同的组织在处理这些情况时有不同的要求。使用 Keycon，您可以使用 IDP 设置中的 First Login Flow 选项为首次从外部 IDP 登录的用户选择工作流。默认情况下，First Login Flow 选项指向第一个代理登录流，但是您可以为不同的标识提供程序使用您的流或不同的流。

The flow is in the Admin Console under the **Authentication** tab. When you choose the `First Broker Login` flow, you see the authenticators used by default. You can re-configure the existing flow. For example, you can disable some authenticators, mark some of them as `required`, or configure some authenticators.

该流在“身份验证”选项卡下的“管理控制台”中。当您选择 First Broker Login 流时，您将看到默认使用的身份验证器。可以重新配置现有流。例如，您可以禁用某些身份验证器，将其中一些标记为必需的，或配置某些身份验证器。

You can also create a new authentication flow, write your own Authenticator implementations, and use it in your flow. See [Server Developer Guide](https://www.keycloak.org/docs/latest/server_development/) (https://www.keycloak.org/docs/latest/server_development/) for more information.

您还可以创建一个新的身份验证流，编写自己的 Authenticator 实现，并在流中使用它。有关更多信息，请参见服务器开发人员指南。

Default first login flow authenticators 默认的第一个登录流身份验证器

Review Profile 检讨概况

- This authenticator displays the profile information page, so the users can review their profile that Keycloak retrieves from an identity provider.

此身份验证程序显示配置文件信息页，这样用户就可以查看从标识提供程序检索的 Keycloak 的配置文件。

- You can set the `Update Profile On First Login` option in the **Actions** menu.

可以在“操作”菜单中设置“首次登录时更新配置文件”选项。

- When **ON**, users are presented with the profile page requesting additional information to federate the user's identities.

当 ON 时，将向用户显示要求附加信息以联合用户身份的配置文件页面。

- When **missing**, users are presented with the profile page if the identity provider does not provide mandatory information, such as email, first name, or last name.

如果缺少，如果身份提供程序不提供强制性信息(如电子邮件、名字或姓氏)，则向用户显示配置文件页。

- When **OFF**, the profile page does not display unless the user clicks in a later phase on the `Review profile info` link in the page displayed by the `Confirm Link Existing Account` authenticator.

当 OFF 时，配置文件页面不会显示，除非用户在稍后阶段单击由 ConfirmLink 现有帐户身份验证器显示的页面中的 Review profile info 链接。

Create User If Unique 创建用户如果唯一

This authenticator checks if there is already an existing Keycloak account with the same email or username like the account from the identity provider. If it's not, then the authenticator just creates a new local Keycloak account and links it with the identity provider and the whole flow is finished.

Otherwise it goes to the next `Handle Existing Account` subflow. If you always want to ensure that there is no duplicated account, you can mark this authenticator as `REQUIRED`. In this case, the user will see the error page if there is an existing Keycloak account and the user will need to link the identity provider account through Account management.

此身份验证程序检查是否已经有一个与身份提供程序的帐户具有相同的电子邮件或用户名的现有 Keycloak 帐户。如果不是，那么身份验证程序只需创建一个新的本地钥匙斗篷帐户，并将其与身份提供程序链接，整个流程就完成了。否则，它将转到下一个 HandleExistingAccount 子流。如果您总是希望确保没有重复的帐户，可以将此身份验证器标记为 REQUIRED。在这种情况下，如果有一个现有的密钥斗篷帐户，并且用户需要通过帐户管理链接身份提供者帐户，则用户将看到错误页面。

- This authenticator verifies that there is already a Keycloak account with the same email or username as the identity provider's account.

此身份验证程序验证是否已有与身份提供程序的帐户具有相同电子邮件或用户名的密钥斗篷帐户。

- If an account does not exist, the authenticator creates a local Keycloak account, links this account with the identity provider, and terminates the flow.

如果帐户不存在，身份验证程序将创建一个本地 Keycloak 帐户，将此帐户与标识提供程序链接，并终止流。

- If an account exists, the authenticator implements the next `Handle Existing Account` sub-flow.

如果存在帐户，身份验证程序将实现下一个 HandleInternetAccount 子流。

- To ensure there is no duplicated account, you can mark this authenticator as `REQUIRED`. The user sees the error page if a Keycloak account exists, and users must link their identity provider account through Account management.

为了确保没有重复的帐户，您可以将这个身份验证器标记为 REQUIRED。用户可以查看错误页面(如果存在 Keycloak 覆盖帐户)，并且用户必须通过帐户管理链接他们的身份提供者帐户。

Confirm Link Existing Account 确认链接现有帐户

- On the information page, users see a Keycloak account with the same email. Users can review their profile again and use a different email or username. The flow restarts and goes back to the `Review Profile` authenticator.

在信息页面上，用户可以看到具有相同电子邮件的“钥匙斗篷”帐户。用户可以再次查看他们的个人资料，并使用不同的电子邮件或用户名。流重新启动并返回到 Review Profile 身份验证器。

- Alternatively, users can confirm that they want to link their identity provider account with their existing Keycloak account.

或者，用户可以确认他们想要将他们的身份提供者帐户与他们现有的密钥斗篷帐户连接起来。

- Disable this authenticator if you do not want users to see this confirmation page and go straight to linking identity provider account by email verification or re-authentication.

如果您不希望用户看到此确认页面，请禁用此身份验证程序，然后直接通过电子邮件验证或重新验证链接身份提供者帐户。

Verify Existing Account By Email 通过电子邮件核实现有帐户

- This authenticator is `ALTERNATIVE` by default. Keycloak uses this authenticator if the realm has an SMTP setup configured.

默认情况下，这个身份验证器是 ALTERNATIVE。如果领域已配置了 SMTP 设置，Keycon 将使用这个身份验证器。

- The authenticator sends an email to users to confirm that they want to link the identity provider with their Keycloak account.

身份验证程序向用户发送一封电子邮件，以确认他们希望将身份提供程序与其密钥斗篷帐户连接起来。

- Disable this authenticator if you do not want to confirm linking by email, but want users to reauthenticate with their password.

如果您不想通过电子邮件确认链接，但希望用户使用其密码重新进行身份验证，则禁用此身份验证程序。

Verify Existing Account By Re-authentication 重新验证现有帐户

- Use this authenticator if the email authenticator is not available. For example, you have not configured SMTP for your realm. This authenticator displays a login screen for users to authenticate to link their Keycloak account with the Identity Provider.

如果电子邮件身份验证器不可用，请使用此身份验证器。例如，您还没有为您的领域配置 SMTP。此身份验证器显示一个登录屏幕，供用户进行身份验证，以将其 Key 帐户与身份提供程序连接起来。

- Users can also re-authenticate with another identity provider already linked to their Keycloak account.

用户还可以使用已经链接到他们的“钥匙斗篷”帐户的另一个身份提供者进行重新身份验证。

- You can force users to use OTP. Otherwise, it is optional and used if you have set OTP for the user account.

您可以强制用户使用 OTP。否则，它是可选的，如果您已经为用户帐户设置了 OTP，则可以使用它。

Automatically link existing first login flow 自动链接现有的第一个登录流



The AutoLink authenticator is dangerous in a generic environment where users can register themselves using arbitrary usernames or email addresses. Do not use this authenticator unless you are carefully curating user registration and assigning usernames and email addresses.

AutoLink 身份验证器在通用环境中是危险的，因为用户可以使用任意用户名或电子邮件地址注册自己。不要使用此身份验证器，除非您正在仔细管理用户注册并分配用户名和电子邮件地址。

To configure a first login flow that links users automatically without prompting, create a new flow with the following two authenticators:

若要配置第一个登录流，在不提示的情况下自动链接用户，请使用以下两个身份验证器创建一个新流：

Create User If Unique 创建用户如果唯一

This authenticator ensures Keycloak handles unique users. Set the authenticator requirement to **Alternative**.

此身份验证器可确保“钥匙斗篷”处理唯一用户。将身份验证器要求设置为 Alternative。

Automatically Set Existing User 自动设置现有用户

This authenticator sets an existing user to the authentication context without verification. Set the authenticator requirement to "Alternative".

此身份验证器将现有用户设置为未经验证的身份验证上下文。将身份验证器要求设置为“Alternative”。



This setup is the simplest setup available, but it is possible to use other authenticators. For example: * You can add the Review Profile authenticator to the beginning of the flow if you want end users to confirm their profile information. * You can add authentication mechanisms to this flow, forcing a user to verify their credentials. Adding authentication mechanisms requires a complex flow. For example, you can set the "Automatically Set Existing User" and "Password Form" as "Required" in an "Alternative" sub-flow.

此设置是可用的最简单设置，但可以使用其他身份验证器。例如：* 如果希望最终用户确认其配置文件信息，可以将 Review Profile 身份验证器添加到流的开头。* 可以向此流添加身份验证机制，强制用户验证其凭据。添加身份验证机制需要复杂的流。例如，可以在“可选”子流程中将“自动设置现有用户”和“密码表单”设置为“必需”。

Disabling automatic user creation 禁用自动用户创建

The Default first login flow looks up the Keycloak account matching the external identity and offers to link them. If no matching Keycloak account exists, the flow automatically creates one.

默认的第一个登录流查找与外部标识匹配的密钥斗篷帐户并提供链接。如果不存在匹配的密钥斗篷帐户，则流将自动创建一个密钥斗篷帐户。

This default behavior may be unsuitable for some setups. One example is when you use a read-only LDAP user store, where all users are pre-created. In this case, you must switch off automatic user creation.

此默认行为可能不适合某些设置。一个例子是当您使用只读 LDAP 用户存储时，其中所有用户都是预先创建的。在这种情况下，必须关闭自动用户创建。

To disable user creation:

禁用用户创建：

Procedure 程序

1. Click **Authentication** in the menu.

单击菜单中的“身份验证”。

2. Select **First Broker Login** from the list.

从列表中选择 First Broker Login。

3. Set **Create User If Unique** to **DISABLED**.

设置创建用户如果唯一到禁用。

4. Set **Confirm Link Existing Account** to **DISABLED**.

将确认链接现有帐户设置为禁用。

This configuration also implies that Keycloak itself won't be able to determine which internal account would correspond to the external identity. Therefore, the `Verify Existing Account By Re-authentication` authenticator will ask the user to provide both username and password.

这种配置还意味着 Keycloak 本身将无法确定哪个内部帐户将对应于外部身份。因此，通过重新认证验证器验证现有帐户将要求用户同时提供用户名和密码。

Detect existing user first login flow 检测现有用户首次登录流

In order to configure a first login flow in which:

为了配置第一个登录流，其中：

- only users already registered in this realm can log in,
只有已经在此领域注册的用户才能登录，
- users are automatically linked without being prompted,
用户会自动链接，而无需提示，

create a new flow with the following two authenticators:

使用以下两个身份验证器创建一个新流：

Detect Existing Broker User 检测现有代理用户

This authenticator ensures that unique users are handled. Set the authenticator requirement to `Mandatory`.

此验证器确保处理唯一用户。将验证器要求设置为 Mandatory。

Automatically Set Existing User 自动设置现有用户

Automatically sets an existing user to the authentication context without any verification. Set the authenticator requirement to `Mandatory`.

无需任何验证即可自动将现有用户设置为身份验证上下文。

You have to set the `First Login Flow` of the identity provider configuration to that flow. You could set the also set `Sync Mode` to `force` if you want to update the user profile (Last Name, First Name...) with the identity provider attributes.

您必须将标识提供程序配置的 First Login Flow 设置为该流。如果要使用标识提供程序属性更新用户配置文件（姓氏，名字...），可以将同步模式设置为强制。



This flow can be used if you want to delegate the identity to other identity providers (such as GitHub, Facebook ...) but you want to manage which users that can log in. 如果希望将身份委托给其他身份提供者(如 GitHub、Facebook...)，但希望管理可以登录的用户，则可以使用此流

With this configuration, Keycloak is unable to determine which internal account corresponds to the external identity. The **Verify Existing Account By Re-authentication** authenticator asks the provider for the username and password.

使用此配置，Keycloak 无法确定哪个内部帐户对应于外部标识。通过重新身份验证来验证现有帐户身份验证器向提供程序询问用户名和密码。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/identity-broker/tokens.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/identity-broker/tokens.adoc)

With Keycloak, you can store tokens and responses from the authentication process with the external IDP using the **Store Token** configuration option on the IDP's settings page.

使用 Keycloak，您可以使用 IDP 设置页面上的 **Store Token** 配置选项来存储来自外部 IDP 的身份验证过程的令牌和响应。

Application code can retrieve these tokens and responses to import extra user information or to request the external IDP securely. For example, an application can use the Google token to use other Google services and REST APIs. To retrieve a token for a particular identity provider, send a request as follows:

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安全地请求外部 IDP。例如，应用程序可以使用 Google 令牌来使用其他 Google 服务和 REST API。若要检索特定标识提供程序的令牌，请按如下方式发送请求：

```
GET /realms/{realm}/broker/{provider_alias}/token HTTP/1.1
Host: localhost:8080
Authorization: Bearer <KEYCLOAK ACCESS TOKEN>
```

An application must authenticate with Keycloak and receive an access token. This access token must have the **broker** client-level role **read-token** set, so the user must have a role mapping for this role, and the client application must have that role within its scope. In this case, since you are accessing a protected service in Keycloak, send the access token issued by Keycloak during the user authentication. You can assign this role to newly imported users in the broker configuration page by setting the **Stored Tokens Readable** switch to **ON**.

应用程序必须使用 Keycloak 进行身份验证并接收访问令牌。此访问令牌必须具有代理客户端级别的角色读令牌集，因此用户必须具有此角色的角色映射，并且客户端应用程序必须在其范围内具有该角色。在这种情况下，因为您正在访问 Keycloak 的受保护服务，所以在用户身份验证期间发送由 Keycloak 发出的访问令牌。通过将 Stored Tokens Readable 开关设置为 ON，可以将此角色分配给代理配置页中新导入的用户。

These external tokens can be re-established by logging in again through the provider or using the client-initiated account linking API.

可以通过提供程序或使用客户端启动的帐户链接 API 再次登录来重新建立这些外部令牌。

[Edit this section](#) 编辑这部分

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/identity-broker/logout.adoc)

[Report an issue](#) 报告一个问题

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/identity-broker/logout.adoc)

When logging out, Keycloak sends a request to the external identity provider that is used to log in initially and logs the user out of this identity provider. You can skip this behavior and avoid logging out of the external identity provider. See [adapter logout documentation](#)

(https://www.keycloak.org/docs/latest/securing_apps/#_java_adapter_logout) for more information.

当注销时，Keycloak 向最初用于登录的外部标识提供程序发送一个请求，并将用户从此标识提供程序中注销。您可以跳过此行为，避免注销外部标识提供程序。有关更多信息，请参见适配器注销文档。

[Edit this section](#) 编辑这部分

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/sso-protocols.adoc)

[Report an issue](#) 报告一个问题

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/sso-protocols.adoc)

This section discusses authentication protocols, the Keycloak authentication server and how applications, secured by the Keycloak authentication server, interact with these protocols.

本节将讨论身份验证协议、钥匙斗篷身份验证服务器以及由钥匙斗篷身份验证服务器保护的应用程序如何与这些协议进行交互。

[Edit this section](#) 编辑这部分

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/sso-protocols/con-oidc.adoc)

[Report an issue](#) 报告一个问题

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/sso-protocols/con-oidc.adoc)

[OpenID Connect](#) (<https://openid.net/connect/>) (OIDC) 是一个身份验证协议，是 OAuth 2.0 (<https://datatracker.ietf.org/doc/html/rfc6749>) 的扩展。

OpenID Connect (OIDC)是一种身份验证协议，是 OAuth 2.0的扩展。

OAuth 2.0 is a framework for building authorization protocols and is incomplete. OIDC, however, is a full authentication and authorization protocol that uses the [Json Web Token](https://jwt.io) (<https://jwt.io>) (JWT) standards. The JWT standards define an identity token JSON format and methods to digitally sign and encrypt data in a compact and web-friendly way.

OAuth 2.0是一个用于构建授权协议的框架，它是不完整的。但是，OIDC 是使用 JsonWebToken (JWT) 标准的完全身份验证和授权协议。JWT 标准定义了一种标识令牌 JSON 格式和方法，用于以紧凑和 Web 友好的方式对数据进行数字签名和加密。

In general, OIDC implements two use cases. The first case is an application requesting that a Keycloak server authenticates a user. Upon successful login, the application receives an *identity token* and an *access token*. The *identity token* contains user information including user name, email, and profile information. The realm digitally signs the *access token* which contains access information (such as user role mappings) that applications use to determine the resources users can access in the application.

通常，OIDC 实现两个用例。第一种情况是一个应用程序，它请求一个钥匙斗篷服务器对用户进行身份验证。成功登录后，应用程序将收到一个标识令牌和一个访问令牌。标识令牌包含用户信息，包括用户名、电子邮件和配置文件信息。领域对包含访问信息(例如用户角色映射)的访问令牌进行数字签名，应用程序使用这些信息来确定用户可以在应用程序中访问的资源。

The second use case is a client accessing remote services.

第二个用例是访问远程服务的客户端。

- The client requests an *access token* from Keycloak to invoke on remote services on behalf of the user.
客户端从 Keycloak 请求一个访问令牌来代表用户调用远程服务。
- Keycloak authenticates the user and asks the user for consent to grant access to the requesting client.

Key斗篷对用户进行身份验证，并请求用户同意授予对请求客户端的访问权限。

- The client receives the *access token* which is digitally signed by the realm.

客户端接收由域进行数字签名的访问令牌。

- The client makes REST requests on remote services using the *access token*.

客户机使用访问令牌对远程服务发出 REST 请求。

- The remote REST service extracts the *access token*.

远程 REST 服务提取访问令牌。

- The remote REST service verifies the tokens signature.

远程 REST 服务验证令牌签名。

- The remote REST service decides, based on access information within the token, to process or reject the request.

远程 REST 服务根据令牌中的访问信息决定处理或拒绝请求。

[Edit this section](#) 编辑这部分

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/sso-protocols/con-oidc-auth-flows.adoc)

[Report an issue](#) 报告一个问题

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/sso-protocols/con-oidc-auth-flows.adoc)

OIDC has several methods, or flows, that clients or applications can use to authenticate users and receive *identity* and *access* tokens. The method depends on the type of application or client requesting access.

OIDC 有几种方法或流，客户机或应用程序可以使用这些方法或流对用户进行身份验证并接收标识和访问令牌。该方法取决于请求访问的应用程序或客户端的类型。

Authorization Code Flow 授权代码流

The Authorization Code Flow is a browser-based protocol and suits authenticating and authorizing browser-based applications. It uses browser redirects to obtain *identity* and *access* tokens.

授权代码流是一种基于浏览器的协议，适合对基于浏览器的应用程序进行身份验证和授权。它使用浏览器重定向来获取标识和访问令牌。

1. A user connects to an application using a browser. The application detects the user is not logged into the application.

用户使用浏览器连接到应用程序。应用程序检测到用户未登录到应用程序。

2. The application redirects the browser to Keycloak for authentication.

应用程序将浏览器重定向到 Keycloak 进行身份验证。

3. The application passes a callback URL as a query parameter in the browser redirect. Keycloak uses the parameter upon successful authentication.

应用程序将回调 URL 作为查询参数在浏览器重定向中传递。在成功身份验证时，Keycloak 使用该参数。

4. Keycloak authenticates the user and creates a one-time, short-lived, temporary code.

Keycloak 对用户进行身份验证，并创建一个一次性的、短期的临时代码。

5. Keycloak redirects to the application using the callback URL and adds the temporary code as a query parameter in the callback URL.

使用回调 URL 重定向到应用程序，并将临时代码作为查询参数添加到回调 URL 中。

6. The application extracts the temporary code and makes a background REST invocation to Keycloak to exchange the code for an *identity* and *access* and *refresh* token. To prevent replay attacks, the temporary code cannot be used more than once.

应用程序提取临时代码，并对 Keycloak 进行后台 REST 调用，以便将代码交换为身份、访问和刷新令牌。为了防止重播攻击，临时代码不能使用一次以上。



A system is vulnerable to a stolen token for the lifetime of that token. For security and scalability reasons, access tokens are generally set to expire quickly so subsequent token requests fail. If a token expires, an application can obtain a new access token using the additional *refresh* token sent by the login protocol.

在令牌的生命周期内，系统容易受到被盗的令牌的攻击。出于安全性和可伸缩性的原因，访问令牌通常设置为快速过期，以便后续令牌请求失败。如果令牌过期，应用程序可以使用登录协议发送的附加刷新令牌获取新的访问令牌。

Confidential clients provide client secrets when they exchange the temporary codes for tokens. *Public* clients are not required to provide client secrets. *Public* clients are secure when HTTPS is strictly enforced and redirect URIs registered for the client are strictly controlled. HTML5/JavaScript clients have to be *public* clients because there is no way to securely transmit the client secret to HTML5/JavaScript clients. For more details, see the Managing Clients chapter.

机密客户机在交换令牌的临时代码时提供客户机机密。公共客户不需要提供客户秘密。当严格执行 HTTPS 并严格控制为客户端注册的重定向 URI 时，公共客户端是安全的。HTML5/JavaScript 客户端必须是公共客户端，因为没有办法安全地将客户端机密传输给 HTML5/JavaScript 客户端。有关详细信息，请参阅管理客户端一章。

Keycloak also supports the [Proof Key for Code Exchange](https://datatracker.ietf.org/doc/html/rfc7636) (<https://datatracker.ietf.org/doc/html/rfc7636>) specification.

密钥斗篷还支持代码交换规范的证明密钥。

Implicit Flow 隐性心流

The Implicit Flow is a browser-based protocol. It is similar to the Authorization Code Flow but with fewer requests and no refresh tokens.

隐式流是一种基于浏览器的协议。它类似于 AuthorizationCodeFlow，但是请求更少，没有刷新令牌。



The possibility exists of access tokens leaking in the browser history when tokens are transmitted via redirect URLs (see below).

当令牌通过重定向 URI 传输时，存在访问令牌在浏览器历史记录中泄漏的可能性(见下文)。

Also, this flow does not provide clients with refresh tokens. Therefore, access tokens have to be long-lived or users have to re-authenticate when they expire.

此外，此流不向客户端提供刷新令牌。因此，访问令牌必须是长期存在的，或者当它们过期时，用户必须重新进行身份验证。

We do not advise using this flow. This flow is supported because it is in the OIDC and OAuth 2.0 specification.

我们不建议使用这个流程。支持这个流程是因为它在 OIDC 和 OAuth 2.0 规范中。

The protocol works as follows:

该协议的工作原理如下：

1. A user connects to an application using a browser. The application detects the user is not logged into the application.

用户使用浏览器连接到应用程序。应用程序检测到用户未登录到应用程序。

2. The application redirects the browser to Keycloak for authentication.

应用程序将浏览器重定向到 Keycloak 进行身份验证。

3. The application passes a callback URL as a query parameter in the browser redirect. Keycloak uses the query parameter upon successful authentication.

应用程序将回调 URL 作为查询参数在浏览器重定向中传递。在成功身份验证时，Keycloak 使用查询参数。

4. Keycloak authenticates the user and creates an *identity* and *access token*. Keycloak redirects to the application using the callback URL and additionally adds the *identity* and *access tokens* as a query parameter in the callback URL.

Keycloak 对用户进行身份验证，并创建标识和访问令牌。Keycloak 使用回调 URL 重定向到应用程序，并在回调 URL 中添加标识和访问令牌作为查询参数。

5. The application extracts the *identity* and *access tokens* from the callback URL.

应用程序从回调 URL 中提取标识和访问令牌。

Resource owner password credentials grant (Direct Access Grants) 资源所有者密码凭证授予(直接访问授予)

Direct Access Grants are used by REST clients to obtain tokens on behalf of users. It is a HTTP POST request that contains:

REST 客户端使用直接访问授权来代表用户获取令牌。这是一个 HTTP POST 请求，其中包含：

- The credentials of the user. The credentials are sent within form parameters.
用户的凭据。凭据在表单参数中发送。
- The id of the client.
客户的 ID。
- The clients secret (if it is a confidential client).
客户保密(如果是保密客户)。

The HTTP response contains the *identity*, *access*, and *refresh* tokens.

HTTP 响应包含标识、访问和刷新令牌。

Client credentials grant 客户凭证授予

The *Client Credentials Grant* creates a token based on the metadata and permissions of a service account associated with the client instead of obtaining a token that works on behalf of an external user. *Client Credentials Grants* are used by REST clients.

客户端凭据授予基于与客户端关联的服务帐户的元数据和权限创建令牌，而不是获取代表外部用户工作的令牌。REST 客户端使用客户端凭据授予。

See the Service Accounts chapter for more information.

有关更多信息，请参见服务帐户一章。

Device authorization grant 设备授权授权

This is used by clients running on internet-connected devices that have limited input capabilities or lack a suitable browser. Here's a brief summary of the protocol:

客户端在连接互联网的设备上运行，这些设备的输入能力有限或缺乏合适的浏览器。以下是该协议的简要概述：

1. The application requests Keycloak a device code and a user code. Keycloak creates a device code and a user code. Keycloak returns a response including the device code and the user code to the application.

该应用程序请求“钥匙斗篷”一个设备代码和一个用户代码。钥匙斗篷创建一个设备代码和一个用户代码。密钥斗篷返回一个响应，包括应用程序的设备代码和用户代码。

2. The application provides the user with the user code and the verification URI. The user accesses a verification URI to be authenticated by using another browser.

应用程序向用户提供用户代码和验证 URI。用户访问一个验证 URI，使用另一个浏览器进行身份验证。

3. The application repeatedly polls Keycloak to find out if the user completed the user authorization. If user authentication is complete, the application exchanges the device code for an *identity*, *access* and *refresh* token.

该应用程序重复轮询 Keycloak，以查明用户是否完成了用户授权。如果用户身份验证完成，应用程序将设备代码交换为标识、访问和刷新令牌。

Client initiated backchannel authentication grant 客户端启动的反向通道身份验证授权

This feature is used by clients who want to initiate the authentication flow by communicating with the OpenID Provider directly without redirect through the user's browser like OAuth 2.0's authorization code grant. Here's a brief summary of the protocol:

这个特性被客户机使用，客户机希望通过与 OpenID 提供者直接通信来启动身份验证流程，而不需要通过用户的浏览器(如 OAuth 2.0的授权代码)进行重定向。以下是该协议的简要概述：

1. The client requests Keycloak an auth_req_id that identifies the authentication request made by the client. Keycloak creates the auth_req_id.

客户端请求 Keycloak 一个 auth_req_id 来标识客户端发出的身份验证请求。

2. After receiving this auth_req_id, this client repeatedly needs to poll Keycloak to obtain an Access Token, Refresh Token and ID Token from Keycloak in return for the auth_req_id until the user is authenticated.

在接收到 auth_req_id 之后，这个客户端需要反复轮询 Keycloak，以从 Keycloak 获得访问令牌、刷新令牌和 ID 令牌，作为 auth_req_id 的回报，直到用户得到身份验证。

An administrator can configure Client Initiated Backchannel Authentication (CIBA) related operations as CIBA Policy per realm.

管理员可以将客户端启动的后台通道身份验证(CIBA)相关操作配置为每个领域的 CIBA 策略。

Also please refer to other places of Keycloak documentation like [Backchannel Authentication Endpoint section](https://www.keycloak.org/docs/latest/securing_apps/#_backchannel_authentication_endpoint) (https://www.keycloak.org/docs/latest/securing_apps/#_backchannel_authentication_endpoint) of Securing Applications and Services Guide and [Client Initiated Backchannel Authentication Grant section](https://www.keycloak.org/docs/latest/securing_apps/#_client_initiated_backchannel_authentication_grant) (https://www.keycloak.org/docs/latest/securing_apps/#_client_initiated_backchannel_authentication_grant) of Securing Applications and Services Guide.

此外，请参考其他地方的密钥斗篷文档，如安全应用程序和服务指南的反向通道身份验证端点部分和客户端启动的反向通道身份验证授予部分的安全应用程序和服务指南。

CIBA Policy CIBA 政策

An administrator carries out the following operations on the Admin Console :

管理员在管理控制台上执行以下操作：

- Open the Authentication → CIBA Policy tab.

打开“身份验证→ CIBA 策略”选项卡。

- Configure items and click `Save`.

配置项目并单击“保存”。

The configurable items and their description follow.

下面是可配置项及其说明。

Configuration 配置	Description 描述
Backchannel Token Delivery Mode 反向通道令牌传递模式	<p>Specifying how the CD (Consumption Device) gets the authentication result and related tokens. There are three modes, "poll", "ping" and "push". Keycloak only supports "poll". The default setting is "poll". This configuration is required. For more details, see CIBA Specification (https://openid.net/specs/openid-client-initiated-backchannel-authentication-core-1_0.html#rfc.section.5)</p> <p>.</p> <p>指定 CD (消费设备)如何获取身份验证结果和相关令牌。有三种模式，“投票”，“平”和“推”。钥匙斗篷只支持“民意测验”。默认设置为“ poll ”。这个配置是必需的。有关详细信息，请参阅 CIBA 规范。</p>
Expires In 过期	<p>The expiration time of the "auth_req_id" in seconds since the authentication request was received. The default setting is 120. This configuration is required. For more details, see CIBA Specification (https://openid.net/specs/openid-client-initiated-backchannel-authentication-core-1_0.html#successful_authentication_request_acknowledgment)</p> <p>.</p> <p>“ auth _ req _ id ”自接收身份验证请求以来的过期时间(秒)。默认设置为120。这个配置是必需的。有关详细信息，请参阅 CIBA 规范。</p>

Configuration 配置	Description 描述
Interval 间隔	<p>The interval in seconds the CD (Consumption Device) needs to wait for between polling requests to the token endpoint. The default setting is 5. This configuration is optional. For more details, see CIBA Specification (https://openid.net/specs/openid-client-initiated-backchannel-authentication-core-1_0.html#successful_authentication_request_acknowledgment)</p> <p>.</p> <p>CD (消费设备)在轮询到令牌端点的请求之间需要等待的时间间隔(秒)。默认设置为5。此配置是可选的。有关详细信息，请参阅 CIBA 规范。</p>
Authentication Requested User Hint 身份验证请求的用户提示	<p>The way of identifying the end-user for whom authentication is being requested. The default setting is "login_hint". There are three modes, "login_hint", "login_hint_token" and "id_token_hint". Keycloak only supports "login_hint". This configuration is required.</p> <p>For more details, see CIBA Specification (https://openid.net/specs/openid-client-initiated-backchannel-authentication-core-1_0.html#rfc.section.7.1)</p> <p>.</p> <p>标识正在为其请求身份验证的最终用户的方法。默认设置是“login_tip”。有三种模式，“login_tip”、“login_tip_token”和“id_token_tip”。密钥斗篷只支持“login_tip”。这个配置是必需的。有关详细信息，请参阅 CIBA 规范。</p>

Provider Setting 提供程序设置

The CIBA grant uses the following two providers.

CIBA 补助使用以下两个提供者。

1. Authentication Channel Provider : provides the communication between Keycloak and the entity that actually authenticates the user via AD (Authentication Device).

身份验证通道提供者: 提供钥匙斗篷和实际通过 AD (身份验证设备)对用户进行身份验证的实体之间的通信。

2. User Resolver Provider : get `UserModel` of Keycloak from the information provided by the client to identify the user.

用户解析器提供程序: 从客户端提供的信息中获取钥匙斗篷的用户模型以识别用户。

Keycloak has both default providers. However, the administrator needs to set up Authentication Channel Provider like this:

钥匙斗篷具有两个默认提供程序。但是，管理员需要像下面这样设置身份验证通道提供程序：

```
kc.[sh|bat] start --spi-ciba-auth-channel-ciba-http-auth-channel-http-authentication-channel-  
uri=https://backend.internal.example.com
```

The configurable items and their description follow.

下面是可配置项及其说明。

Configuration 配置	Description 描述
http-authentication-channel-uri	Specifying URI of the entity that actually authenticates the user via AD (Authentication Device).
Http-entication-channel-uri	指定实际通过 AD (身份验证设备)对用户进行身份验证的实体的 URI。

Authentication Channel Provider 认证频道提供者

CIBA standard document does not specify how to authenticate the user by AD. Therefore, it might be implemented at the discretion of products. Keycloak delegates this authentication to an external authentication entity. To communicate with the authentication entity, Keycloak provides Authentication Channel Provider.

CIBA 标准文档没有指定如何通过 AD 对用户进行身份验证。因此，它可以根据产品的需要来实现。Key斗篷将此身份验证委托给外部身份验证实体。为了与身份验证实体进行通信，Keycon 提供了身份验证通道提供程序。

Its implementation of Keycloak assumes that the authentication entity is under the control of the administrator of Keycloak so that Keycloak trusts the authentication entity. It is not recommended to use the authentication entity that the administrator of Keycloak cannot control.

它的实现假设身份验证实体处于管理员的控制之下，这样 Key 就可以信任身份验证实体。不建议使用 Keycloak 管理人无法控制的认证实体。

Authentication Channel Provider is provided as SPI provider so that users of Keycloak can implement their own provider in order to meet their environment. Keycloak provides its default provider called HTTP Authentication Channel Provider that uses HTTP to communicate with the authentication entity.

认证频道供应商是以数码声音广播服务供应商的身份提供服务，让 Keycloak 用户可以实施自己的认证频道供应商，以配合他们的环境。Keycover 提供了名为 HTTP 身份验证通道提供程序的默认提供程序，该提供程序使用 HTTP 与身份验证实体进行通信。

If a user of Keycloak user want to use the HTTP Authentication Channel Provider, they need to know its contract between Keycloak and the authentication entity consisting of the following two parts.

如果 Keycloak 用户希望使用 HTTP 认证频道提供者，他们需要知道 Keycloak 与认证实体之间的合约，该合约包括以下两部分。

Authentication Delegation Request/Response 身份验证委托请求/响应

Keycloak sends an authentication request to the authentication entity.

Keycover 向身份验证实体发送身份验证请求。

Authentication Result Notification/ACK 认证结果通知/ACK

The authentication entity notifies the result of the authentication to Keycloak.

认证实体会将认证结果通知 Keycloak。

Authentication Delegation Request/Response consists of the following messaging.

身份验证委托请求/响应由以下消息组成。

Authentication Delegation Request 认证委托请求

The request is sent from Keycloak to the authentication entity to ask it for user authentication by AD.

请求从 Keycloak 发送到认证实体，请求该实体使用 AD 进行用户认证。

POST [delegation_reception]

- Headers

标题

Name 姓名	Value 价值	Description 描述
Content-Type 内容类别	application/json 应用程序/json	The message body is json formatted. 消息正文是 json 格式的。

Name 姓名	Value 价值	Description 描述
Authorization 授权	Bearer [token] 持有人[信物]	The [token] is used when the authentication entity notifies the result of the authentication to Keycloak. 当身份验证实体将身份验证结果通知 Keycloak 时，会使用[标记]。

- Parameters

参数

Type 类型	Name 姓名	Description 描述
Path 路径	delegation_reception 代表团_接待	The endpoint provided by the authentication entity to receive the delegation request 身份验证实体提供的接收委托请求的端点

- Body

尸体

Name 姓名	Description 描述
login_hint Login_tip 登录提示	It tells the authentication entity who is authenticated by AD. By default, it is the user's "username". This field is required and was defined by CIBA standard document. 它告诉认证实体谁被 AD 认证。默认情况下，它是用户的“用户名”。此字段是必需的，并由 CIBA 标准文档定义。

Name 姓名	Description 描述
scope 范围	<p>It tells which scopes the authentication entity gets consent from the authenticated user. This field is required and was defined by CIBA standard document.</p> <p>它告诉身份验证实体获得经过身份验证的用户的同意的作用域。此字段是必需的，并由 CIBA 标准文档定义。</p>
is Consent Required 必须征得同意	<p>It shows whether the authentication entity needs to get consent from the authenticated user about the scope. This field is required.</p> <p>它显示身份验证实体是否需要获得经过身份验证的用户对范围的同意。这个字段是必需的。</p>
binding Message Bind _ message	<p>Its value is intended to be shown in both CD and AD's UI to make the user recognize that the authentication by AD is triggered by CD. This field is optional and was defined by CIBA standard document.</p> <p>它的值将在 CD 和 AD 的 UI 中显示，以使用户认识到 AD 的身份验证是由 CD 触发的。该字段是可选的，由 CIBA 标准文档定义。</p>
acr Values Acr _ values	<p>It tells the requesting Authentication Context Class Reference from CD. This field is optional and was defined by CIBA standard document.</p> <p>它从 CD 告诉请求身份验证上下文类引用。该字段是可选的，由 CIBA 标准文档定义。</p>

Authentication Delegation Response 身份验证委托响应

The response is returned from the authentication entity to Keycloak to notify that the authentication entity received the authentication request from Keycloak.

认证机构向 Keycloak 发出回应，通知认证机构收到来自 Keycloak 的认证要求。

- Responses

回答

HTTP Status Code HTTP状态码	Description 描述
201	<p>It notifies Keycloak of receiving the authentication delegation request.</p> <p>它将接收身份验证委托请求的消息通知 Keycloak。</p>

Authentication Result Notification/ACK consists of the following messaging.

身份验证结果通知/ACK 由以下消息组成。

Authentication Result Notification 认证结果通知书

The authentication entity sends the result of the authentication request to Keycloak.

身份验证实体将身份验证请求的结果发送到 Keycloak。

```
POST /realms/[realm]/protocol/openid-connect/ext/ciba/auth/callback
```

- Headers

标题

Name 姓名	Value 价值	Description 描述
Content-Type	application/json	The message body is json formatted.
内容类别	应用程序/json	消息正文是 json 格式的。
Authorization	Bearer [token]	The [token] must be the one the authentication entity has received from Keycloak in Authentication Delegation Request.
授权	持有人[信物]	[令牌]必须是认证实体在认证委托要求中从 Keycloak 收到的令牌。

- Parameters

参数

Type 类型	Name 姓名	Description 描述
Path 路径	realm 王国	The realm name 王国的名字
• Body 尸体		

Name 姓名	Description 描述
status 地位	<p>It tells the result of user authentication by AD. It must be one of the following status.</p> <p>SUCCEED : The authentication by AD has been successfully completed.</p> <p>UNAUTHORIZED : The authentication by AD has not been completed.</p> <p>CANCELLED : The authentication by AD has been cancelled by the user.</p> <p>它通过 AD 告知用户身份验证的结果。它必须是下列状态之一。成功: AD 认证已经成功完成。未授权: AD 的身份验证尚未完成。取消: 由广告诉认已被用户取消。</p>

Authentication Result ACK 认证结果

The response is returned from Keycloak to the authentication entity to notify Keycloak received the result of user authentication by AD from the authentication entity.

响应从 Keycloak 返回给认证实体，以通知 Keycloak 从认证实体收到 AD 的用户认证结果。

- Responses

回答

HTTP Status Code HTTP状态码	Description 描述
200	<p>It notifies the authentication entity of receiving the notification of the authentication result.</p> <p>它通知身份验证实体接收身份验证结果的通知。</p>

Even if the same user, its representation may differ in each CD, Keycloak and the authentication entity.

即使是同一个用户，它的表示在每张 CD、钥匙斗篷和身份验证实体中也可能有所不同。

For CD, Keycloak and the authentication entity to recognize the same user, this User Resolver Provider converts their own user representations among them.

为了使 CD、钥匙斗篷和身份验证实体识别同一个用户，这个用户解析器提供程序在它们之间转换它们自己的用户表示。

User Resolver Provider is provided as SPI provider so that users of Keycloak can implement their own provider in order to meet their environment. Keycloak provides its default provider called Default User Resolver Provider that has the following characteristics.

用户解析器提供程序是以 SPI 提供程序的形式提供的，因此 Keycloak 的用户可以实现自己的提供程序，以满足他们的环境。钥匙斗篷提供名为默认用户解析器提供程序的默认提供程序，该提供程序具有以下特征。

- Only support `login_hint` parameter and is used as default.
只支持 `login_tip` 参数，并作为默认参数使用。
- `username` of UserModel in Keycloak is used to represent the user on CD, Keycloak and the authentication entity.

Keycloak 的用户名是用来代表光碟、钥匙斗篷及认证实体的用户。

OIDC Logout 注销

OIDC has four specifications relevant to logout mechanisms. These specifications are in draft status:

OIDC 有四个与注销机制相关的规范，这些规范处于草案状态：

1. [Session Management](https://openid.net/specs/openid-connect-session-1_0.html) (https://openid.net/specs/openid-connect-session-1_0.html)

会议管理

2. [RP-Initiated Logout](https://openid.net/specs/openid-connect-rpinitiated-1_0.html) (https://openid.net/specs/openid-connect-rpinitiated-1_0.html)

RP 启动注销

3. [Front-Channel Logout](https://openid.net/specs/openid-connect-frontchannel-1_0.html) (https://openid.net/specs/openid-connect-frontchannel-1_0.html)

前台注销

4. [Back-Channel Logout](https://openid.net/specs/openid-connect-backchannel-1_0.html) (https://openid.net/specs/openid-connect-backchannel-1_0.html)

后台注销

Again since all of this is described in the OIDC specification we will only give a brief overview here.

同样，由于所有这些都在 OIDC 规范中进行了描述，因此我们在这里只给出一个简要的概述。

Session Management 会议管理

This is a browser-based logout. The application obtains session status information from Keycloak at a regular basis. When the session is terminated at Keycloak the application will notice and trigger its own logout.

这是一个基于浏览器的注销。应用程式定期从 Keycloak 取得会话状态资料。当会话在 Keycloak 终止时，应用程序会发现并触发自己的注销。

RP-Initiated Logout RP 启动注销

This is also a browser-based logout where the logout starts by redirecting the user to a specific endpoint at Keycloak. This redirect usually happens when the user clicks the `Log Out` link on the page of some application, which previously used Keycloak to authenticate the user.

这也是一个基于浏览器的注销，注销开始于将用户重定向到 Keycloak 的特定端点。这种重定向通常发生在用户单击某个应用程序页面上的注销链接时，这个应用程序以前曾使用 Keycloak 对用户进行身份验证。

Once the user is redirected to the logout endpoint, Keycloak is going to send logout requests to clients to let them invalidate their local user sessions, and potentially redirect the user to some URL once the logout process is finished. The user might be optionally requested to confirm the logout in case the `id_token_hint` parameter was not used. After logout, the user is automatically redirected to the specified `post_logout_redirect_uri` as long as it is provided as a parameter. Note that you need to include either the `client_id` or `id_token_hint` parameter in case the `post_logout_redirect_uri` is included. Also the `post_logout_redirect_uri` parameter needs to match one of the `Valid Post Logout Redirect URIs` specified in the client configuration.

一旦用户被重定向到注销端点，Keycloak 将向客户机发送注销请求，让客户机使其本地用户会话失效，并且一旦注销过程结束，可能会将用户重定向到某个 URL。可以选择请求用户确认注销，以防未使用 `id_token_tip` 参数。注销后，用户会自动重定向到指定的 `post_logout_redirect_uri`，只要它作为参数提供。请注意，如果包含 `post_logout_redirect_uri`，则需要包含 `client_id` 或 `id_token_tip` 参数。另外，`Post Logout Redirect Uri` 参数需要与客户机配置中指定的有效 Post Logout 重定向 URI 之一匹配。

Depending on the client configuration, logout requests can be sent to clients through the front-channel or through the back-channel. For the frontend browser clients, which rely on the Session Management described in the previous section, Keycloak does not need to send any logout requests to them; these clients automatically detect that SSO session in the browser is logged out.

根据客户端配置，注销请求可以通过前端通道或后端通道发送给客户端。对于前端浏览器客户端，它们依赖于前面部分中描述的 Session Management，Keycloak 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 会话已经注销。

Front-channel Logout 前台注销

To configure clients to receive logout requests through the front-channel, look at the Front-Channel Logout client setting. When using this method, consider the following:

若要配置客户端以通过前端通道接收注销请求，请查看 Front-Channel Logout 客户端设置。使用此方法时，请考虑以下因素：

- Logout requests sent by Keycloak to clients rely on the browser and on embedded `iframes` that are rendered for the logout page.

键斗篷发送给客户端的注销请求依赖于浏览器和为注销页面呈现的嵌入式 iframe。

- By being based on `iframes`, front-channel logout might be impacted by Content Security Policies (CSP) and logout requests might be blocked.

通过基于 iframe，前端通道注销可能受到内容安全策略(CSP)的影响，注销请求可能被阻塞。

- If the user closes the browser prior to rendering the logout page or before logout requests are actually sent to clients, their sessions at the client might not be invalidated.

如果用户在呈现注销页面之前或在实际向客户端发送注销请求之前关闭浏览器，则客户端的会话可能不会失效。

 Consider using Back-Channel Logout as it provides a more reliable and secure approach to log out users and terminate their sessions on the clients.

考慮使用 Back-Channel Logout，因为它提供了一种更可靠、更安全的方法来注销用户并终止客户机上的会话。

If the client is not enabled with front-channel logout, then Keycloak is going to try first to send logout requests through the back-channel using the Back-Channel Logout URL. If not defined, the server is going to fall back to using the Admin URL.

如果客户端没有启用前端通道注销，那么 Keycloak 将首先尝试使用后端通道注销 URL 通过后端通道发送注销请求。如果没有定义，服务器将退回到使用管理 URL。

Backchannel Logout 反向通道注销

This is a non-browser-based logout that uses direct backchannel communication between Keycloak and clients. Keycloak sends a HTTP POST request containing a logout token to all clients logged into Keycloak. These requests are sent to a registered backchannel logout URLs at Keycloak and are supposed to trigger a logout at client side.

这是一个非基于浏览器的注销，它使用 Keycloak 和客户端之间的直接反向通道通信。钥匙斗篷发送一个包含注销令牌的 HTTP POST 请求给所有登录到钥匙斗篷的客户端。这些请求被发送到 Keycloak 的一个已注册的反向通道注销 URL，并应该在客户端触发一个注销。

[Edit this section 编辑这部分](#)
(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/sso-protocols/con-server-oidc-uri-endpoints.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/sso-protocols/con-server-oidc-uri-endpoints.adoc)

The following is a list of OIDC endpoints that Keycloak publishes. These endpoints can be used when a non-Keycloak client adapter uses OIDC to communicate with the authentication server. They are all relative URLs. The root of the URL consists of the HTTP(S) protocol, hostname, and optionally the path: For example

下面列出了钥匙斗篷发布的 OIDC 端点。这些端点可以用于非钥匙斗篷客户端适配器使用 OIDC 与身份验证服务器通信时。它们都是相对 URL。URL 的根由 HTTP (S) 协议、主机名和可选的路径组成: 例如

`https://localhost:8080`

`/realms/{realm-name}/protocol/openid-connect/auth` / realms / { realm-name } / protocol / openid-connect / auth

Used for obtaining a temporary code in the Authorization Code Flow or obtaining tokens using the Implicit Flow, Direct Grants, or Client Grants.

用于获取授权代码流中的临时代码或使用隐式流、直接授予或客户端授予获取令牌。

`/realms/{realm-name}/protocol/openid-connect/token` / realms / { realm-name } / protocol / openid-connect / token

Used by the Authorization Code Flow to convert a temporary code into a 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` / fields / { domain-name } / protocol / openid-connect / logout / 领域 / { 域名 } / 协议 / openid-连接 / 注销

Used for performing logouts.

用于执行注销。

`/realms/{realm-name}/protocol/openid-connect/userinfo` 领域 / { domain-name } / protocol / openid-connect / userinfo

Used for the User Info service described in the OIDC specification.

用于 OIDC 规范中描述的 UserInfo 服务。

`/realms/{realm-name}/protocol/openid-connect/revoke` / realms / { realm-name } / protocol / openid-connect / revoke

Used for OAuth 2.0 Token Revocation described in [RFC7009](https://datatracker.ietf.org/doc/html/rfc7009) (<https://datatracker.ietf.org/doc/html/rfc7009>).

用于 RFC7009 中描述的 OAuth 2.0 令牌撤销。

/realms/{realm-name}/protocol/openid-connect/certs 领域/{ domain-name }/protocol/openid-connect/certs

Used for the JSON Web Key Set (JWKS) containing the public keys used to verify any JSON Web Token (jwks_uri)

用于包含用于验证任何 JSON Web 令牌(JWKS _ uri)的公钥的 JSON Web 密钥集(JWKS)

/realms/{realm-name}/protocol/openid-connect/auth/device /fields/{ domain-name }/protocol/openid-connect/auth/device//领域/{领域名}/协议/openid-连接/授权/设备

Used for Device Authorization Grant to obtain a device code and a user code.

用于设备授权授权以获取设备代码和用户代码。

/realms/{realm-name}/protocol/openid-connect/ext/ciba/auth /fields/{ domain-name }/protocol/openid-connect/ext/ciba/auth/领域/{领域名}/协议/openid-connect/ext/ciba/auth

This is the URL endpoint for Client Initiated Backchannel Authentication Grant to obtain an auth_req_id that identifies the authentication request made by the client.

这是 Client Initiated Backchannel Authentication Grant 的 URL 端点，用于获取 auth_req_id，该 auth_req_id 标识客户端发出的身份验证请求。

/realms/{realm-name}/protocol/openid-connect/logout/backchannel-logout 领域/{ domain-name }/protocol/openid-connect/logout/backchannel-logout

This is the URL endpoint for performing backchannel logouts described in the OIDC specification.

这是用于执行 OIDC 规范中描述的反向通道注销的 URL 端点。

In all of these, replace {realm-name} with the name of the realm.

在所有这些内容中，用领域的名称替换{ domain-name }。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/sso-protocols/con-saml.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/sso-protocols/con-saml.adoc)

SAML 2.0 (<http://saml.xml.org/saml-specifications>) is a similar specification to OIDC but more mature. It is descended from SOAP and web service messaging specifications so is generally more verbose than OIDC. SAML 2.0 is an authentication protocol that exchanges XML documents between authentication servers and applications. XML signatures and encryption are used to verify requests and responses.

SAML 2.0与OIDC类似，但更加成熟。它是SOAP和Web服务消息传递规范的后裔，因此通常比OIDC更冗长。SAML 2.0是一种在身份验证服务器和应用程序之间交换XML文档的身份验证协议。XML签名和加密用于验证请求和响应。

In general, SAML implements two use cases.

通常，SAML实现两个用例。

The first use case is an application that requests the Keycloak server authenticates a user. Upon successful login, the application will receive an XML document. This document contains an SAML assertion that specifies user attributes. The realm digitally signs the document which contains access information (such as user role mappings) that applications use to determine the resources users are allowed to access in the application.

第一个用例是一个应用程序，它请求钥匙斗篷服务器对用户进行身份验证。成功登录后，应用程序将收到一个XML文档。此文档包含指定用户属性的SAML断言。领域对包含访问信息(例如用户角色映射)的文档进行数字签名，应用程序使用这些信息来确定应用程序中允许用户访问的资源。

The second use case is a client accessing remote services. The client requests a SAML assertion from Keycloak to invoke on remote services on behalf of the user.

第二个用例是访问远程服务的客户端。客户端从Keycloak请求一个SAML断言来代表用户调用远程服务。

[Edit this section](#) 编辑这部分

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/sso-protocols/con-saml-bindings.adoc)

[Report an issue](#) 报告一个问题

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/sso-protocols/con-saml-bindings.adoc)

Keycloak supports three binding types.

密钥斗篷支持三种绑定类型。

Redirect binding 重定向绑定

Redirect binding uses a series of browser redirect URIs to exchange information.

重定向绑定使用一系列浏览器重定向URI来交换信息。

1. A user connects to an application using a browser. The application detects the user is not authenticated.

用户使用浏览器连接到应用程序。应用程序检测到用户未经身份验证。

2. The application generates an XML authentication request document and encodes it as a query parameter in a URI. The URI is used to redirect to the Keycloak server. Depending on your settings, the application can also digitally sign the XML document and include the signature as a query

parameter in the redirect URI to Keycloak. This signature is used to validate the client that sends the request.

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 被用来重定向到钥匙斗篷服务器。根据您的设置，应用程序还可以对 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 Keycloak 中。此签名用于验证发送请求的客户端。

3. The browser redirects to Keycloak.

浏览器重定向到“钥匙斗篷”。

4. The server extracts the XML auth request document and verifies the digital signature, if required.

服务器提取 XML 身份验证请求文档，并在需要时验证数字签名。

5. The user enters their authentication credentials.

用户输入其身份验证凭据。

6. After authentication, the server generates an XML authentication response document. The document contains a SAML assertion that holds metadata about the user, including name, address, email, and any role mappings the user has. The document is usually digitally signed using XML signatures, and may also be encrypted.

身份验证之后，服务器生成 XML 身份验证响应文档。该文档包含一个 SAML 断言，该断言包含有关用户的元数据，包括用户的名称、地址、电子邮件和任何角色映射。文档通常使用 XML 签名进行数字签名，也可能进行加密。

7. The XML authentication response document is encoded as a query parameter in a redirect URI. The URI brings the browser back to the application. The digital signature is also included as a query parameter.

XML 身份验证响应文档被编码为重定向 URI 中的查询参数。URI 将浏览器带回应用程序。数字签名也包含在查询参数中。

8. The application receives the redirect URI and extracts the XML document.

应用程序接收重定向 URI 并提取 XML 文档。

9. The application verifies the realm's signature to ensure it is receiving a valid authentication response. The information inside the SAML assertion is used to make access decisions or display user data.

应用程序验证领域的签名，以确保它正在接收有效的身份验证响应。SAML 断言中的信息用于做出访问决策或显示用户数据。

POST binding POST 绑定

POST binding is similar to *Redirect* binding but POST binding exchanges XML documents using POST requests instead of using GET requests. POST Binding uses JavaScript to make the browser send a POST request to the Keycloak server or application when exchanging documents. HTTP responds with an HTML document which contains an HTML form containing embedded JavaScript. When the page loads, the JavaScript automatically invokes the form.

POST 绑定类似于重定向绑定，但 POST 绑定使用 POST 请求而不是 GET 请求交换 XML 文档。POST 绑定使用 JavaScript 使浏览器在交换文档时向 Keycloak 服务器或应用程序发送 POST 请求。HTTP 使用一个 HTML 文档进行响应，该文档包含一个包含嵌入式 JavaScript 的 HTML 表单。当页面加载时，JavaScript 会自动调用表单。

POST binding is recommended due to two restrictions:

由于两个限制，建议使用 POST 绑定：

- **Security**—With *Redirect* binding, the SAML response is part of the URL. It is less secure as it is possible to capture the response in logs.
安全性——通过重定向绑定，SAML 响应是 URL 的一部分。它不太安全，因为在日志中可以捕获响应。
- **Size**—Sending the document in the HTTP payload provides more scope for large amounts of data than in a limited URL.
大小——使用 HTTP 有效负载发送文档比使用有限的 URL 发送大量数据提供了更大的空间。

ECP 心电图

Enhanced Client or Proxy (ECP) is a SAML v.2.0 profile which allows the exchange of SAML attributes outside the context of a web browser. It is often used by REST or SOAP-based clients.

增强型客户端或代理(ECP)是一个 SAML v. 2.0概要文件，它允许在 Web 浏览器上下文之外交换 SAML 属性。它通常由 REST 或基于 SOAP 的客户端使用。

Keycloak Server SAML URI Endpoints 密钥斗篷服务器 SAML URI 端点

Keycloak has one endpoint for all SAML requests.

Key斗篷对所有 SAML 请求都有一个端点。

```
http(s)://authserver.host/realms/{realm-name}/protocol/saml
```

Http (s)://authserver.host/fields/{ domain-name }/protocol/saml

All bindings use this endpoint.

所有绑定都使用此端点。

[Edit this section 编辑这部分](https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/sso-protocols/ref-saml-vs-oidc.adoc)
(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/sso-protocols/ref-saml-vs-oidc.adoc)

[Report an issue 报告一个问题](https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/sso-protocols/ref-saml-vs-oidc.adoc)
(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/sso-protocols/ref-saml-vs-oidc.adoc)

The following lists a number of factors to consider when choosing a protocol.

下面列出了在选择协议时需要考虑的一些因素。

For most purposes, Keycloak recommends using OIDC.

对于大多数情况，Keycloak建议使用OIDC。

OIDC

- OIDC is specifically designed to work with the web.

OIDC 是专门设计来与网络协同工作的。

- OIDC is suited for HTML5/JavaScript applications because it is easier to implement on the client side than SAML.

OIDC 适合于 HTML5/JavaScript 应用程序，因为它比 SAML 更容易在客户端实现。

- OIDC tokens are in the JSON format which makes them easier for Javascript to consume.

OIDC 令牌采用 JSON 格式，这使得 Javascript 更容易使用它们。

- OIDC has features to make security implementation easier. For example, see the [iframe trick](#) (https://openid.net/specs/openid-connect-session-1_0.html#ChangeNotification) that the specification uses to determine a user's login status.

OIDC 具有使安全实现更容易的特性。例如，请参见规范用于确定用户登录状态的 iframe 技巧。

SAML

- SAML is designed as a layer to work on top of the web.

SAML 被设计成一个工作在 Web 之上的层。

- SAML can be more verbose than OIDC.

SAML 可能比 OIDC 更冗长。

- Users pick SAML over OIDC because there is a perception that it is mature.

用户选择 SAML 而不是 OIDC，因为他们认为 SAML 是成熟的。

- Users pick SAML over OIDC existing applications that are secured with it.

用户选择 SAML 而不是使用 SAML 保护的 OIDC 现有应用程序。

[Edit this section](#) 编辑这部分

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/sso-protocols/con-sso-docker.adoc)

[Report an issue](#) 报告一个问题

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/sso-protocols/con-sso-docker.adoc)



Docker authentication is disabled by default. To enable docker authentication, see the [Enabling and disabling features](https://www.keycloak.org/server/features) (<https://www.keycloak.org/server/features>) guide.

Docker 身份验证在默认情况下是禁用的。要启用 Docker 身份验证，请参阅启用和禁用特性指南。

[Docker Registry V2 Authentication](https://docs.docker.com/registry/spec/auth/) (<https://docs.docker.com/registry/spec/auth/>) is a protocol, similar to OIDC, that authenticates users against Docker registries. Keycloak's implementation of this protocol lets Docker clients use a Keycloak authentication server authenticate against a registry. This protocol uses standard token and signature mechanisms but it does deviate from a true OIDC implementation. It deviates by using a very specific JSON format for requests and responses as well as mapping repository names and permissions to the OAuth scope mechanism.

Docker Registry V2 Authentication 是一个类似于 OIDC 的协议，用于根据 Docker 注册中心对用户进行身份验证。这个协议的实现允许 Docker 客户端使用针对注册表进行身份验证的 Keycloak 身份验证服务器。这个协议使用标准的令牌和签名机制，但它确实偏离了真正的 OIDC 实现。它偏离了对请求和响应使用非常特定的 JSON 格式，并将存储库名称和权限映射到 OAuth 范围机制。

Docker authentication flow Docker 认证流程

The authentication flow is described in the [Docker API documentation](https://docs.docker.com/registry/spec/auth/token/) (<https://docs.docker.com/registry/spec/auth/token/>). The following is a summary from the perspective of the Keycloak authentication server:

Docker API 文档中描述了身份验证流程。以下是从钥匙斗篷身份验证服务器的角度进行的总结：

- Perform a `docker login`.

执行 Docker 登录。

- The Docker client requests a resource from the Docker registry. If the resource is protected and no authentication token is in the request, the Docker registry server responds with a 401 HTTP message with some information on the permissions that are required and the location of the authorization server.

Docker 客户端从 Docker 注册表请求资源。如果资源受到保护，并且请求中没有身份验证令牌，则 Docker 注册表服务器将用一条 401 HTTP 消息进行响应，其中包含关于所需权限和授权服务器位置的一些信息。

- The Docker client constructs an authentication request based on the 401 HTTP message from the Docker registry. The client uses the locally cached credentials (from the `docker login` command) as part of the [HTTP Basic Authentication](https://datatracker.ietf.org/doc/html/rfc2617) (<https://datatracker.ietf.org/doc/html/rfc2617>) request to the Keycloak authentication server.

Docker 客户端基于来自 Docker 注册中心的 401 HTTP 消息构造身份验证请求。客户端使用本地缓存的凭据(来自 docker login 命令)作为向 Keycloak 身份验证服务器发出的 HTTP 基本身份验证请求的一部分。

- The Keycloak authentication server attempts to authenticate the user and return a JSON body containing an OAuth-style Bearer token.

Key斗篷身份验证服务器尝试对用户进行身份验证，并返回一个包含 OAuth 风格 Bearer 令牌的 JSON 主体。

- The Docker client receives a bearer token from the JSON response and uses it in the authorization header to request the protected resource.

Docker 客户机从 JSON 响应接收一个持有者令牌，并在授权头中使用它来请求受保护的资源。

- The Docker registry receives the new request for the protected resource with the token from the Keycloak server. The registry validates the token and grants access to the requested resource (if appropriate).

Docker 注册表接收对受保护资源的新请求，该请求带有来自 Keycover 服务器的令牌。注册中心验证令牌并授予对请求资源的访问权(如果适当)。

 Keycloak does not create a browser SSO session after successful authentication with the Docker protocol. The browser SSO session does not use the Docker protocol as it cannot refresh tokens or obtain the status of a token or session from the Keycloak server; therefore a browser SSO session is not necessary. For more details, see the 在成功使用 Docker 协议进行身份验证之后，Keycover 不会创建浏览器 SSO 会话。浏览器 SSO 会话不使用 Docker 协议，因为它不能刷新令牌，也不能从 Keycloak 服务器获得令牌或会话的状态；因此，浏览器 SSO 会话是不必要的。有关更多详细信息，请参见 transient session 暂时性会话 section. 部分

Keycloak Docker Registry v2 Authentication Server URI Endpoints 钥匙斗篷 Docker 注册表 v2 身份验证服务器 URI 端点

Keycloak has one endpoint for all Docker auth v2 requests.

对于所有 Docker auth v2 请求，Key斗篷都有一个端点。

`http(s)://authserver.host/realm/{realm-name}/protocol/docker-v2`

`Http (s)://authserver.host/fields/{ domain-name }/protocol/docker-v2`

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/admin-console-permissions.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/admin-console-permissions.adoc)

Each realm created on the Keycloak has a dedicated Admin Console from which that realm can be managed. The `master` realm is a special realm that allows admins to manage more than one realm on the system. You can also define fine-grained access to users in different realms to manage the server. This chapter goes over all the scenarios for this.

在 Keycloak 上創建的每個領域都有一個專用的管理控制台，可以從中管理這個領域。主域是一个特殊的域，允许管理员管理系统上的多个域。还可以定义对不同领域中的用户的细粒度访问，以管理服务器。这一章介绍了所有这方面的情况。

[Edit this section](#) 编辑这部分

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/admin-console-permissions/master-realm.adoc)

[Report an issue](#) 报告一个问题

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/admin-console-permissions/master-realm.adoc)

The `master` realm in Keycloak is a special realm and treated differently than other realms. Users in the Keycloak `master` realm can be granted permission to manage zero or more realms that are deployed on the Keycloak server. When a realm is created, Keycloak automatically creates various roles that grant fine-grain permissions to access that new realm. Access to The Admin Console and Admin REST endpoints can be controlled by mapping these roles to users in the `master` realm. It's possible to create multiple superusers, as well as users that can only manage specific realms.

Keycloak 的大师境界是一个特殊的境界，与其他境界不同。可以授予钥匙斗篷主域中的用户管理部署在钥匙斗篷服务器上的零个或多个领域的权限。在创建领域时，Keycloak 会自动创建各种角色，这些角色授予访问新领域的细粒度权限。可以通过将这些角色映射到主领域中的用户来控制对管理控制台和管理 REST 端点的访问。可以创建多个超级用户，也可以创建只能管理特定领域的用户。

Global roles 全球角色

There are two realm-level roles in the `master` realm. These are:

在主领域中有两个领域级别的角色，它们是：

- `admin`
管理员
- `create-realm`
创造领域

Users with the `admin` role are superusers and have full access to manage any realm on the server. Users with the `create-realm` role are allowed to create new realms. They will be granted full access to any new realm they create.

具有管理员角色的用户是超级用户，可以完全访问服务器上的任何领域。具有创建领域角色的用户可以创建新的领域。它们将被授予对它们创建的任何新领域的完全访问权限。

Realm specific roles 特定领域的角色

Admin users within the `master` realm can be granted management privileges to one or more other realms in the system. Each realm in Keycloak is represented by a client in the `master` realm. The name of the client is `<realm name>-realm`. These clients each have client-level roles defined which define varying level of access to manage an individual realm.

主领域中的管理用户可以被授予系统中一个或多个其他领域的管理权限。Keycloak 中的每个领域都由主领域中的一个客户端表示。客户端的名称是 `< domain name >-domain`。这些客户端都定义了客户端级别的角色，这些角色定义了管理单个领域的不同访问级别。

The roles available are:

可供选择的角色包括：

- `view-realm`
视野领域
- `view-users`
浏览用户
- `view-clients`
浏览-客户
- `view-events`
观看事件
- `manage-realm`
管理领域管理领域
- `manage-users`
管理用户
- `create-client`
创建-客户端
- `manage-clients`
管理客户
- `manage-events`
管理事件
- `view-identity-providers`
视图-识别-提供者
- `manage-identity-providers`

管理-身份-提供者

- impersonation

模仿

Assign the roles you want to your users and they will only be able to use that specific part of the administration console.

将您想要的角色分配给您的用户，他们将只能使用管理控制台的特定部分。



Admins with the 管理员与 `manage-users` role will only be able to assign admin roles to users that they themselves have. So, if an admin has the Role 只能将管理员角色分配给他们自己拥有的用户。所以，如果管理员有 `manage-users` role but doesn't have the 角色，但没有 `manage-realm` role, they will not be able to assign this role. 角色，则它们将无法分配此角色

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/admin-console-permissions/per-realm.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/admin-console-permissions/per-realm.adoc)

Each realm has a dedicated Admin Console that can be accessed by going to the url `/admin/{realm-name}/console`. Users within that realm can be granted realm management permissions by assigning specific user role mappings.

每个领域都有一个专用的管理控制台，可以通过访问 url/Admin/{ domain-name }/sole 进行访问。通过分配特定的用户角色映射，可以授予该领域中的用户领域管理权限。

Each realm has a built-in client called `realm-management`. You can view this client by going to the `Clients` left menu item of your realm. This client defines client-level roles that specify permissions that can be granted to manage the realm.

每个领域都有一个称为领域管理的内置客户机。您可以通过转到领域的 `Clients` left 菜单项来查看此客户端。此客户端定义客户端级别的角色，这些角色指定可以授予管理领域的权限。

- `view-realm`

视野领域

- `view-users`

浏览用户

- `view-clients`

浏览-客户

- view-events
观看事件
- manage-realm
管理领域管理领域
- manage-users
管理用户
- create-client
创建-客户端
- manage-clients
管理客户
- manage-events
管理事件
- view-identity-providers
视图-识别-提供者
- manage-identity-providers
管理-身份-提供者
- impersonation
模仿

Assign the roles you want to your users and they will only be able to use that specific part of the administration console.

将您想要的角色分配给您的用户，他们将只能使用管理控制台的特定部分。

[Edit this section 编辑这部分](#)
(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/admin-console-permissions/fine-grain.adoc)

[Report an issue 报告一个问题](#)
(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/admin-console-permissions/fine-grain.adoc)

Fine Grain Admin Permissions is **Technology Preview** and is not fully supported. This feature is disabled by default.

细粒度管理权限是技术预览，不完全支持。默认情况下禁用此特性。



To enable start the server with `--features=preview` or `--features=admin-fine-grained-authz`

要启用以下命令启动服务器，请使用—— Features = 预览或—— Features = admin-fine-graines-authz

Sometimes roles like `manage-realm` or `manage-users` are too coarse grain and you want to create restricted admin accounts that have more fine grain permissions. Keycloak allows you to define and assign restricted access policies for managing a realm. Things like:

有时候，像管理领域或管理用户这样的角色过于粗粒度，您希望创建具有更细粒度权限的受限管理帐户。Key斗篷允许您定义和分配用于管理领域的受限访问策略。比如：

- Managing one specific client
管理一个特定的客户
- Managing users that belong to a specific group
管理属于特定组的用户
- Managing membership of a group
管理组的成员资格
- Limited user management.
有限的用户管理。
- Fine grain impersonation control
细粒度模拟控制
- Being able to assign a specific restricted set of roles to users.
能够为用户分配特定的受限角色集。
- Being able to assign a specific restricted set of roles to a composite role.
能够将特定的受限角色集分配给复合角色。
- Being able to assign a specific restricted set of roles to a client's scope.
能够将特定的受限角色集分配到客户端的作用域。
- New general policies for viewing and managing users, groups, roles, and clients.
用于查看和管理用户、组、角色和客户端的新通用策略。

There are some important things to note about fine grain admin permissions:

关于细粒度管理权限，有一些重要的事情需要注意：

- Fine grain admin permissions were implemented on top of [Authorization Services](https://www.keycloak.org/docs/latest/authorization_services/) (https://www.keycloak.org/docs/latest/authorization_services/). It is highly recommended that you read up on those features before diving into fine grain permissions.

在 AuthorizationServices 之上实现了细粒度管理权限。强烈建议您在深入研究细粒度权限之前阅读这些特性。

- Fine grain permissions are only available within dedicated admin consoles and admins defined within those realms. You cannot define cross-realm fine grain permissions.

细粒度权限只能在这些领域内定义的专用管理控制台和管理员中使用。不能定义跨领域细粒度权限。

- Fine grain permissions are used to grant additional permissions. You cannot override the default behavior of the built-in admin roles.

细粒度权限用于授予其他权限。不能重写内置管理角色的默认行为。

Managing one specific client 管理一个特定的客户

Let's look first at allowing an admin to manage one client and one client only. In our example, we have a realm called `test` and a client called `sales-application`. In the realm `test` we will give a user in that realm permission to only manage that application.

让我们首先看一下允许管理员只管理一个客户机和一个客户机。在我们的示例中，我们有一个名为 `test` 的领域和一个名为 `sales-application` 的客户机。在领域测试中，我们将授予该领域中的用户仅管理该应用程序的权限。



You cannot do cross realm fine grain permissions. Admins in the `master` realm are limited to the predefined admin roles defined in previous chapters.

不能执行跨域细粒度权限。领域仅限于前面章节中定义的预定义的管理角色

Permission setup 许可设置

The first thing we must do is login to the Admin Console so we can set up permissions for that client. We navigate to the management section of the client, we want to define fine-grain permissions for.

我们必须做的第一件事是登录到管理控制台，以便我们可以设置该客户端的权限。我们导航到客户端的管理部分，我们希望为。

Client management 客户管理

The screenshot shows the 'Clients' section of the Keycloak interface. On the left sidebar, under 'Clients', the 'sales-application' client is selected. The main content area shows the 'General Settings' tab for this client. The 'Client ID' is set to 'sales-application'. Under 'Access settings', 'Client authentication' is turned 'On' and 'Authorization' is turned 'Off'. In the 'Authentication flow' section, 'Standard flow' and 'Direct access grants' are checked, while 'Implicit flow' and 'Service accounts roles' are unchecked. A vertical sidebar on the right lists other tabs: 'General Settings', 'Access settings', 'Capability config', 'Login settings', and 'Logout settings'. At the top right, there is an 'Enabled' switch (set to 'Enabled') and an 'Action' dropdown menu.

You should see a tab menu item called **Permissions**. Click on that tab.

您应该会看到一个名为“权限”的选项卡菜单项。单击该选项卡。

Client permissions tab 客户端权限选项卡

The screenshot shows the 'Clients' section of the Keycloak interface. The 'sales-application' client is selected. The 'Permissions' tab is now active. A note states: 'Fine grained permissions for administrators that want to manage this client or apply roles defined by this client.' Below this, a 'Permissions enabled' switch is turned 'Off'. The left sidebar remains the same as the previous screenshot.

By default, each client is not enabled to do fine grain permissions. So turn the **Permissions Enabled** switch to on to initialize permissions.

默认情况下，不允许每个客户端执行细粒度权限。因此，打开“启用权限”开关以初始化权限。



If you turn the **Permissions Enabled** switch to off, it will delete any and all permissions you have defined for this client. 关闭时，它将删除您为此客户端定义的所有权限

Client permissions tab 客户端权限选项卡

The screenshot shows the Keycloak interface for managing clients. On the left, a sidebar lists various management options like Manage, Clients, Client scopes, etc. The 'Clients' option is selected. The main area shows the 'Clients' section for a client named 'sales-application' (OpenID Connect). A toggle switch labeled 'Enabled' is set to 'On'. Below it, a sub-section titled 'Permissions' is active, showing a message about fine-grained permissions for administrators. A 'Permissions enabled' switch is also set to 'On'. The 'Permission list' section contains a table with eight rows, each representing a permission scope name and its description. The columns are 'Scope-name' and 'Description'. The rows are:

Scope-name	Description
configure	Reduced management permissions for administrator. Cannot set scope, template, or protocol mappers.
manage	Policies that decide if an administrator can manage this client
map-roles	Policies that decide if an administrator can map roles defined by this client
map-roles-client-scope	Policies that decide if an administrator can apply roles defined by this client to the client scope of another client
map-roles-composite	Policies that decide if an administrator can apply roles defined by this client as a composite to another role
token-exchange	Policies that decide which clients are allowed exchange tokens for a token that is targeted to this client.
view	Policies that decide if an administrator can view this client

When you switch **Permissions Enabled** to on, it initializes various permission objects behind the scenes using [Authorization Services](https://www.keycloak.org/docs/latest/authorization_services/) (https://www.keycloak.org/docs/latest/authorization_services/). For this example, we're interested in the `manage` permission for the client. Clicking on that will redirect you to the permission that handles the `manage` permission for the client. All authorization objects are contained in the `realm-management` client's **Authorization** tab.

当您将“启用的权限”切换到“打开”时，它将使用授权服务在幕后初始化各种权限对象。对于本例，我们感兴趣的是客户端的管理权限。单击此选项将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领域管理客户端的 Authorization 选项卡中。

Client manage permission 客户端管理权限

The screenshot shows the 'Permission details' page for a client permission named 'manage.permission.client'. The 'Name' field is populated with 'manage.permission.client.0b0590a6-eea1-49f7-a73a-b1202b107be6'. The 'Resources' field contains 'client.resource.0b0590a6-eea1-49f7-a73a-b1202b107be6'. The 'Authorization scopes' field contains 'manage'. The 'Decision strategy' dropdown is set to 'Unanimous'. At the bottom, there are 'Save' and 'Cancel' buttons.

When first initialized the `manage` permission does not have any policies associated with it. You will need to create one by going to the policy tab. To get there fast, click on the `Authorization` link shown in the above image. Then click on the policies tab.

初始化管理权限时，该权限没有任何与之关联的策略。您需要通过转到 `policy` 选项卡来创建一个。要快速到达那里，请单击上图中显示的 `Authorization` 链接。然后单击 `policy` 选项卡。

There's a pull down menu on this page called `Create policy`. There's a multitude of policies you can define. You can define a policy that is associated with a role or a group or even define rules in JavaScript. For this simple example, we're going to create a `User Policy`.

这个页面上有一个下拉菜单，名为 `Create policy`。你可以定义很多政策。您可以定义与角色或组相关联的策略，甚至可以在 JavaScript 中定义规则。对于这个简单的示例，我们将创建一个用户策略。

User policy 用户策略

Test

Clients > Client details > Permission details

manage.permission.client.0b0590a6-eea1-49f7-a73a-b1202b107be6

Name *	manage.permission.client.0b0590a6-eea1-49f7-a73a-b1202b107be6
Description	
Resources *	client.resource.0b0590a6-eea1-49f7-a73a-b1202b107be6
Authorization scopes *	manage
Policies	sales-app-admin
Decision strategy	<input checked="" type="radio"/> Unanimous <input type="radio"/> Affirmative <input type="radio"/> Consensus
<button>Save</button> <button>Cancel</button>	

This policy will match a hard-coded user in the user database. In this case, it is the `sales-admin` user. We must then go back to the `sales-application` client's `manage` permission page and assign the policy to the permission object.

此策略将匹配用户数据库中的硬编码用户。在本例中，它是 `sales-admin` 用户。然后，我们必须返回销售应用程序客户端的管理权限页面，并将策略分配给权限对象。

Assign user policy 分配用户策略

Test

Clients > Client details > Permission details

manage.permission.client.0b0590a6-eea1-49f7-a73a-b1202b107be6

Name *	manage.permission.client.0b0590a6-eea1-49f7-a73a-b1202b107be6
Description	
Resources *	client.resource.0b0590a6-eea1-49f7-a73a-b1202b107be6
Authorization scopes *	manage
Policies	sales-app-admin
Decision strategy	<input checked="" type="radio"/> Unanimous <input type="radio"/> Affirmative <input type="radio"/> Consensus
<button>Save</button> <button>Cancel</button>	

The `sales-admin` user can now has permission to manage the `sales-application` client.

Sales-admin 用户现在可以拥有管理 sales-application 客户端的权限。

There's one more thing we have to do. Go to the `Role Mappings` tab and assign the `query-clients` role to the `sales-admin`.

我们还有一件事要做。转到“角色映射”选项卡，并将 `query-client` 角色分配给 `sales-admin`。

Assign query-clients 分配查询客户端

The screenshot shows the Keycloak Admin Console interface. In the top navigation bar, 'Users' is selected. Under 'User details' for the 'sales-admin' user, the 'Role mapping' tab is active. Below the tabs, there are search and filter fields: 'Search by name' and 'Filter by Origin' set to 'query'. A list of roles is displayed, with 'realm-management query-clients' being checked. At the bottom of the modal are 'Assign' and 'Cancel' buttons.

Why do you have to do this? This role tells the Admin Console what menu items to render when the `sales-admin` visits the Admin Console. The `query-clients` role tells the Admin Console that it should render client menus for the `sales-admin` user.

你为什么要这么做？这个角色告诉管理控制台当销售管理员访问管理控制台时要呈现哪些菜单项。Query-client 角色告诉管理控制台它应该为 sales-Admin 用户呈现客户端菜单。

IMPORTANT If you do not set the `query-clients` role, restricted admins like `sales-admin` will not see any menu options when they log into the Admin Console

重要信息: 如果不设置查询客户端角色，受限制的管理员(如 sales-Admin)在登录到管理控制台时将看不到任何菜单选项

Testing it out 测试一下

Next, we log out of the master realm and re-login to the dedicated admin console for the `test` realm using the `sales-admin` as a username. This is located under `/admin/test/console`.

接下来，我们注销主领域，并使用 sales-admin 作为用户名重新登录到测试领域的专用管理控制台。它位于/admin/test/sole 下。

Sales admin login 销售管理登录

Client ID	Type	Description	Home URL
sales-application	OpenID Connect	-	-

This admin is now able to manage this one client.

这个管理员现在可以管理这个客户端。

Restrict user role mapping 限制用户角色映射

Another thing you might want to do is to restrict the set of roles an admin is allowed to assign to a user. Continuing our last example, let's expand the permission set of the 'sales-admin' user so that he can also control which users are allowed to access this application. Through fine grain permissions, we can enable it so that the `sales-admin` can only assign roles that grant specific access to the `sales-application`. We can also restrict it so that the admin can only map roles and not perform any other types of user administration.

您可能想要做的另一件事情是限制管理员允许分配给用户的角色集。继续我们上一个示例，让我们展开“sales-admin”用户的权限集，以便他还可以控制哪些用户被允许访问该应用程序。通过细粒度权限，我们可以启用它，以便 sales-admin 只能分配授予销售应用程序特定访问权限的角色。我们还可以对其进行限制，以便管理员只能映射角色，而不能执行任何其他类型的用户管理。

The `sales-application` has defined three different client roles.

销售应用程序定义了三种不同的客户端角色。

Sales application roles 销售应用程序角色

The screenshot shows the Keycloak 'Clients' interface. On the left, a sidebar menu includes 'Manage', 'Clients', 'Client scopes', 'Realm roles', 'Users', 'Groups', 'Sessions', and 'Events'. The 'Clients' item is selected. In the main area, the title is 'Clients > Client details' for 'sales-application' (OpenID Connect). A search bar and a 'Create role' button are at the top. Below is a table of roles:

Role name	Composite
admin	False
leader-creator	False
viewLeads	False

We want the `sales-admin` user to be able to map these roles to any user in the system. The first step to do this is to allow the role to be mapped by the admin. If we click on the `viewLeads` role, you'll see that there is a `Permissions` tab for this role.

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。这样做的第一步是允许管理员映射角色。如果我们单击 viewLeads 角色，您将看到该角色有一个 Permission 选项卡。

View leads role permission tab 视图引导角色权限选项卡

The screenshot shows the 'viewLeads' role details page. The sidebar is identical to the previous screenshot. The main area shows the 'viewLeads' role with a 'Role name' of 'viewLeads'. There is a 'Description' field which is empty. At the bottom are 'Save' and 'Revert' buttons. The 'Permissions' tab is highlighted in blue.

If we click on that tab and turn the `Permissions Enabled` on, you'll see that there are a number of actions we can apply policies to.

如果我们单击该选项卡并打开“启用的权限”，您将看到有许多操作可以应用策略。

View leads permissions 查看引导权限

Test

Manage

Clients

Client scopes

Realm roles

Users

Groups

Sessions

Events

Configure

Realm settings

Authentication

Identity providers

Clients > Client details > Role details

viewLeads

Action

Details Attributes Users in role Permissions

Permissions

Determines if fine grained permissions are enabled for managing this role. Disabling will delete all current permissions that have been set up.

Permissions enabled On

Permission list

Edit the permission list by clicking the scope-name. It then redirects to the permission details page of the client named **realm-management**

Scope-name	Description	⋮
map-role	Policies that decide if an administrator can map this role to a user or group	⋮
map-role-client-scope	Policies that decide if an administrator can apply this role to the client scope of a client	⋮
map-role-composite	Policies that decide if an administrator can apply this role as a composite to another role	⋮

The one we are interested in is `map-role`. Click on this permission and add the same User Policy that was created in the earlier example.

我们感兴趣的是 `map-role`。单击此权限并添加与前面示例中创建的相同的用户策略。

Map-roles permission 映射角色权限

Test

Manage

Clients

Client scopes

Realm roles

Users

Groups

Sessions

Events

Configure

Realm settings

Authentication

Identity providers

User federation

Clients > Client details > Permission details

map-roles.permission.client.3c7a3696-25d9-4a28-a863-5dcf5dcabd62

Action

Name *

Description

Apply to resource type Off

Resources

Authorization scopes

Policies

Decision strategy Unanimous
 Affirmative
 Consensus

Save Cancel

What we've done is say that the `sales-admin` can map the `viewLeads` role. What we have not done is specify which users the admin is allowed to map this role too. To do that we must go to the `Users` section of the admin console for this realm. Clicking on the `Users` left menu item brings us to the users interface of the realm. You should see a `Permissions` tab. Click on that and enable it.

我们所做的就是说销售管理员可以映射 `viewLeads` 角色。我们还没有做的是指定管理员也可以映射这个角色的用户。为此，我们必须转到此领域的管理控制台的 `Users` 部分。单击 `Users` left 菜单项将我们带到领域的用户界面。您应该会看到一个“权限”选项卡。点击并启用它。

Users permissions 用户权限

Scope-name	Description	⋮
impersonate	Policies that decide if administrator can impersonate other users	⋮
manage	Policies that decide if an administrator can manage all users in the realm	⋮
manage-group-membership	Policies that decide if an administrator can manage group membership for all users in the realm. This is used in conjunction with specific group policy	⋮
map-roles	Policies that decide if administrator can map roles for all users	⋮
user-impersonated	Policies that decide which users can be impersonated. These policies are applied to the user being impersonated.	⋮
view	Policies that decide if an administrator can view all users in realm	⋮

The permission we are interested in is `map-roles`. This is a restrictive policy in that it only allows admins the ability to map roles to a user. If we click on the `map-roles` permission and again add the User Policy we created for this, our `sales-admin` will be able to map roles to any user.

我们感兴趣的权限是 `map-role`。这是一个限制性策略，因为它只允许管理员将角色映射到用户。如果我们单击 `map-role` 权限并再次添加为此创建的用户策略，则我们的 `sales-admin` 将能够将角色映射到任何用户。

The last thing we have to do is add the `view-users` role to the `sales-admin`. This will allow the admin to view users in the realm he wants to add the `sales-application` roles to.

我们必须做的最后一件事是将 `view-users` 角色添加到 `sales-admin`。这将允许管理员在他想要添加销售应用程序角色的领域中查看用户。

Add view-users 添加视图用户

The screenshot shows the Keycloak Admin Console interface. At the top, there's a navigation bar with 'Users' > 'User details'. Below it, the user 'sales-admin' is selected. The main menu has tabs: Details, Attributes, Credentials, Role mapping (which is currently active), Groups, Consents, and Identity provider links. In the center, a modal window titled 'Assign roles to sales-admin account' is displayed. This modal contains a search bar and a table with columns 'Name' and 'Description'. The table lists several roles, each with a checkbox. One role, 'realm-management view-users', has a checked checkbox. At the bottom of the modal are 'Assign' and 'Cancel' buttons.

Testing it out 测试一下

Next, we log out of the master realm and re-login to the dedicated admin console for the `test` realm using the `sales-admin` as a username. This is located under `/admin/test/console`.

接下来，我们注销主领域，并使用 `sales-admin` 作为用户名重新登录到测试领域的专用管理控制台。它位于`/admin/test/sole`下。

You will see that now the `sales-admin` can view users in the system. If you select one of the users you'll see that each user detail page is read only, except for the `Role Mappings` tab. Going to this tab you'll find that there are no `Available` roles for the admin to map to the user except when we browse the `sales-application` roles.

您将看到现在销售管理员可以查看系统中的用户。如果选择其中一个用户，您将看到除了“角色映射”选项卡之外，每个用户详细信息页都是只读的。进入此选项卡后，您会发现除了浏览销售应用程序角色之外，管理员没有可映射到用户的可用角色。

Add viewleads 添加视角

The screenshot shows the Keycloak admin console interface. On the left, there's a sidebar with various management tabs like Master, Manage, Clients, etc. The main area shows a user named 'salesman'. Under the 'User details' tab, the 'Role mapping' tab is selected. A modal window titled 'Assign roles to salesman account' is displayed, showing a list of roles from the 'sales-application' client. One role, 'viewLeads', is selected and mapped to the 'sales-admin' user. The modal also has 'Assign' and 'Cancel' buttons.

We've only specified that the `sales-admin` can map the `viewLeads` role.

我们只指定 sales-admin 可以映射 viewLeads 角色。

Per client map-roles shortcut 每个客户端映射角色的快捷方式

It would be tedious if we had to do this for every client role that the `sales-application` published. To make things easier, there's a way to specify that an admin can map any role defined by a client. If we log back into the admin console to our master realm admin and go back to the `sales-application` permissions page, you'll see the `map-roles` permission.

如果我们必须为销售应用程序发布的每个客户角色执行此操作，那将非常单调乏味。为了使事情变得更容易，有一种方法可以指定管理员可以映射客户端定义的任何角色。如果我们重新登录到管理控制台到我们的主领域管理员，并返回到销售应用程序权限页面，您将看到 map-role 权限。

Client map-roles permission 客户端映射-角色权限

The screenshot shows the Keycloak admin interface under the 'Clients' section for a client named 'sales-application'. The 'Permissions' tab is active. A table lists various permission scopes and their descriptions:

Scope-name	Description
configure	Reduced management permissions for administrator. Cannot set scope, template, or protocol mappers.
manage	Policies that decide if an administrator can manage this client
map-roles	Policies that decide if an administrator can map roles defined by this client
map-roles-client-scope	Policies that decide if an administrator can apply roles defined by this client to the client scope of another client
map-roles-composite	Policies that decide if an administrator can apply roles defined by this client as a composite to another role
token-exchange	Policies that decide which clients are allowed exchange tokens for a token that is targeted to this client.
view	Policies that decide if an administrator can view this client

If you grant access to this particular permission to an admin, that admin will be able map any role defined by the client.

如果您授予管理员对此特定权限的访问权，该管理员将能够映射客户端定义的任何角色。

Full list of permissions 完整的权限列表

You can do a lot more with fine grain permissions beyond managing a specific client or the specific roles of a client. This chapter defines the whole list of permission types that can be described for a realm.

除了管理特定的客户机或客户机的特定角色之外，您还可以使用细粒度权限执行更多操作。本章定义了可以为领域描述的权限类型的完整列表。

Role 角色

When going to the `Permissions` tab for a specific role, you will see these permission types listed.

当转到特定角色的“权限”选项卡时，您将看到列出了这些权限类型。

map-role 地图角色

Policies that decide if an admin can map this role to a user. These policies only specify that the role can be mapped to a user, not that the admin is allowed to perform user role mapping tasks. The admin will also have to have manage or role mapping permissions. See [Users Permissions](#) for more information.

决定管理员是否可以将此角色映射到用户的策略。这些策略只指定可以将角色映射到用户，而不允许管理员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。有关更多信息，请参见用户权限。

map-role-composite 地图-角色-复合

Policies that decide if an admin can map this role as a composite to another role. An admin can define roles for a client if he has to manage permissions for that client but he will not be able to add composites to those roles unless he has the `map-role-composite` privileges for the role he wants to add as a composite.

决定管理员是否可以将此角色作为复合角色映射到另一个角色的策略。如果管理员必须管理客户端的权限，那么他可以为客户端定义角色，但是他不能向这些角色添加组合，除非他拥有要作为组合添加的角色的 map-role-Composite 特权。

map-role-client-scope

Policies that decide if an admin can apply this role to the scope of a client. Even if the admin can manage the client, he will not have permission to create tokens for that client that contain this role unless this privilege is granted.

决定管理员是否可以将此角色应用于客户端范围的策略。即使管理员可以管理客户端，他也没有权限为包含此角色的客户端创建令牌，除非授予此权限。

Client 客户

When going to the `Permissions` tab for a specific client, you will see these permission types listed.

当转到特定客户端的“权限”选项卡时，您将看到列出了这些权限类型。

view 风景

Policies that decide if an admin can view the client's configuration.

决定管理员是否可以查看客户端配置的策略。

manage 管理

Policies that decide if an admin can view and manage the client's configuration. There are some issues with this in that privileges could be leaked unintentionally. For example, the admin could define a protocol mapper that hardcoded a role even if the admin does not have privileges to map the role to the client's scope. This is currently the limitation of protocol mappers as they don't have a way to assign individual permissions to them like roles do.

决定管理员是否可以查看和管理客户端配置的策略。这样做有一些问题，因为特权可能会无意中泄露。例如，管理员可以定义一个硬编码角色的协议映射器，即使管理员没有将角色映射到客户机范围的特权。这是目前协议映射程序的局限性，因为它们没有像角色那样分配个人权限的方法。

configure 配置

Reduced set of privileges to manage the client. It is like the `manage` scope except the admin is not allowed to define protocol mappers, change the client template, or the client's scope.

减少了管理客户端的权限集。它类似于管理范围，只是管理员不允许定义协议映射器、更改客户端模板或客户端范围。

map-roles 地图角色

Policies that decide if an admin can map any role defined by the client to a user. This is a shortcut, easy-of-use feature to avoid having to define policies for each and every role defined by the client.

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、易用的特性，可以避免为客户机定义的每个角色定义策略。

map-roles-composite 映射-角色-合成

Policies that decide if an admin can map any role defined by the client as a composite to another role. This is a shortcut, easy-of-use feature to avoid having to define policies for each and every role defined by the client.

策略，用于确定管理员是否可以将客户端定义为组合的任何角色映射到另一个角色。这是一个快捷、易用的特性，可以避免为客户端定义的每个角色定义策略。

map-roles-client-scope Map-role-client-scope

Policies that decide if an admin can map any role defined by the client to the scope of another client. This is a shortcut, easy-of-use feature to avoid having to define policies for each and every role defined by the client.

策略，决定管理员是否可以将客户端定义的任何角色映射到另一个客户端的范围。这是一个快捷、易用的特性，可以避免为客户机定义的每个角色定义策略。

Users 用户

When going to the `Permissions` tab for all users, you will see these permission types listed.

当转到所有用户的“权限”选项卡时，您将看到列出了这些权限类型。

view 风景

Policies that decide if an admin can view all users in the realm.

决定管理员是否可以查看域中所有用户的策略。

manage 管理

Policies that decide if an admin can manage all users in the realm. This permission grants the admin the privilege to perform user role mappings, but it does not specify which roles the admin is allowed to map. You'll need to define the privilege for each role you want the admin to be able to map.

策略，用于确定管理员是否可以管理域中的所有用户。此权限授予管理员执行用户角色映射的权限，但是没有指定允许管理员映射哪些角色。您需要为希望管理员能够映射的每个角色定义特权。

map-roles 地图角色

This is a subset of the privileges granted by the `manage` scope. In this case the admin is only allowed to map roles. The admin is not allowed to perform any other user management operation. Also, like `manage`, the roles that the admin is allowed to apply must be specified per role or per set of roles if dealing with client roles.

这是由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。管理员不允许执行任何其他用户管理操作。此外，与管理一样，如果要处理客户端角色，则必须为每个角色或每组角色指定管理员允许应用的角色。

manage-group-membership 管理-团体-成员资格

Similar to `map-roles` except that it pertains to group membership: which groups a user can be added or removed from. These policies just grant the admin permission to manage group membership, not which groups the admin is allowed to manage membership for. You'll have to specify policies for each group's `manage-members` permission.

类似于 `map-role`，只不过它与组成员关系有关：用户可以从哪个组中添加或删除。这些策略只授予管理员管理组成员资格的权限，而不授予管理员管理组成员资格的权限。您必须为每个组的管理成员权限指定策略。

impersonate 假扮

Policies that decide if the admin is allowed to impersonate other users. These policies are applied to the administrator's attributes and role mappings.

决定是否允许管理员模拟其他用户的策略。这些策略应用于管理员的属性和角色映射。

user-impersonated 用户模拟

Policies that decide which users can be impersonated. These policies will be applied to the user being impersonated. For example, you might want to define a policy that will forbid anybody from impersonating a user that has admin privileges.

决定可以模拟哪些用户的策略。这些策略将应用于被模拟的用户。例如，您可能希望定义一个策略，该策略将禁止任何人模拟具有管理特权的用户。

Group 小组

When going to the `Permissions` tab for a specific group, you will see these permission types listed.

当转到特定组的“权限”选项卡时，您将看到列出了这些权限类型。

view 风景

Policies that decide if the admin can view information about the group.

策略，用于确定管理员是否可以查看有关组的信息。

manage 管理

Policies that decide if the admin can manage the configuration of the group.

决定管理员是否可以管理组的配置的策略。

view-members 观众

Policies that decide if the admin can view the user details of members of the group.

策略，用于确定管理员是否可以查看组成员的用户详细信息。

manage-members 管理会员

Policies that decide if the admin can manage the users that belong to this group.

策略，用于确定管理员是否可以管理属于此组的用户。

manage-membership 管理会员制

Policies that decide if an admin can change the membership of the group. Add or remove members from the group.

决定管理员是否可以更改组的成员资格的策略。从组中添加或删除成员。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/assembly-managing-clients.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/assembly-managing-clients.adoc)

Clients are entities that can request authentication of a user. Clients come in two forms. The first type of client is an application that wants to participate in single-sign-on. These clients just want Keycloak to provide security for them. The other type of client is one that is requesting an access token so that it can invoke other services on behalf of the authenticated user. This section discusses various aspects around configuring clients and various ways to do it.

客户端是可以请求用户身份验证的实体。客户有两种形式。第一种类型的客户端是希望参与单点登录的应用程序。这些客户只是想让“钥匙斗篷”为他们提供安全性。另一种类型的客户端请求访问令牌，以便它可以代表经过身份验证的用户调用其他服务。本节讨论围绕配置客户端的各个方面以及实现这一点的各种方法。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/clients/assembly-client-oidc.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/clients/assembly-client-oidc.adoc)

OpenID Connect is the recommended protocol to secure applications. It was designed from the ground up to be web friendly and it works best with HTML5/JavaScript applications.

OpenIDConnect 是保护应用程序安全的推荐协议。它从头到尾都是为了网络友好而设计的，并且它在 HTML5/JavaScript 应用程序中工作得最好。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/clients/oidc/proc-creating-oidc-client.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/clients/oidc/proc-creating-oidc-client.adoc)

To protect an application that uses the OpenID connect protocol, you create a client.

若要保护使用 OpenID 连接协议的应用程序，请创建一个客户端。

Procedure 程序

1. Click **Clients** in the menu.

单击菜单中的 Clients。

2. Click **Create client**.

单击 Create client。

Create client 创建客户端

The screenshot shows the 'Create client' dialog in Keycloak. On the left is a sidebar with options: Master, Manage, Clients (which is selected), Client scopes, Realm roles, Users, Groups, Sessions, Events, Configure, Realm settings, and Authentication. The main area has a breadcrumb 'Clients > Create client' and a title 'Create client'. A sub-header says 'Clients are applications and services that can request authentication of a user.' Below this is a step indicator '1 General Settings'. On the right, there are fields for 'Client type' (set to 'OpenID Connect'), 'Client ID' (marked with a red asterisk), 'Name', and 'Description'. At the bottom are 'Next', 'Back', and 'Cancel' buttons.

3. Leave Client type set to **OpenID Connect**.

将客户端类型设置为 OpenID 连接。

4. Enter a Client ID.

输入客户端 ID。

This ID is an alphanumeric string that is used in OIDC requests and in the Keycloak database to identify the client.

这个 ID 是一个字母数字字符串，用于 OIDC 请求和钥匙斗篷数据库中来标识客户端。

5. Supply a Name for the client.

为客户端提供一个 Name。

If you plan to localize this name, set up a replacement string value. For example, a string value such as \${myapp}. See the [Server Developer Guide](#) (https://www.keycloak.org/docs/latest/server_development/) for more information.

如果计划本地化此名称，请设置替换字符串值。例如，一个字符串值，如 \${ myapp }。有关更多信息，请参见服务器开发人员指南。

6. Click Save.

单击“保存”。

This action creates the client and bring you to the **Settings** tab, where you can perform Basic configuration.

此操作将创建客户端并将您带到 Settings 选项卡，您可以在其中执行基本配置。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/clients/oidc/con-basic-settings.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/clients/oidc/con-basic-settings.adoc)

The **Settings** tab includes many options to configure this client.

“设置”选项卡包含许多配置此客户端的选项。

Settings tab 设置标签

The screenshot shows the Keycloak administration interface. On the left, a sidebar menu includes 'Master', 'Manage', 'Clients' (which is selected), 'Client scopes', 'Realm roles', 'Users', 'Groups', 'Sessions', 'Events', 'Configure', 'Realm settings', 'Authentication', 'Identity providers', and 'User federation'. The main content area is titled 'Client details' for 'Test' (OpenID Connect). It shows 'General Settings' with 'Client ID' set to 'test', 'Name' and 'Description' fields, and a toggle for 'Always display in console' which is off. A sidebar on the right lists 'Jump to section' options: General Settings, Access settings, Capability config, Login settings, and Logout settings. At the bottom are 'Save' and 'Revert' buttons.

General Settings 一般設定

Client ID 客户名

The alphanumeric ID string that is used in OIDC requests and in the Keycloak database to identify the client.

用于 OIDC 请求以及用于钥匙斗篷数据库以标识客户端的字母数字 ID 字符串。

Name 姓名

The name for the client in Keycloak UI screen. To localize the name, set up a replacement string value. For example, a string value such as \${myapp}. See the [Server Developer Guide](#) (https://www.keycloak.org/docs/latest/server_development/) for more information.

键斗 UI 屏幕中的客户端名称。若要本地化名称，请设置替换字符串值。例如，一个字符串值，如 \${myapp}。有关更多信息，请参见服务器开发人员指南。

Description 描述

The description of the client. This setting can also be localized.

客户端的描述。此设置也可以本地化。

Always Display in Console 始终在控制台中显示

Always list this client in the Account Console even if this user does not have an active session.

始终在帐户控制台中列出此客户端，即使此用户没有活动会话。

Access Settings 访问设置

Root URL 根网址

If Keycloak uses any configured relative URLs, this value is prepended to them.

如果“钥匙斗篷”使用任何已配置的相对 URL，则为它们预先设置此值。

Home URL 主页网址

Provides the default URL for when the auth server needs to redirect or link back to the client.

为身份验证服务器需要重定向或链接回客户端时提供默认 URL。

Valid Redirect URIs 有效重定向 URI

Required field. Enter a URL pattern and click + to add and - to remove existing URLs and click Save.

You can use wildcards at the end of the URL pattern. For example `http://host.com/*`

需要的领域。输入 URL 模式并单击 + 添加和-删除现有 URL 并单击 Save。您可以在 URL 模式的末尾使用通配符。比如 `http://host.com/*`

Exclusive redirect URL patterns are typically more secure. See Unspecific Redirect URIs for more information.

独占重定向 URL 模式通常更安全。有关更多信息，请参见非特定重定向 URI。

Web Origins 网站起源

Enter a URL pattern and click + to add and - to remove existing URLs. Click Save.

输入一个 URL 模式，单击 + 添加和-删除现有的 URL。单击 Save。

This option handles [Cross-Origin Resource Sharing \(CORS\)](https://fetch.spec.whatwg.org/) (<https://fetch.spec.whatwg.org/>). If browser JavaScript attempts an AJAX HTTP request to a server whose domain is different from the one that the JavaScript code came from, the request must use CORS. The server must handle CORS requests, otherwise the browser will not display or allow the request to be processed. This protocol protects against XSS, CSRF, and other JavaScript-based attacks.

此选项可处理跨来源资源共享(CORS)。如果浏览器 JavaScript 尝试向其域与 JavaScript 代码所来自的域不同的服务器发出 AJAX HTTP 请求，则请求必须使用 CORS。服务器必须处理 CORS 请求，否则浏览器将不显示或不允许处理请求。此协议可防止 XSS、CSRF 和其他基于 JavaScript 的攻击。

Domain URLs listed here are embedded within the access token sent to the client application. The client application uses this information to decide whether to allow a CORS request to be invoked on it. Only Keycloak client adapters support this feature. See [Securing Applications and Services Guide](https://www.keycloak.org/docs/latest/securing_apps/) (https://www.keycloak.org/docs/latest/securing_apps/) for more information.

这里列出的域 URL 嵌入在发送到客户端应用程序的访问令牌中。客户机应用程序使用这些信息来决定是否允许对其调用 CORS 请求。只有钥匙斗篷客户端适配器支持此特性。有关更多信息，请参见保护应用程序和服务指南。

Admin URL 管理网址

Callback endpoint for a client. The server uses this URL to make callbacks like pushing revocation policies, performing backchannel logout, and other administrative operations. For Keycloak servlet adapters, this URL can be the root URL of the servlet application. For more information, see [Securing Applications and Services Guide](https://www.keycloak.org/docs/latest/securing_apps/) (https://www.keycloak.org/docs/latest/securing_apps/).

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、执行反向通道注销和其他管理操作。对于 Keycloak servlet 适配器，此 URL 可以是 servlet 应用程序的根 URL。有关详细信息，请参阅保护应用程序和服务指南。

Capability Config 能力配置

Client authentication 客户身份验证

The type of OIDC client.

OIDC 客户端的类型。

- *ON*

开始

For server-side clients that perform browser logins and require client secrets when making an Access Token Request. This setting should be used for server-side applications.

用于执行浏览器登录并在发出访问令牌请求时需要客户端机密的服务器端客户端。此设置应用于服务器端应用程序。

- *OFF*

关掉

For client-side clients that perform browser logins. As it is not possible to ensure that secrets can be kept safe with client-side clients, it is important to restrict access by configuring correct redirect URIs.

用于执行浏览器登录的客户端客户端。由于不可能确保客户端客户机能够保密，因此通过配置正确的重定向 URI 来限制访问非常重要。

Authorization 授权

Enables or disables fine-grained authorization support for this client.

启用或禁用此客户端的细粒度授权支持。

Standard Flow 标准流程

If enabled, this client can use the OIDC Authorization Code Flow.

如果启用，此客户端可以使用 OIDC 授权代码流。

Direct Access Grants 直接进入补助金

If enabled, this client can use the OIDC Direct Access Grants.

如果启用，此客户端可以使用 OIDC 直接访问授权。

Implicit Flow 隐性心流

If enabled, this client can use the OIDC Implicit Flow.

如果启用，此客户端可以使用 OIDC 隐式流。

Service account roles 服务帐户角色

If enabled, this client can authenticate to Keycloak and retrieve access token dedicated to this client.

In terms of OAuth2 specification, this enables support of `Client Credentials Grant` for this client.

如果启用，该客户端可以向 Keycloak 验证并检索专用于该客户端的访问令牌。根据 OAuth2 规范，这使得该客户端能够支持客户端凭据授予。

Auth 2.0 Device Authorization Grant Auth 2.0设备授权授权

If enabled, this client can use the OIDC Device Authorization Grant.

如果启用，此客户端可以使用 OIDC 设备授权授权。

OIDC CIBA Grant CIBA 拨款

If enabled, this client can use the OIDC Client Initiated Backchannel Authentication Grant.

如果启用，此客户端可以使用 OIDC 客户端启动的反向通道身份验证授权。

Login settings 登录设置

Login theme 登录主题

A theme to use for login, OTP, grant registration, and forgotten password pages.

用于登录、OTP、授予注册和忘记密码页的主题。

Consent required 需要同意

If enabled, users have to consent to client access.

如果启用，用户必须同意客户端访问。

For client-side clients that perform browser logins. As it is not possible to ensure that secrets can be kept safe with client-side clients, it is important to restrict access by configuring correct redirect URIs.

用于执行浏览器登录的客户端客户端。由于不可能确保客户端客户机能够保密，因此通过配置正确的重定向 URI 来限制访问非常重要。

Display client on screen 在屏幕上显示客户端

This switch applies if `Consent Required` is Off.

如果“必须同意”关闭，则此开关适用。

- *Off*

关掉

The consent screen will contain only the consents corresponding to configured client scopes.

同意屏幕将只包含与配置的客户端范围相对应的同意。

- *On*

开始

There will be also one item on the consent screen about this client itself.

同意屏幕上还会有一条关于这个客户端本身的内容。

Client consent screen text 客户同意屏幕文本

Applies if Consent required and Display client on screen are enabled. Contains the text that will be on the consent screen about permissions for this client.

如果启用了同意和屏幕上显示客户端，则应用。包含将在同意屏幕上显示的有关此客户端权限的文本。

Logout settings 注销设置

Front channel logout 前台注销

If **Front Channel Logout** is enabled, the application should be able to log out users through the front channel as per [OpenID Connect Front-Channel Logout](#) (https://openid.net/specs/openid-connect-frontchannel-1_0.html) specification. If enabled, you should also provide the `Front-Channel Logout URL`.

如果启用了前端通道注销，应用程序应该能够按照 OpenID 连接前端通道注销规范通过前端通道注销用户。如果启用，还应该提供前端通道注销 URL。

Front-channel logout URL 前端通道注销网址

URL that will be used by Keycloak to send logout requests to clients through the front-channel.

将被钥匙斗篷用来通过前端通道向客户端发送注销请求的 URL。

Backchannel logout URL 反向通道注销 URL

URL that will cause the client to log itself out when a logout request is sent to this realm (via `end_session_endpoint`). If omitted, no logout requests are sent to the client.

当一个注销请求被发送到这个领域(通过 `end_session_endpoint`)时，将导致客户端注销自身的 URL。如果省略，则不向客户端发送注销请求。

Backchannel logout session required 需要反向通道注销会话

Specifies whether a session ID Claim is included in the Logout Token when the **Backchannel Logout URL** is used.

指定使用 Backchannel Logout URL 时是否在注销令牌中包含会话 ID 声明。

Backchannel logout revoke offline sessions 反向通道注销撤销脱机会话

Specifies whether a revoke_offline_access event is included in the Logout Token when the Backchannel Logout URL is used. Keycloak will revoke offline sessions when receiving a Logout Token with this event.

指定在使用 Backchannel Logout URL 时，Logout 令牌中是否包含 revoke _ off _ access 事件。当收到带有此事件的注销令牌时，键隐藏将撤销脱机会话。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/clients/oidc/con-advanced-settings.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/clients/oidc/con-advanced-settings.adoc)

After completing the fields on the **Settings** tab, you can use the other tabs to perform advanced configuration. For example, you can use the **Permissions** and **Roles** tabs to configure fine-grained authentication for administrators. See Fine grain admin permissions. Also, see the remaining sections in this chapter for other capabilities.

完成“设置”选项卡上的字段后，可以使用其他选项卡执行高级配置。例如，可以使用“权限”和“角色”选项卡为管理员配置细粒度身份验证。请参阅细粒度管理权限。此外，请参阅本章其他章节了解其他功能。

Advanced tab 高级标签

When you click the **Advanced** tab, additional fields are displayed. For details on a specific field, click the question mark icon for that field. However, certain fields are described in detail in this section.

单击“高级”选项卡时，将显示其他字段。有关特定字段的详细信息，请单击该字段的问号图标。但是，本节将详细描述某些字段。

Fine grain OpenID Connect configuration 细粒度 OpenID 连接配置

Logo URL

标志网址

URL that references a logo for the Client application.

引用客户端应用程序徽标的 URL。

Policy URL

政策网址

URL that the Relying Party Client provides to the End-User to read about how the profile data will be used.

依赖方客户端提供给最终用户的 URL，用于读取将如何使用配置文件数据。

Terms of Service URL

服务条款网址

URL that the Relying Party Client provides to the End-User to read about the Relying Party's terms of service.

信赖方客户端提供给最终用户的 URL，用于阅读信赖方的服务条款。

Signed and Encrypted ID Token Support

签名和加密的 ID 令牌支持

Keycloak can encrypt ID tokens according to the [Json Web Encryption \(JWE\)](#) (<https://datatracker.ietf.org/doc/html/rfc7516>) specification. The administrator determines if ID tokens are encrypted for each client.

密钥斗篷可以根据 JsonWebEncryption (JWE) 规范对 ID 令牌进行加密。管理员确定是否为每个客户端加密了 ID 令牌。

The key used for encrypting the ID token is the Content Encryption Key (CEK). Keycloak and a client must negotiate which CEK is used and how it is delivered. The method used to determine the CEK is the Key Management Mode. The Key Management Mode that Keycloak supports is Key Encryption.

用于加密 ID 令牌的密钥是 ContentEncryptionKey (CEK)。钥匙斗篷和客户端必须协商使用哪个 CEK 以及如何交付。确定 CEK 的方法是密钥管理模式。密钥斗篷支持的密钥管理模式是密钥加密。

In Key Encryption:

密码匙加密:

1. The client generates an asymmetric cryptographic key pair.

客户端生成一个不对称的单密钥对。

2. The public key is used to encrypt the CEK.

公钥用于加密 CEK。

3. Keycloak generates a CEK per ID token

密钥斗篷为每个 ID 标记生成一个 CEK

4. Keycloak encrypts the ID token using this generated CEK

密钥斗篷使用这个生成的 CEK 加密 ID 令牌

5. Keycloak encrypts the CEK using the client's public key.

密钥斗篷使用客户端的公钥对 CEK 进行加密。

6. The client decrypts this encrypted CEK using their private key

客户端使用其私钥解密这个加密的 CEK

7. The client decrypts the ID token using the decrypted CEK.

客户端使用解密的 CEK 解密 ID 令牌。

No party, other than the client, can decrypt the ID token.

除了客户端之外，任何一方都不能解密 ID 令牌。

The client must pass its public key for encrypting CEK to Keycloak. Keycloak supports downloading public keys from a URL provided by the client. The client must provide public keys according to the [Json Web Keys \(JWK\)](#) (<https://datatracker.ietf.org/doc/html/rfc7517>) specification.

客户端必须传递其公钥，以便将 CEK 加密到 Keycloak。Key斗篷支持从客户端提供的 URL 下载公钥。客户端必须根据 JsonWebKeys (JWK) 规范提供公钥。

The procedure is:

程序是：

1. Open the client's **Keys** tab.

打开客户端的“键”选项卡。

2. Toggle JWKS URL to ON.

将 JWKS URL 切换到 ON。

3. Input the client's public key URL in the JWKS URL textbox.

在 JWKS URL 文本框中输入客户端的公钥 URL。

Key Encryption's algorithms are defined in the [Json Web Algorithm \(JWA\)](#)

(<https://datatracker.ietf.org/doc/html/rfc7518#section-4.1>) specification. Keycloak supports:

密钥加密的算法是在 Json Web 算法 (JWA) 规范中定义的：

- RSAES-PKCS1-v1_5(RSA1_5)

RSAES-PKCS1-v1_5(RSA1_5)

- RSAES OAEP using default parameters (RSA-OAEP)

使用默认参数的 RSAES OAEP (RSA-OAEP)

- RSAES OAEP 256 using SHA-256 and MFG1 (RSA-OAEP-256)

使用 SHA-256 和 MFG1(RSA-OAEP-256) 的 RSAES OAEP 256

The procedure to select the algorithm is:

选择算法的程序是：

1. Open the client's **Advanced** tab.

打开客户端的 Advanced 选项卡。

2. Open **Fine Grain OpenID Connect Configuration**.

开放式细粒度 OpenID 连接配置。

3. Select the algorithm from **ID Token Encryption Content Encryption Algorithm** pulldown menu.

从 ID 令牌加密内容加密算法下拉菜单中选择该算法。

Open ID Connect Compatibility Modes 开放 ID 连接兼容模式

This section exists for backward compatibility. Click the question mark icons for details on each field.

此部分适用于向下兼容。请单击问号图标查看每个字段的详细信息。

OAuth 2.0 Mutual TLS Certificate Bound Access Tokens Enabled

OAuth 2.0 相互 TLS 证书绑定访问令牌启用

Mutual TLS binds an access token and a refresh token together with a client certificate, which is exchanged during a TLS handshake. This binding prevents an attacker from using stolen tokens.

相互 TLS 将访问令牌和刷新令牌与客户端证书绑定在一起，客户端证书在 TLS 握手过程中进行交换。此绑定可防止攻击者使用被窃取的令牌。

This type of token is a holder-of-key token. Unlike bearer tokens, the recipient of a holder-of-key token can verify if the sender of the token is legitimate.

这种类型的令牌是密钥持有者令牌。与持有者令牌不同，持有者令牌的接收者可以验证令牌的发送者是否合法。

If this setting is on, the workflow is:

如果打开此设置，则工作流为：

1. A token request is sent to the token endpoint in an authorization code flow or hybrid flow.

令牌请求被发送到授权代码流或混合流中的令牌端点。

2. Keycloak requests a client certificate.

密钥斗篷请求客户端证书。

3. Keycloak receives the client certificate.

密钥斗篷接收客户端证书。

4. Keycloak successfully verifies the client certificate.

密钥斗篷成功验证了客户端证书。

If verification fails, Keycloak rejects the token.

如果验证失败，Keycloak 拒绝令牌。

In the following cases, Keycloak will verify the client sending the access token or the refresh token:

在以下情况下，Keycloak 将验证发送访问令牌或刷新令牌的客户端：

- A token refresh request is sent to the token endpoint with a holder-of-key refresh token.

令牌刷新请求通过一个键持有者刷新令牌发送到令牌端点。

- A UserInfo request is sent to UserInfo endpoint with a holder-of-key access token.

UserInfo 请求通过一个密钥持有者访问令牌发送到 UserInfo 端点。

- A logout request is sent to Logout endpoint with a holder-of-key refresh token.

注销请求通过一个键持有者刷新令牌发送到注销端点。

See [Mutual TLS Client Certificate Bound Access Tokens](#)

(<https://datatracker.ietf.org/doc/html/draft-ietf-oauth-mtls-08#section-3>) in the OAuth 2.0 Mutual TLS Client Authentication and Certificate Bound Access Tokens for more details.

有关详细信息，请参阅 OAuth 2.0 Mutual TLS 客户端身份验证和证书绑定访问令牌中的 Mutual TLS 客户端证书绑定访问令牌。

Currently, Keycloak client adapters do not support holder-of-key token verification.
Keycloak adapters treat access and refresh tokens as bearer tokens.



目前，钥匙斗篷客户端适配器不支持钥匙持有者令牌验证。钥匙斗篷适配器将访问和刷新令牌视为持有者令牌。

Proof Key for Code Exchange Code Challenge Method

代码交换代码质疑方法的证明密钥

If an attacker steals an authorization code of a legitimate client, Proof Key for Code Exchange (PKCE) prevents the attacker from receiving the tokens that apply to the code.

如果攻击者窃取合法客户端的授权代码，则代码交换的证明密钥(Proof Key for Code Exchange, PKCE)将阻止攻击者接收应用于代码的令牌。

An administrator can select one of these options:

管理员可以选择以下选项之一：

(blank) (空白)

Keycloak does not apply PKCE unless the client sends appropriate PKCE parameters to Keycloaks authorization endpoint.

除非客户机将适当的 PKCE 参数发送到 Keycloaks 授权终结点，否则不会应用 PKCE。

S256

Keycloak applies to the client PKCE whose code challenge method is S256.

密钥斗篷适用于客户机 PKCE，其代码质疑方法是 S256。

plain 平淡无奇

Keycloak applies to the client PKCE whose code challenge method is plain.

密钥斗篷应用于客户机 PKCE，其代码询问方法是普通的。

See [RFC 7636 Proof Key for Code Exchange by OAuth Public Clients](#)

(<https://datatracker.ietf.org/doc/html/rfc7636>) for more details.

有关更多详细信息，请参见 OAuth 公共客户机代码交换的 RFC 7636证明密钥。

ACR to Level of Authentication (LoA) Mapping

ACR 到认证级别(LoA)映射

In the advanced settings of a client, you can define which `Authentication Context Class Reference (ACR)` value is mapped to which `Level of Authentication (LoA)`. This mapping can be specified also at the realm as mentioned in the ACR to LoA Mapping. A best practice is to configure this mapping at the realm level, which allows to share the same settings across multiple clients.

在客户端的高级设置中，可以定义将哪个身份验证上下文类引用(ACR)值映射到哪个身份验证级别(LoA)。此映射也可以在 ACR 到 LoA Mapping 中提到的领域中指定。最佳实践是在领域级别配置此映射，这允许跨多个客户端共享相同的设置。

The `Default ACR Values` can be used to specify the default values when the login request is sent from this client to Keycloak without `acr_values` parameter and without a `claims` parameter that has an `acr` claim attached. See [official OIDC dynamic client registration specification](#) (https://openid.net/specs/openid-connect-registration-1_0.html#ClientMetadata).

当登录请求从这个客户机发送到 Keycloak 时，可以使用默认的 ACR 值来指定默认值，而不需要 ACR_value 参数，也不需要附带 ACR 声明的声明参数。请参阅官方的 OIDC 动态客户端注册规范。



Note that default ACR values are used as the default level, however it cannot be reliably used to enforce login with the particular level. For example, assume that you configure the 请注意，默认 ACR 值被用作默认级别，但它不能可靠地用于强制特定级别的登录。例如，假设您配置了 Default ACR Values to level 2. Then by default, users will be required to authenticate with level 2. However when the user explicitly attaches the parameter into login request such as 到第二层。然后，默认情况下，用户将被要求使用级别2进行身份验证。但是，当用户显式地将参数附加到登录请求中时，例如 acr_values=1 , then the level 1 will be used. As a result, if the client really requires level 2, the client is encouraged to check the presence of the , 那么第一级将被使用。因此，如果客户机确实需要级别2，则鼓励客户机检查 acr claim inside ID Token and double-check that it contains the requested level 2. 在 ID 令牌内声明，并仔细检查它是否包含请求的级别2

ACR to LoA Mapping	Key	Value
	Type a key	Type a value

Default ACR Values ?

Add

Save **Revert**

For further details see Step-up Authentication and [the official OIDC specification](#) (https://openid.net/specs/openid-connect-core-1_0.html#acrSemantics).

有关详细信息，请参阅升级身份验证和官方 OIDC 规范。

[Edit this section 编辑这部分](#)
(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/clients/oidc/con-confidential-client-credentials.adoc)

[Report an issue 报告一个问题](#)
(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/clients/oidc/con-confidential-client-credentials.adoc)

If the Client authentication of the client is set to **ON**, the credentials of the client must be configured under the **Credentials** tab.

如果客户端的 Client 身份验证设置为 ON，则必须在“凭据”选项卡下配置客户端的凭据。

Credentials tab 证件标签

The screenshot shows the Keycloak 'Clients' configuration interface. On the left sidebar, 'Clients' is selected. In the main area, a client named 'myapp' (OpenID Connect) is selected. The 'Credentials' tab is active. The 'Client Authenticator' dropdown is set to 'Client Id and Secret'. The 'Client secret' field contains a long string of asterisks, with 'Regenerate' and 'Copy' buttons. The 'Registration access token' field is also present with its own 'Regenerate' button. Other tabs like 'Settings', 'Keys', 'Roles', etc., are visible but inactive.

The **Client Authenticator** drop-down list specifies the type of credential to use for your client.

“客户端身份验证器”下拉列表指定要为客户端使用的凭据类型。

Client ID and Secret

客户身份和机密

This choice is the default setting. The secret is automatically generated. Click **Regenerate** to recreate the secret if necessary.

此选项是默认设置。秘密将自动生成。如果需要，单击“重新生成”重新生成秘密。

Signed JWT 签名 JWT

The screenshot shows the Keycloak interface for managing clients. On the left sidebar, 'Clients' is selected. In the main area, under the 'myapp' client, the 'Credentials' tab is active. It displays two dropdown menus: 'Client Authenticator' set to 'Signed Jwt' and 'Signature algorithm' set to 'Any algorithm'. A blue 'Save' button is located below these fields. Further down, there's a section for 'Registration access token' with a 'Regenerate' button.

Signed JWT is "Signed Json Web Token".

Signed JWT 是“ Signed Json Web Token”。

When choosing this credential type you will have to also generate a private key and certificate for the client in the tab **Keys**. The private key will be used to sign the JWT, while the certificate is used by the server to verify the signature.

选择此凭据类型时，还必须在选项卡 **Keys** 中为客户端生成私钥和证书。私钥将用于对 JWT 进行签名，而证书则由服务器用于验证签名。

Keys tab 钥匙卡

The screenshot shows the 'Keys' tab of the 'Clients' configuration page for the 'myapp' client. The 'JWKS URL configs' section contains a note: 'If "Use JWKS URL switch" is on, you need to fill a valid JWKS URL. After saving, admin can download keys from the JWKS URL or keys will be downloaded automatically by Keycloak server when see the stuff signed by the unknown KID'. Below this is a 'Use JWKS URL' toggle switch which is currently off. At the bottom of the page are three buttons: 'Save', 'Generate new keys' (which is highlighted in blue), and 'Import'.

Click on the **Generate new keys** button to start this process.

单击 **Generate new keys** 按钮启动此过程。

Generate keys 生成密钥

Generate keys?

If you generate new keys, you can download the keystore with the private key automatically and save it on your client's side. Keycloak server will save just the certificate and public key, but not the private key.

Archive format ?

JKS

Key alias * ?

myapp

Key password * ?

Store password * ?

Generate **Cancel**

1. Select the archive format you want to use.

选择要使用的归档格式。

2. Enter a **key password**.

输入密钥密码。

3. Enter a **store password**.

输入存储密码。

4. Click **Generate**.

单击 Generate。

When you generate the keys, Keycloak will store the certificate and you download the private key and certificate for your client.

当您生成密钥时，Keycon 将存储证书，并为您的客户端下载私钥和证书。

You can also generate keys using an external tool and then import the client's certificate by clicking **Import Certificate**.

还可以使用外部工具生成密钥，然后通过单击“导入证书”导入客户端的证书。

Generate keys?

If you generate new keys, you can download the keystore with the private key automatically and save it on your client's side. Keycloak server will save just the certificate and public key, but not the private key.

Archive format ?

JKS

Key alias * ?

Store password * ?

Import file

Drag a file here or browse to upload

Browse... Clear

Import Cancel

1. Select the archive format of the certificate.

选择证书的归档格式。

2. Enter the store password.

输入商店密码。

3. Select the certificate file by clicking Import File.

通过单击 Import File 选择证书文件。

4. Click Import.

单击 Import。

Importing a certificate is unnecessary if you click **Use JWKS URL**. In this case, you can provide the URL where the public key is published in **JWK** (<https://self-issued.info/docs/draft-ietf-jose-json-web-key.html>) format. With this option, if the key is ever changed, Keycloak reimports the key.

如果单击“使用 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。使用这个选项，如果密钥曾经更改过，Keycloak 将重新导入密钥。

If you are using a client secured by Keycloak adapter, you can configure the JWKS URL in this format, assuming that <https://myhost.com/myapp> is the root URL of your client application:

如果你使用的客户端使用的是钥匙斗篷适配器，你可以配置这种格式的 Jwks URL，假设 https://myhost.com/myapp 是你的客户端应用程序的根 URL：

```
https://myhost.com/myapp/k_jwks
```

BASH

See [Server Developer Guide](https://www.keycloak.org/docs/latest/server_development/) (https://www.keycloak.org/docs/latest/server_development/) for more details.

有关详细信息，请参阅服务器开发人员指南。

Signed JWT with Client Secret

带有客户机秘密的签名 JWT

If you select this option, you can use a JWT signed by client secret instead of the private key.

如果选择此选项，则可以使用由客户机机密签名的 JWT 代替私钥。

The client secret will be used to sign the JWT by the client.

客户机机密将用于客户机对 JWT 进行签名。

X509 Certificate

X509证书

Keycloak will validate if the client uses proper X509 certificate during the TLS Handshake.

如果客户端在 TLS 握手过程中使用了正确的 X509证书，则键掩码将进行验证。

X509 certificate X509证书

The screenshot shows the Keycloak administration interface. On the left, a sidebar menu is open with 'Clients' selected. The main content area shows the 'Clients' section with 'myapp' selected. The 'Credentials' tab is active. The 'Client Authenticator' dropdown is set to 'X509 Certificate'. The 'Subject DN' input field contains 'cn=localhost, ou=keycloak'. A 'Save' button is visible at the bottom of the form.

The validator also checks the Subject DN field of the certificate with a configured regexp validation expression. For some use cases, it is sufficient to accept all certificates. In that case, you can use `(.*?)` `(? : $)` expression.

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，接受所有证书就足够了。在这种情况下，您可以使用`(.*?)` `(? : $)`表达式。

Two ways exist for Keycloak to obtain the Client ID from the request:

有两种方法可以让 Keycloak 从请求中获取客户端 ID:

- The `client_id` parameter in the query (described in Section 2.2 of the [OAuth 2.0 Specification](#) (<https://datatracker.ietf.org/doc/html/rfc6749>)).
查询中的 `client_id` 参数(在 OAuth 2.0 规范的 2.2 节中描述)。
- Supply `client_id` as a form parameter.
提供 `client_id` 作为表单参数。

[Edit this section](#) 编辑这部分

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/clients/oidc/con-secret-rotation.adoc)

[Report an issue](#) 报告一个问题

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/clients/oidc/con-secret-rotation.adoc)

 Please note that Client Secret Rotation support is in development. Use this feature experimentally.

请注意，客户端秘密旋转支持正在开发中。试验性地使用这个特性。

For a client with Confidential Client authentication Keycloak supports the functionality of rotating client secrets through Client Policies.

对于具有保密客户端身份验证的客户端，Keycloak 支持通过客户端策略来旋转客户端机密的功能。

The client secrets rotation policy provides greater security in order to alleviate problems such as secret leakage. Once enabled, Keycloak supports up to two concurrently active secrets for each client. The policy manages rotations according to the following settings:

客户机机密轮换策略提供了更高的安全性，以减轻机密泄漏等问题。启用后，Keycloak 为每个客户机支持最多两个并发活动的机密。该策略根据以下设置管理轮换:

- **Secret expiration:** [seconds] - When the secret is rotated, this is the expiration of time of the new secret. The amount, *in seconds*, added to the secret creation date. Calculated at policy execution time.

秘密过期时间: [秒]-当秘密被旋转时, 这是新秘密的过期时间。以秒为单位添加到秘密创建日期的数量。在策略执行时计算。

- **Rotated secret expiration:** [seconds] - When the secret is rotated, this value is the remaining expiration time for the old secret. This value should be always smaller than Secret expiration. When the value is 0, the old secret will be immediately removed during client rotation. The amount, *in seconds*, added to the secret rotation date. Calculated at policy execution time.

旋转秘密过期时间: [秒]-当秘密被旋转时, 这个值是旧秘密的剩余过期时间。此值应始终小于 Secret 过期值。当值为0时, 旧的秘密将在客户端旋转期间立即删除。以秒为单位的数量添加到秘密轮换日期。在策略执行时计算。

- **Remaining expiration time for rotation during update:** [seconds] - Time period when an update to a dynamic client should perform client secret rotation. Calculated at policy execution time.

更新期间轮换的剩余过期时间: [秒]-对动态客户端的更新应执行客户端秘密轮换的时间段。在策略执行时计算。

When a client secret rotation occurs, a new main secret is generated and the old client main secret becomes the secondary secret with a new expiration date.

当客户机机密轮换发生时, 将生成一个新的主机密, 旧的客户机机密将成为次机密, 并且有一个新的过期日期。

Rules for client secret rotation 客户机秘密轮换规则

Rotations do not occur automatically or through a background process. In order to perform the rotation, an update action is required on the client, either through the Keycloak Admin Console through the function of **Regenerate Secret**, in the client's credentials tab or Admin REST API. When invoking a client update action, secret rotation occurs according to the rules:

旋转不会自动或通过后台进程进行。为了执行旋转, 需要在客户机上执行更新操作, 可以通过在客户机的凭据选项卡或管理 REST API 中的“再生秘密”函数, 通过 Keycloak “隐藏管理控制台”执行。当调用客户端更新操作时, 根据规则进行秘密轮换:

- When the value of **Secret expiration** is less than the current date.
当 Secret 过期值小于当前日期时。
- During dynamic client registration client-update request, the client secret will be automatically rotated if the value of **Remaining expiration time for rotation during update** match the period between the current date and the **Secret expiration**.

在动态客户端注册客户端更新请求期间, 如果更新期间的剩余到期时间值与当前日期和 Secret 过期之间的时间段相匹配, 则客户端秘密将自动旋转。

Additionally it is possible through Admin REST API to force a client secret rotation at any time.

此外, 还可以通过 Admin REST API 在任何时候强制客户机进行秘密轮换。



During the creation of new clients, if the client secret rotation policy is active, the behavior will be applied automatically.

在创建新客户端期间，如果客户端秘密轮换策略处于活动状态，则将自动应用该行为。



To apply the secret rotation behavior to an existing client, update that client after you define the policy so that the behavior is applied. 若要将秘密轮换行为应用于现有客户端，请在定义策略后更新该客户端，以便应用该行为

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/clients/oidc/proc-secret-rotation.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/clients/oidc/proc-secret-rotation.adoc)

The following is an example of defining a secret rotation policy:

以下是定义秘密轮调政策的一个例子：

Procedure 程序

1. Click **Realm Settings** in the menu.

单击菜单中的“域设置”。

2. Click **Client Policies** tab.

单击“客户端策略”选项卡。

3. On the **Profiles** page, click **Create client profile**.

在“配置文件”页上，单击“创建客户端配置文件”。

Create a profile 创建一个配置文件

Realm settings > Client policies > Create client profile

Create client profile

Action ▾

Client profile name * Weekly client secret rotation profile
The name must be unique within the realm

Description Updates the client secret weekly

Save Cancel

4. Enter any name for Name.

为 Name 输入任何名称。

5. Enter a description that helps you identify the purpose of the profile for **Description**.

输入有助于确定 Description 配置文件用途的说明。

6. Click **Save**.

单击“保存”。

This action creates the profile and enables you to configure executors.

此操作将创建配置文件并使您能够配置执行器。

7. Click **Add executor** to configure an executor for this profile.

单击“添加执行器”可为此配置文件配置执行器。

Create a profile executor 创建配置文件执行器

Master

Manage

Clients

Client scopes

Realm roles

Users

Groups

Sessions

Events

Configure

Realm settings > Client policies > Add executor

Add executor

Executor type: secret-rotation

Secret expiration: 604800

Rotated Secret expiration: 172800

Remain Expiration Time: 864000

Add Cancel

8. Select *secret-rotation* for **Executor Type**.

为“执行器类型”选择“秘密旋转”。

9. Enter the maximum duration time of each secret, in seconds, for **Secret Expiration**.

输入每个秘密的最长持续时间(秒) , 用于秘密到期。

10. Enter the maximum duration time of each rotated secret, in seconds, for **Rotated Secret Expiration**.

输入每个旋转秘密的最长持续时间(秒) , 用于旋转秘密有效期。



Remember that the **Rotated Secret Expiration** 旋转秘密有效期 value must always be less than **Secret Expiration** 秘密到期. 价值必须总是小于Secret Expiration 秘密到期.

11. Enter the amount of time, in seconds, after which any update action will update the client for **Remain Expiration Time**.

输入时间量(以秒为单位) , 在此之后, 任何更新操作都将更新客户端的剩余过期时间。

12. Click **Add**.

单击“添加”。

In the example above:

在上面的例子中:

- Each secret is valid for one week.

每个秘密有效期为一周。

- The rotated secret expires after two days.

这个秘密两天后就过期了。

- The window for updating dynamic clients starts one day before the secret expires.

用于更新动态客户端的窗口在秘密过期前一天开始。

13. Return to the Client Policies tab.

返回到客户策略选项卡。

14. Click Policies.

单击策略。

15. Click Create client policy.

单击“创建客户端策略”。

Create the Client Secret Rotation Policy 创建客户端秘密轮换策略

Master

Realm settings > Client policies > Create policy

Enabled

Name *

Weekly client secret rotation policy

Description

Enables secret rotation behavior for confidential clients.

Save Cancel

16. Enter any name for Name.

为 Name 输入任何名称。

17. Enter a description that helps you identify the purpose of the policy for Description.

输入有助于识别 Description 策略用途的说明。

18. Click Save.

单击“保存”。

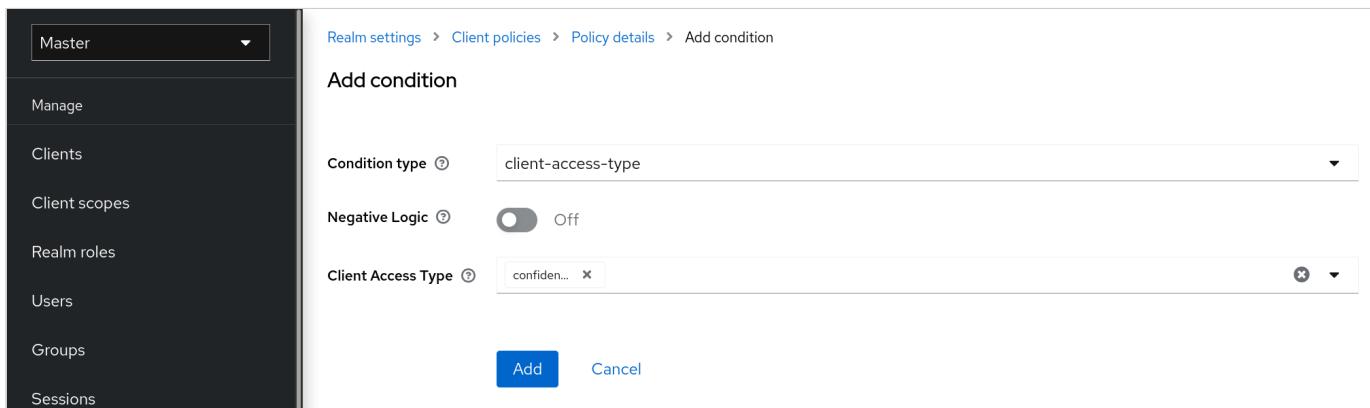
This action creates the policy and enables you to associate policies with profiles. It also allows you to configure the conditions for policy execution.

此操作将创建策略并使您能够将策略与配置文件关联。它还允许您配置策略执行的条件。

19. Under Conditions, click Add condition.

在“条件”下，单击“添加条件”。

Create the Client Secret Rotation Policy Condition 创建客户端秘密轮换策略条件



20. To apply the behavior to all confidential clients select *client-access-type* in the Condition Type field

若要将此行为应用于所有机密客户端，请在“条件类型”字段中选择“客户端访问类型”



To apply to a specific group of clients, another approach would be to select the *client-roles* type in the **Condition Type** field. In this way, you could create specific roles and assign a custom rotation configuration to each role.

要应用于特定的客户端组，另一种方法是在条件类型字段中选择客户端角色类型。通过这种方式，您可以创建特定的角色，并为每个角色分配自定义的轮换配置。

21. Add *confidential* to the field Client Access Type.

向字段“客户端访问类型”添加机密。

22. Click Add.

单击“添加”。

23. Back in the policy setting, under *Client Profiles*, click Add client profile and then select **Weekly Client Secret Rotation Profile** from the list and then click Add.

回到策略设置中，在“客户端配置文件”下，单击“添加客户端配置文件”，然后从列表中选择“每周客户端秘密轮换配置文件”，再单击“添加”。

Client Secret Rotation Policy 客户保密轮换政策

The screenshot shows the Keycloak Admin Console interface. On the left, a sidebar menu is visible with options like Master, Manage, Clients, Client scopes, Realm roles, Users, Groups, Sessions, Events, Configure, Realm settings (which is selected), and Authentication. The main content area is titled 'Weekly client secret rotation policy'. It shows a name field set to 'Weekly client secret rotation policy' and a description field containing 'Enables secret rotation behavior for confidential clients.' There are 'Save' and 'Revert' buttons at the bottom. Below this, there are sections for 'Conditions' (with a 'client-access-type' entry) and 'Client profiles' (with a 'Weekly client secret rotation profile' entry). A top navigation bar indicates the path: Realm settings > Client policies > Policy details. A toggle switch labeled 'Enabled' is set to 'Enabled', and an 'Action' dropdown menu is open.

To apply the secret rotation behavior to an existing client, follow the following steps:

要将秘密轮换行为应用于现有客户端，请按照下列步骤操作：

Using the Admin Console 使用管理控制台

1. Click **Clients** in the menu.

单击菜单中的 Clients。

2. Click a client.

点击客户端。

3. Click the **Credentials** tab.

单击“凭据”选项卡。

4. Click **Re-generate** of the client secret.

单击“重新生成客户端机密”。



Using client REST services it can be executed in two ways: 使用客户机 REST 服务可以通过两种方式执行：

- Through an update operation on a client

通过客户端上的更新操作

- Through the regenerate client secret endpoint

通过重新生成客户端秘密端点

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/clients/oidc/proc-using-a-service-account.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/clients/oidc/proc-using-a-service-account.adoc)

Each OIDC client has a built-in *service account*. Use this *service account* to obtain an access token.

每个 OIDC 客户端都有一个内置的服务帐户。使用此服务帐户获取访问令牌。

Procedure 程序

1. Click **Clients** in the menu.

单击菜单中的 Clients。

2. Select your client.

选择你的客户。

3. Click the **Settings** tab.

单击“设置”选项卡。

4. Toggle Client authentication to **On**.

将客户端身份验证切换到“打开”。

5. Select **Service accounts roles**.

选择服务帐户角色。

6. Click **Save**.

单击“保存”。

7. Configure your client credentials.

配置您的客户端凭据。

8. Click the **Scope** tab.

单击 Scope 选项卡。

9. Verify that you have roles or toggle **Full Scope Allowed** to **ON**.

验证是否有角色或切换“允许开启的全范围”。

10. Click the **Service Account Roles** tab

单击“服务帐户角色”选项卡

11. Configure the roles available to this service account for your client.

为客户端配置此服务帐户可用的角色。

Roles from access tokens are the intersection of:

访问令牌中的角色是以下几个方面的交集:

- Role scope mappings of a client combined with the role scope mappings inherited from linked client scopes.

客户端的角色范围映射与从链接的客户端范围继承的角色范围映射相结合。

- Service account roles.

服务帐户角色。

The REST URL to invoke is `/realms/{realm-name}/protocol/openid-connect/token`. This URL must be invoked as a POST request and requires that you post the client credentials with the request.

要调用的 REST URL 是`/fields/{ domain-name }/protocol/openid-connect/token`。此 URL 必须作为 POST 请求调用，并要求您将客户端凭据与请求一起发布。

By default, client credentials are represented by the `clientId` and `clientSecret` of the client in the **Authorization: Basic** header but you can also authenticate the client with a signed JWT assertion or any other custom mechanism for client authentication.

默认情况下，客户端凭据在 `Authorization: Basic` 头中由客户端的 `clientId` 和 `clientSecret` 表示，但是您也可以使用签名的 JWT 断言或任何其他用于客户端身份验证的自定义机制对客户端进行身份验证。

You also need to set the `grant_type` parameter to "client_credentials" as per the OAuth2 specification.

您还需要根据 OAuth2 规范将 `Grant_type` 参数设置为“client凭据”。

For example, the POST invocation to retrieve a service account can look like this:

例如，检索服务帐户的 POST 调用可以如下所示：

```
POST /realms/demo/protocol/openid-connect/token
Authorization: Basic cHJvZHvjdC1zYS1jbG11bnQ6cGFzc3dvcmQ=
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials
```

The response would be similar to this [Access Token Response](#)

(<https://datatracker.ietf.org/doc/html/rfc6749#section-4.4.3>) from the OAuth 2.0 specification.

响应类似于 OAuth 2.0 规范中的 Access Token Response。

```
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
}
```

Only the access token is returned by default. No refresh token is returned and no user session is created on the Keycloak side upon successful authentication by default. Due to the lack of a refresh token, re-authentication is required when the access token expires. However, this situation does not mean any additional overhead for the Keycloak server because sessions are not created by default.

默认情况下只返回访问令牌。默认情况下，身份验证成功后，不会返回刷新令牌，也不会在 Key (斗篷)端创建用户会话。由于缺少刷新令牌，当访问令牌过期时需要重新身份验证。但是，这种情况并不意味着 Keycloak 服务器会有额外的开销，因为默认情况下不会创建会话。

In this situation, logout is unnecessary. However, issued access tokens can be revoked by sending requests to the OAuth2 Revocation Endpoint as described in the OpenID Connect Endpoints section.

在这种情况下，注销是不必要的。但是，可以通过向 OAuth2 撤销端点发送请求来撤销已发出的访问令牌，如 OpenID 连接端点部分所述。

Additional resources 额外资源

For more details, see Client Credentials Grant.

有关更多详细信息，请参见客户端凭据授予。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/clients/oidc/con-audience.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/clients/oidc/con-audience.adoc)

Typically, the environment where Keycloak is deployed consists of a set of *confidential* or *public* client applications that use Keycloak for authentication.

通常，部署 Key 的环境包含一个使用 Key 斗篷进行认证的保密或公共客户端应用程序集合。

Services (Resource Servers in the OAuth 2 specification)

(<https://datatracker.ietf.org/doc/html/draft-ietf-oauth-mtls-08#section-4.2>) are also available that serve requests from client applications and provide resources to these applications. These services require an *Access token* (Bearer token) to be sent to them to authenticate a request. This token is obtained by the frontend application upon login to Keycloak.

服务(OAuth 2规范中的资源服务器)也可用于服务来自客户端应用程序的请求并为这些应用程序提供资源。这些服务需要向它们发送一个 Access 令牌(Bearer 令牌)来验证请求。这个令牌由前端应用程序在登录到钥匙斗篷时获得。

In the environment where trust among services is low, you may encounter this scenario:

在服务间信任度较低的环境中，您可能会遇到以下场景：

1. A frontend client application requires authentication against Keycloak.

前端客户端应用程序需要针对 Keycloak 进行身份验证。

2. Keycloak authenticates a user.

钥匙斗篷对用户进行身份验证。

3. Keycloak issues a token to the application.

钥匙斗篷向应用程序发出一个令牌。

4. The application uses the token to invoke an untrusted service.

应用程序使用令牌调用不受信任的服务。

5. The untrusted service returns the response to the application. However, it keeps the applications token.

不受信任的服务将响应返回给应用程序。

6. The untrusted service then invokes a trusted service using the applications token. This results in broken security as the untrusted service misuses the token to access other services on behalf of the client application.

然后，不受信任的服务使用应用程序令牌调用受信任的服务。由于不受信任的服务滥用令牌来代表客户端应用程序访问其他服务，这将导致安全性受损。

This scenario is unlikely in environments with a high level of trust between services but not in environments where trust is low. In some environments, this workflow may be correct as the untrusted service may have to retrieve data from a trusted service to return data to the original client application.

在服务之间具有高度信任的环境中不太可能出现这种情况，但在信任度较低的环境中则不会出现这种情况。在某些环境中，此工作流可能是正确的，因为不受信任的服务可能必须从受信任的服务检索数据，以将数据返回给原始客户端应用程序。

An unlimited audience is useful when a high level of trust exists between services. Otherwise, the audience should be limited. You can limit the audience and, at the same time, allow untrusted services to retrieve data from trusted services. In this case, ensure that the untrusted service and the trusted service are added as audiences to the token.

当服务之间存在高水平的信任时，无限的受众是有用的。否则，观众应该是有限的。您可以限制受众，同时允许不受信任的服务从受信任的服务检索数据。在这种情况下，请确保将不受信任的服务和受信任的服务作为受众添加到令牌。

To prevent any misuse of the access token, limit the audience on the token and configure your services to verify the audience on the token. The flow will change as follows:

若要防止访问令牌的任何误用，请限制令牌上的访问对象，并配置服务以验证令牌上的访问对象。流程将改变如下：

1. A frontend application authenticates against Keycloak.

一个前端应用程序针对钥匙斗篷进行身份验证。

2. Keycloak authenticates a user.

钥匙斗篷对用户进行身份验证。

3. Keycloak issues a token to the application. The application knows that it will need to invoke an untrusted service so it places `scope=<untrusted service>` in the authentication request sent to Keycloak (see Client Scopes section for more details about the scope parameter).

钥匙斗篷向应用程序发出一个令牌。应用程序知道它将需要调用一个不受信任的服务，因此它将 `scope = <untrust service>` 放置在发送给 Keycon 的身份验证请求中(有关 `scope` 参数的更多细节，请参见 Client Scope 部分)。

The token issued to the application contains a reference to the untrusted service in its audience ("audience": ["<untrusted service>"]) which declares that the client uses this access token to invoke the untrusted service.

发给应用程序的令牌包含对其受众中的不受信任服务的引用("受众": ["<untrust service>"])，该引用声明客户端使用此访问令牌调用不受信任服务。

4. The untrusted service invokes a trusted service with the token. Invocation is not successful because the trusted service checks the audience on the token and find that its audience is only for the untrusted service. This behavior is expected and security is not broken.

不受信任的服务使用令牌调用受信任的服务。调用未能成功，因为受信任的服务检查令牌上的受众，并发现其受众仅针对不受信任的服务。这种行为是预料之中的，安全性没有被破坏。

If the client wants to invoke the trusted service later, it must obtain another token by reissuing the SSO login with `scope=<trusted service>`. The returned token will then contain the trusted service as an audience:

如果客户机希望稍后调用受信任的服务，它必须通过重新发出 SSO 登录名(`scope = < trust service >`)来获得另一个令牌。然后，返回的令牌将包含作为受众的受信任服务：

```
"audience": [ "<trusted service>" ]
```

JSON

Use this value to invoke the <trusted service>.

使用此值调用 <受信任的服务>。

Setup 设置

When setting up audience checking:

设置观众检查时:

- Ensure that services are configured to check audience on the access token sent to them by adding the flag **verify-token-audience** in the adapter configuration. See [Adapter configuration](https://www.keycloak.org/docs/latest/securing_apps/#_java_adapter_config) (https://www.keycloak.org/docs/latest/securing_apps/#_java_adapter_config) for details.

通过在适配器配置中添加验证令牌-观众标志，确保服务被配置为检查发送给它们的访问令牌的观众。有关详细信息，请参阅适配器配置。

- Ensure that access tokens issued by Keycloak contain all necessary audiences. Audiences can be added using the client roles as described in the next section or hardcoded. See [Hardcoded audience](#).
确保 Keycloak 发出的访问令牌包含所有必要的访问对象。可以使用下一节中描述的客户端角色或硬编码添加受众。看到硬编码的观众。

Automatically add audience 自动添加观众

An *Audience Resolve* protocol mapper is defined in the default client scope *roles*. The mapper checks for clients that have at least one client role available for the current token. The client ID of each client is then added as an audience, which is useful if your service (usually bearer-only) clients rely on client roles.

听众解析协议映射器在默认客户端范围角色中定义。映射器检查当前令牌至少有一个客户端角色可用的客户端。然后将每个客户机的客户机 ID 作为受众添加，如果您的服务(通常是仅持有者)客户机依赖于客户机角色，那么这将非常有用。

For example, for a bearer-only client and a confidential client, you can use the access token issued for the confidential client to invoke the bearer-only client REST service. The bearer-only client will be automatically added as an audience to the access token issued for the confidential client if the following are true:

例如，对于只承载客户机和机密客户机，您可以使用为机密客户机发出的访问令牌来调用只承载客户机 REST 服务。如果下列情况属实，只有持有人客户端将自动添加为机密客户端发出的访问令牌的受众:

- The bearer-only client has any client roles defined on itself.

只有持有者的客户端具有自身定义的任何客户端角色。

- Target user has at least one of those client roles assigned.

目标用户至少分配了其中一个客户端角色。

- Confidential client has the role scope mappings for the assigned role.

机密客户端具有指定角色的角色范围映射。

If you want to ensure that the audience is not added automatically, do not configure role scope mappings directly on the confidential client. Instead, you can create a dedicated client scope that contains the role scope mappings for the client roles of your dedicated client scope.

如果希望确保不自动添加受众，请不要直接在机密客户端上配置角色范围映射。相反，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的客户端角色的角色范围映射。

Assuming that the client scope is added as an optional client scope to the confidential client, the client roles and the audience will be added to the token if explicitly requested by the `scope=<trusted service>` parameter.

假设将客户端作用域作为可选的客户端作用域添加到机密客户端，如果 `scope = < trust service >` 参数显式请求，则客户端角色和受众将被添加到令牌中。

The frontend client itself is not automatically added to the access token audience, therefore allowing easy differentiation between the access token and the ID token, since the access token will not contain the client for which the token is issued as an audience.

前端客户端本身不会自动添加到访问令牌访问者中，因此可以容易地区分访问令牌和 ID 令牌，因为访问令牌不包含作为访问令牌发出的客户端。

If you need the client itself as an audience, see the hardcoded audience option. However, using the same client as both frontend and REST service is not recommended.

如果您需要客户端本身作为受众，请参见硬编码的受众选项。但是，不建议使用同一个客户端作为前端和 REST 服务。

Hardcoded audience 硬编码的观众

When your service relies on realm roles or does not rely on the roles in the token at all, it can be useful to use a hardcoded audience. A hardcoded audience is a protocol mapper, that will add the client ID of the specified service client as an audience to the token. You can use any custom value, for example a URL, if you want to use a different audience than the client ID.

当您的服务依赖于领域角色或者根本不依赖于令牌中的角色时，使用硬编码的受众会很有用。硬编码的访问接口是一个协议映射器，它将指定服务客户端的客户端 ID 作为访问接口添加到令牌中。如果希望使用与客户端 ID 不同的受众，可以使用任何自定义值，例如 URL。

You can add the protocol mapper directly to the frontend client. If the protocol mapper is added directly, the audience will always be added as well.

您可以将协议映射器直接添加到前端客户端。如果直接添加了协议映射程序，那么也将始终添加受众。

For more control over the protocol mapper, you can create the protocol mapper on the dedicated client scope, which will be called for example **good-service**.

为了对协议映射器进行更多的控制，您可以在专用的客户机作用域上创建协议映射器，例如 **good-service**。

Audience protocol mapper 受众协议映射器

The screenshot shows the 'Mapper details' page for an 'Audience' type protocol mapper. The left sidebar is titled 'Client scopes' and lists various client-related sections. The main form has the following fields:

- Mapper type:** Audience
- Name:** Audience for good-client
- Included Client Audience:** good-client
- Included Custom Audience:** (empty)
- Add to ID token:** Off (disabled)
- Add to access token:** On (selected)

At the bottom are 'Save' and 'Cancel' buttons.

- From the Installation tab of the **good-service** client, you can generate the adapter configuration and you can confirm that *verify-token-audience* option will be set to **true**. This forces the adapter to verify the audience if you use this configuration.

从好的服务客户机的 Installationtab 中，您可以生成适配器配置，并且您可以确认 verifier- 令牌-受众选项将被设置为 true。如果您使用此配置，这将强制适配器验证受众。

- You need to ensure that the confidential client is able to request **good-service** as an audience in its tokens.

您需要确保机密客户端能够请求良好的服务，作为其令牌的受众。

On the confidential client:

关于机密客户：

- Click the *Client Scopes* tab.

单击 ClientScope 选项卡。

- Assign **good-service** as an optional (or default) client scope.

将良好服务分配为可选(或默认)客户端范围。

See Client Scopes Linking section for more details.

有关详细信息，请参阅客户端作用域链接部分。

- You can optionally Evaluate Client Scopes and generate an example access token. **good-service** will be added to the audience of the generated access token if **good-service** is included in the *scope* parameter, when you assigned it as an optional client scope.

您可以选择评估客户端作用域并生成示例访问令牌。如果在将 **good-service** 作为可选客户端作用域分配时，*scope* 参数中包含 **good-service**，那么将向生成的访问令牌的受众添加 **good-service**。

- In your confidential client application, ensure that the *scope* parameter is used. The value **good-service** must be included when you want to issue the token for accessing **good-service**.

在您的机密客户端应用程序中，确保使用 *scope* 参数。当您想要发出访问好服务的令牌时，必须包括好服务的价值。

See:

参见:

- [parameters forwarding section](https://www.keycloak.org/docs/latest/securing_apps/#_params_forwarding) (https://www.keycloak.org/docs/latest/securing_apps/#_params_forwarding) if your application uses the servlet adapter.

如果应用程序使用 servlet 适配器，则参数转发部分。

- [javascript adapter section](https://www.keycloak.org/docs/latest/securing_apps/#_javascript_adapter) (https://www.keycloak.org/docs/latest/securing_apps/#_javascript_adapter) if your application uses the javascript adapter.

如果应用程序使用 javascript 适配器，则在 javascript 适配器部分。

Both the *都是 Audience 观众* and *还有 Audience Resolve 观众决心* protocol mappers add the audiences to the access token only, by default. The ID Token typically contains only a single audience, the client ID for which the token was issued, a requirement of the OpenID Connect specification. However, the access token does not necessarily have the client ID, which was the token issued for, unless the audience mappers added it. 默认情况下，协议映射程序仅将访问群体添加到访问令牌。ID 令牌通常只包含单个受众，即为其发出令牌的客户机 ID，这是 OpenID Connect 规范的要求。但是，访问令牌不一定具有客户端 ID，这是为其发出的令牌，除非受众映射程序添加了它

Edit this section 编辑这部分
(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/clients/saml/proc-creating-saml-client.adoc)

Report an issue 报告一个问题
(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/clients/saml/proc-creating-saml-client.adoc)

Keycloak supports SAML 2.0 for registered applications. POST and Redirect bindings are supported. You can choose to require client signature validation. You can have the server sign and/or encrypt responses as well.

对于已注册的应用程序，Key斗篷支持 SAML 2.0。支持 POST 和重定向绑定。您可以选择要求客户端签名验证。您还可以让服务器签名和/或加密响应。

Procedure 程序

1. Click **Clients** in the menu.

单击菜单中的 Clients。

2. Click **Create client** to go to the **Create client** page.

单击 Create client 转到 Create client 页面。

3. Set **Client type** to **SAML**.

将客户端类型设置为 SAML。

Create client 创建客户端

The screenshot shows the 'Create client' page in the Keycloak interface. The left sidebar has a dropdown menu with 'Master' selected, followed by a list of management options: Manage, Clients (which is highlighted with a blue bar), Client scopes, Realm roles, Users, Groups, Sessions, Events, Configure, Realm settings, Authentication, Identity providers, and User federation. The main content area is titled 'Create client' and includes a sub-section 'General Settings'. It displays the following fields:

- Client type**: SAML
- Client ID**: mysamllapp
- Name**: saml
- Description**: (empty)

At the bottom of the form are three buttons: 'Save' (highlighted in blue), 'Back', and 'Cancel'.

4. Enter the **Client ID** of the client. This is often a URL and is the expected **issuer** value in SAML requests sent by the application.

输入客户端的客户端 ID。这通常是一个 URL，并且是应用程序发送的 SAML 请求中预期的发行者值。

5. Click **Save**. This action creates the client and brings you to the **Settings** tab.

单击 Save。此操作将创建客户端并将您带到设置选项卡。

The following sections describe each setting on this tab.

以下各节描述此选项卡上的每个设置。

Settings tab 设置标签

The **Settings** tab includes many options to configure this client.

“设置”选项卡包含许多配置此客户端的选项。

Client settings 客户端设置

The screenshot shows the Keycloak interface with the left sidebar expanded. The 'Clients' section is selected. On the right, the 'SAML' client details page is displayed. The 'General Settings' tab is active. The 'Client ID' field contains 'SAML'. The 'Name' and 'Description' fields are empty. A toggle switch for 'Always display in console' is set to 'Off'. At the bottom are 'Save' and 'Revert' buttons. A vertical sidebar on the right lists other configuration sections: General Settings (selected), Access settings, SAML capabilities, Signature and Encryption, and Logout settings.

General settings 一般设置

Client ID 客户名

The alphanumeric ID string that is used in OIDC requests and in the Keycloak database to identify the client. This value must match the issuer value sent with AuthNRequests. Keycloak pulls the issuer from the Authn SAML request and match it to a client by this value.

用于 OIDC 请求以及用于钥匙斗篷数据库以标识客户端的字母数字 ID 字符串。此值必须与使用 AuthNRequest 发送的发行者值匹配。Key斗篷从 Authn SAML 请求中提取发行者，并通过此值将其与客户端匹配。

Name 姓名

The name for the client in a Keycloak UI screen. To localize the name, set up a replacement string value. For example, a string value such as \${myapp}. See the [Server Developer Guide](#) (https://www.keycloak.org/docs/latest/server_development/) for more information.

钥匙斗篷 UI 屏幕中的客户端名称。若要本地化名称，请设置替换字符串值。例如，一个字符串值，如 \${myapp}。有关更多信息，请参见服务器开发人员指南。

Description 描述

The description of the client. This setting can also be localized.

客户端的描述。此设置也可以本地化。

Always Display in Console 始终在控制台中显示

Always list this client in the Account Console even if this user does not have an active session.

始终在帐户控制台中列出此客户端，即使此用户没有活动会话。

Access Settings 访问设置

Root URL 根网址

When Keycloak uses a configured relative URL, this value is prepended to the URL.

当“钥匙斗篷”使用已配置的相对 URL 时，此值将被预置到 URL 之前。

Home URL 主页网址

If Keycloak needs to link to a client, this URL is used.

如果钥匙斗篷需要链接到客户端，则使用此 URL。

Valid Redirect URIs 有效重定向 URI

Enter a URL pattern and click the + sign to add. Click the - sign to remove. Click **Save** to save these changes. Wildcards values are allowed only at the end of a URL. For example, `http://host.com/*$$.` This field is used when the exact SAML endpoints are not registered and Keycloak pulls the Assertion Consumer URL from a request.

输入一个 URL 模式并单击要添加的 + 号。单击-sign 以删除。单击 Save 保存这些更改。通配符值只允许在 URL 的末尾。比如 `http://host.com/*$$.` 如果没有注册确切的 SAML 端点，并且 Keycloak 从请求中提取断言消费者 URL，则使用此字段。

IDP-Initiated SSO URL name 启动的 SSO URL 名称

URL fragment name to reference client when you want to do IDP Initiated SSO. Leaving this empty will disable IDP Initiated SSO. The URL you will reference from your browser will be: `server-root/realm/{realm}/protocol/saml/clients/{client-url-name}`

当您想要执行 IDP 初始化 SSO 时，URL 片段名称引用客户端。保留此空白将禁用 IDP 初始化的 SSO。您将从浏览器中引用的 URL 将是: `server-root/fields/{ domain }/protocol/saml/client/{ client-URL-name }`

IDP Initiated SSO Relay State 启动 SSO 中继状态

Relay state you want to send with SAML request when you want to do IDP Initiated SSO.

当您要执行 IDPInitiatedSSO 时，要随 SAML 请求发送的中继状态。

Master SAML Processing URL 主 SAML 处理 URL

This URL is used for all SAML requests and the response is directed to the SP. It is used as the Assertion Consumer Service URL and the Single Logout Service URL.

此 URL 用于所有 SAML 请求，响应直接发送到 SP。它用作断言使用者服务 URL 和单一注销服务 URL。

If login requests contain the Assertion Consumer Service URL then those login requests will take precedence. This URL must be validated by a registered Valid Redirect URI pattern.

如果登录请求包含断言使用者服务 URL，那么这些登录请求将优先。此 URL 必须通过已注册的 ValidRedirect URI 模式进行验证。

SAML capabilities SAML 功能

Name ID Format 名称 ID 格式

The Name ID Format for the subject. This format is used if no name ID policy is specified in a request, or if the Force Name ID Format attribute is set to ON.

主题的名称 ID 格式。如果请求中没有指定名称 ID 策略，或者如果 ForceNameIDFormat 属性设置为 ON，则使用此格式。

Force Name ID Format 部队名称 ID 格式

If a request has a name ID policy, ignore it and use the value configured in the Admin Console under Name ID Format.

如果请求具有名称 ID 策略，请忽略它，并使用在“名称 ID 格式”下的管理控制台中配置的值。

Force POST Binding 强制后绑定

By default, Keycloak responds using the initial SAML binding of the original request. By enabling Force POST Binding, Keycloak responds using the SAML POST binding even if the original request used the redirect binding.

默认情况下，Keycon 使用原始请求的初始 SAML 绑定进行响应。通过启用强制 POST 绑定，即使原始请求使用了重定向绑定，Keycon 也会使用 SAML POST 绑定进行响应。

Force artifact binding 强制神器绑定

If enabled, response messages are returned to the client through the SAML ARTIFACT binding system.

如果启用，响应消息将通过 SAML ARTIFACT 绑定系统返回到客户机。

Include AuthnStatement 包括 AuthnStatement

SAML login responses may specify the authentication method used, such as password, as well as timestamps of the login and the session expiration. **Include AuthnStatement** is enabled by default, so that the **AuthnStatement** element will be included in login responses. Setting this to OFF prevents clients from determining the maximum session length, which can create client sessions that do not expire.

SAML 登录响应可以指定所使用的身份验证方法，如密码，以及登录和会话过期的时间戳。默认情况下启用了 **Include AuthnStatement**，因此 **AuthnStatement** 元素将包含在登录响应中。将此设置为 OFF 可防止客户端确定最大会话长度，从而可以创建不会过期的客户端会话。

Include OneTimeUse Condition 包括一次性使用条件

If enable, a OneTimeUse Condition is included in login responses.

如果启用，登录响应中将包含 OneTimeUse 条件。

Optimize REDIRECT signing key lookup 优化 REDIRECT 签名密钥查找

When set to ON, the SAML protocol messages include the Keycloak native extension. This extension contains a hint with the signing key ID. The SP uses the extension for signature validation instead of attempting to validate the signature using keys.

当设置为 ON 时，SAML 协议消息将包含本机扩展 Key 。此扩展包含带有签名密钥 ID 的提示。SP 使用扩展名进行签名验证，而不是试图使用密钥来验证签名。

This option applies to REDIRECT bindings where the signature is transferred in query parameters and this information is not found in the signature information. This is contrary to POST binding messages where key ID is always included in document signature.

此选项适用于 REDIRECT 绑定，其中签名在查询参数中传输，并且在签名信息中找不到此信息。这与 POST 绑定消息相反，在 POST 绑定消息中，密钥 ID 始终包含在文档签名中。

This option is used when Keycloak server and adapter provide the IDP and SP. This option is only relevant when **Sign Documents** is set to ON.

此选项用于 Keycloak 服务器和适配器提供 IDP 和 SP 时。只有在“签名文档”设置为 ON 时，此选项才相关。

Signature and Encryption 签名及加密

Sign Documents 签署文件

When set to ON, Keycloak signs the document using the realms private key.

当设置为 ON 时，Key斗篷使用领域私钥对文档进行签名。

Sign Assertions 签署断言

The assertion is signed and embedded in the SAML XML Auth response.

断言被签名并嵌入到 SAMLXMLAuth 响应中。

Signature Algorithm 签名算法

The algorithm used in signing SAML documents.

用于 SAML 文档签名的算法。

SAML Signature Key Name 签名密钥名称

Signed SAML documents sent using POST binding contain the identification of the signing key in the **KeyName** element. This action can be controlled by the **SAML Signature Key Name** option. This option controls the contents of the **Keyname**.

使用 POST 绑定发送的签名 SAML 文档包含 KeyName 元素中签名密钥的标识。此操作可由 SAML 签名密钥名称选项控制。此选项控制 Keyname 的内容。

- **KEY_ID** The KeyName contains the key ID. This option is the default option.
KEY_ID KeyName 包含密钥 ID。此选项是默认选项。
- **CERT SUBJECT** The KeyName contains the subject from the certificate corresponding to the realm key. This option is expected by Microsoft Active Directory Federation Services.
KeyName 包含与领域密钥对应的证书中的主题。这个选项是微软活动目录联合服务所期待的。
- **NONE** The KeyName hint is completely omitted from the SAML message.
没有从 SAML 消息中完全省略 KeyName 提示。

Canonicalization Method 规范化方法

The canonicalization method for XML signatures.

XML 签名的规范化方法。

Login settings 登录设置

Login theme 登录主题

A theme to use for login, OTP, grant registration, and forgotten password pages.

用于登录、OTP、授予注册和忘记密码页的主题。

Consent required 需要同意

If enabled, users have to consent to client access.

如果启用，用户必须同意客户端访问。

For client-side clients that perform browser logins. As it is not possible to ensure that secrets can be kept safe with client-side clients, it is important to restrict access by configuring correct redirect URLs.

用于执行浏览器登录的客户端客户端。由于不可能确保客户端客户机能够保密，因此通过配置正确的重定向 URI 来限制访问非常重要。

Display client on screen 在屏幕上显示客户端

This switch applies if Consent Required is Off.

如果“必须同意”关闭，则此开关适用。

- **Off**

关掉

The consent screen will contain only the consents corresponding to configured client scopes.

同意屏幕将只包含与配置的客户端范围相对应的同意。

- *On*

开始

There will be also one item on the consent screen about this client itself.

同意屏幕上还会有一条关于这个客户端本身的内容。

Client consent screen text 客户同意屏幕文本

Applies if **Consent required** and **Display client on screen** are enabled. Contains the text that will be on the consent screen about permissions for this client.

如果启用了同意和屏幕上显示客户端，则应用。包含将在同意屏幕上显示的有关此客户端权限的文本。

Logout settings 注销设置

Front channel logout 前台注销

If **Front Channel Logout** is enabled, the application requires a browser redirect to perform a logout. For example, the application may require a cookie to be reset which could only be done via a redirect. If **Front Channel Logout** is disabled, Keycloak invokes a background SAML request to log out of the application.

如果启用了 FrontChannel 注销，则应用程序需要浏览器重定向来执行注销。例如，应用程序可能需要重置 Cookie，而这只能通过重定向来完成。如果前端通道注销被禁用，那么 Keycloak 将调用一个后台 SAML 请求来注销应用程序。

Keys tab 钥匙卡

Encrypt Assertions 加密断言

Encrypts the assertions in SAML documents with the realms private key. The AES algorithm uses a key size of 128 bits.

使用领域私钥加密 SAML 文档中的断言。AES 算法使用的密钥大小为128位。

Client Signature Required 需要客户签名

If **Client Signature Required** is enabled, documents coming from a client are expected to be signed. Keycloak will validate this signature using the client public key or cert set up in the **Keys** tab.

如果启用客户端签名要求，来自客户端的文档将被签名。“钥匙斗篷”将使用“钥匙”选项卡中设置的客户端公钥或证书来验证此签名。

Allow ECP Flow 允许 ECP 流动

If true, this application is allowed to use SAML ECP profile for authentication.

如果为 true，则允许此应用程序使用 SAML ECP 配置文件进行身份验证。

Advanced tab 高级标签

This tab has many fields for specific situations. Some fields are covered in other topics. For details on other fields, click the question mark icon.

此选项卡有许多特定情况的字段。其他主题涵盖了一些字段。有关其他字段的详细信息，请单击问号图标。

Fine Grain SAML Endpoint Configuration 细粒度 SAML 端点配置

Logo URL 标志网址

URL that references a logo for the Client application.

引用客户端应用程序徽标的 URL。

Policy URL 政策网址

URL that the Relying Party Client provides to the End-User to read about how the profile data will be used.

依赖方客户端提供给最终用户的 URL，用于读取将如何使用配置文件数据。

Terms of Service URL 服务条款网址

URL that the Relying Party Client provides to the End-User to read about the Relying Party's terms of service.

信赖方客户端提供给最终用户的 URL，用于阅读信赖方的服务条款。

Assertion Consumer Service POST Binding URL 断言消费者服务 POST 绑定 URL

POST Binding URL for the Assertion Consumer Service.

断言使用者服务的 POST 绑定 URL。

Assertion Consumer Service Redirect Binding URL 断言使用者服务重定向绑定 URL

Redirect Binding URL for the Assertion Consumer Service.

为断言使用者服务重定向绑定 URL。

Logout Service POST Binding URL 注销服务 POST 绑定 URL

POST Binding URL for the Logout Service.

注销服务的 POST 绑定 URL。

Logout Service Redirect Binding URL 注销服务重定向绑定 URL

Redirect Binding URL for the Logout Service.

重定向注销服务的绑定 URL。

Logout Service Artifact Binding URL **注销服务工件绑定 URL**

Artifact Binding URL for the Logout Service. When set together with the **Force Artifact Binding** option, *Artifact* binding is forced for both login and logout flows. *Artifact* binding is not used for logout unless this property is set.

注销服务的工件绑定 URL。当与强制工件绑定选项一起设置时，工件绑定对于登录和注销流都是强制的。工件绑定不用于注销，除非设置了此属性。

Artifact Binding URL **工件绑定 URL**

URL to send the HTTP artifact messages to.

将 HTTP 工件消息发送到的 URL。

Artifact Resolution Service **文物辨认服务**

URL of the client SOAP endpoint where to send the **ArtifactResolve** messages to.

客户端 SOAP 端点的 URL，将 **ArtifactResolve** 消息发送到该端点。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/clients/saml/idp-initiated-login.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/clients/saml/idp-initiated-login.adoc)

IDP Initiated Login is a feature that allows you to set up an endpoint on the Keycloak server that will log you into a specific application/client. In the **Settings** tab for your client, you need to specify the **IDP Initiated SSO URL Name**. This is a simple string with no whitespace in it. After this you can reference your client at the following URL: `root/realms/{realm}/protocol/saml/clients/{url-name}`

IDP Initiated Login 是一个特性，它允许您在钥匙斗篷服务器上设置一个端点，该端点将使您登录到特定的应用程序/客户端。在客户端的 Settings 选项卡中，需要指定 IDP Initiated SSO URL Name。这是一个没有空格的简单字符串。在此之后，您可以在以下 URL 引用您的客户端：根/领域/{ domain }/protocol/saml/client/{ URL-name }

The IDP initiated login implementation prefers *POST* over *REDIRECT* binding (check saml bindings for more information). Therefore the final binding and SP URL are selected in the following way:

IDP 发起的登录实现更喜欢 POST 而不是 REDIRECT 绑定(检查 saml 绑定以获得更多信息)。因此，最终绑定和 SP URL 的选择方式如下：

1. If the specific Assertion Consumer Service **POST Binding URL** is defined (inside **Fine Grain SAML Endpoint Configuration** section of the client settings) *POST* binding is used through that URL.

如果定义了特定的断言消费者服务 POST 绑定 URL (在客户端设置的精细粒度 SAML 端点配置部分中)，则通过该 URL 使用 POST 绑定。

2. If the general **Master SAML Processing URL** is specified then *POST* binding is used again throughout this general URL.

如果指定了通用的 MasterSAML 处理 URL，则在整个通用 URL 中再次使用 POST 绑定。

3. As the last resort, if the **Assertion Consumer Service Redirect Binding URL** is configured (inside **Fine Grain SAML Endpoint Configuration**) *REDIRECT* binding is used with this URL.

作为最后的手段，如果配置了断言消费者服务重定向绑定 URL (在精细粒度 SAML 端点配置中)，则该 URL 将使用 REDIRECT 绑定。

If your client requires a special relay state, you can also configure this on the **Settings** tab in the **IDP Initiated SSO Relay State** field. Alternatively, browsers can specify the relay state in a **RelayState** query parameter, i.e. `root/realm/{realm}/protocol/saml/clients/{url-name}?RelayState=thestate`.

如果您的客户端需要一个特殊的中继状态，您也可以在 IDP Initiated SSO Relay State 字段的 Settings 选项卡上配置它。或者，浏览器可以在 RelayState 查询参数中指定中继状态，也就是根/领域/{ domain }/protocol/saml/client/{ url-name } ? RelayState = 状态。

When using identity brokering, it is possible to set up an IDP Initiated Login for a client from an external IDP. The actual client is set up for IDP Initiated Login at broker IDP as described above. The external IDP has to set up the client for application IDP Initiated Login that will point to a special URL pointing to the broker and representing IDP Initiated Login endpoint for a selected client at the brokering IDP. This means that in client settings at the external IDP:

在使用身份代理时，可以从外部 IDP 为客户端设置 IDP 初始登录。实际的客户机是在代理 IDP 上为 IDPInitiatedLogin 设置的，如上所述。外部 IDP 必须为应用程序 IDP Initiated Login 设置客户端，该客户端将指向一个特殊的 URL，该 URL 指向代理，并表示代理 IDP 中所选客户端的 IDP Initiated Login 端点。这意味着在外部 IDP 的客户端设置中：

- **IDP Initiated SSO URL Name** is set to a name that will be published as IDP Initiated Login initial point,
IDP Initiated SSO URL Name 被设置为将作为 IDP Initiated Login 起始点发布的名称,
- **Assertion Consumer Service POST Binding URL** in the **Fine Grain SAML Endpoint Configuration** section has to be set to the following URL: `broker-root/realm/{broker-realm}/broker/{idp-name}/endpoint/clients/{client-id}` , where:

细粒度 SAML 端点配置部分中的断言消费者服务 POST 绑定 URL 必须设置为以下 URL:

- *broker-root* is base broker URL

Broker-root 是基本的 Broker URL

- *broker-realm* is name of the realm at broker where external IDP is declared

代理领域是在代理中声明外部 IDP 的领域的名称

- *idp-name* is name of the external IDP at broker
IDP-name 是代理上的外部 IDP 的名称
- *client-id* is the value of IDP Initiated SSO URL Name attribute of the SAML client defined at broker.
It is this client, which will be made available for IDP Initiated Login from the external IDP.
Client-id 是在代理上定义的 SAML 客户机的 IDP Initiated SSO URL Name 属性的值。正是这个客户机，将从外部 IDP 提供 IDP 初始化登录。

Please note that you can import basic client settings from the brokering IDP into client settings of the external IDP - just use SP Descriptor available from the settings of the identity provider in the brokering IDP, and add `clients/client-id` to the endpoint URL.

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——只需使用可从代理 IDP 中的身份提供者设置中获得的 SP 描述符，并将客户端/客户端 ID 添加到端点 URL 中。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/clients/saml/proc-using-an-entity-descriptor.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/clients/saml/proc-using-an-entity-descriptor.adoc)

Instead of registering a SAML 2.0 client manually, you can import the client using a standard SAML Entity Descriptor XML file.

您可以使用标准的 SAML 实体描述符 XML 文件导入客户机，而不必手动注册 SAML 2.0 客户机。

The Client page includes an **Import client** option.

Client 页面包含一个 Import 客户端选项。

Add client 添加客户端

The screenshot shows the Keycloak administration interface under the 'Clients' section. On the left sidebar, 'Clients' is selected. The main area is titled 'Import client' with the sub-instruction 'Clients are applications and services that can request authentication of a user.' Below this, there's a 'Resource file' input field containing a JSON file named 'mysamlapp.json' which defines a client with ID 'mysamlapp'. There are also fields for 'Client ID' (set to 'mysamlapp'), 'Name', 'Description', 'Type' (set to 'saml'), and two toggle switches for 'Encrypt assertions' (Off) and 'Client signature' (On).

Procedure 程序

1. Click **Browse**.

点击浏览。

2. Load the file that contains the XML entity descriptor information.

加载包含 XML 实体描述符信息的文件。

3. Review the information to ensure everything is set up correctly.

检查信息以确保一切正确设置。

Some SAML client adapters, such as *mod-auth-mellon*, need the XML Entity Descriptor for the IDP. You can find this descriptor by going to this URL:

一些 SAML 客户机适配器(如 mod-auth-mellon)需要用于 IDP 的 XML 实体描述符。你可以通过这个 URL 找到这个描述符:

```
root/realms/{realm}/protocol/saml/descriptor
```

where *realm* is the realm of your client.

其中域是您的客户端的域。

[Edit this section 编辑这部分](#)
(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/clients/con-client-links.adoc)

[Report an issue 报告一个问题](#)
(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/clients/con-client-links.adoc)

To link from one client to another, Keycloak provides a redirect endpoint:

`/realms/realm_name/clients/{client-id}/redirect`.

为了从一个客户端链接到另一个客户端，Keycloak 提供了一个重定向端点: `/realms/realm_name/clients/{client-id}/redirect`.

If a client accesses this endpoint using a `HTTP GET` request, Keycloak returns the configured base URL for the provided Client and Realm in the form of an `HTTP 307 (Temporary Redirect)` in the response's `Location` header. As a result of this, a client needs only to know the Realm name and the Client ID to link to them. This indirection avoids hard-coding client base URLs.

如果客户端使用 `HTTP GET` 请求访问这个端点，在响应的 `Location` 报头中，Keycloak 以 `HTTP 307 (临时重定向)` 的形式返回所提供的 Client 和 Domain 的已配置的基 URL。因此，客户端只需要知道域名和客户端 ID 就可以链接到它们。这种间接方式避免了硬编码客户机基 URL。

As an example, given the realm `master` and the client-id `account`:

例如，给定领域主机和客户机 ID 帐户：

```
http://host:port/realms/master/clients/account/redirect
```

This URL temporarily redirects to: `http://host:port/realms/master/account`

这个网址会暂时重定向至: `http://host:port/realms/master/account`

[Edit this section](#) 编辑这部分

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/clients/con-protocol-mappers.adoc)

[Report an issue](#) 报告一个问题

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/clients/con-protocol-mappers.adoc)

Applications receiving ID tokens, access tokens, or SAML assertions may require different roles and user metadata.

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。

You can use Keycloak to:

你可以使用钥匙斗篷：

- Hardcode roles, claims and custom attributes.

硬编码角色、索赔和自定义属性。

- Pull user metadata into a token or assertion.

将用户元数据拖放到令牌或断言中。

- Rename roles.

重命名角色。

You perform these actions in the **Mappers** tab in the Admin Console.

您可以在管理控制台的 Mappers 选项卡中执行这些操作。

Mappers tab 地图标签

New clients do not have built-in mappers but they can inherit some mappers from client scopes. See the client scopes section for more details.

新客户机没有内置映射器，但是它们可以从客户机作用域继承一些映射器。有关详细信息，请参阅客户端范围一节。

Protocol mappers map items (such as an email address, for example) to a specific claim in the identity and access token. The function of a mapper should be self-explanatory from its name. You add pre-configured mappers by clicking **Add Builtin**.

协议映射器将项(例如电子邮件地址)映射到标识和访问令牌中的特定声明。映射器的功能名称应该是不言而喻的。通过单击 AddBuiltin 添加预配置的映射器。

Each mapper has a set of common settings. Additional settings are available, depending on the mapper type. Click **Edit** next to a mapper to access the configuration screen to adjust these settings.

每个映射器都有一组公共设置。根据映射器类型，还可以使用其他设置。单击映射器旁边的 **Edit** 访问配置屏幕以调整这些设置。

Mapper config 映射器配置

Mapper type: User Realm Role

Name:

Realm Role prefix:

Multivalued: On

Token Claim Name:

Claim JSON Type: String

Add to ID token: On

Add to access token: On

Add to userinfo: On

Details on each option can be viewed by hovering over its tooltip.

可以通过将鼠标悬停在其工具提示上查看每个选项的详细信息。

You can use most OIDC mappers to control where the claim gets placed. You opt to include or exclude the claim from the *id* and *access* tokens by adjusting the **Add to ID token** and **Add to access token** switches.

您可以使用大多数 OIDC 映射器来控制声明放置的位置。通过调整 Add to ID 令牌和 Add to access 令牌开关，您可以选择从 ID 和访问令牌中包含或排除声明。

You can add mapper types as follows:

您可以添加映射器类型如下：

Procedure 程序

1. Go to the **Mappers** tab.

转到 Mapper 选项卡。

2. Click **Configure a new mapper**.

单击“配置新映射器”。

Add mapper 添加映射器

Master

Manage

Clients

Client scopes

Realm roles

Users

Groups

Sessions

Events

Configure

Realm settings

Authentication

Clients > Client details > Dedicated scopes > Mapper details

Add mapper

If you want more fine-grain control, you can create protocol mapper on this client

Mapper type	Group Membership
Name *	
Token Claim Name	
Full group path	<input checked="" type="checkbox"/> Off
Add to ID token	<input checked="" type="checkbox"/> Off
Add to access token	<input checked="" type="checkbox"/> Off
Add to userinfo	<input checked="" type="checkbox"/> Off

Save Cancel

3. Select a Mapper Type from the list box.

[Edit this section](#) 编辑这部分

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/clients/proc-creating-mappers.adoc)

[Report an issue](#) 报告一个问题

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/clients/proc-creating-mappers.adoc)

Priority order 优先顺序

Mapper implementations have *priority order*. *Priority order* is not the configuration property of the mapper. It is the property of the concrete implementation of the mapper.

映射器实现具有优先级顺序。优先级顺序不是映射器的配置属性。它是映射器具体实现的属性。

Mappers are sorted by the order in the list of mappers. The changes in the token or assertion are applied in that order with the lowest applying first. Therefore, the implementations that are dependent on other implementations are processed in the necessary order.

映射器按照映射器列表中的顺序排序。令牌或断言中的更改按照这个顺序应用，最小的更改首先应用。因此，依赖于其他实现的实现将按照必要的顺序进行处理。

For example, to compute the roles which will be included with a token:

例如，要计算将包含在令牌中的角色：

1. Resolve audiences based on those roles.

根据这些角色解析受众。

2. Process a JavaScript script that uses the roles and audiences already available in the token.

处理使用令牌中已有的角色和访问者的JavaScript脚本。

OIDC user session note mappers OIDC 用户会话注释映射器

User session details are defined using mappers and are automatically included when you use or enable a feature on a client. Click **Add builtin** to include session details.

用户会话详细信息是使用映射器定义的，当您在客户机上使用或启用某个特性时，会自动包含用户会话详细信息。单击 **Add builtin** 以包含会话详细信息。

Impersonated user sessions provide the following details:

模拟用户会话提供以下详细信息:

- **IMPERSONATOR_ID**: The ID of an impersonating user.

IMPERSONATOR_ID: 模拟用户的 ID。

- **IMPERSONATOR_USERNAME**: The username of an impersonating user.

IMPERSONATOR_USERNAME: 模拟用户的用户名。

Service account sessions provide the following details:

服务帐户会议提供以下详情:

- **clientId**: The client ID of the service account.

ClientId: 服务帐户的客户端 ID。

- **clientAddress**: The remote host IP of the service account's authenticated device.

ClientAddress: 服务帐户的已验证设备的远程主机 IP。

- **clientHost**: The remote host name of the service account's authenticated device.

ClientHost: 服务帐户的已验证设备的远程主机名。

Script mapper 脚本映射器

Use the **Script Mapper** to map claims to tokens by running user-defined JavaScript code. For more details about deploying scripts to the server, see [JavaScript Providers](#) (https://www.keycloak.org/docs/latest/server_development/#_script_providers).

通过运行用户定义的 JavaScript 代码，使用 ScriptMapper 将声明映射到令牌。有关将脚本部署到服务器的详细信息，请参见 JavaScript 提供程序。

When scripts deploy, you should be able to select the deployed scripts from the list of available mappers.

当脚本部署时，您应该能够从可用映射器列表中选择已部署的脚本。

[Report an issue](#) 报告一个问题

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/clients/proc-generating-client-adapter-config.adoc)

Keycloak can generate configuration files that you can use to install a client adapter in your application's deployment environment. A number of adapter types are supported for OIDC and SAML.

密钥斗篷可以生成配置文件，您可以使用这些文件在应用程序的部署环境中安装客户端适配器。OIDC 和 SAML 支持许多适配器类型。

1. Click on the **Action** menu and select the **Download adapter config** option

单击 Action 菜单并选择“下载适配器配置”选项

Download adaptor configs

description

```
keycloak.json file used by the Keycloak OIDC client adapter to configure clients. This must be saved to a keycloak.json file and put in your WEB-INF directory of your WAR file. You may also want to tweak this file after you download it.
```

Format option ⓘ

Keycloak OIDC JSON

Details ⓘ

```
{  
  "realm": "master",  
  "auth-server-url": "http://localhost:8180/",  
  "ssl-required": "external",  
  "resource": "myapp",  
  "credentials": {  
    "secret": "36gLgP28Ak4Czp9o3JetORPOqCZQ3jwX"  
  },  
  "confidential-port": 0  
}
```

Download **Cancel**

2. Select the **Format Option** you want configuration generated for.

选择要为其生成配置的格式选项。

All Keycloak client adapters for OIDC and SAML are supported. The mod-auth-mellon Apache HTTPD adapter for SAML is supported as well as standard SAML entity descriptor files.

支持用于 OIDC 和 SAML 的所有 Keycloak 客户端适配器。支持用于 SAML 的 mod-auth-mellon Apache HTTPD 适配器以及标准 SAML 实体描述符文件。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/clients/con-client-scopes.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/clients/con-client-scopes.adoc)

Use Keycloak to define a shared client configuration in an entity called a *client scope*. A *client scope* configures protocol mappers and role scope mappings for multiple clients.

在称为客户端作用域的实体中，使用 Keycloak 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。

Client scopes also support the OAuth 2 **scope** parameter. Client applications use this parameter to request claims or roles in the access token, depending on the requirement of the application.

客户端作用域还支持 OAuth 2 作用域参数。客户端应用程序使用此参数请求访问令牌中的声明或角色，具体取决于应用程序的需求。

To create a client scope, follow these steps:

要创建客户端范围，请执行以下步骤：

1. Click **Client Scopes** in the menu.

在菜单中单击“客户端作用域”。

Client scopes list 客户端范围列表

Name	Assigned type	Protocol	Display order	Description
acr	Default	OpenID Connect	-	OpenID Connect scope for add acr (authentication context class reference) to the token
address	Optional	OpenID Connect	-	OpenID Connect built-in scope: address
email	Default	OpenID Connect	-	OpenID Connect built-in scope: email
micropause-jwt	Optional	OpenID Connect	-	Micropause - JWT built-in scope

2. Click **Create**.

单击 Create。

3. Name your client scope.

命名客户端范围。

4. Click Save.

单击“保存”。

A *client scope* has similar tabs to regular clients. You can define protocol mappers and role scope mappings. These mappings can be inherited by other clients and are configured to inherit from this client scope.

客户端作用域与常规客户端类似。您可以在协议映射器和角色范围映射器上进行配置。此作用域可以由其他客户端继承。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/clients/proc-creating-client-scopes.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/clients/proc-creating-client-scopes.adoc)

Protocol 规定

When you create a client scope, choose the **Protocol**. Clients linked in the same scope must have the same protocol.

创建客户端作用域时，请选择 Protocol。在同一作用域中链接的客户端必须具有相同的协议。

Each realm has a set of pre-defined built-in client scopes in the menu.

每个领域在菜单中都有一组预定义的内置客户端作用域。

- SAML protocol: The **role_list**. This scope contains one protocol mapper for the roles list in the SAML assertion.

SAML 协议: role _ list。此范围包含 SAML 断言中角色列表的一个协议映射器。

- OpenID Connect protocol: Several client scopes are available:

OpenID 连接协议: 有几个客户端范围可用:

- **roles**

角色

This scope is not defined in the OpenID Connect specification and is not added automatically to the **scope** claim in the access token. This scope has mappers, which are used to add the roles of the user to the access token and add audiences for clients that have at least one client role. These mappers are described in more detail in the Audience section.

此范围未在 OpenIDConnect 规范中定义，也未自动添加到访问令牌中的范围声明中。此作用域具有映射器，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访问群体。这些映射器将在 Audience 部分进行更详细的描述。

- **web-origins**

网络起源

This scope is also not defined in the OpenID Connect specification and not added to the **scope** claiming the access token. This scope is used to add allowed web origins to the access token **allowed-origins** claim.

这个作用域也没有在 OpenIDConnect 规范中定义，也没有添加到声明访问令牌的作用域中。此作用域用于将允许的 Web 起源添加到访问令牌允许起源声明中。

- **microprofile-jwt**

微轮廓

This scope handles claims defined in the [MicroProfile/JWT Auth Specification](https://wiki.eclipse.org/MicroProfile/JWT_Auth_Specification) (https://wiki.eclipse.org/MicroProfile/JWT_Auth). This scope defines a user property mapper for the **upn** claim and a realm role mapper for the **groups** claim. These mappers can be changed so different properties can be used to create the MicroProfile/JWT specific claims.

此范围处理 MicroProfile/JWT Auth 规范中定义的声明。此范围为 upn 声明定义了一个用户属性映射器，为组声明定义了一个领域角色映射器。可以更改这些映射程序，以便可以使用不同的属性来创建特定于 MicroProfile/JWT 的声明。

- **offline_access**

离线访问

This scope is used in cases when clients need to obtain offline tokens. More details on offline tokens is available in the Offline Access section and in the [OpenID Connect specification](https://openid.net/specs/openid-connect-core-1_0.html#OfflineAccess) (https://openid.net/specs/openid-connect-core-1_0.html#OfflineAccess).

此范围用于客户端需要获取脱机令牌的情况。有关脱机令牌的详细信息，请参阅脱机访问部分和 OpenIDConnect 规范。

- **profile**

侧写

- **email**

电子邮件

- **address**

地址

- **phone**

电话

The client scopes **profile**, **email**, **address** and **phone** are defined in the [OpenID Connect specification](https://openid.net/specs/openid-connect-core-1_0.html#ScopeClaims) (https://openid.net/specs/openid-connect-core-1_0.html#ScopeClaims). These scopes do not have any role scope mappings defined but they do have protocol mappers defined. These mappers correspond to the

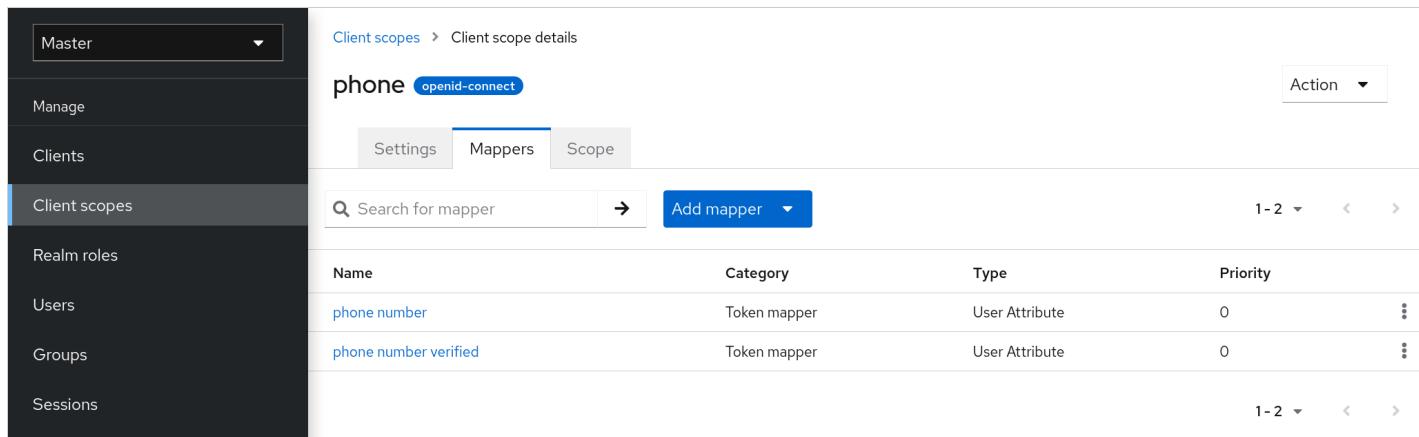
claims defined in the OpenID Connect specification.

客户端范围配置文件、电子邮件、地址和电话在 OpenIDConnect 规范中定义。这些作用域没有定义任何角色作用域映射，但是它们定义了协议映射。这些映射器对应于 OpenIDConnect 规范中定义的声明。

For example, when you open the **phone** client scope and open the **Mappers** tab, you will see the protocol mappers which correspond to the claims defined in the specification for the scope **phone**.

例如，当您打开手机客户机范围并打开 Mappers 选项卡时，您将看到与范围手机规范中定义的声明相对应的协议映射程序。

Client scope mappers 客户端范围映射程序



The screenshot shows the 'Client scopes' section of a management interface. The left sidebar has a 'Client scopes' item selected. The main area shows the 'phone' scope details, specifically the 'Mappers' tab. It includes a search bar, an 'Add mapper' button, and a table listing two mappers:

Name	Category	Type	Priority
phone number	Token mapper	User Attribute	0
phone number verified	Token mapper	User Attribute	0

When the **phone** client scope is linked to a client, the client automatically inherits all the protocol mappers defined in the **phone** client scope. Access tokens issued for this client contain the phone number information about the user, assuming that the user has a defined phone number.

当电话客户端作用域链接到客户端时，客户端将自动继承在电话客户端作用域中定义的所有协议映射器。为此客户机发出的访问令牌包含有关该用户的电话号码信息，假设该用户具有已定义的电话号码。

Built-in client scopes contain the protocol mappers as defined in the specification. You are free to edit client scopes and create, update, or remove any protocol mappers or role scope mappings.

内置的客户机作用域包含规范中定义的协议映射器。您可以自由地编辑客户端范围并创建、更新或删除任何协议映射器或角色范围映射。

Consent related settings 与同意有关的设定

Client scopes contain options related to the consent screen. Those options are useful if the linked client if **Consent Required** is enabled on the client.

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链接的客户端(如果需要同意)，那么这些选项非常有用。

Display On Consent Screen 在同意屏幕上显示

If **Display On Consent Screen** is enabled, and the scope is added to a client that requires consent, the text specified in **Consent Screen Text** will be displayed on the consent screen. This text is shown when the user is authenticated and before the user is redirected from Keycloak to the client. If **Display On Consent Screen** is disabled, this client scope will not be displayed on the consent screen.

如果启用了“在同意屏幕上显示”，并将范围添加到需要同意的客户端，则“同意屏幕文本”中指定的文本将显示在同意屏幕上。当用户经过身份验证后，并在用户从 Keycloak 重定向到客户端之前显示此文本。如果禁用“在同意屏幕上显示”，此客户端范围将不会显示在同意屏幕上。

Consent Screen Text 同意画面文字

The text displayed on the consent screen when this client scope is added to a client when consent required defaults to the name of client scope. The value for this text can be customised by specifying a substitution variable with `${var-name}` strings. The customised value is configured within the property files in your theme. See the [Server Developer Guide](#) (https://www.keycloak.org/docs/latest/server_development/) for more information on customisation.

当这个客户端范围被添加到需要同意的客户端时，同意屏幕上显示的文本默认为客户端范围的名称。可以通过使用 `${ var-name }` 字符串指定替换变量来自定义此文本的值。自定义值在主题的属性文件中配置。有关自定义的更多信息，请参见服务器开发人员指南。

Link client scope with the client 将客户端作用域与客户端链接起来

Linking between a client scope and a client is configured in the **Client Scopes** tab of the client. Two ways of linking between client scope and client are available.

客户端作用域和客户端之间的链接是在客户端的 ClientScope 选项卡中配置的。有两种连接客户端作用域和客户端的方法。

Default Client Scopes 默认客户端作用域

This setting is applicable to the OpenID Connect and SAML clients. Default client scopes are applied when issuing OpenID Connect tokens or SAML assertions for a client. The client will inherit Protocol Mappers and Role Scope Mappings that are defined on the client scope. For the OpenID Connect Protocol, the Mappers and Role Scope Mappings are always applied, regardless of the value used for the scope parameter in the OpenID Connect authorization request.

此设置适用于 OpenIDConnect 和 SAML 客户端。当为客户机发出 OpenIDConnect 令牌或 SAML 断言时，将应用默认客户机作用域。客户端将继承在客户端作用域上定义的 Protocol Mapper 和 Role Scope 映射。对于 OpenID Connect 协议，映射器和角色作用域映射始终应用，而不管 OpenID Connect 授权请求中作用域参数的值是什么。

Optional Client Scopes 可选的客户端作用域

This setting is applicable only for OpenID Connect clients. Optional client scopes are applied when issuing tokens for this client but only when requested by the `scope` parameter in the OpenID Connect authorization request.

此设置仅适用于 OpenIDConnect 客户端。可选的客户端范围在为此客户端发出令牌时应用，但只有在 OpenIDConnect 授权请求中的 scope 参数请求时才应用。

Example 例子

For this example, assume the client has **profile** and **email** linked as default client scopes, and **phone** and **address** linked as optional client scopes. The client uses the value of the scope parameter when sending a request to the OpenID Connect authorization endpoint.

对于这个示例，假设客户端将配置文件和电子邮件链接为默认客户端范围，将电话和地址链接为可选的客户端范围。客户机在向 OpenIDConnect 授权端点发送请求时使用 scope 参数的值。

```
scope=openid phone
```

BASH

The scope parameter contains the string, with the scope values divided by spaces. The value **openid** is the meta-value used for all OpenID Connect requests. The token will contain mappers and role scope mappings from the default client scopes **profile** and **email** as well as **phone**, an optional client scope requested by the scope parameter.

Scope 参数包含字符串，其范围值除以空格。OpenID 值是用于所有 OpenIDConnect 请求的元值。令牌将包含来自默认客户机范围配置文件、电子邮件和电话的映射器和角色范围映射，这是 scope 参数请求的一个可选客户机范围。

Evaluating Client Scopes 评估客户范围

The **Mappers** tab contains the protocol mappers and the **Scope** tab contains the role scope mappings declared for this client. They do not contain the mappers and scope mappings inherited from client scopes. It is possible to see the effective protocol mappers (that is the protocol mappers defined on the client itself as well as inherited from the linked client scopes) and the effective role scope mappings used when generating a token for a client.

Mappers 选项卡包含协议映射器，Scope 选项卡包含为此客户机声明的角色范围映射。它们不包含从客户端作用域继承的映射器和作用域映射。可以看到有效的协议映射器(即在客户机本身上定义的以及从链接的客户机作用域继承的协议映射器)和在为客户机生成令牌时使用的有效角色作用域映射。

Procedure 程序

1. Click the **Client Scopes** tab for the client.

单击客户端的 ClientScope 选项卡。

2. Open the sub-tab **Evaluate**.

打开了选项卡“评估”。

3. Select the optional client scopes that you want to apply.

选择要应用的可选客户端作用域。

This will also show you the value of the **scope** parameter. This parameter needs to be sent from the application to the Keycloak OpenID Connect authorization endpoint.

这还将显示 scope 参数的值。这个参数需要从应用程序发送到 KeycloakOpenIDConnect 授权端点。

Evaluating client scopes 评估客户端范围

The screenshot shows the Keycloak Admin UI with the 'Clients' menu item selected. On the right, the 'Client details' page for a client named 'account' (OpenID Connect) is displayed. The 'Client scopes' tab is selected, and the 'Evaluate' sub-tab is active. A user 'john' is selected. The results pane shows a JSON access token with various claims like exp, iat, jti, iss, sub, and tvp. There are also links to 'Effective protocol mappers', 'Effective role scope mappings', and 'Generated access token'.

[Edit this section 编辑这部分](#)
(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/clients/proc-evaluating-client-scopes.adoc)

[Report an issue 报告一个问题](#)
(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/clients/proc-evaluating-client-scopes.adoc)



To send a custom value for a **scope** parameter from your application, see the [parameters forwarding section](#) (https://www.keycloak.org/docs/latest/securing_apps/#_params_forwarding), for servlet adapters or the [javascript adapter section](#) (https://www.keycloak.org/docs/latest/securing_apps/#_javascript_adapter), for javascript adapters.

要从应用程序发送作用域参数的自定义值，请参阅有关 servlet 适配器的参数转发部分或有关 javascript 适配器的参数转发部分。

All examples are generated for the particular user and issued for the particular client, with the specified value of the **scope** parameter. The examples include all of the claims and role mappings used.

所有示例都是为特定用户生成的，并为特定客户机发出，其中包含 scope 参数的指定值。这些示例包括所有使用的声明和角色映射。

Client scopes permissions 客户端范围权限

When issuing tokens to a user, the client scope applies only if the user is permitted to use it.

当向用户发出令牌时，客户端范围只有在用户被允许使用时才应用。

When a client scope does not have any role scope mappings defined, each user is permitted to use this client scope. However, when a client scope has role scope mappings defined, the user must be a member of at least one of the roles. There must be an intersection between the user roles and the roles of the client scope. Composite roles are factored into evaluating this intersection.

当客户端作用域没有定义任何角色作用域映射时，允许每个用户使用此客户端作用域。但是，当客户端范围定义了角色范围映射时，用户必须是至少一个角色的成员。用户角色和客户端范围的角色之间必须有交集。评估这个交集时考虑了复合角色。

If a user is not permitted to use the client scope, no protocol mappers or role scope mappings will be used when generating tokens. The client scope will not appear in the *scope* value in the token.

如果不允许用户使用客户端作用域，则在生成令牌时将不使用协议映射器或角色作用域映射。客户端作用域将不会出现在令牌中的作用域值中。

Realm default client scopes 域默认客户端作用域

Use **Realm Default Client Scopes** to define sets of client scopes that are automatically linked to newly created clients.

使用“域默认客户端作用域”定义自动链接到新创建的客户端的客户端作用域集。

Procedure 程序

1. Click the **Client Scopes** tab for the client.

单击客户端的 ClientScope 选项卡。

From here, select the client scopes that you want to add as **Default Client Scopes** to newly created clients and **Optional Client Scopes**.

从这里，选择要作为默认客户端作用域添加到新创建的客户端和可选客户端作用域的客户端作用域。

Default client scopes 默认客户端作用域

The screenshot shows the Keycloak administration interface under the 'Clients' section. A sidebar on the left lists various management options like 'Manage', 'Clients', 'Client scopes', 'Realm roles', 'Users', 'Groups', 'Sessions', 'Events', 'Configure', 'Realm settings', 'Authentication', 'Identity providers', and 'User federation'. The 'Clients' option is currently selected. In the main content area, a client named 'myclient' (OpenID Connect) is being edited. The 'Client scopes' tab is active, and the 'Setup' sub-tab is selected. A table displays the assigned client scopes for this client. The table has columns for 'Assigned client scope', 'Assigned type', and 'Description'. The rows show the following assignments:

Assigned client scope	Assigned type	Description
myclient-dedicated	none	Dedicated scope and mappers for this client
acr	Default	OpenID Connect scope for add acr (authentication context class reference) to the token
email	Default	OpenID Connect built-in scope: email
profile	Default	OpenID Connect built-in scope: profile
roles	Default	OpenID Connect scope for add user roles to the access token
web-origins	Default	OpenID Connect scope for add allowed web origins to the access token

When a client is created, you can unlink the default client scopes, if needed. This is similar to removing Default Roles.

[Edit this section](#) 编辑这部分
(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/clients/proc-updating-default-scopes.adoc)

[Report an issue](#) 报告一个问题
(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/clients/proc-updating-default-scopes.adoc)

Scopes explained 解释了范围

Client scope 客户端范围

Client scopes are entities in Keycloak that are configured at the realm level and can be linked to clients. Client scopes are referenced by their name when a request is sent to the Keycloak authorization endpoint with a corresponding value of the **scope** parameter. See the client scopes linking section for more details.

客户端作用域是在 Keycloak 中配置的实体，可以链接到客户端。当请求发送到具有对应的 scope 参数值的 Keycloak 授权终结点时，客户端作用域通过其名称引用。有关详细信息，请参阅客户端作用域链接部分。

Role scope mapping 角色范围映射

This is available under the **Scope** tab of a client or client scope. Use **Role scope mapping** to limit the roles that can be used in the access tokens. See the Role Scope Mappings section for more details.

这在客户端或客户端范围的 Scope 选项卡下可用。使用角色范围映射来限制可以在访问令牌中使用的角色。有关更多详细信息，请参见“角色范围映射”部分。

Authorization scopes 授权范围

The Authorization Scope covers the actions that can be performed in the application. See the [Authorization Services Guide](https://www.keycloak.org/docs/latest/authorization_services/) (https://www.keycloak.org/docs/latest/authorization_services/) for more details.

授权范围涵盖了可以在应用程序中执行的操作。有关详细信息，请参阅[授权服务指南](#)。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/clients/client-policies.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/clients/client-policies.adoc)

To make it easy to secure client applications, it is beneficial to realize the following points in a unified way.

为了方便客户端应用程序的安全，统一实现以下几点是有益的。

- Setting policies on what configuration a client can have
设置客户端可以拥有的配置的策略
- Validation of client configurations
验证客户端配置
- Conformance to a required security standards and profiles such as Financial-grade API (FAPI)
符合所需的安全标准和配置文件，例如金融级 API (FAPI)

To realize these points in a unified way, *Client Policies* concept is introduced.

为了统一实现这些要点，引入了客户端策略的概念。

Use-cases 用例

Client Policies realize the following points mentioned as follows.

客户端策略实现下列要点。

Setting policies on what configuration a client can have 设置客户端可以具有的配置的策略

Configuration settings on the client can be enforced by client policies during client creation/update, but also during OpenID Connect requests to Keycloak server, which are related to particular client.

Keycloak supports similar thing also through the Client Registration Policies described in the [Securing Applications and Services Guide](#)

(https://www.keycloak.org/docs/latest/securing_apps/#_client_registration_policies). However, Client Registration Policies can only cover OIDC Dynamic Client Registration. Client Policies cover not only what Client Registration Policies can do, but other client registration and configuration ways. The current plans are for Client Registration to be replaced by Client Policies.

客户端上的配置设置可以在客户端创建/更新期间由客户端策略强制执行，也可以在 OpenID Connect 请求到 Keycover 服务器期间强制执行，这些请求与特定的客户端相关。Keycon 还通过安全应用程序和服务指南中描述的客户端注册策略支持类似的东西。但是，客户端注册策略只能覆盖 OIDC 动态客户端注册。客户端策略不仅包括客户端注册策略可以做的事情，还包括其他客户端注册和配置方式。当前的计划是用客户策略取代客户注册。

Validation of client configurations 验证客户端配置

Keycloak supports validation whether the client follows settings like Proof Key for Code Exchange, Request Object Signing Algorithm, Holder-of-Key Token, and so on some endpoints like Authorization Endpoint, Token Endpoint, and so on. These can be specified by each setting item (on Admin Console, switch, pull-down menu and so on). To make the client application secure, the administrator needs to set many settings in the appropriate way, which makes it difficult for the administrator to secure the client application. Client Policies can do these validation of client configurations mentioned just above and they can also be used to autoconfigure some client configuration switches to meet the advanced security requirements. In the future, individual client configuration settings may be replaced by Client Policies directly performing required validations.

密钥斗篷支持验证客户端是否遵循诸如代码交换的证明密钥、请求对象签名算法、密钥持有者令牌等设置，以及授权端点、令牌端点等端点。这些可以由每个设置项指定(在管理控制台，开关，下拉菜单等)。为了使客户端应用程序安全，管理员需要以适当的方式设置许多设置，这使得管理员很难保护客户端应用程序。客户端策略可以对上面提到的客户端配置进行这些验证，它们还可以用于自动配置一些客户端配置开关，以满足高级安全需求。将来，单个客户端配置设置可能会被直接执行所需验证的客户端策略所替代。

Conformance to a required security standards and profiles such as FAPI 符合所需的安全标准和配置文件(如 FAPI)

The *Global client profiles* are client profiles pre-configured in Keycloak by default. They are pre-configured to be compliant with standard security profiles like [FAPI](#) (https://www.keycloak.org/docs/latest/securing_apps/#_fapi-support), which makes it easy for the administrator to secure their client application to be compliant with the particular security profile. At this moment, Keycloak has global profiles for the support of FAPI 1 specification. The administrator will just need to configure the client policies to specify which clients should be compliant with the FAPI. The administrator can configure client profiles and client policies, so that Keycloak clients can be easily made compliant with various other security profiles like SPA, Native App, Open Banking and so on.

Global 客户端配置文件是默认情况下在 Keycloak 预先配置的客户端配置文件。它们被预先配置为与标准的安全配置文件(如 FAPI)兼容，这使得管理员很容易保护他们的客户端应用程序与特定的安全配置文件兼容。目前，为了支持 FAPI 1规范，Key枕具有全局配置文件。管理员只需要配置客户端策略，以指定哪些客户端应该与 FAPI 兼容。管理员可以配置客户端配置文件和客户端策略，这样 Keycover 客户端可以很容易地与 SPA、NativeApp、Open Banking 等各种其他安全配置文件兼容。

Protocol 规定

The client policy concept is independent of any specific protocol. However, Keycloak currently supports it only just for the [OpenID Connect \(OIDC\) protocol](https://www.keycloak.org/docs/latest/securing_apps/#_oidc) (https://www.keycloak.org/docs/latest/securing_apps/#_oidc) .

客户端策略概念独立于任何特定的协议。但是，目前只有 OpenID 连接(OIDC)协议才支持它。

Architecture 建筑学

Client Policies consists of the four building blocks: Condition, Executor, Profile and Policy.

客户端策略由四个构建块组成: 条件、执行器、配置文件和策略。

Condition 环境状况

A condition determines to which client a policy is adopted and when it is adopted. Some conditions are checked at the time of client create/update when some other conditions are checked during client requests (OIDC Authorization request, Token endpoint request and so on). The condition checks whether one specified criteria is satisfied. For example, some condition checks whether the access type of the client is confidential.

条件确定采用策略的客户端和采用策略的时间。当在客户端请求期间检查其他一些条件(OIDC 授权请求、令牌端点请求等)时，在客户端创建/更新时检查一些条件。该条件检查是否满足一个指定的条件。例如，某些条件检查客户端的访问类型是否保密。

The condition can not be used solely by itself. It can be used in a policy that is described afterwards.

该条件不能单独使用。它可以在后面描述的策略中使用。

A condition can be configurable the same as other configurable providers. What can be configured depends on each condition's nature.

条件可以像其他可配置提供程序一样进行配置。可配置的内容取决于每个条件的性质。

The following conditions are provided:

提供下列条件:

The way of creating/updating a client 创建/更新客户端的方法

- Dynamic Client Registration (Anonymous or Authenticated with Initial access token or Registration access token)

动态客户端注册(使用初始访问令牌或注册访问令牌进行匿名或身份验证)

- Admin REST API (Admin Console and so on)

管理 REST API (管理控制台等)

So for example when creating a client, a condition can be configured to evaluate to true when this client is created by OIDC Dynamic Client Registration without initial access token (Anonymous Dynamic Client Registration). So this condition can be used for example to ensure that all clients registered through OIDC Dynamic Client Registration are FAPI compliant.

因此，例如，在创建客户端时，当 OIDC 动态客户端注册创建了一个没有初始访问令牌(匿名动态客户端注册)的客户端时，可以将条件配置为计算结果为 true。因此，这个条件可以用来确保通过 OIDC 动态客户端注册的所有客户端都符合 FAPI。

Author of a client (Checked by presence to the particular role or group) 客户端的作者(通过对特定角色或组的存在进行检查)

On OpenID Connect dynamic client registration, an author of a client is the end user who was authenticated to get an access token for generating a new client, not Service Account of the existing client that actually accesses the registration endpoint with the access token. On registration by Admin REST API, an author of a client is the end user like the administrator of the Keycloak.

在 OpenID Connect 动态客户端注册中，客户端的作者是经过身份验证以获得用于生成新客户端的访问令牌的最终用户，而不是用访问令牌实际访问注册端点的现有客户端的服务帐户。在通过 Admin REST API 注册时，客户端的作者就是最终用户，就像 Key something something something 恨恨。

Client Access Type (confidential, public, bearer-only) 客户端访问类型(机密的、公共的、仅持有者)

For example when a client sends an authorization request, a policy is adopted if this client is confidential.

例如，当客户端发送授权请求时，如果该客户端是保密的，则采用策略。

Client Scope 客户端范围

Evaluates to true if the client has a particular client scope (either as default or as an optional scope used in current request). This can be used for example to ensure that OIDC authorization requests with scope `fapi-example-scope` need to be FAPI compliant.

如果客户端具有特定的客户端范围(作为默认范围或当前请求中使用的可选范围)，则计算结果为 true。例如，这可以用来确保具有范围 FAPI-example-scope 的 OIDC 授权请求需要与 FAPI 兼容。

Client Role 客户角色

Applies for clients with the client role of the specified name

应用于具有指定名称的客户端角色的客户端

Client Domain Name, Host or IP Address 客户端域名、主机或 IP 地址

Applied for specific domain names of client. Or for the cases when the administrator registers/updates client from particular Host or IP Address.

应用于客户端的特定域名。或者当管理员从特定的主机或 IP 地址注册/更新客户端时。

Any Client 任何客户

This condition always evaluates to true. It can be used for example to ensure that all clients in the particular realm are FAPI compliant.

此条件的计算结果总是为 true。例如，可以使用它来确保特定领域中的所有客户机都符合 FAPI。

Executor 执行人

An executor specifies what action is executed on a client to which a policy is adopted. The executor executes one or several specified actions. For example, some executor checks whether the value of the parameter `redirect_uri` in the authorization request matches exactly with one of the pre-registered redirect URIs on Authorization Endpoint and rejects this request if not.

执行器指定在采用策略的客户端上执行什么操作。执行者执行一个或多个指定的操作。例如，一些执行器检查授权请求中的参数 `redirect_uri` 的值是否与 Authorization Endpoint 上的一个预注册重定向 URI 完全匹配，如果不匹配，则拒绝该请求。

The executor can not be used solely by itself. It can be used in a profile that is described afterwards.

执行程序不能单独使用。它可以用在后面描述的配置文件中。

An executor can be configurable the same as other configurable providers. What can be configured depends on the nature of each executor.

执行器可以像其他可配置提供程序一样进行配置。可以配置什么取决于每个执行器的性质。

An executor acts on various events. An executor implementation can ignore certain types of events (For example, executor for checking OIDC `request` object acts just on the OIDC authorization request). Events are:

遗嘱执行人对各种事件采取行动。执行器实现可以忽略某些类型的事件(例如，检查 OIDC 请求对象的执行器仅作用于 OIDC 授权请求)。活动包括：

- Creating a client (including creation through dynamic client registration)
创建客户端(包括通过动态客户端注册创建)
- Updating a client
更新客户信息
- Sending an authorization request
发送授权请求
- Sending a token request
发送令牌请求
- Sending a token refresh request

发送令牌刷新请求

- Sending a token revocation request

发送令牌撤销请求

- Sending a token introspection request

发送令牌内省请求

- Sending a userinfo request

发送用户信息请求

- Sending a logout request with a refresh token

使用刷新令牌发送注销请求

On each event, an executor can work in multiple phases. For example, on creating/updating a client, the executor can modify the client configuration by autoconfigure specific client settings. After that, the executor validates this configuration in validation phase.

对于每个事件，执行者可以分多个阶段工作。例如，在创建/更新客户端时，执行程序可以通过自动配置特定的客户端设置来修改客户端配置。然后，执行程序在验证阶段验证此配置。

One of several purposes for this executor is to realize the security requirements of client conformance profiles like FAPI. To do so, the following executors are needed:

这个执行器的目的之一是实现客户端一致性配置文件(如 FAPI)的安全需求。为此，需要以下遗嘱执行人：

- Enforce secure Client Authentication method is used for the client

对客户端使用强制安全的客户端身份验证方法

- Enforce Holder-of-key tokens are used

强制使用“键的持有者”令牌

- Enforce Proof Key for Code Exchange (PKCE) is used

使用代码交换(PKCE)的强制证明密钥

- Enforce secure signature algorithm for Signed JWT client authentication (private-key-jwt) is used

使用签名 JWT 客户端身份验证(private-key-JWT)的安全签名算法

- Enforce HTTPS redirect URI and make sure that configured redirect URI does not contain wildcards

强制 HTTPS 重定向 URI，并确保配置的重定向 URI 不包含通配符

- Enforce OIDC `request` object satisfying high security level

执行满足高安全级别的 OIDC 请求对象

- Enforce Response Type of OIDC Hybrid Flow including ID Token used as *detached signature* as described in the FAPI 1 specification, which means that ID Token returned from Authorization response won't contain user profile data

强制执行 OIDC 混合流的响应类型，包括在 FAPI 1 规范中描述的作为分离签名使用的 ID 令牌，这意味着从 Authorization 响应返回的 ID 令牌不包含用户配置文件数据

- Enforce more secure `state` and `nonce` parameters treatment for preventing CSRF

实施更安全的状态和临时参数处理以防止 CSRF

- Enforce more secure signature algorithm when client registration

客户端注册时实施更安全的签名算法

- Enforce `binding_message` parameter is used for CIBA requests

强制 `bind_message` 参数用于 CIBA 请求

- Enforce Client Secret Rotation

执行客户保密轮换

- Enforce checking if a client is the one to which an intent was issued in a use case where an intent is issued before starting an authorization code flow to get an access token like UK OpenBanking

在启动授权代码流以获取类似 UK OpenBanking 的访问令牌之前，在发出意图的用例中，强制检查客户端是否是发出意图的客户端

Profile 侧写

A profile consists of several executors, which can realize a security profile like FAPI. Profile can be configured by the Admin REST API (Admin Console) together with its executors. Three *global profiles* exist and they are configured in Keycloak by default with pre-configured executors compliant with the FAPI Baseline, FAPI Advanced and FAPI CIBA specifications. More details exist in the FAPI section of the [Securing Applications and Services Guide](#) (https://www.keycloak.org/docs/latest/securing_apps/#_fapi-support).

一个配置文件由几个执行器组成，它们可以实现类似于 FAPI 的安全配置文件。配置文件可以由管理 REST API (管理控制台)及其执行程序配置。有三个全局配置文件，它们默认在 Keycloak 配置，预先配置的执行器符合 FAPI 基线、FAPI 高级和 FAPI CIBA 规范。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。

Policy 政策

A policy consists of several conditions and profiles. The policy can be adopted to clients satisfying all conditions of this policy. The policy refers several profiles and all executors of these profiles execute their task against the client that this policy is adopted to.

策略由几个条件和配置文件组成。该策略可以被满足该策略所有条件的客户所采用。策略引用几个概要文件，这些概要文件的所有执行者针对采用此策略的客户机执行任务。

Configuration 配置

Policies, profiles, conditions, executors can be configured by Admin REST API, which means also the Admin Console. To do so, there is a tab *Realm* → *Realm Settings* → *Client Policies*, which means the administrator can have client policies per realm.

策略、配置文件、条件、执行器可以通过管理 REST API 进行配置，也就是管理控制台。要做到这一点，有一个标签领域→领域设置→客户端策略，这意味着管理员可以有每个领域的客户端策略。

The *Global Client Profiles* are automatically available in each realm. However there are no client policies configured by default. This means that the administrator is always required to create any client policy if they want for example the clients of his realm to be FAPI compliant. Global profiles cannot be updated, but the administrator can easily use them as a template and create their own profile if they want to do some slight changes in the global profile configurations. There is JSON Editor available in the Admin Console, which simplifies the creation of new profile based on some global profile.

全局客户端概要文件在每个领域中都是自动可用的。但是缺省情况下没有配置客户端策略。这意味着，如果管理员希望其领域的客户端遵从 FAPI，那么他们总是需要创建任何客户端策略。全局配置文件不能更新，但是管理员可以很容易地将它们用作模板，并且如果他们想对全局配置文件配置进行一些细微的更改，则可以创建自己的配置文件。管理控制台中有可用的 JSON 编辑器，它简化了基于某些全局概要文件的新概要文件的创建。

Backward Compatibility 向下兼容

Client Policies can replace Client Registration Policies described in the [Securing Applications and Services Guide](#) (https://www.keycloak.org/docs/latest/securing_apps/#_client_registration_policies). However, Client Registration Policies also still co-exist. This means that for example during a Dynamic Client Registration request to create/update a client, both client policies and client registration policies are applied.

客户端策略可以替换安全应用程序和服务指南中描述的客户端注册策略。但是，客户注册策略仍然共存。这意味着，例如，在创建/更新客户端的 Dynamic Client Registry 请求期间，将应用客户端策略和客户端注册策略。

The current plans are for the Client Registration Policies feature to be removed and the existing client registration policies will be migrated into new client policies automatically.

当前的计划是删除客户注册策略功能，现有的客户注册策略将自动迁移到新的客户策略中。

Client Secret Rotation Example 客户端秘密轮换示例

See an example configuration for client secret rotation.

请参见客户机秘密轮换的示例配置。

[Edit this section 编辑这部分](#) (https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/vault.adoc)

[Report an issue 报告一个问题](#) (https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/vault.adoc)

To obtain a secret from a vault rather than entering it directly, enter the following specially crafted string into the appropriate field:

要从保险库中获得秘密而不是直接进入，请在相应的字段中输入以下特制字符串：

```
**${vault._key_*}**
```

where the `key` is the name of the secret recognized by the vault.

其中的钥匙是保险库认可的秘密的名称。

To prevent secrets from leaking across realms, Keycloak combines the realm name with the `key` obtained from the vault expression. This method means that the `key` does not directly map to an entry in the vault but creates the final entry name according to the algorithm used to combine the `key` with the realm name.

为了防止跨领域泄露机密，Key斗篷将领域名称与从保险库表达式获得的密钥结合起来。这种方法意味着密钥不直接映射到保险库中的条目，而是根据用于将密钥与领域名称组合在一起的算法创建最终的条目名称。

You can obtain the secret from the vault in the following fields:

您可以在以下字段中从保险库中获得这个秘密：

SMTP password SMTP 密码

In the realm SMTP settings

在领域 SMTP 设置中

LDAP bind credential LDAP 绑定凭据

In the LDAP settings of LDAP-based user federation.

在基于 LDAP 的用户联合的 LDAP 设置中。

OIDC identity provider secret OIDC 身份提供商机密

In the *Client Secret* inside identity provider OpenID Connect Config

在身份提供程序 OpenID 连接配置中的 Client Secret 中

Key resolvers 钥匙分解器

All built-in providers support the configuration of key resolvers. A key resolver implements the algorithm or strategy for combining the realm name with the key, obtained from the `${vault.key}` expression, into the final entry name used to retrieve the secret from the vault. Keycloak uses the

`keyResolvers` property to configure the resolvers that the provider uses. The value is a comma-separated list of resolver names. An example of the configuration for the `files-plaintext` provider follows:

所有内置提供程序都支持密钥解析器的配置。密钥解析器实现将领域名称与从 \${ vault.key } 表达式获得的密钥组合到用于从保险库检索秘密的最终条目名称中的算法或策略。Keycon 使用 `keyResolvers` 属性配置提供程序使用的冲突解决程序。该值是以逗号分隔的解析器名称列表。下面是文件明文提供程序的配置示例：

```
kc.[sh.bat] start --spi-vault-file-key-resolvers=REALM_UNDERSCORE_KEY,KEY_ONLY
```

BASH

The resolvers run in the same order you declare them in the configuration. For each resolver, Keycloak uses the last entry name the resolver produces, which combines the realm with the vault key to search for the vault's secret. If Keycloak finds a secret, it returns the secret. If not, Keycloak uses the next resolver. This search continues until Keycloak finds a non-empty secret or runs out of resolvers. If Keycloak finds no secret, Keycloak returns an empty secret.

解析器按照您在配置中声明它们的顺序运行。对于每个冲突解决程序，Key枕使用冲突解决程序生成的最后一个条目名称，该名称将领域与保险库密钥结合起来，以搜索保险库的秘密。如果钥匙斗篷发现了一个秘密，它将返回该秘密。如果没有，则钥匙斗篷使用下一个解析器。这种搜索一直持续到钥匙斗篷找到一个非空的秘密或用完解析器。如果钥匙斗篷没有发现任何秘密，钥匙斗篷返回一个空的秘密。

In the previous example, Keycloak uses the `REALM_UNDERSCORE_KEY` resolver first. If Keycloak finds an entry in the vault that using that resolver, Keycloak returns that entry. If not, Keycloak searches again using the `KEY_ONLY` resolver. If Keycloak finds an entry by using the `KEY_ONLY` resolver, Keycloak returns that entry. If Keycloak uses all resolvers, Keycloak returns an empty secret.

在上一个示例中，Keycon 首先使用 `REALM _ UNDERSCORE _ KEY` 解析器。如果钥匙斗篷在使用该解析器的保险库中找到一个条目，钥匙斗篷将返回该条目。如果没有，则使用 `KEY _ ONLY` 解析程序再次搜索。如果钥匙斗篷通过使用 `KEY _ ONLY` 解析器找到一个条目，则钥匙斗篷返回该条目。如果钥匙斗篷使用所有的解析器，钥匙斗篷返回一个空的秘密。

A list of the currently available resolvers follows:

现有解决方案如下：

Name 姓名	Description 描述
KEY_ONLY	Keycloak ignores the realm name and uses the key from the vault expression.
KEY _ ONLY	Key斗篷忽略领域名称并使用来自保险库表达式的密钥。

Name 姓名	Description 描述
REALM_UNDERSCORE_KEY	Keycloak combines the realm and key by using an underscore character. Keycloak escapes occurrences of underscores in the realm or key with another underscore character. For example, if the realm is called <code>master_realm</code> and the key is <code>smtp_key</code> , the combined key is <code>master__realm_smtp__key</code> .
REALM_UNDERSCORE_KEY	Key斗篷通过使用下划线字符组合领域和键。在域或键中使用另一个下划线字符可以避免出现下划线。例如，如果领域名为 <code>master_domain</code> ，而键为 <code>smtp_key</code> ，则组合键为 <code>master__domain_smtp__key</code> 。
REALM_FILESEPARATOR_KEY 文件分离器	Keycloak combines the realm and key by using the platform file separator character. Keycon 通过使用平台文件分隔符字符组合领域和键。
FACTORY_PROVIDED 工厂_提供	Keycloak combines the realm and key by using the vault provider factory's <code>VaultKeyResolver</code> , allowing the creation of a custom key resolver by extending an existing factory and implementing the <code>getFactoryResolver</code> method. Keycon 通过使用保险库提供商工厂的 <code>VaultKeyResolver</code> 组合领域和密钥，允许通过扩展现有工厂和实现 <code>getFactoryResolver</code> 方法创建自定义密钥解析器。

If you have not configured a resolver for the built-in providers, Keycloak selects the `REALM_UNDERSCORE_KEY`.

如果尚未为内置提供程序配置冲突解决程序，则 Keycloak 将选择 `REALM_UNDERSCORE_KEY`。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/events.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/events.adoc)

Keycloak includes a suite of auditing capabilities. You can record every login and administrator action and review those actions in the Admin Console. Keycloak also includes a Listener SPI that listens for events and can trigger actions. Examples of built-in listeners include log files and sending emails if an event occurs.

Key斗篷包括一套审计功能。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。Key斗篷还包括一个侦听器 SPI，它侦听事件并可以触发操作。内置侦听器的示例包括日志文件和在发生事件时发送电子邮件。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/events/login.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/events/login.adoc)

You can record and view every event that affects users. Keycloak triggers login events for actions such as successful user login, a user entering an incorrect password, or a user account updating. By default, Keycloak does not store or display events in the Admin Console. Only the error events are logged to the Admin Console and the server's log file.

您可以记录和查看影响用户的每个事件。键斗篷触发用户登录成功、用户输入错误密码或用户帐户更新等操作的登录事件。默认情况下，“钥匙斗篷”不在管理控制台中存储或显示事件。只有错误事件被记录到管理控制台和服务器的日志文件中。

Procedure 程序

Use this procedure to start auditing user events.

使用此过程开始审核用户事件。

1. Click **Realm settings** in the menu.

单击菜单中的“域”设置。

2. Click the **Events** tab.

单击 Events 选项卡。

3. Click the **User events settings** tab.

单击“用户事件设置”选项卡。

4. Toggle **Save events** to ON.

将保存事件切换为 ON。

User events settings 用户事件设置

[Event listeners](#)[User events settings](#)[Admin events settings](#)

User events configuration

Save events [?](#)

On

Expiration [?](#)

90

Minutes [▼](#)[Save](#)[Revert](#)[Clear user events](#) [?](#)[Clear user events](#)

Search saved event type

[Add saved types](#)**Event saved type****Description**

Login

Login

5. Specify the length of time to store events in the **Expiration** field.

指定在 Exiration 字段中存储事件的时间长度。

6. Click **Add saved types** to see other events you can save.

单击“添加保存的类型”以查看可以保存的其他事件。

Add types 添加类型

Add types

refresh

	Description
<input checked="" type="checkbox"/>	Event saved type
<input checked="" type="checkbox"/>	Refresh token
<input checked="" type="checkbox"/>	Refresh token error

1 - 2

7. Click Add.

单击“添加”。

Click **Clear user events** when you want to delete all saved events.

当要删除所有保存的事件时，单击“清除用户事件”。

Procedure 程序

You can now view events.

现在可以查看事件。

1. Click the **Events** tab in the menu.

单击菜单中的 Events 选项卡。

User events 用户事件

User events	Admin events				
Search user event		Refresh	1 - 4	<	>
Time	User	Event type	IP address	Client	
May 2, 2022 4:00 PM	a4d4e4a8-3440-46f0-b29f-4bc12ec45e44	CODE_TO_TOKEN	127.0.0.1	security-admin-console	
May 2, 2022 4:00 PM	a4d4e4a8-3440-46f0-b29f-4bc12ec45e44	LOGIN	127.0.0.1	security-admin-console	
May 2, 2022 2:55 PM	a4d4e4a8-3440-46f0-b29f-4bc12ec45e44	CODE_TO_TOKEN	127.0.0.1	security-admin-console	
May 2, 2022 2:55 PM	a4d4e4a8-3440-46f0-b29f-4bc12ec45e44	LOGIN	127.0.0.1	security-admin-console	

2. To filter events, click **Search user event**.

要筛选事件，请单击“搜索用户事件”。

Search user event 搜索用户事件

The screenshot shows a search interface for user events. At the top, there are tabs for "User events" and "Admin events". Below them is a search bar labeled "Search user event" with a "Refresh" button. On the left, there are several filter fields: "User ID" (empty), "Event type" (set to "LOGIN"), "Client" (empty), "Date(from)" (empty), and "Date(to)" (empty). A dropdown menu for "Event type" is open, showing options: LOGIN (selected), LOGIN_ERROR, REGISTER, REGISTER_ERROR, and LOGOUT. At the bottom of this menu is a blue "Search events" button.

Event types 事件类型

Login events:

登入事件:

Event 事件	Description 描述
Login	A user logs in.
登陆	用户登录。
Register	A user registers.
登记册	用户注册。
Logout	A user logs out.
注销	用户注销。

Event 事件	Description 描述
Code to Token Token 的密码	An application, or client, exchanges a code for a token. 应用程序或客户端用代码交换令牌。
Refresh Token 刷新令牌	An application, or client, refreshes a token. 应用程序或客户端刷新令牌。

Account events:

帐户事项:

Event 事件	Description 描述
Social Link 社会联系	A user account links to a social media provider. 用户帐户链接到社交媒体提供商。
Remove Social Link 删除社交链接	The link from a social media account to a user account severs. 从社交媒体帐户到用户帐户的链接断开。
Update Email 更新邮件	An email address for an account changes. 帐户更改的电子邮件地址。
Update Profile 更新配置文件	A profile for an account changes. 帐户的配置文件更改。
Send Password Reset 发送密码重置	Keycloak sends a password reset email. 钥匙斗篷发送密码重置电子邮件。
Update Password 更新密码	The password for an account changes. 帐户的密码更改。

Event 事件	Description 描述
Update TOTP 更新 TOTP	The Time-based One-time Password (TOTP) settings for an account changes. 帐户的基于时间的一次性密码(TOTP)设置发生了变化。
Remove TOTP 移除 TOTP	Keycloak removes TOTP from an account. 密钥斗篷从帐户中移除 TOTP。
Send Verify Email 发送验证邮件	Keycloak sends an email verification email. 钥匙斗篷发送一封电子邮件验证电子邮件。
Verify Email 确认电子邮件	Keycloak verifies the email address for an account. 密钥斗篷验证帐户的电子邮件地址。

Each event has a corresponding error event.

每个事件都有相应的错误事件。

Event listener 事件监听器

Event listeners listen for events and perform actions based on that event. Keycloak includes two built-in listeners, the Logging Event Listener and Email Event Listener.

事件监听器侦听事件并根据该事件执行操作。Key斗篷包括两个内置监听器，即日志事件监听器和电子邮件事件监听器。

The logging event listener 日志事件监听器

When the Logging Event Listener is enabled, this listener writes to a log file when an error event occurs.

启用日志事件监听器后，当发生错误事件时，此监听器将写入日志文件。

An example log message from a Logging Event Listener:

来自日志事件监听器的日志消息示例:

```
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
```

You can use the Logging Event Listener to protect against hacker bot attacks:

您可以使用日志事件监听器来防止黑客机器人的攻击:

1. Parse the log file for the `LOGIN_ERROR` event.

解析 `LOGIN_ERROR` 事件的日志文件。

2. Extract the IP Address of the failed login event.

提取失败的登录事件的 IP 地址。

3. Send the IP address to an intrusion prevention software framework tool.

将 IP 地址发送到一个入侵防御软件框架工具。

The Logging Event Listener logs events to the `org.keycloak.events` log category. Keycloak does not include debug log events in server logs, by default.

Logging Event Listener 将事件记录到 `org.keycloak.events` 日志类别。默认情况下，“钥匙斗篷”不在服务器日志中包含调试日志事件。

To include debug log events in server logs:

在服务器日志中包含调试日志事件:

1. Change the log level for the `org.keycloak.events` category

更改 `org.keycloak.events` 类别的日志级别

2. Change the log level used by the Logging Event listener.

更改日志记录事件侦听器使用的日志级别。

To change the log level used by the Logging Event listener, add the following:

若要更改日志记录事件侦听器使用的日志级别，请添加以下内容:

```
bin/kc.[sh|bat] start --spi-events-listener-jboss-logging-success-level=info --spi-events-listener-jboss-logging-error-level=error
```

The valid values for log levels are `debug`, `info`, `warn`, `error`, and `fatal`.

日志级别的有效值是调试、信息、警告、错误和致命。

The Email Event Listener 电子邮件事件监听器

The Email Event Listener sends an email to the user's account when an event occurs and supports the following events:

电子邮件事件监听器在事件发生时向用户帐户发送电子邮件，并支持以下事件：

- Login Error.
登入错误。
- Update Password.
更新密码。
- Update Time-based One-time Password (TOTP).
更新时间一次性密码(TOTP)。
- Remove Time-based One-time Password (TOTP).
删除基于时间的一次性密码(TOTP)。

Procedure 程序

To enable the Email Listener:

启用电子邮件监听器：

1. Click **Realm settings** in the menu.

单击菜单中的“域”设置。

2. Click the **Events** tab.

单击 Events 选项卡。

3. Click the **Event listeners** field.

单击“事件监听器”字段。

4. Select **email**.

选择电子邮件。

Event listeners 事件侦听器

The screenshot shows the 'Event listeners' section of the Keycloak Admin UI. It has three tabs at the top: 'Event listeners' (which is selected), 'User events settings', and 'Admin events settings'. Below the tabs, there is a list of event listeners. One listener is currently selected, shown in a box with the name 'email'. There is also an 'x' icon to remove the selection. At the bottom of the list are two buttons: 'Save' (in blue) and 'Revert'.

You can exclude events by using the `--spi-events-listener-email-exclude-events` argument. For example:

您可以通过使用—— spi-events-listen-email- 排除事件。例如:

```
kc.[sh|bat] --spi-events-listener-email-exclude-events=UPDATE_TOTP,REMOVE_TOTP
```

BASH

You can set a maximum length of each Event detail in the database by using the `--spi-events-store-jpa-max-detail-length` argument. This setting is useful if a detail (for example, redirect_uri) is long. For example:

可以使用—— spi-events-store-jpa-max-Details-length 参数设置数据库中每个 Event 详细信息的最大长度。如果详细信息(例如 redirect _ uri)很长，那么这个设置非常有用。例如:

```
kc.[sh|bat] --spi-events-store-jpa-max-detail-length=1000
```

BASH

Also you can set a maximum length of all Event's details by using the `--spi-events-store-jpa-max-field-length` argument. This setting is useful if you want to adhere to the underlying storage limitation. For example:

还可以使用—— spi-events-store-jpa-max-field-length 参数设置所有 Event 详细信息的最大长度。如果希望遵守基础存储限制，则此设置非常有用。例如:

```
kc.[sh|bat] --spi-events-store-jpa-max-field-length=2500
```

BASH

[Edit this section](#) 编辑这部分

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/events/admin.adoc)

[Report an issue](#) 报告一个问题

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/events/admin.adoc)

You can record all actions that are performed by an administrator in the Admin Console. The Admin Console performs administrative actions by invoking the Keycloak REST interface and Keycloak audits these REST invocations. You can view the resulting events in the Admin Console.

您可以记录管理员在管理控制台中执行的所有操作。管理控制台执行管理操作，调用 Key斗篷 REST 接口，Key斗篷审计这些 REST 调用。您可以在管理控制台中查看结果事件。

Procedure 程序

Use this procedure to start auditing admin actions.

使用此过程开始审核管理操作。

1. Click **Realm settings** in the menu.

单击菜单中的“域”设置。

2. Click the **Events** tab.

单击 Events 选项卡。

3. Click the **Admin events settings** tab.

单击“管理事件设置”选项卡。

4. Toggle **Save events** to ON.

将保存事件切换为 ON。

Keycloak displays the **Include representation** switch.

键斗篷显示“包含”表示开关。

5. Toggle **Include representation** to ON.

切换包含表示为 ON。

The **Include Representation** switch includes JSON documents sent through the admin REST API so you can view the administrators actions.

包含表示开关包括通过管理员 REST API 发送的 JSON 文档，因此您可以查看管理员操作。

Admin events settings 管理事件设置

The screenshot shows the 'Events' tab selected in the top navigation bar. Below it, the 'Admin events settings' tab is active. On the left, there are two configuration sections: 'Save events' (On) and 'Include representation' (Off). At the bottom, there are 'Save' and 'Revert' buttons, and a 'Clear admin events' button.

Master

Realm settings are settings that control the options for users, applications, roles, and groups in the current realm. [Learn more](#)

Enabled Action ▾

General Login Email Themes Keys Events Localization Security defens >

Event listeners User events settings Admin events settings

Admin events configuration

Save events ⓘ On

Include representation Off ⓘ

Save **Revert**

Clear admin events ⓘ **Clear admin events**

6. Click Save.

单击“保存”。

7. To clear the database of stored actions, click Clear admin events.

若要清除存储操作的数据库，请单击“清除管理事件”。

Procedure 程序

You can now view admin events.

现在可以查看管理事件。

1. Click Events in the menu.

单击菜单中的 Events。

2. Click the Admin events tab.

单击 Admin events 选项卡。

Admin events 管理事件

User events	Admin events			
Search admin event				
Time	Resource path	Resource type	Operation type	User
April 29, 2022 9:06 PM	events/config	REALM	UPDATE	a4d4e4a8-3440-46f0-b29f-4bc12ec45e44
April 29, 2022 8:48 PM	events/config	REALM	UPDATE	a4d4e4a8-3440-46f0-b29f-4bc12ec45e44
April 29, 2022 7:57 PM	events/config	REALM	UPDATE	a4d4e4a8-3440-46f0-b29f-4bc12ec45e44

When the `Include Representation` switch is ON, it can lead to storing a lot of information in the database. You can set a maximum length of the representation by using the `--spi-events-store-jpa-max-field-length` argument. This setting is useful if you want to adhere to the underlying storage limitation. For example:

当包含表示形式开关为 ON 时，可能导致在数据库中存储大量信息。可以使用—— `spi-events-store-jpa-max-field-length` 参数设置表示形式的最大长度。如果希望遵守基础存储限制，则此设置非常有用。例如：

```
kc .[sh|bat] --spi-events-store-jpa-max-field-length=2500
```

BASH

[Edit this section](#) 编辑这部分

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/threat.adoc)

[Report an issue](#) 报告一个问题

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/threat.adoc)

Security vulnerabilities exist in any authentication server. See the Internet Engineering Task Force's (IETF) [OAuth 2.0 Threat Model](#) (<https://datatracker.ietf.org/doc/html/rfc6819>) and the [OAuth 2.0 Security Best Current Practice](#) (<https://datatracker.ietf.org/doc/html/draft-ietf-oauth-security-topics-15>) for more information.

任何身份验证服务器都存在安全漏洞。有关更多信息，请参阅 Internet Engineering Task Force (IETF) OAuth 2.0 威胁模型和 OAuth 2.0 安全性最佳当前实践。

[Edit this section](#) 编辑这部分

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/threat/host.adoc)

[Report an issue](#) 报告一个问题

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/threat/host.adoc)

Keycloak uses the public hostname in several ways, such as within token issuer fields and URLs in password reset emails.

钥匙斗篷以多种方式使用公共主机名，例如在令牌发行者字段中以及在密码重置电子邮件中使用 URL。

By default, the hostname derives from request headers. No validation exists to ensure a hostname is valid. If you are not using a load balancer, or proxy, with Keycloak to prevent invalid host headers, configure the acceptable hostnames.

默认情况下，主机名从请求头派生。不存在确保主机名有效的验证。如果您没有使用带有钥匙斗篷的负载平衡器或代理来防止无效的主机头，请配置可接受的主机名。

The hostname's Service Provider Interface (SPI) provides a way to configure the hostname for requests. You can use this built-in provider to set a fixed URL for frontend requests while allowing backend requests based on the request URI. If the built-in provider does not have the required capability, you can develop a customized provider.

主机名的服务提供者接口(SPI)提供了一种为请求配置主机名的方法。您可以使用此内置提供程序为前端请求设置固定的 URL，同时允许基于请求 URI 的后端请求。如果内置提供程序没有所需的功能，则可以开发定制的提供程序。

[Edit this section](#) 编辑这部分

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/threat/admin.adoc)

[Report an issue](#) 报告一个问题

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/threat/admin.adoc)

Keycloak exposes the administrative REST API and the web console on the same port as non-administrative usage. Do not expose administrative endpoints externally if external access is not necessary.

作为非管理用法，Key斗篷在同一端口上公开管理 REST API 和 Web 控制台。如果不需要外部访问，不要在外部公开管理端点。

[Edit this section](#) 编辑这部分

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/threat/brute-force.adoc)

[Report an issue](#) 报告一个问题

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/threat/brute-force.adoc)

A brute force attack attempts to guess a user's password by trying to log in multiple times. Keycloak has brute force detection capabilities and can temporarily disable a user account if the number of login failures exceeds a specified threshold.

一个穷举法试图通过多次登录来猜测用户的密码。Keycon 具有强力检测功能，如果登录失败的次数超过指定的阈值，它可以暂时禁用用户帐户。



Keycloak disables brute force detection by default. Enable this feature to protect against brute force attacks.

钥匙斗篷在默认情况下禁用暴力检测。启用此功能可防止暴力攻击。

Procedure 程序

To enable this protection:

为了实现这种保护:

1. Click **Realm Settings** in the menu

单击菜单中的“域设置”

2. Click the **Security Defenses** tab.

单击“安全防御”选项卡。

3. Click the **Brute Force Detection** tab.

单击“暴力检测”选项卡。

Brute force detection 蛮力侦测

The screenshot shows the Keycloak 'Master' realm settings under the 'Security defenses' tab. The 'Brute force detection' sub-tab is selected. The configuration includes:

- Enabled:** On (blue toggle switch)
- Max login failures:** 30 (adjustable slider from 1 to 30)
- Permanent lockout:** Off (grey toggle switch)
- Wait increment:** 1 minute (input field: 1, dropdown: Minutes)
- Max wait:** 15 minutes (input field: 15, dropdown: Minutes)
- Failure reset time:** 12 hours (input field: 12, dropdown: Hours)
- Quick login check milliseconds:** 1000 (adjustable slider from 1 to 1000)
- Minimum quick login wait:** 1 minute (input field: 1, dropdown: Minutes)

At the bottom are 'Save' and 'Revert' buttons.

Keycloak can deploy permanent lockout and temporary lockout actions when it detects an attack. Permanent lockout disables a user account until an administrator re-enables it. Temporary lockout disables a user account for a specific period of time. The time period that the account is disabled increases as the attack continues.

当钥匙斗篷检测到攻击时，它可以部署永久锁定和临时锁定操作。永久锁定将禁用用户帐户，直到管理员重新启用该帐户。暂时锁定在特定时间内禁用用户帐户。随着攻击的继续，帐户被禁用的时间段会增加。

 When a user is temporarily locked and attempts to log in, Keycloak displays the default `Invalid username or password` error message. This message is the same error message as the message displayed for an invalid username or invalid password to ensure the attacker is unaware the account is disabled.

当用户被临时锁定并试图登录时，Key枕显示默认的无效用户名或密码错误消息。此消息与显示无效用户名或无效密码的消息相同，以确保攻击者不知道帐户已禁用。

Common Parameters

公共参数

Name 姓名	Description 描述	Default 默认
Max Login Failures 最大登录失败	The maximum number of login failures. 登录失败的最大次数。	30 failures. 30次失败。
Quick Login Check Milliseconds 快速登录检查毫秒	The minimum time between login attempts. 登录尝试之间的最短时间。	1000 milliseconds. 1000毫秒。
Minimum Quick Login Wait 最小快速登录等待	The minimum time the user is disabled when login attempts are quicker than <i>Quick Login Check Milliseconds</i> . 当登录尝试比快速登录检查毫秒更快时，禁用用户的最短时间。	1 minute. 一分钟。

Permanent Lockout Flow

永久锁定流

1. On successful login

成功登入

a. Reset count

重置计数

2. On failed login

登录失败

a. Increment count

增量计数

b. If count greater than Max Login Failures

如果计数大于最大登录失败

i. Permanently disable user

永久禁用用户

c. Else if the time between this failure and the last failure is less than Quick Login Check Milliseconds

否则，如果此故障与最后一次故障之间的时间小于“快速登录检查毫秒”

i. Temporarily disable user for Minimum Quick Login Wait

暂时禁用最小快速登录等待用户

When Keycloak disables a user, the user cannot log in until an administrator enables the user. Enabling an account resets the count.

当“钥匙斗篷”禁用用户时，在管理员启用该用户之前，该用户无法登录。启用帐户将重置计数。

Temporary Lockout Parameters

临时锁定参数

Name 姓名	Description 描述	Default 默认
Wait Increment 等等，增量	The time added to the time a user is temporarily disabled when the user's login attempts exceed Max Login Failures. 当用户的登录尝试超过最大登录失败时，用户被临时禁用的时间所增加的时间。	1 minute. 一分钟。

Name 姓名	Description 描述	Default 默认
Max Wait Max 等等	The maximum time a user is temporarily disabled. 暂时禁用用户的最大时间。	15 minutes. 15分钟。
Failure Reset Time 故障重置时间	The time when the failure count resets. The timer runs from the last failed login. 失败计数重置的时间。计时器从上次失败的登录开始运行。	12 hours. 12小时。

Temporary Lockout Algorithm

临时锁定算法

1. On successful login

成功登入

a. Reset count

重置计数

2. On failed login

登录失败

a. If the time between this failure and the last failure is greater than Failure Reset Time

如果此故障与最后一次故障之间的时间大于故障重置时间

i. Reset count

重置计数

b. Increment count

增量计数

c. Calculate wait using Wait Increment * (count / Max Login Failures). The division is an integer division rounded down to a whole number

使用等待增量 * (计数/最大登录失败)计算等待。除法是一个四舍五入到整数的整数除法

d. If wait equals 0 and the time between this failure and the last failure is less than Quick Login Check Milliseconds, set wait to Minimum Quick Login Wait.

如果 Wait 等于0，且此故障与最后一次故障之间的时间小于 Quick Login Check Milliseconds，则将 Wait 设置为最小 Quick Login Wait。

i. Temporarily disable the user for the smallest of `wait` and `Max Wait` seconds

暂时禁用最小等待时间和最大等待秒数的用户

`count` does not increment when a temporarily disabled account commits a login failure.

当暂时禁用的帐户提交登录失败时，`count` 不会递增。

The downside of Keycloak brute force detection is that the server becomes vulnerable to denial of service attacks. When implementing a denial of service attack, an attacker can attempt to log in by guessing passwords for any accounts it knows and eventually causing Keycloak to disable the accounts.

Keycloak 暴力检测的缺点是服务器容易受到分布式拒绝服务攻击攻击。在实施分布式拒绝服务攻击攻击时，攻击者可以尝试通过猜测已知帐户的密码来登录，最终导致“钥匙斗篷”禁用这些帐户。

Consider using intrusion prevention software (IPS). Keycloak logs every login failure and client IP address failure. You can point the IPS to the Keycloak server's log file, and the IPS can modify firewalls to block connections from these IP addresses.

考虑使用入侵防御软件(IPS)。密钥隐形记录每次登录失败和客户端 IP 地址失败。您可以将 IPS 指向 Keycloak 服务器的日志文件，并且 IPS 可以修改防火墙以阻止来自这些 IP 地址的连接。

Password policies 密码策略

Ensure you have a complex password policy to force users to choose complex passwords. See the Password Policies chapter for more information. Prevent password guessing by setting up the Keycloak server to use one-time-passwords.

确保您有一个复杂的密码策略来强制用户选择复杂的密码。有关更多信息，请参见密码策略一章。通过设置 Keycloak 服务器使用一次性密码来防止猜密码。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/threat/read-only-attributes.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/threat/read-only-attributes.adoc)

Typical users who are stored in Keycloak have various attributes related to their user profiles. Such attributes include email, firstName or lastName. However users may also have attributes, which are not typical profile data, but rather metadata. The metadata attributes usually should be read-only for the users and the typical users never should have a way to update those attributes from the Keycloak user interface or Account REST API. Some of the attributes should be even read-only for the administrators when creating or updating user with the Admin REST API.

存储在 Keycloak 的典型用户具有与其用户配置文件相关的各种属性。这些属性包括 email、firstName 或 lastName。但是，用户也可能拥有属性，这些属性不是典型的配置文件数据，而是元数据。元数据属性对于用户来说通常应该是只读的，典型的用户永远都不应该有办法来更新来自 Keycloak 用户界面或 Account REST API 的那些属性。当管理员使用 Admin REST API 创建或更新用户时，其中一些属性甚至应该是只读的。

The metadata attributes are usually attributes from those groups:

元数据属性通常是来自这些组的属性：

- Various links or metadata related to the user storage providers. For example in case of the LDAP integration, the `LDAP_ID` attribute contains the ID of the user in the LDAP server.

与用户存储提供程序相关的各种链接或元数据。例如，在 LDAP 集成的情况下，`LDAP_ID` 属性包含 LDAP 服务器中用户的 ID。

- Metadata provisioned by User Storage. For example `createdTimestamp` provisioned from the LDAP should be always read-only by user or administrator.

用户存储提供的元数据。例如，LDAP 提供的 `createdTimestamp` 应该始终由用户或管理员只读。

- Metadata related to various authenticators. For example `KERBEROS_PRINCIPAL` attribute can contain the kerberos principal name of the particular user. Similarly attribute `usercertificate` can contain metadata related to binding the user with the data from the X.509 certificate, which is used typically when X.509 certificate authentication is enabled.

与各种身份验证程序相关的元数据。例如，`KERBEROS_PRINCIPAL` 属性可以包含特定用户的 KERBEROS 主体名称。类似的属性 `usercertificate` 可以包含与将用户与来自 X.509 证书的数据绑定相关的元数据，这通常在启用 X.509 证书身份验证时使用。

- Metadata related to the identifier of users by the applications/clients. For example `saml.persistent.name.id.for.my_app` can contain SAML NameID, which will be used by the client application `my_app` as the identifier of the user.

与应用程序/客户端的用户标识符相关的元数据。例如，`SAML.persistent.name.id.For.my_app` 可以包含 SAML NameID，客户机应用程序 `my_app` 将使用 SAML NameID 作为用户的标识符。

- Metadata related to the authorization policies, which are used for the attribute based access control (ABAC). Values of those attributes may be used for the authorization decisions. Hence it is important that those attributes cannot be updated by the users.

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，用户不能更新这些属性非常重要。

From the long term perspective, Keycloak will have a proper User Profile SPI, which will allow fine-grained configuration of every user attribute. Currently this capability is not fully available yet. So Keycloak has the internal list of user attributes, which are read-only for the users and read-only for the administrators configured at the server level.

从长远的角度来看，Keycloak将有一个适当的用户配置文件 SPI，它将允许对每个用户属性进行细粒度配置。目前这种能力还不完全可用。因此 Keycloak 具有用户属性的内部列表，这些属性对于用户是只读的，对于在服务器级别配置的管理员是只读的。

This is the list of the read-only attributes, which are used internally by the Keycloak default providers and functionalities and hence are always read-only:

这是只读属性的列表，它们在内部由 Keycloak 默认提供程序和功能使用，因此总是只读的：

- For users: KERBEROS_PRINCIPAL, LDAP_ID, LDAP_ENTRY_DN, CREATED_TIMESTAMP, createTimestamp, modifyTimestamp, userCertificate, saml.persistent.name.id.for.*, ENABLED, EMAIL_VERIFIED

对于用户: KERBEROS_PRINCIPAL, LDAP_ID, LDAP_ENTRY_DN, CREATED_TIMESTAMP, createTimestamp, modifyTimestamp, userSecurities, saml.persistent.name.id.For.*, ENABLED, EMAIL_VERIFIED

- For administrators: KERBEROS_PRINCIPAL, LDAP_ID, LDAP_ENTRY_DN, CREATED_TIMESTAMP, createTimestamp, modifyTimestamp

对于管理员: KERBEROS_PRINCIPAL, LDAP_ID, LDAP_ENTRY_DN, CREATED_TIMESTAMP, createTimestamp, AmendmentTimestamp

System administrators have a way to add additional attributes to this list. The configuration is currently available at the server level.

系统管理员有办法向此列表添加其他属性。该配置目前在服务器级别可用。

You can add this configuration by using the `spi-user-profile-legacy-user-profile-read-only-attributes` and `'spi-user-profile-legacy-user-profile-admin-read-only-attributes'` options. For example:

您可以通过使用 `spi-user-profile-Heritage-user-profile-read-only-Attribute` 和 `'spi-user-profile-Heritage-user-profile-admin-read-only-Attribute'` 选项来添加此配置：

```
kc.[sh|bat] start --spi-user-profile-legacy-user-profile-read-only-attributes=foo,bar*
```

BASH

For this example, users and administrators would not be able to update attribute `foo`. Users would not be able to edit any attributes starting with the `bar`. So for example `bar` or `barrier`. Configuration is case-insensitive, so attributes like `FOO` or `BarRier` will be denied as well for this example. The wildcard character `*` is supported only at the end of the attribute name, so the administrator can effectively deny all the attributes starting with the specified character. The `*` in the middle of the attribute is considered as a normal character.

对于此示例，用户和管理员将无法更新 foo 属性。用户将无法编辑以工具条开始的任何属性。比如说吧台或者障碍物。配置是不区分大小写的，因此像 FOO 或 BarRier 这样的属性也将被拒绝。只有在属性名的末尾才支持通配符 *，所以管理员可以有效地拒绝从指定字符开始的所有属性。属性中间的 * 被认为是正常字符。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/threat/clickjacking.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/threat/clickjacking.adoc)

Clickjacking is a technique of tricking users into clicking on a user interface element different from what users perceive. A malicious site loads the target site in a transparent iFrame, overlaid on top of a set of dummy buttons placed directly under important buttons on the target site. When a user clicks a visible button, they are clicking a button on the hidden page. An attacker can steal a user's authentication credentials and access their resources by using this method.

点击劫持是一种欺骗用户点击与用户感知不同的用户界面元素的技术。恶意站点以透明的 iFrame 加载目标站点，该 iFrame 覆盖在一组虚拟按钮之上，该虚拟按钮直接放置在目标站点的重要按钮之下。当用户单击一个可见按钮时，他们正在单击隐藏页面上的一个按钮。攻击者可以使用此方法窃取用户的身份验证凭据并访问其资源。

By default, every response by Keycloak sets some specific HTTP headers that can prevent this from happening. Specifically, it sets [X-Frame-Options](https://datatracker.ietf.org/doc/html/rfc7034) (<https://datatracker.ietf.org/doc/html/rfc7034>) and [Content-Security-Policy](http://www.w3.org/TR/CSP/) (<http://www.w3.org/TR/CSP/>). You should take a look at the definition of both of these headers as there is a lot of fine-grain browser access you can control.

默认情况下，Keycloak 的每个响应都设置一些特定的 HTTP 头，可以防止这种情况发生。具体来说，它设置 X-Frame-Options 和 Content-Security-Policy。您应该看一下这两个头的定义，因为您可以控制许多细粒度的浏览器访问。

Procedure 程序

In the Admin Console, you can specify the values of the X-Frame-Options and Content-Security-Policy headers.

在管理控制台中，可以指定 X-Frame-Options 和 Content-Security-Policy 头的值。

1. Click the **Realm Settings** menu item.

单击“域设置”菜单项。

2. Click the **Security Defenses** tab.

单击“安全防御”选项卡。

Security Defenses 安全防御

The screenshot shows the Keycloak administration interface for the 'Master' realm. The left sidebar has a 'Manage' section with links for Clients, Client scopes, Realm roles, Users, Groups, Sessions, Events, Configure, Realm settings, Authentication, Identity providers, and User federation. The 'Configure' link is currently selected. The main content area is titled 'Master' and describes realm settings for users, applications, roles, and groups. It features a tab navigation bar with 'General', 'Login', 'Email', 'Themes', 'Keys', 'Events', 'Localization', 'Security defenses' (which is selected and highlighted in blue), and 'Sessions'. Below the tabs, there are two sub-sections: 'Headers' and 'Brute force detection'. Under 'Headers', several security headers are listed with their current values: X-Frame-Options is set to 'SAMEORIGIN'; Content-Security-Policy is set to 'frame-src \'self\'; frame-ancestors \'self\'; object-src \'none\''; Content-Security-Policy-Report-Only is empty; X-Content-Type-Options is set to 'nosniff'; X-Robots-Tag is set to 'none'; X-XSS-Protection is set to '1; mode=block'; and HTTP Strict Transport Security (HSTS) is set to 'max-age=31536000; includeSubDomains'. At the bottom of the page are 'Save' and 'Revert' buttons.

By default, Keycloak only sets up a *same-origin* policy for iframes.

默认情况下，Keycloak 只为 iframe 设置一个同源策略。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/threat/ssl.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/threat/ssl.adoc)

OAuth 2.0/OpenID Connect uses access tokens for security. Attackers can scan your network for access tokens and use them to perform malicious operations for which the token has permission. This attack is known as a man-in-the-middle attack. Use SSL/HTTPS for communication between the Keycloak auth server and the clients Keycloak secures to prevent man-in-the-middle attacks.

OAuth 2.0/OpenID Connect 使用访问令牌来保证安全性。攻击者可以扫描您的网络寻找访问令牌，并使用它们执行令牌具有权限的恶意操作。这种攻击被称为中间人攻击。使用 SSL/HTTPS 来实现钥匙斗篷身份验证服务器和客户机之间的通信，从而保证钥匙斗篷的安全，防止中间人攻击。

Keycloak has three modes for SSL/HTTPS. SSL is complex to set up, so Keycloak allows non-HTTPS communication over private IP addresses such as localhost, 192.168.x.x, and other private IP addresses. In production, ensure you enable SSL and SSL is compulsory for all operations.

对于 SSL/HTTPS，密钥隐形有三种模式。SSL 的设置非常复杂，因此 Keycloak 允许通过诸如 localhost、192.168.x.x 和其他私有 IP 地址之类的私有 IP 地址进行非 HTTPS 通信。在生产环境中，确保所有操作都必须启用 SSL 和 TLS。

On the adapter/client-side, you can disable the SSL trust manager. The trust manager ensures the client's identity that Keycloak communicates with is valid and ensures the DNS domain name against the server's certificate. In production, ensure that each of your client adapters uses a truststore to prevent DNS man-in-the-middle attacks.

在适配器/客户端，可以禁用 SSL 信任管理器。信任管理器可以确保 Keycloak 通信的客户端标识是有效的，并根据服务器的证书确保 DNS 域名。在生产环境中，确保您的每个客户端适配器都使用信任存储库来防止 DNS 中间人攻击。

[Edit this section](#) 编辑这部分

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/threat/csrf.adoc)

[Report an issue](#) 报告一个问题

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/threat/csrf.adoc)

A Cross-site request forgery (CSRF) attack uses HTTP requests from users that websites have already authenticated. Any site using cookie-based authentication is vulnerable to CSRF attacks. You can mitigate these attacks by matching a state cookie against a posted form or query parameter.

跨站请求伪造攻击使用网站已经认证过的用户的 HTTP 请求。任何使用基于 cookie 的身份验证的站点都很容易受到 CSRF 攻击。可以通过将状态 cookie 与发布的表单或查询参数进行匹配来减轻这些攻击。

The OAuth 2.0 login specification requires that a state cookie matches against a transmitted state parameter. Keycloak fully implements this part of the specification, so all logins are protected.

OAuth 2.0 登录规范要求状态 cookie 与传输的状态参数匹配。钥匙斗篷完全实现了规范的这一部分，因此所有的登录都受到保护。

The Keycloak Admin Console is a JavaScript/HTML5 application that makes REST calls to the backend Keycloak admin REST API. These calls all require bearer token authentication and consist of JavaScript Ajax calls, so CSRF is impossible. You can configure the admin REST API to validate the CORS origins.

钥匙斗篷管理控制台是一个 JavaScript/HTML5 应用程序，它对后端钥匙斗篷管理 REST API 进行 REST 调用。这些调用都需要承载令牌身份验证，并且由 JavaScript Ajax 调用组成，因此 CSRF 是不可能的。您可以配置管理员 REST API 来验证 CORS 起源。

The user account management section in Keycloak can be vulnerable to CSRF. To prevent CSRF attacks, Keycloak sets a state cookie and embeds the value of this cookie in hidden form fields or query parameters within action links. Keycloak checks the query/form parameter against the state cookie to verify that the user makes the call.

Keycloak 的用户帐户管理部门可能会受到 CSRF 的攻击。为了防止 CSRF 攻击，Keycon 设置一个状态 cookie，并将该 cookie 的值嵌入隐藏的表单字段或操作链接中的查询参数中。Keycon 根据状态 cookie 检查查询/表单参数，以验证用户是否进行了调用。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/threat/redirect.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/threat/redirect.adoc)

Make your registered redirect URIs as specific as feasible. Registering vague redirect URIs for Authorization Code Flows can allow malicious clients to impersonate another client with broader access. Impersonation can happen if two clients live under the same domain, for example.

尽可能使您注册的重定向 URI 具体化。为授权代码流注册模糊的重定向 URI 可以允许恶意客户机模拟具有更广泛访问权限的另一个客户机。例如，如果两个客户端位于同一个域下，则可能发生模拟。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/threat/fapi-compliance.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/threat/fapi-compliance.adoc)

To make sure that Keycloak server will validate your client to be more secure and FAPI compliant, you can configure client policies for the FAPI support. Details are described in the FAPI section of [Securing Applications and Services Guide](#) (https://www.keycloak.org/docs/latest/securing_apps/#_fapi-support). Among other things, this ensures some security best practices described above like SSL required for clients, secure redirect URI used and more of similar best practices.

为了确保 Keycloak 服务器将验证您的客户端以使其更加安全和符合 FAPI，您可以为 FAPI 支持配置客户端策略。详细信息在安全应用程序和服务指南的 FAPI 部分中描述。除其他外，这确保了上面描述的一些安全最佳实践，如客户端所需的 SSL、使用的安全重定向 URI 以及更多类似的最佳实践。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/threat/compromised-tokens.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/threat/compromised-tokens.adoc)

Keycloak includes several actions to prevent malicious actors from stealing access tokens and refresh tokens. The crucial action is to enforce SSL/HTTPS communication between Keycloak and its clients and applications. Keycloak does not enable SSL by default.

Keycloak 包括几个防止恶意角色窃取访问令牌和刷新令牌的操作。关键的措施是加强 Keycloak 与其客户端和应用程序之间的 SSL/HTTPS 通信。密钥斗篷默认情况下不启用 SSL。

Another action to mitigate damage from leaked access tokens is to shorten the token's lifespans. You can specify token lifespans within the timeouts page. Short lifespans for access tokens force clients and applications to refresh their access tokens after a short time. If an admin detects a leak, the admin can log out all user sessions to invalidate these refresh tokens or set up a revocation policy.

减轻泄漏访问令牌损害的另一个操作是缩短令牌的生命周期。您可以在超时页面中指定令牌生命周期。访问令牌的短生命周期迫使客户端和应用程序在短时间内刷新它们的访问令牌。如果管理员检测到泄漏，管理员可以注销所有用户会话以使这些刷新令牌失效或设置撤销策略。

Ensure refresh tokens always stay private to the client and are never transmitted.

确保刷新令牌始终对客户端保持私有，并且永远不会被传输。

You can mitigate damage from leaked access tokens and refresh tokens by issuing these tokens as holder-of-key tokens. See OAuth 2.0 Mutual TLS Client Certificate Bound Access Token for more information.

通过将这些令牌作为密钥持有者令牌发出，可以减轻泄漏访问令牌造成的损害，并刷新令牌。有关更多信息，请参见 OAuth 2.0 Mutual TLS 客户端证书绑定访问令牌。

If an access token or refresh token is compromised, access the Admin Console and push a not-before revocation policy to all applications. Pushing a not-before policy ensures that any tokens issued before that time become invalid. Pushing a new not-before policy ensures that applications must download new public keys from Keycloak and mitigate damage from a compromised realm signing key. See the keys chapter for more information.

如果访问令牌或刷新令牌受到损害，请访问管理控制台，并向所有应用程序推送一个“不前撤销”策略。推动一个 not-before 策略可以确保在该时间之前发出的任何令牌都无效。推行一个新的“不在前”策略可以确保应用程序必须从 Keycloak 下载新的公钥，并减轻受损领域签名密钥造成的损害。有关更多信息，请参见键章节。

You can disable specific applications, clients, or users if they are compromised.

如果特定的应用程序、客户端或用户受到威胁，则可以禁用它们。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/threat/compromised-codes.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/threat/compromised-codes.adoc)

For the OIDC Auth Code Flow, Keycloak generates a cryptographically strong random value for its authorization codes. An authorization code is used only once to obtain an access token.

对于 OIDC 授权代码流，Keycloak 为其授权代码生成一个加密强随机值。授权代码只使用一次，以获取访问令牌。

On the timeouts page in the Admin Console, you can specify the length of time an authorization code is valid. Ensure that the length of time is less than 10 seconds, which is long enough for a client to request a token from the code.

在管理控制台的超时页面上，可以指定授权代码有效的时间长度。确保时间长度小于10秒，这足以让客户端从代码中请求令牌。

You can also defend against leaked authorization codes by applying Proof Key for Code Exchange (PKCE) to clients.

还可以通过对客户端应用代码交换证明密钥(Proof Key for Code Exchange, PKCE)来防止授权代码泄漏。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/threat/open-redirect.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/threat/open-redirect.adoc)

An open redirector is an endpoint using a parameter to automatically redirect a user agent to the location specified by the parameter value without validation. An attacker can use the end-user authorization endpoint and the redirect URI parameter to use the authorization server as an open redirector, using a user's trust in an authorization server to launch a phishing attack.

开放重定向器是一个端点，它使用一个参数自动将用户代理重定向到参数值指定的位置，而不需要进行验证。攻击者可以使用最终用户授权端点和重定向 URI 参数将授权服务器用作打开的重定向器，使用用户对授权服务器的信任来发动钓鱼攻击。

Keycloak requires that all registered applications and clients register at least one redirection URI pattern. When a client requests that Keycloak performs a redirect, Keycloak checks the redirect URI against the list of valid registered URI patterns. Clients and applications must register as specific a URI pattern as possible to mitigate open redirector attacks.

密钥斗篷要求所有已注册的应用程序和客户端至少注册一个重定向 URI 模式。当客户端请求 Key枕执行重定向时，Key枕根据有效的已注册 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特定的 URI 模式，以减轻开放式重定向攻击。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/threat/password-db-compromised.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/threat/password-db-compromised.adoc)

Keycloak does not store passwords in raw text but as hashed text, using the PBKDF2 hashing algorithm. Keycloak performs 27,500 hashing iterations, the number of iterations recommended by the security community. This number of hashing iterations can adversely affect performance as PBKDF2 hashing

uses a significant amount of CPU resources.

密钥斗篷不以原始文本形式存储密码，而是使用 PBKDF2哈希算法以哈希文本形式存储密码。Key斗篷执行27,500次哈希迭代，这是安全社区推荐的迭代次数。由于 PBKDF2散列使用了大量的 CPU 资源，这个数量的散列迭代可能会对性能产生不利影响。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/threat/scope.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/threat/scope.adoc)

By default, new client applications have unlimited `role scope mappings`. Every access token for that client contains all permissions that the user has. If an attacker compromises the client and obtains the client's access tokens, each system that the user can access is compromised.

默认情况下，新客户端应用程序具有无限的角色范围映射。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得客户端的访问令牌，则用户可以访问的每个系统都会受到破坏。

Limit the roles of an access token by using the Scope menu for each client. Alternatively, you can set role scope mappings at the Client Scope level and assign Client Scopes to your client by using the Client Scope menu.

通过使用每个客户端的 Scope 菜单来限制访问令牌的角色。或者，您可以在 ClientScope 级别设置角色范围映射，并使用 ClientScope 菜单将 ClientScope 分配给客户端。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/threat/audience-limit.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/threat/audience-limit.adoc)

In environments with low levels of trust among services, limit the audiences on the token. See the [OAuth2 Threat Model](#) (<https://datatracker.ietf.org/doc/html/rfc6819#section-5.1.5.5>) and the Audience Support section for more information.

在服务间信任度较低的环境中，限制令牌上的受众。有关更多信息，请参见 OAuth2威胁模型和受众支持部分。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/threat/auth-sessions-limit.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/threat/auth-sessions-limit.adoc)

When a login page is opened for the first time in a web browser, Keycloak creates an object called authentication session that stores some useful information about the request. Whenever a new login page is opened from a different tab in the same browser, Keycloak creates a new record called authentication sub-session that is stored within the authentication session. Authentication requests can come from any type of clients such as the Admin CLI. In that case, a new authentication session is also created with one authentication sub-session. Please note that authentication sessions can be created also in other ways than using a browser flow. The text below is applicable regardless of the source flow.

当登录页面第一次在 Web 浏览器中打开时，Keycloak 创建一个名为身份验证会话的对象，该对象存储有关请求的一些有用信息。无论何时从同一浏览器中的不同选项卡打开新的登录页，Keycloak 都会创建一个名为身份验证子会话的新记录，该记录存储在身份验证会话中。身份验证请求可以来自任何类型的客户机，例如 AdminCLI。在这种情况下，还将创建一个带有一个身份验证子会话的新身份验证会话。请注意，除了使用浏览器流之外，还可以通过其他方式创建身份验证会话。无论源流如何，下面的文本都是适用的。



This section describes deployments that use the Infinispan provider for authentication sessions. 本节描述使用 Infinispan 提供程序进行身份验证会话的部署

Authentication session is internally stored as `RootAuthenticationSessionEntity`. Each `RootAuthenticationSessionEntity` can have multiple authentication sub-sessions stored within the `RootAuthenticationSessionEntity` as a collection of `AuthenticationSessionEntity` objects. Keycloak stores authentication sessions in a dedicated Infinispan cache. The number of `AuthenticationSessionEntity` per `RootAuthenticationSessionEntity` contributes to the size of each cache entry. Total memory footprint of authentication session cache is determined by the number of stored `RootAuthenticationSessionEntity` and by the number of `AuthenticationSessionEntity` within each `RootAuthenticationSessionEntity`.

身份验证会话在内部存储为 `RootAuthenticationSessionEntity`。每个 `RootAuthenticationSessionEntity` 可以在 `RootAuthenticationSessionEntity` 中存储多个身份验证子会话，作为 `AuthenticationSessionEntity` 对象的集合。密钥斗篷将身份验证会话存储在专用的 Infinispan 缓存中。每个 `RootAuthenticationSessionEntity` 的 `AuthenticationSessionEntity` 的数量决定了每个缓存条目的大小。身份验证会话缓存的总内存占用由存储的 `RootAuthenticationSessionEntity` 的数量和每个 `RootAuthenticationSessionEntity` 中的 `AuthenticationSessionEntity` 的数量决定。

The number of maintained `RootAuthenticationSessionEntity` objects corresponds to the number of unfinished login flows from the browser. To keep the number of `RootAuthenticationSessionEntity` under control, using an advanced firewall control to limit ingress network traffic is recommended.

维护的 `RootAuthenticationSessionEntity` 对象的数量对应于来自浏览器的未完成登录流的数量。为了控制 `RootAuthenticationSessionEntity` 的数量，建议使用高级防火墙控制来限制进入网络的流量。

Higher memory usage may occur for deployments where there are many active `RootAuthenticationSessionEntity` with a lot of `AuthenticationSessionEntity`. If the load balancer does not support or is not configured for session stickiness, the load over network in a cluster can increase significantly. The reason for this load is that each request that lands on a node that does not own the appropriate authentication session needs to retrieve and update the authentication session record in the owner node which involves a separate network transmission for both the retrieval and the storage.

对于有许多活动 `RootAuthenticationSessionEntity` 和许多 `AuthenticationSessionEntity` 的部署，可能会出现更高的内存使用率。如果负载平衡器不支持或者没有配置会话粘性，集群中的网络负载可能会显著增加。造成这种负载的原因是，每个到达不拥有适当身份验证会话的节点的请求都需要检索和更新所有者节点中的身份验证会话记录，这涉及到检索和存储的单独网络传输。

The maximum number of `AuthenticationSessionEntity` per `RootAuthenticationSessionEntity` can be configured in `authenticationSessions` SPI by setting property `authSessionsLimit`. The default value is set to 300 `AuthenticationSessionEntity` per a `RootAuthenticationSessionEntity`. When this limit is reached, the oldest authentication sub-session will be removed after a new authentication session request.

每个 `RootAuthenticationSessionEntity` 的 `AuthenticationSessionEntity` 的最大数量可以通过设置属性 `authSessionslimit` 在 `AuthenticationSessions` SPI 中配置。每个 `RootAuthenticationSessionEntity` 的默认值设置为300 `AuthenticationSessionEntity`。当达到此限制时，将在新的身份验证会话请求之后删除最老的身份验证子会话。

The following example shows how to limit the number of active `AuthenticationSessionEntity` per a `RootAuthenticationSessionEntity` to 100.

下面的示例演示如何将每个 `RootAuthenticationSessionEntity` 的活动 `AuthenticationSessionEntity` 的数量限制为100。

```
bin/kc.[sh|bat] start --spi-authentication-sessions-infinispan-auth-sessions-limit=100
```

BASH

The equivalent command for the new map storage:

新映射存储的等效命令：

```
bin/kc.[sh|bat] start --spi-authentication-sessions-map-auth-sessions-limit=100
```

BASH

[Edit this section 编辑这部分](#)
(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/threat/sql.adoc)

[Report an issue 报告一个问题](#)
(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/threat/sql.adoc)

Currently, Keycloak has no known SQL injection vulnerabilities.

目前，“钥匙斗篷”没有已知的 SQL 注入漏洞。

[Edit this section 编辑这部分](#)

(https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/account.adoc)

[Report an issue 报告一个问题](#)

(https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/account.adoc)

Keycloak users can manage their accounts through the Account Console. Users can configure their profiles, add two-factor authentication, include identity provider accounts, and oversee device activity.

密钥斗篷用户可以通过帐户控制台管理他们的帐户。用户可以配置他们的配置文件，添加双重身份验证，包括身份提供者帐户，并监督设备活动。

Additional resources 额外资源

- The Account Console can be configured in terms of appearance and language preferences. An example is adding attributes to the **Personal info** page by clicking **Personal info** link and completing and saving details. For more information, see reference:[https://www.keycloak.org/docs/latest/server_development/\[Server Developer Guide\]](https://www.keycloak.org/docs/latest/server_development/[Server Developer Guide]).

帐户控制台可以根据外观和语言首选项进行配置。例如，通过单击 Personal info 链接并完成和保存详细信息，将属性添加到 Personal info 页面。有关详细信息，请参阅参考 [https://www.keycloak.org/docs/latest/server_development/\[服务器开发人员指南\]](https://www.keycloak.org/docs/latest/server_development/[服务器开发人员指南])。

Accessing the Account Console 访问帐户控制台

Any user can access the Account Console.

任何用户都可以访问帐户控制台。

Procedure 程序

1. Make note of the realm name and IP address for the Keycloak server where your account exists.

请注意您帐户所在的密钥斗篷服务器的领域名称和 IP 地址。

2. In a web browser, enter a URL in this format: `server-root/realms/{realm-name}/account`.

在 Web 浏览器中，输入以下格式的 URL: `server-root/fields/{ domain-name }/account`。

3. Enter your login name and password.

输入您的登录名和密码。

Account Console 帐户控制台

 Personal info Manage your basic information Personal info	 Account security Control your password and account access Signing in Device activity Linked accounts
 Applications Track and manage your app permission to access your account Applications	 Resources Share your resources among team members Resources

Configuring ways to sign in 配置登录方式

You can sign in to this console using basic authentication (a login name and password) or two-factor authentication. For two-factor authentication, use one of the following procedures.

您可以使用基本身份验证(登录名和密码)或双因素身份验证登录到此控制台。对于双因素身份验证，请使用下列过程之一。

Two-factor authentication with OTP 使用 OTP 的双因素身份验证

Prerequisites 先决条件

- OTP is a valid authentication mechanism for your realm.

OTP 是您的领域的有效身份验证机制。

Procedure 程序

1. Click **Account security** in the menu.

单击菜单中的“帐户安全性”。

2. Click **Signing in**.

单击“登录”。

3. Click **Set up authenticator application**.

单击“设置验证程序应用程序”。

Signing in 正在登录

Signing in

Configure ways to sign in.

Basic authentication

Password

Sign in by entering your password.

My password

Created May 3, 2022,
11:56 AM

Update

Two-factor authentication

Authenticator application

Enter a verification code from authenticator application.

⋮

Authenticator application is not set up.

4. Follow the directions that appear on the screen to use either [FreeOTP](https://freeotp.github.io/) (<https://freeotp.github.io/>) or [Google Authenticator](https://play.google.com/store/apps/details?id=com.google.android.apps.authenticator2) (<https://play.google.com/store/apps/details?id=com.google.android.apps.authenticator2>) on your mobile device as your OTP generator.

按照屏幕上显示的指示，在移动设备上使用 FreeOTP 或 Google Authenticator 作为 OTP 生成器。

5. Scan the QR code in the screen shot into the OTP generator on your mobile device.

将屏幕截图中的二维码扫描到移动设备上的 OTP 生成器中。

6. Log out and log in again.

退出并再次登录。

7. Respond to the prompt by entering an OTP that is provided on your mobile device.

通过输入移动设备上提供的 OTP 来响应提示。

Two-factor authentication with WebAuthn 使用 WebAuthn 的双因素身份验证

Prerequisites 先决条件

- WebAuthn is a valid two-factor authentication mechanism for your realm. Please follow the WebAuthn section for more details.

对于您的领域，WebAuthn 是一种有效的双因素身份验证机制。

Procedure 程序

1. Click Account Security in the menu.

单击菜单中的“帐户安全性”。

2. Click Signing In.

单击“登录”。

3. Click Set up Security Key.

单击“设置安全密钥”。

Signing In 登录

Basic Authentication

Password

Log in by entering your password.

My Password

Created: August 19, 2021,
11:26 AM

Update

Two-Factor Authentication

Authenticator Application

[Set up Authenticator Application](#)

Enter a verification code from authenticator application.

Authenticator Application is not set up.

Security Key

[Set up Security Key](#)

Use your security key to log in.

Security Key is not set up.

4. Prepare your WebAuthn Security Key. How you prepare this key depends on the type of WebAuthn security key you use. For example, for a USB based Yubikey, you may need to put your key into the USB port on your laptop.

准备 WebAuthn 安全密钥。如何准备此密钥取决于所使用的 WebAuthn 安全密钥的类型。例如，对于一个基于 USB 的 Yubikey，您可能需要将您的密钥放入笔记本电脑的 USB 端口。

5. Click Register to register your security key.

单击“注册”注册安全密钥。

6. Log out and log in again.

退出并再次登录。

7. Assuming authentication flow was correctly set, a message appears asking you to authenticate with your Security Key as second factor.

假设身份验证流设置正确，将出现一条消息，要求您使用 SecurityKey 作为第二因素进行身份验证。

Passwordless authentication with WebAuthn 使用 WebAuthn 进行无密码身份验证

Prerequisites 先决条件

- WebAuthn is a valid passwordless authentication mechanism for your realm. Please follow the Passwordless WebAuthn section for more details.

WebAuthn 是您的领域的一种有效的无密码身份验证机制。

Procedure 程序

1. Click Account Security in the menu.

单击菜单中的“帐户安全性”。

2. Click Signing In.

单击“登录”。

3. Click Set up Security Key in the Passwordless section.

单击“无密码”部分中的“设置安全密钥”。

Signing In 登录

Basic Authentication

Password

Log in by entering your password.

My Password

Created: August 19, 2021,
11:26 AM

Update

Two-Factor Authentication

Authenticator Application

[Set up Authenticator Application](#)

Enter a verification code from authenticator application.

Authenticator Application is not set up.

Passwordless

Security Key

[Set up Security Key](#)

Use your security key for passwordless log in.

Security Key is not set up.

4. Prepare your WebAuthn Security Key. How you prepare this key depends on the type of WebAuthn security key you use. For example, for a USB based Yubikey, you may need to put your key into the USB port on your laptop.

准备 WebAuthn 安全密钥。如何准备此密钥取决于所使用的 WebAuthn 安全密钥的类型。例如，对于一个基于 USB 的 Yubikey，您可能需要将您的密钥放入笔记本电脑的 USB 端口。

5. Click **Register** to register your security key.

单击“注册”注册安全密钥。

6. Log out and log in again.

退出并再次登录。

7. Assuming authentication flow was correctly set, a message appears asking you to authenticate with your Security Key as second factor. You no longer need to provide your password to log in.

假设身份验证流设置正确，将出现一条消息，要求您使用 SecurityKey 作为第二因素进行身份验证。您不再需要提供登录密码。

Viewing device activity 查看设备活动

You can view the devices that are logged in to your account.

您可以查看登录到您的帐户的设备。

Procedure 程序

1. Click **Account security** in the menu.

单击菜单中的“帐户安全性”。

2. Click **Device activity**.

单击设备活动。

3. Log out a device if it looks suspicious.

如果设备看起来可疑，就注销它。

Devices 设备

Device activity

Sign out of any unfamiliar devices.

Signed in devices		Refresh		
IP address	Last accessed	Clients	Started	Expires
127.0.0.1	June 1, 2022, 11:36 AM	security-admin-console-v2, Account, Account Console	June 1, 2022, 10:54 AM	June 1, 2022, 8:54 PM

Adding an identity provider account 添加标识提供程序帐户

You can link your account with an identity broker. This option is often used to link social provider accounts.

您可以将您的帐户与身份代理链接。此选项通常用于连接社会服务提供者帐户。

Procedure 程序

1. Log into the Admin Console.

登录到管理控制台。

2. Click **Identity providers** in the menu.

单击菜单中的“身份提供程序”。

3. Select a provider and complete the fields.

选择提供程序并完成字段。

4. Return to the Account Console.

返回帐户控制台。

5. Click Account security in the menu.

单击菜单中的“帐户安全性”。

6. Click Linked accounts.

点击链接帐户。

The identity provider you added appears in this page.

您添加的标识提供程序显示在此页中。

Linked Accounts 关联帐户

Linked accounts

Manage logins through third-party accounts.

Linked login providers

No linked providers

Unlinked login providers



Github



Accessing other applications 访问其他应用程序

The Applications menu item shows users which applications you can access. In this case, only the Account Console is available.

“应用程序”菜单项显示用户可以访问的应用程序。在这种情况下，只有帐户控制台可用。

Applications 申请表

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

Viewing group memberships 查看组成员资格

You can view the groups you are associated with by clicking the **Groups** menu. If you select **Direct membership** checkbox, you will see only the groups you are direct associated with.

通过单击“组”菜单，可以查看与之关联的组。如果选中“直接成员资格”复选框，将只看到与您直接关联的组。

Prerequisites 先决条件

- You need to have the **view-groups** account role for being able to view **Groups** menu.

您需要具有视图组帐户角色，以便能够查看“组”菜单。

View group memberships 查看组成员资格

Groups		
<input type="checkbox"/> direct membership		
Name	path	direct membership
gov	/parent/gov	<input checked="" type="checkbox"/>
test	/test	<input checked="" type="checkbox"/>
parent	/parent	<input type="checkbox"/>
Edit this section 编辑这部分 (https://github.com/keycloak/keycloak-documentation/blob/master/server_admin/topics/admin-cli.adoc)		
Report an issue 报告一个问题 (https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12313920&components=12323375&issuetype=1&priority=3&description=File:%20server_admin/topics/admin-cli.adoc)		

With Keycloak, you can perform administration tasks from the command-line interface (CLI) by using the Admin CLI command-line tool.

通过使用“钥匙斗篷”，您可以使用“管理命令行界面”命令行工具执行来自管理程序(CLI)的管理任务。

Installing the Admin CLI 安装管理 CLI

Keycloak packages the Admin CLI server distribution with the execution scripts in the `bin` directory.

Keycover 使用 bin 目录中的执行脚本对 Admin CLI 服务器发行版进行打包。

The Linux script is called `kcadm.sh`, and the script for Windows is called `kcadm.bat`. Add the Keycloak server directory to your `PATH` to use the client from any location on your file system.

Linux 脚本称为 `kcadm.sh`, Windows 脚本称为 `kcadm.bat`。在 `PATH` 中添加“钥匙斗篷”服务器目录，以便从文件系统上的任何位置使用客户端。

For example:

例如:

- Linux:

```
$ export PATH=$PATH:$KEYCLOAK_HOME/bin  
$ kcadm.sh
```

- Windows:

视窗:

```
c:\> set PATH=%PATH%;%KEYCLOAK_HOME%\bin  
c:\> kcadm
```

You must set the `KEYCLOAK_HOME` environment variable to the path where you extracted the Keycloak Server distribution.

您必须将 `KEYCLOAK_HOME` 环境变量设置为解压出 Key斗篷服务器分发版的路径。



To avoid repetition, the rest of this document only uses Windows examples in places where the CLI differences are more than just in the `kcadm` command name.

为了避免重复，本文剩余部分只在 CLI 差异超过 `kcadm` 命令名的地方使用 Windows 示例。

Using the Admin CLI 使用管理 CLI

The Admin CLI makes HTTP requests to Admin REST endpoints. Access to the Admin REST endpoints requires authentication.

管理 CLI 向管理 REST 端点发出 HTTP 请求。访问管理 REST 端点需要身份验证。



Consult the Admin REST API documentation for details about JSON attributes for specific endpoints.

有关特定端点的 JSON 属性的详细信息，请参阅 Admin REST API 文档。

1. Start an authenticated session by logging in. You can now perform create, read, update, and delete (CRUD) operations.

通过登录启动一个经过身份验证的会话。您现在可以执行创建、读取、更新和删除(CRUD)操作。

For example:

例如:

- o Linux:

```
$ kcadm.sh config credentials --server http://localhost:8080 --realm demo --user admin --password $PASSWORD  
$ 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:8080/auth/realms/demo/protocol/openid-connect/auth"]')  
$ kcadm.sh get clients/$CID/installation/providers/keycloak-oidc-keycloak-json
```

- o Windows:

视窗:

```
c:\> kcadm config credentials --server http://localhost:8080 --realm demo --user admin --password $PASSWORD  
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:8080/auth/realms/demo/protocol/openid-connect/auth\"]"  
c:\> set /p CID=<clientid.txt  
c:\> kcadm get clients/%CID%/installation/providers/keycloak-oidc-keycloak-json
```

2. In a production environment, access Keycloak by using `https:` to avoid exposing tokens. If a trusted certificate authority, included in Java's default certificate truststore, has not issued a server's certificate, prepare a `truststore.jks` file and instruct the Admin CLI to use it.

在生产环境中，通过使用 `https:` 来访问 Keycloak，以避免暴露令牌。如果 Java 的默认证书信任库中包含的受信任的证书颁发机构没有颁发服务器的证书，那么准备一个 `truststore.jks` 文件并指示 Admin CLI 使用它。

For example:

例如:

- o Linux:

```
$ kcadm.sh config truststore --trustpass $PASSWORD ~/.keycloak/truststore.jks
```

- o Windows:

视窗:

```
c:\> kcadm config truststore --trustpass %PASSWORD% %HOMEPATH%\keycloak\truststore.jks
```

Authenticating 鉴定中

When you log in with the Admin CLI, you specify:

使用 Admin CLI 登录时，需要指定:

- A server endpoint URL

服务器端点 URL

- A realm

一个王国

- A user name

用户名

Another option is to specify a clientId only, which creates a unique service account for you to use.

另一种选择是仅指定 clientId，这将创建一个惟一的服务帐户供您使用。

When you log in using a user name, use a password for the specified user. When you log in using a clientId, you need the client secret only, not the user password. You can also use the `Signed JWT` rather than the client secret.

使用用户名登录时，请为指定用户使用密码。当您使用 clientId 登录时，只需要客户机机密，而不需要用户密码。您还可以使用 Signed JWT 而不是客户端机密。

Ensure the account used for the session has the proper permissions to invoke Admin REST API operations. For example, the `realm-admin` role of the `realm-management` client can administer the realm of the user.

确保用于会话的帐户具有调用 AdminRESTAPI 操作的适当权限。例如，领域管理客户机的领域管理角色可以管理用户的领域。

Two primary mechanisms are available for authentication. One mechanism uses `kcadm config credentials` to start an authenticated session.

有两种主要的机制可用于身份验证。一种机制使用 `kcadm` 配置凭据来启动一个经过身份验证的会话。

```
$ kcadm.sh config credentials --server http://localhost:8080 --realm master --user admin --passw
```

This mechanism maintains an authenticated session between the `kcadm` command invocations by saving the obtained access token and its associated refresh token. It can maintain other secrets in a private configuration file. See the next chapter for more information.

此机制通过保存获得的访问令牌及其相关的刷新令牌，在 `kcadm` 命令调用之间维护经过身份验证的会话。它可以在私有配置文件中维护其他机密。有关更多信息，请参见下一章。

The second mechanism authenticates each command invocation for the duration of the invocation. This mechanism increases the load on the server and the time spent on round trips obtaining tokens. The benefit of this approach is that it is unnecessary to save tokens between invocations, so nothing is saved to disk. Keycloak uses this mode when the `--no-config` argument is specified.

第二种机制在调用期间对每个命令调用进行身份验证。这种机制增加了服务器上的负载以及获取令牌往返所花费的时间。这种方法的好处是不需要在调用之间保存令牌，因此不会将任何东西保存到磁盘。当指定`--no-config`参数时，键隐形将使用此模式。

For example, when performing an operation, specify all the information required for authentication.

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 --realm master --user admin --p
```

Run the `kcadm.sh help` command for more information on using the Admin CLI.

运行 `kcadm.sh` 帮助命令以获得有关使用 Admin CLI 的更多信息。

Run the `kcadm.sh config credentials --help` command for more information about starting an authenticated session.

运行 `kcadm.sh` 配置凭据——`help` 命令以获得有关启动经过身份验证的会话的更多信息。

Working with alternative configurations 使用替代配置

By default, the Admin CLI maintains a configuration file named `kcadm.config`. Keycloak places this file in the user's home directory. In Linux-based systems, the full pathname is

`$HOME/.keycloak/kadm.config`. In Windows, the full pathname is

`%HOMEPATH%\keycloak\kadm.config`.

默认情况下，Admin CLI 维护一个名为 `kadm.config` 的配置文件。钥匙斗篷将此文件放在用户的主目录中。在基于 Linux 的系统中，完整路径名是 `$HOME/.keycloak/kadm.config`。在 Windows 中，完整路径名为 `%HOMEPATH%\keycloak\kadm.config`。

You can use the `--config` option to point to a different file or location so you can maintain multiple authenticated sessions in parallel.

可以使用—— config 选项指向不同的文件或位置，以便可以并行维护多个经过身份验证的会话。



Perform operations tied to a single configuration file from a single thread.

执行从单个线程绑定到单个配置文件的操作。

Ensure the configuration file is invisible to other users on the system. It contains access tokens and secrets that must be private. Keycloak creates the `~/.keycloak` directory and its contents automatically with proper access limits. If the directory already exists, Keycloak does not update the directory's permissions.

确保配置文件对系统上的其他用户是不可见的。它包含访问令牌和必须是私有的机密。密钥斗篷创建 `~/.keycloak` 目录及其内容自动适当的访问限制。如果目录已经存在，则“钥匙斗篷”不更新目录的权限。

It is possible to avoid storing secrets inside a configuration file, but doing so is inconvenient and increases the number of token requests. Use the `--no-config` option with all commands and specify the authentication information the `config credentials` command requires with each invocation of `kcadm`.

可以避免在配置文件中存储机密，但是这样做不方便，并且会增加令牌请求的数量。在所有命令中使用`--no-config` 选项，并在每次调用 `kcadm` 时指定配置凭据命令所需的身份验证信息。

Basic operations and resource URIs 基本操作和资源 URI

The Admin CLI can generically perform CRUD operations against Admin REST API endpoints with additional commands that simplify particular tasks.

通过使用简化特定任务的附加命令，Admin CLI 通常可以对 Admin REST API 端点执行 CRUD 操作。

The main usage pattern is listed here:

这里列出了主要的使用模式：

```
$ kcadm.sh create ENDPOINT [ARGUMENTS]
$ kcadm.sh get ENDPOINT [ARGUMENTS]
$ kcadm.sh update ENDPOINT [ARGUMENTS]
$ kcadm.sh delete ENDPOINT [ARGUMENTS]
```

The `create`, `get`, `update`, and `delete` commands map to the HTTP verbs `POST`, `GET`, `PUT`, and `DELETE`, respectively. `ENDPOINT` is a target resource URI and can be absolute (starting with `http:` or `https:`) or relative, that Keycloak uses to compose absolute URLs in the following format:

Create、GET、update 和 DELETE 命令分别映射到 HTTP 谓词 POST、GET、PUT 和 DELETE。ENDPOINT 是一个目标资源 URI，它可以是绝对的(从 `http:` 或 `https:` 开始)或相对的，Keycon 使用它以下列格式组合绝对 URL:

SERVER_URI/admin/realms/REALM/ENDPOINT

For example, if you authenticate against the server http://localhost:8080 and realm is master , using users as ENDPOINT creates the http://localhost:8080/admin/realms/master/users resource URL.

例如，如果您根据服务器 http://localhost:8080 进行身份验证，而域是 master，则使用用户作为 ENDPOINT 创建 http://localhost:8080/admin/realms/master/users 资源 URL。

If you set ENDPOINT to clients , the effective resource URI is http://localhost:8080/admin/realms/master/clients.

如果将 ENDPOINT 设置为客户端，则有效的资源 URI 是 http://localhost:8080/admin/realms/master/clients 的。

Keycloak has a realms endpoint that is the container for realms. It resolves to:

Key斗篷有一个领域端点，它是领域的容器。它解析为：

SERVER_URI/admin/realms

Keycloak has a serverinfo endpoint. This endpoint is independent of realms.

钥匙斗篷有一个 serverinfo 端点。这个端点独立于领域。

When you authenticate as a user with realm-admin powers, you may need to perform commands on multiple realms. If so, specify the -r option to tell the CLI which realm the command is to execute against explicitly. Instead of using REALM as specified by the --realm option of kcadm.sh config credentials , the command uses TARGET_REALM .

当您以具有领域管理能力的用户身份进行身份验证时，可能需要在多个领域上执行命令。如果是这样，请指定-r 选项来告诉 CLI 要显式地对哪个领域执行命令。该命令使用 TARGET_REALM，而不是使用 kadm.sh 配置凭据的— domain 选项指定的 REALM。

SERVER_URI/admin/realms/TARGET_REALM/ENDPOINT

For example:

例如：

```
$ kcadm.sh config credentials --server http://localhost:8080 --realm master --user admin --password
$ kcadm.sh create users -s username=testuser -s enabled=true -r demorealm
```

In this example, you start a session authenticated as the `admin` user in the `master` realm. You then perform a POST call against the resource URL
`http://localhost:8080/admin/realms/demorealm/users`.

在这个示例中，您启动一个在主领域中作为管理员用户进行身份验证的会话。然后对资源 URL `http://localhost:8080/admin/realms/demorealm/users` 执行 POST 调用。

The `create` and `update` commands send a JSON body to the server. You can use `-f FILENAME` to read a pre-made document from a file. When you can use the `-f -` option, Keycloak reads the message body from the standard input. You can specify individual attributes and their values, as seen in the `create users` example. Keycloak composes the attributes into a JSON body and sends them to the server.

创建和更新命令将一个 JSON 主体发送到服务器。您可以使用`-f FILENAME`从文件中读取预制的文档。当您可以使用`-f`选项时，Keycloak 从标准输入读取消息正文。您可以指定各个属性及其值，如创建用户示例所示。Keycloak 将属性组合到一个 JSON 主体中，并将它们发送到服务器。

Several methods are available in Keycloak to update a resource using the `update` command. You can determine the current state of a resource and save it to a file, edit that file, and send it to the server for an update.

在 Keycloak，有几种方法可以使用 `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，然后将其发送到服务器进行更新。

For example:

例如：

```
$ kcadm.sh get realms/demorealm > demorealm.json
$ vi demorealm.json
$ kcadm.sh update realms/demorealm -f demorealm.json
```

This method updates the resource on the server with the attributes in the sent JSON document.

此方法使用已发送的 JSON 文档中的属性更新服务器上的资源。

Another method is to perform an on-the-fly update by using the `-s`, `--set` options to set new values.

另一种方法是通过使用`-s`, `--set` 选项设置新值来执行动态更新。

For example:

例如：

```
$ kcadm.sh update realms/demorealm -s enabled=false
```

This method sets the `enabled` attribute to `false`.

此方法将启用的属性设置为 `false`。

By default, the `update` command performs a `get` and then merges the new attribute values with existing values. In some cases, the endpoint may support the `put` command but not the `get` command. You can use the `-n` option to perform a no-merge update, which performs a `put` command without first running a `get` command.

默认情况下，`update` 命令执行 `get`，然后将新属性值与现有值合并。在某些情况下，端点可能支持 `put` 命令，但不支持 `get` 命令。可以使用 `-n` 选项执行无合并更新，该更新不需要首先运行 `get` 命令即可执行 `put` 命令。

Realm operations 王国行动

Creating a new realm

建立一个新的国度

Use the `create` command on the `realms` endpoint to create a new enabled realm. Set the attributes to `realm` and `enabled`.

使用领域端点上的 `create` 命令创建新的已启用领域。将属性设置为 `domain` 并启用。

```
$ kcadm.sh create realms -s realm=demorealm -s enabled=true
```

Keycloak disables realms by default. You can use a realm immediately for authentication by enabling it.

密钥斗篷在默认情况下禁用域。您可以通过启用一个域来立即使用它进行身份验证。

A description for a new object can also be in JSON format.

新对象的描述也可以是 JSON 格式的。

```
$ kcadm.sh create realms -f demorealm.json
```

You can send a JSON document with realm attributes directly from a file or pipe the document to standard input.

您可以直接从文件中发送带领域属性的 JSON 文档，或者将文档导入标准输入。

For example:

例如：

- Linux:

```
$ kcadm.sh create realms -f - << EOF
{ "realm": "demorealm", "enabled": true }
EOF
```

- Windows:

视窗:

```
c:\> echo { "realm": "demorealm", "enabled": true } | kcadm create realms -f -
```

Listing existing realms

列出现有的领域

This command returns a list of all realms.

此命令返回所有领域的列表。

```
$ kcadm.sh get realms
```



Keycloak filters the list of realms on the server to return realms a user can see only.

Key斗篷过滤服务器上的领域列表，以返回用户只能看到的领域。

The list of all realm attributes can be verbose, and most users are interested in a subset of attributes, such as the realm name and the enabled status of the realm. You can specify the attributes to return by using the `--fields` option.

所有领域属性的列表可能非常冗长，而且大多数用户都对属性的子集感兴趣，例如领域名称和领域的启用状态。可以使用`--fields`选项指定要返回的属性。

```
$ kcadm.sh get realms --fields realm(enabled)
```

You can display the result as comma-separated values.

你可以把结果显示为逗号分隔值。

```
$ kcadm.sh get realms --fields realm --format csv --noquotes
```

Getting a specific realm

得到一个特定的领域

Append a realm name to a collection URI to get an individual realm.

将领域名称追加到集合 URI 以获取单个领域。

```
$ kcadm.sh get realms/master
```

Updating a realm

- 更新领域
1. Use the `-s` option to set new values for the attributes when you do not want to change all of the realm's attributes.

如果不想更改领域的所有属性，可以使用 `-s` 选项为属性设置新值。

For example:

例如:

```
$ kcadm.sh update realms/demorealm -s enabled=false
```

2. If you want to set all writable attributes to new values:

如果要将所有可写属性设置为新值:

- a. Run a `get` command.

运行 `get` 命令。

- b. Edit the current values in the JSON file.

编辑 JSON 文件中的当前值。

- c. Resubmit.

重新提交。

For example:

例如:

```
$ kcadm.sh get realms/demorealm > demorealm.json
$ vi demorealm.json
$ kcadm.sh update realms/demorealm -f demorealm.json
```

Deleting a realm

删除一个领域

Run the following command to delete a realm:

运行以下命令删除一个域:

```
$ kcadm.sh delete realms/demorealm
```

Turning on all login page options for the realm

打开领域的所有登录页面选项

Set the attributes that control specific capabilities to `true`.

将控制特定功能的属性设置为 `true`。

For example:

例如:

```
$ kcadm.sh update realms/demorealm -s registrationAllowed=true -s registrationEmailAsUsername=true
```

Listing the realm keys

列出领域键

Use the `get` operation on the `keys` endpoint of the target realm.

在目标领域的键端点上使用 `get` 操作。

```
$ kcadm.sh get keys -r demorealm
```

Generating new realm keys

生成新的领域密钥

1. Get the ID of the target realm before adding a new RSA-generated key pair.

在添加新的 RSA 生成的密钥对之前获取目标领域的 ID。

For example:

例如:

```
$ kcadm.sh get realms/demorealm --fields id --format csv --noquotes
```

2. Add a new key provider with a higher priority than the existing providers as revealed by `kcadm.sh get keys -r demorealm`.

添加一个新的密钥提供程序，该提供程序的优先级高于现有的提供程序，如 `kcadm.sh get keys -r demorealm` 所示。

For example:

例如:

- o Linux:

```
$ kcadm.sh create components -r demorealm -s name=rsa-generated -s providerId=rsa-generated
```

- Windows:

视窗:

```
c:\> kcadm create components -r demorealm -s name=rsa-generated -s providerId=rsa-generated
```

3. Set the `parentId` attribute to the value of the target realm's ID.

将 ParentId 属性设置为目标领域的 ID 的值。

The newly added key is now the active key, as revealed by `kcadm.sh get keys -r demorealm`.

新添加的键现在是活动键，正如 `kcadm.sh get keys -r demorealm` 所显示的那样。

Adding new realm keys from a Java Key Store file

从 JavaKeyStore 文件添加新的领域密钥

1. Add a new key provider to add a new key pair pre-prepared as a JKS file.

添加一个新的密钥提供程序，以添加一个作为 JKS 文件预先准备好的新密钥对。

For example, on:

例如:

- Linux:

```
$ kcadm.sh create components -r demorealm -s name=java-keystore -s providerId=java-keystor
```

- Windows:

视窗:

```
c:\> kcadm create components -r demorealm -s name=java-keystore -s providerId=java-keystor
```

2. Ensure you change the attribute values for `keystore`, `keystorePassword`, `keyPassword`, and `alias` to match your specific keystore.

确保更改 `keystore`、`keystorePassword`、`keyPassword` 和别名的属性值以匹配特定的 `keystore`。

3. Set the `parentId` attribute to the value of the target realm's ID.

将 ParentId 属性设置为目标领域的 ID 的值。

Making the key passive or disabling the key

使密钥被动或禁用密钥

1. Identify the key you want to make passive.

确定你想要被动的关键。

```
$ kcadm.sh get keys -r demorealm
```

2. Use the key's `providerId` attribute to construct an endpoint URI, such as `components/PROMISER_ID`.

使用密钥的 `provisionerId` 属性构造端点 URI，例如组件/PROMISER_ID。

3. Perform an `update`.

执行更新。

For example:

例如：

- Linux:

```
$ kcadm.sh update components/PROMISER_ID -r demorealm -s 'config.active=["false"]'
```

- Windows:

视窗：

```
c:\> kcadm update components/PROMISER_ID -r demorealm -s "config.active=[\"false\"]"
```

You can update other key attributes:

您可以更新其他关键属性：

4. Set a new `enabled` value to disable the key, for example, `config.enabled=["false"]`.

设置一个新的启用值以禁用密钥，例如 `config.enabled = ["false"]`。

5. Set a new `priority` value to change the key's priority, for example, `config.priority=["110"]`.

设置一个新的优先级值来更改密钥的优先级，例如 `config.priority = ["110"]`。

Deleting an old key

删除旧密钥

1. Ensure the key you are deleting is inactive and you have disabled it. This action is to prevent existing tokens held by applications and users from failing.

确保正在删除的键处于非活动状态，并且已禁用该键。此操作是为了防止应用程序和用户持有的现有令牌失败。

2. Identify the key to delete.

确定要删除的密钥。

```
$ kcadm.sh get keys -r demorealm
```

3. Use the `providerId` of the key to perform the delete.

使用密钥的 `providerId` 执行删除操作。

```
$ kcadm.sh delete components/PROVIDER_ID -r demorealm
```

Configuring event logging for a realm

为领域配置事件日志记录

Use the `update` command on the `events/config` endpoint.

在 `events/config` 端点上使用 `update` 命令。

The `eventsListeners` attribute contains a list of EventListenerProviderFactory IDs, specifying all event listeners that receive events. Attributes are available that control built-in event storage, so you can query past events using the Admin REST API. Keycloak has separate control over the logging of service calls (`eventsEnabled`) and the auditing events triggered by the Admin Console or Admin REST API (`adminEventsEnabled`). You can set up the `eventsExpiration` event to expire to prevent your database from filling. Keycloak sets `eventsExpiration` to time-to-live expressed in seconds.

EventsListener 属性包含 EventListenerProviderFactory ID 列表，指定接收事件的所有事件侦听器。属性可用于控制内置事件存储，因此可以使用 AdminRESTAPI 查询过去的事件。对于服务调用的日志记录(`eventsEnable`)和由管理控制台或管理 REST API (`adminEventsEnable`)触发的审计事件，Key枕具有单独的控制权。您可以将 `events sExpiration` 事件设置为过期，以防止数据库被填充。密钥斗篷设置 `events` 以秒为单位的生存时间过期。

You can set up a built-in event listener that receives all events and logs the events through JBoss-logging. Using the `org.keycloak.events` logger, Keycloak logs error events as `WARN` and other events as `DEBUG`.

您可以设置一个内置的事件侦听器，该侦听器接收所有事件并通过 JBoss 日志记录事件。通过使用 `org.Keycloak.events` 日志记录器，Key枕记录错误事件作为 `WARN`，其他事件作为 `DEBUG`。

For example:

例如:

- Linux:

```
$ kcadm.sh update events/config -r demorealm -s 'eventsListeners=["jboss-logging"]'
```

- Windows:

视窗:

```
c:\> kcadm update events/config -r demorealm -s "eventsListeners=[\"jboss-logging\"]"
```

For example:

例如:

You can turn on storage for all available ERROR events, not including auditing events, for two days so you can retrieve the events through Admin REST.

可以为所有可用的 ERROR 事件(不包括审计事件)打开存储两天，以便通过 AdminREST 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", "CLIENT_LOGIN_ERROR", "FEDERATED_IDENTITY_LINK_ERROR", "REMOVE_FEDERATED_IDENTITY_ERROR", "UPDATE_EMAIL_ERROR", "UPDATE_PROFILE_ERROR", "UPDATE_PASSWORD_ERROR", "UPDATE_TOTP_ERROR", "VERIFY_EMAIL_ERROR", "REMOVE_TOTP_ERROR", "SEND_VERIFY_EMAIL_ERROR", "SEND_RESET_PASSWORD_ERROR", "SEND_IDENTITY_PROVIDER_LINK_ERROR", "RESET_PASSWORD_ERROR", "IDENTITY_PROVIDER_FIRST_LOGIN_ERROR", "IDENTITY_PROVIDER_POST_LOGIN_ERROR", "CUSTOM_REQUIRED_ACTION_ERROR", "EXECUTE_ACTIONS_ERROR", "CLIENT_REGISTER_ERROR", "CLIENT_UPDATE_ERROR", "CLIENT_DELETE_ERROR"]' -s eventsExpiration=172800
```

- Windows:

视窗:

```
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_IDENTITY_ERROR\", \"UPDATE_EMAIL_ERROR\", \"UPDATE_PROFILE_ERROR\", \"UPDATE_PASSWORD_ERROR\", \"UPDATE_TOTP_ERROR\", \"VERIFY_EMAIL_ERROR\", \"REMOVE_TOTP_ERROR\", \"SEND_VERIFY_EMAIL_ERROR\", \"SEND_RESET_PASSWORD_ERROR\", \"SEND_IDENTITY_PROVIDER_LINK_ERROR\", \"RESET_PASSWORD_ERROR\", \"IDENTITY_PROVIDER_FIRST_LOGIN_ERROR\", \"IDENTITY_PROVIDER_POST_LOGIN_ERROR\", \"CUSTOM_REQUIRED_ACTION_ERROR\", \"EXECUTE_ACTIONS_ERROR\", \"CLIENT_REGISTER_ERROR\", \"CLIENT_UPDATE_ERROR\", \"CLIENT_DELETE_ERROR\"]" -s eventsExpiration=172800
```

You can reset stored event types to **all available event types**. Setting the value to an empty list is the same as enumerating all.

可以将存储的事件类型重置为所有可用的事件类型。将值设置为空列表与枚举所有。

```
$ kcadm.sh update events/config -r demorealm -s enabledEventTypes=[]
```

You can enable storage of auditing events.

您可以启用审核事件的存储。

```
$ kcadm.sh update events/config -r demorealm -s adminEventsEnabled=true -s adminEventsDetailsEna
```

You can get the last 100 events. The events are ordered from newest to oldest.

您可以得到最后的100个事件。这些事件按照从最新到最旧的顺序排列。

```
$ kcadm.sh get events --offset 0 --limit 100
```

You can delete all saved events.

您可以删除所有保存的事件。

```
$ kcadm delete events
```

Flushing the caches

冲走隐藏的东西

1. Use the `create` command with one of these endpoints to clear caches:

使用具有下列端点之一的 `create` 命令清除缓存:

- `clear-realm-cache`

Clear-domain-cache (清除领域缓存)

- `clear-user-cache`

- `clear-keys-cache`

清除键-缓存

2. Set `realm` to the same value as the target realm.

将域设置为与目标域相同的值。

For example:

例如:

```
$ kcadm.sh create clear-realm-cache -r demorealm -s realm=domorealm  
$ kcadm.sh create clear-user-cache -r demorealm -s realm=domorealm  
$ kcadm.sh create clear-keys-cache -r demorealm -s realm=domorealm
```

Importing a realm from exported .json file

从导出的.json文件导入域

1. Use the `create` command on the `partialImport` endpoint.

在 `partialImport` 端点上使用 `create` 命令。

2. Set `ifResourceExists` to `FAIL`, `SKIP`, or `OVERWRITE`.

将 `ifResourceExists` 设置为 `FAIL`、`SKIP` 或 `OVERWRITE`。

3. Use `-f` to submit the exported realm `.json` file.

使用-f 提交导出的 `realm.json` 文件。

For example:

例如:

```
$ kcadm.sh create partialImport -r demorealm2 -s ifResourceExists=FAIL -o -f demorealm.json
```

If the realm does not yet exist, create it first.

如果领域还不存在，首先创建它。

For example:

例如:

```
$ kcadm.sh create realms -s realm=demorealm2 -s enabled=true
```

Role operations 角色操作

Creating a realm role

创建领域角色

Use the `roles` endpoint to create a realm role.

使用角色端点创建领域角色。

```
$ kcadm.sh create roles -r demorealm -s name=user -s 'description=Regular user with a limited se
```

Creating a client role

创建客户角色

1. Identify the client.

确认客户身份。

2. Use the `get` command to list the available clients.

使用 get 命令列出可用的客户端。

```
$ kcadm.sh get clients -r demorealm --fields id,clientId
```

3. Create a new role by using the `clientId` attribute to construct an endpoint URI, such as `clients/ID/roles`.

通过使用 `clientId` 属性来构造端点 URI (例如 `client/ID/role`)，创建一个新角色。

For example:

例如:

```
$ kcadm.sh create clients/a95b6af3-0bdc-4878-ae2e-6d61a4eca9a0/roles -r demorealm -s name=ed
```

Listing realm roles

列出领域角色

Use the `get` command on the `roles` endpoint to list existing realm roles.

使用角色端点上的 `get` 命令列出现有的领域角色。

```
$ kcadm.sh get roles -r demorealm
```

You can use the `get-roles` command also.

您还可以使用 `get-role` 命令。

```
$ kcadm.sh get-roles -r demorealm
```

Listing client roles

列出客户端角色

Keycloak has a dedicated `get-roles` command to simplify the listing of realm and client roles. The command is an extension of the `get` command and behaves the same as the `get` command but with additional semantics for listing roles.

Keycloak 有一个专用的 `get-role` 命令，用于简化领域和客户端角色的列表。该命令是 `get` 命令的扩展，其行为与 `get` 命令相同，但具有用于列出角色的附加语义。

Use the `get-roles` command by passing it the `clientId` (`--cclientid`) option or the `id` (`--cid`) option to identify the client to list client roles.

通过向 `get-role` 命令传递 `clientId` (— `cclientid`) 选项或 `id` (— `cid`) 选项来标识客户端以列出客户端角色。

For example:

例如:

```
$ kcadm.sh get-roles -r demorealm --cclientid realm-management
```

Getting a specific realm role

获取特定的领域角色

Use the `get` command and the role `name` to construct an endpoint URI for a specific realm role, `roles/ROLE_NAME`, where `user` is the existing role's name.

使用 `get` 命令和角色名称为特定领域角色(`ROLE/ROLE_NAME`)构造端点 URI, 其中 `user` 是现有角色的名称。

For example:

例如:

```
$ kcadm.sh get roles/user -r demorealm
```

You can use the `get-roles` command, passing it a role name (`--rolename` option) or ID (`--roleid` option).

您可以使用 `get-role` 命令, 将角色名称(—`rolename` 选项)或 ID (—`roleid` 选项)传递给它。

For example:

例如:

```
$ kcadm.sh get-roles -r demorealm --rolename user
```

Getting a specific client role

获得一个特定的客户角色

Use the `get-roles` command, passing it the clientId attribute (`--cclientid` option) or ID attribute (`--cid` option) to identify the client, and pass the role name (`--rolename` option) or the role ID attribute (`--roleid`) to identify a specific client role.

使用 `get-role` 命令, 将 `clientId` 属性(—`cclientid` 选项)或 `ID` 属性(—`cid` 选项)传递给它以标识客户端, 并将角色名称(—`rolename` 选项)或角色 `ID` 属性(—`roleid`)传递给它以标识特定的客户端角色。

For example:

例如:

```
$ kcadm.sh get-roles -r demorealm --clientid realm-management --rolename manage-clients
```

Updating a realm role

更新领域角色

Use the `update` command with the endpoint URI you used to get a specific realm role.

对用于获取特定领域角色的端点 URI 使用 `update` 命令。

For example:

例如:

```
$ kcadm.sh update roles/user -r demorealm -s 'description=Role representing a regular user'
```

Updating a client role

更新客户端角色

Use the `update` command with the endpoint URI that you used to get a specific client role.

将 `update` 命令与用于获取特定客户端角色的端点 URI 一起使用。

For example:

例如:

```
$ kcadm.sh update clients/a95b6af3-0bdc-4878-ae2e-6d61a4eca9a0/roles/editor -r demorealm -s 'des
```

Deleting a realm role

删除领域角色

Use the `delete` command with the endpoint URI that you used to get a specific realm role.

对用于获取特定领域角色的端点 URI 使用 `delete` 命令。

For example:

例如:

```
$ kcadm.sh delete roles/user -r demorealm
```

Deleting a client role

删除客户端角色

Use the `delete` command with the endpoint URI that you used to get a specific client role.

对用于获取特定客户端角色的端点 URI 使用 `delete` 命令。

For example:

例如:

```
$ kcadm.sh delete clients/a95b6af3-0bdc-4878-ae2e-6d61a4eca9a0/roles/editor -r demorealm
```

Listing assigned, available, and effective realm roles for a composite role

列出为复合角色分配的、可用的和有效的领域角色

Use the `get-roles` command to list assigned, available, and effective realm roles for a composite role.

使用 `get-role` 命令列出为复合角色分配的、可用的和有效的领域角色。

1. To list **assigned** realm roles for the composite role, specify the target composite role by name (`--rname` option) or ID (`--rid` option).

若要列出为复合角色分配的领域角色，请按名称(—— `rname` 选项)或 ID (—— `rid` 选项)指定目标复合角色。

For example:

例如:

```
$ kcadm.sh get-roles -r demorealm --rname testrole
```

2. Use the `--effective` option to list **effective** realm roles.

使用—— `effect` 选项列出有效的领域角色。

For example:

例如:

```
$ kcadm.sh get-roles -r demorealm --rname testrole --effective
```

3. Use the `--available` option to list realm roles that you can add to the composite role.

使用—— `able` 选项列出可以添加到复合角色的领域角色。

For example:

例如:

```
$ kcadm.sh get-roles -r demorealm --rname testrole --available
```

Listing assigned, available, and effective client roles for a composite role

为复合角色列出已分配、可用和有效的客户端角色

Use the `get-roles` command to list assigned, available, and effective client roles for a composite role.

使用 `get-role` 命令列出为复合角色分配的、可用的和有效的客户端角色。

1. To list assigned client roles for the composite role, you can specify the target composite role by name (`--rname` option) or ID (`--rid` option) and client by the `clientId` attribute (`--cclientid` option) or ID (`--cid` option).

要为复合角色列出分配的客户机角色，可以通过名称(— rname 选项)或 ID (— rid 选项)指定目标复合角色，通过 `clientId` 属性(— cclientid 选项)或 ID (— cid 选项)指定客户机。

For example:

例如:

```
$ kcadm.sh get-roles -r demorealm --rname testrole --cclientid realm-management
```

2. Use the `--effective` option to list effective realm roles.

使用— effect 选项列出有效的领域角色。

For example:

例如:

```
$ kcadm.sh get-roles -r demorealm --rname testrole --cclientid realm-management --effective
```

3. Use the `--available` option to list realm roles that you can add to the target composite role.

使用— able 选项列出可以添加到目标复合角色的领域角色。

For example:

例如:

```
$ kcadm.sh get-roles -r demorealm --rname testrole --cclientid realm-management --available
```

Adding realm roles to a composite role

向复合角色添加领域角色

Keycloak provides an `add-roles` command for adding realm roles and client roles.

Key斗篷提供了一个添加角色命令，用于添加领域角色和客户端角色。

This example adds the `user` role to the composite role `testrole`.

此示例将用户角色添加到复合角色 `testrole`。

```
$ kcadm.sh add-roles --rname testrole --rolename user -r demorealm
```

Removing realm roles from a composite role

从复合角色中移除领域角色

Keycloak provides a `remove-roles` command for removing realm roles and client roles.

Key斗篷提供了一个删除角色命令，用于删除领域角色和客户端角色。

The following example removes the `user` role from the target composite role `testrole`.

下面的示例从目标复合角色 `testrole` 中删除用户角色。

```
$ kcadm.sh remove-roles --rname testrole --rolename user -r demorealm
```

Adding client roles to a realm role

将客户端角色添加到领域角色

Keycloak provides an `add-roles` command for adding realm roles and client roles.

Key斗篷提供了一个添加角色命令，用于添加领域角色和客户端角色。

The following example adds the roles defined on the client `realm-management`, `create-client`, and `view-users`, to the `testrole` composite role.

下面的示例将在客户机领域管理、创建客户机和视图用户上定义的角色添加到 `testrole` 组合角色。

```
$ kcadm.sh add-roles -r demorealm --rname testrole --cclientid realm-management --rolename creat
```

Adding client roles to a client role

向客户端角色添加客户端角色

1. Determine the ID of the composite client role by using the `get-roles` command.

通过使用 `get-role` 命令确定复合客户端角色的 ID。

For example:

例如:

```
$ kcadm.sh get-roles -r demorealm --cclientid test-client --rolename operations
```

2. Assume that a client exists with a clientId attribute named `test-client`, a client role named `support`, and a client role named `operations` which becomes a composite role that has an ID of "fc400897-ef6a-4e8c-872b-1581b7fa8a71".

假设客户端具有名为 `test-client` 的 clientId 属性、名为 `support` 的客户端角色和名为 `operations` 的客户端角色，后者成为一个 ID 为“`fc400897-ef6a-4e8c-872b-1581b7fa8a71`”的复合角色。

3. Use the following example to add another role to the composite role.

使用下面的示例向复合角色添加另一个角色。

```
$ kcadm.sh add-roles -r demorealm --cclientid test-client --rid fc400897-ef6a-4e8c-872b-1581b7fa8a71
```

4. List the roles of a composite role by using the `get-roles --all` command.

使用 `get-role --all` 命令列出复合角色的角色。

For example:

例如:

```
$ kcadm.sh get-roles --rid fc400897-ef6a-4e8c-872b-1581b7fa8a71 --all
```

Removing client roles from a composite role

从复合角色中移除客户端角色

Use the `remove-roles` command to remove client roles from a composite role.

使用 `move-role` 命令可以从复合角色中移除客户端角色。

Use the following example to remove two roles defined on the client `realm-management`, the `create-client` role and the `view-users` role, from the `testrole` composite role.

使用以下示例从 `testrole` 组合角色中删除在客户机领域管理中定义的两个角色，即 `create-client` 角色和 `view-users` 角色。

```
$ kcadm.sh remove-roles -r demorealm --rname testrole --cclientid realm-management --rolename cr
```

Adding client roles to a group

向组中添加客户端角色

Use the `add-roles` command to add realm roles and client roles.

使用 add-role 命令添加领域角色和客户端角色。

The following example adds the roles defined on the client `realm-management`, `create-client` and `view-users`, to the `Group` group (`--gname` option). Alternatively, you can specify the group by ID (`--gid` option).

下面的示例将在客户机领域管理、`create-client` 和 `view-users` 上定义的角色添加到 `Group` 组(—— `gname` 选项)。或者，可以通过 ID (—— `gid` 选项)指定组。

See Group operations for more information.

有关更多信息，请参见组操作。

```
$ kcadm.sh add-roles -r demorealm --gname Group --cclientid realm-management --rolename create-c
```

Removing client roles from a group

从组中移除客户端角色

Use the `remove-roles` command to remove client roles from a group.

使用 `move-role` 命令可以从组中删除客户端角色。

The following example removes two roles defined on the client `realm management`, `create-client` and `view-users`, from the `Group` group.

下面的示例从 `Group` 组中移除在客户端领域管理上定义的两个角色 `create-client` 和 `view-users`。

See Group operations for more information.

有关更多信息，请参见组操作。

```
$ kcadm.sh remove-roles -r demorealm --gname Group --cclientid realm-management --rolename creat
```

Client operations 客户操作

Creating a client

创造一个客户

1. Run the `create` command on a `clients` endpoint to create a new client.

在客户端端点上运行 `create` 命令以创建新客户端。

For example:

例如:

```
$ kcadm.sh create clients -r demorealm -s clientId=myapp -s enabled=true
```

2. Specify a secret if to set a secret for adapters to authenticate.

如果要设置用于适配器身份验证的机密，请指定一个机密。

For example:

例如:

```
$ kcadm.sh create clients -r demorealm -s clientId=myapp -s enabled=true -s clientAuthentica
```

Listing clients

上市客户

Use the `get` command on the `clients` endpoint to list clients.

使用客户端端点上的 `get` 命令列出客户端。

This example filters the output to list only the `id` and `clientId` attributes:

这个示例将输出过滤为只列出 `id` 和 `clientId` 属性:

```
$ kcadm.sh get clients -r demorealm --fields id,clientId
```

Getting a specific client

找一个特定的客户

Use the client ID to construct an endpoint URI that targets a specific client, such as `clients/ID`.

使用客户端 ID 构造以特定客户端为目标的端点 URI，例如 `client/ID`。

For example:

例如:

```
$ kcadm.sh get clients/c7b8547f-e748-4333-95d0-410b76b3f4a3 -r demorealm
```

Getting the current secret for a specific client

获取特定客户端的当前秘密

Use the client ID to construct an endpoint URI, such as `clients/ID/client-secret`.

使用客户端 ID 构造端点 URI，例如 `client/ID/client-secret`。

For example:

例如:

```
$ kcadm.sh get clients/$CID/client-secret
```

Generate a new secret for a specific client

为特定的客户端生成新的机密

Use the client ID to construct an endpoint URI, such as `clients/ID/client-secret`.

使用客户端 ID 构造端点 URI, 例如 `client/ID/client-secret`。

For example:

例如:

```
$ kcadm.sh create clients/$CID/client-secret
```

Updating the current secret for a specific client

为特定客户端更新当前机密

Use the client ID to construct an endpoint URI, such as `clients/ID`.

使用客户端 ID 构造端点 URI, 如 `client/ID`。

For example:

例如:

```
$ kcadm.sh update clients/$CID -s "secret=newSecret"
```

Getting an adapter configuration file (`keycloak.json`) for a specific client

Use the client ID to construct an endpoint URI that targets a specific client, such as `clients/ID/installation/providers/keycloak-oidc-keycloak-json`.

使用客户端 ID 构造一个端点 URI, 该 URI 的目标是特定的客户端, 例如: `client/ID/install/ 提供者/key 密匙斗篷-oidc-密匙斗篷-json`。

For example:

例如:

```
$ kcadm.sh get clients/c7b8547f-e748-4333-95d0-410b76b3f4a3/installation/providers/keycloak-oidc
```

Getting a WildFly subsystem adapter configuration for a specific client

为特定客户端获取 WildFly 子系统适配器配置

Use the client ID to construct an endpoint URI that targets a specific client, such as `clients/ID/installation/providers/keycloak-oidc-jboss-subsystem`.

使用客户端 ID 构造一个端点 URI，该 URI 的目标是特定的客户端，比如客户端/ID/安装/提供程序/密钥斗篷 - oidc-jboss-子系统。

For example:

例如:

```
$ kcadm.sh get clients/c7b8547f-e748-4333-95d0-410b76b3f4a3/installation/providers/keycloak-oidc
```

Getting a Docker-v2 example configuration for a specific client

获取特定客户端的 Docker-v2示例配置

Use the client ID to construct an endpoint URI that targets a specific client, such as `clients/ID/installation/providers/docker-v2-compose-yaml`.

使用客户端 ID 构造一个端点 URI，该 URI 的目标是特定的客户端，例如 client/ID/install/vers/docker-v2-compose-yaml。

The response is in `.zip` format.

响应是`.zip`格式的。

For example:

例如:

```
$ kcadm.sh get http://localhost:8080/admin/realm/demorealm/clients/8f271c35-44e3-446f-8953-b089
```

Updating a client

更新客户信息

Use the `update` command with the same endpoint URI that you use to get a specific client.

使用与获取特定客户端相同的端点 URI 的 update 命令。

For example:

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:

视窗:

```
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
```

Deleting a client

删除客户端

Use the `delete` command with the same endpoint URI that you use to get a specific client.

使用具有与获取特定客户端相同的端点 URI 的 `delete` 命令。

For example:

例如:

```
$ kcadm.sh delete clients/c7b8547f-e748-4333-95d0-410b76b3f4a3 -r demorealm
```

Adding or removing roles for client's service account

为客户端的服务帐户添加或删除角色

A client's service account is a user account with username `service-account-CLIENT_ID`. You can perform the same user operations on this account as a regular account.

客户端的服务帐户是具有用户名 `service-account-CLIENT_ID` 的用户帐户。您可以在此帐户上执行与常规帐户相同的用户操作。

User operations 用户操作

Creating a user

创建用户

Run the `create` command on the `users` endpoint to create a new user.

在用户端点上运行 `create` 命令以创建新用户。

For example:

例如:

```
$ kcadm.sh create users -r demorealm -s username=testuser -s enabled=true
```

Listing users

列出用户

Use the `users` endpoint to list users. The target user must change their password the next time they log in.

使用用户端点列出用户。目标用户下次登录时必须更改密码。

For example:

例如:

```
$ kcadm.sh get users -r demorealm --offset 0 --limit 1000
```

You can filter users by `username`, `firstName`, `lastName`, or `email`.

您可以根据用户名、`firstName`、`lastName` 或电子邮件筛选用户。

For example:

例如:

```
$ kcadm.sh get users -r demorealm -q email=google.com  
$ kcadm.sh get users -r demorealm -q username=testuser
```

 Filtering does not use exact matching. This example matches the value of the `username` attribute against the `*testuser*` pattern.

筛选不使用精确匹配。

You can filter across multiple attributes by specifying multiple `-q` options. Keycloak returns users that match the condition for all the attributes only.

您可以通过指定 `multi-q` 选项来过滤多个属性。键隐藏只返回与所有属性的条件匹配的用户。

Getting a specific user

获取特定用户

Use the user ID to compose an endpoint URI, such as `users/USER_ID`.

使用用户 ID 组成端点 URI, 例如 `USER/USER_ID`。

For example:

例如:

```
$ kcadm.sh get users/0ba7a3fd-6fd8-48cd-a60b-2e8fd82d56e2 -r demorealm
```

Updating a user

更新用户

Use the `update` command with the same endpoint URI that you use to get a specific user.

使用与获取特定用户所使用的端点 URI 相同的 `update` 命令。

For example:

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:

视窗:

```
c:\> kcadm update users/0ba7a3fd-6fd8-48cd-a60b-2e8fd82d56e2 -r demorealm -s  
"requiredActions=  
[\"VERIFY_EMAIL\", \"UPDATE_PROFILE\", \"CONFIGURE_TOTP\", \"UPDATE_PASSWORD\"]"
```

Deleting a user

删除用户

Use the `delete` command with the same endpoint URI that you use to get a specific user.

使用具有与获取特定用户相同的端点 URI 的 `delete` 命令。

For example:

例如:

```
$ kcadm.sh delete users/0ba7a3fd-6fd8-48cd-a60b-2e8fd82d56e2 -r demorealm
```

Resetting a user's password

重置用户密码

Use the dedicated `set-password` command to reset a user's password.

使用专用的 `set-password` 命令重置用户的密码。

For example:

例如:

```
$ kcadm.sh set-password -r demorealm --username testuser --new-password NEWPASSWORD --temporary
```

This command sets a temporary password for the user. The target user must change the password the next time they log in.

此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。

You can use `--userid` to specify the user by using the `id` attribute.

您可以使用—— `userid` 通过 `id` 属性指定用户。

You can achieve the same result using the `update` command on an endpoint constructed from the one you used to get a specific user, such as `users/USER_ID/reset-password`.

您可以在端点上使用 `update` 命令实现相同的结果，该端点由您用来获取特定用户的端点构造而成，例如 `USER/USER_ID/reset-password`。

For example:

例如:

```
$ kcadm.sh update users/0ba7a3fd-6fd8-48cd-a60b-2e8fd82d56e2/reset-password -r demorealm -s type
```

The `-n` parameter ensures that Keycloak performs the `PUT` command without performing a `GET` command before the `PUT` command. This is necessary because the `reset-password` endpoint does not support `GET`.

`N` 参数确保 Keycloak 在 `PUT` 命令之前执行 `PUT` 命令，而不执行 `GET` 命令。这是必要的，因为重置密码端点不支持 `GET`。

Listing assigned, available, and effective realm roles for a user

列出为用户分配的、可用的和有效的领域角色

You can use a `get-roles` command to list assigned, available, and effective realm roles for a user.

可以使用 `get-role` 命令列出为用户分配的、可用的和有效的领域角色。

1. Specify the target user by user name or ID to list the user's **assigned** realm roles.

通过用户名或 ID 指定目标用户，以列出用户分配的领域角色。

For example:

例如:

```
$ kcadm.sh get-roles -r demorealm --username testuser
```

2. Use the `--effective` option to list **effective** realm roles.

使用`--effect` 选项列出有效的领域角色。

For example:

例如:

```
$ kcadm.sh get-roles -r demorealm --username testuser --effective
```

3. Use the `--available` option to list realm roles that you can add to a user.

使用`--able` 选项列出可以添加到用户的领域角色。

For example:

例如:

```
$ kcadm.sh get-roles -r demorealm --username testuser --available
```

Listing assigned, available, and effective client roles for a user

列出为用户分配的、可用的和有效的客户端角色

Use a `get-roles` command to list assigned, available, and effective client roles for a user.

使用 `get-role` 命令列出为用户分配的、可用的和有效的客户端角色。

1. Specify the target user by user name (`--username` option) or ID (`--uid` option) and client by a `clientId` attribute (`--cclientid` option) or an ID (`--cid` option) to list **assigned** client roles for the user.

通过用户名(`--username` 选项)或 ID (`--uid` 选项)指定目标用户，通过 `clientId` 属性(`--cclientid` 选项)或 ID (`--cid` 选项)指定客户端，以列出为用户分配的客户端角色。

For example:

例如:

```
$ kcadm.sh get-roles -r demorealm --username testuser --clientid realm-management
```

2. Use the `--effective` option to list **effective** realm roles.

使用—— effect 选项列出有效的领域角色。

For example:

例如:

```
$ kcadm.sh get-roles -r demorealm --username testuser --clientid realm-management --effect
```

3. Use the `--available` option to list realm roles that you can add to a user.

使用—— able 选项列出可以添加到用户的领域角色。

For example:

例如:

```
$ kcadm.sh get-roles -r demorealm --username testuser --clientid realm-management --availa
```

Adding realm roles to a user

向用户添加领域角色

Use an `add-roles` command to add realm roles to a user.

使用 `add-role` 命令向用户添加领域角色。

Use the following example to add the `user` role to user `testuser`:

使用以下示例向用户 `testuser` 添加用户角色:

```
$ kcadm.sh add-roles --username testuser --rolename user -r demorealm
```

Removing realm roles from a user

从用户移除领域角色

Use a `remove-roles` command to remove realm roles from a user.

使用删除角色命令从用户删除领域角色。

Use the following example to remove the `user` role from the user `testuser`:

使用以下示例从用户 testuser 中删除用户角色:

```
$ kcadm.sh remove-roles --username testuser --rolename user -r demorealm
```

Adding client roles to a user

向用户添加客户端角色

Use an `add-roles` command to add client roles to a user.

使用 `add-role` 命令向用户添加客户端角色。

Use the following example to add two roles defined on the client `realm management`, the `create-client` role and the `view-users` role, to the user `testuser`.

使用以下示例向用户 testuser 添加在客户端领域管理中定义的两个角色: `create-client` 角色和 `view-users` 角色。

```
$ kcadm.sh add-roles -r demorealm --username testuser --clientid realm-management --rolename c
```

Removing client roles from a user

从用户移除客户端角色

Use a `remove-roles` command to remove client roles from a user.

使用删除角色命令从用户删除客户端角色。

Use the following example to remove two roles defined on the realm management client:

使用以下示例删除在领域管理客户端上定义的两个角色:

```
$ kcadm.sh remove-roles -r demorealm --username testuser --clientid realm-management --rolename
```

Listing a user's sessions

列出用户的会话

1. Identify the user's ID,

识别用户的 ID,

2. Use the ID to compose an endpoint URI, such as `users/ID/sessions`.

使用 ID 组成端点 URI, 如用户/ID/会话。

3. Use the `get` command to retrieve a list of the user's sessions.

使用 get 命令检索用户会话的列表。

For example:

例如:

```
$kcadm get users/6da5ab89-3397-4205-afaa-e201ff638f9e/sessions
```

Logging out a user from a specific session

从特定会话注销用户

1. Determine the session's ID as described earlier.

如前所述确定会话的 ID。

2. Use the session's ID to compose an endpoint URI, such as `sessions/ID`.

使用会话的 ID 组成端点 URI, 如会话/ID。

3. Use the `delete` command to invalidate the session.

使用 `delete` 命令使会话无效。

For example:

例如:

```
$ kcadm.sh delete sessions/d0eaa7cc-8c5d-489d-811a-69d3c4ec84d1
```

Logging out a user from all sessions

从所有会话中注销用户

Use the user's ID to construct an endpoint URI, such as `users/ID/logout`.

使用用户 ID 构造端点 URI, 例如 `user/ID/logout`。

Use the `create` command to perform `POST` on that endpoint URI.

使用 `create` 命令对该端点 URI 执行 POST。

For example:

例如:

```
$ kcadm.sh create users/6da5ab89-3397-4205-afaa-e201ff638f9e/logout -r demorealm -s realm=domore
```

Group operations 集体行动

Creating a group

建立一个团队

Use the `create` command on the `groups` endpoint to create a new group.

使用组端点上的 `create` 命令创建新组。

For example:

例如:

```
$ kcadm.sh create groups -r demorealm -s name=Group
```

Listing groups

列表组别

Use the `get` command on the `groups` endpoint to list groups.

使用组端点上的 `get` 命令列出组。

For example:

例如:

```
$ kcadm.sh get groups -r demorealm
```

Getting a specific group

找一个特定的团队

Use the group's ID to construct an endpoint URI, such as `groups/GROUP_ID`.

使用组的 ID 构造端点 URI, 例如 `GROUP/GROUP_ID`。

For example:

例如:

```
$ kcadm.sh get groups/51204821-0580-46db-8f2d-27106c6b5ded -r demorealm
```

Updating a group

更新群组

Use the `update` command with the same endpoint URI that you use to get a specific group.

使用与获取特定组所使用的端点 URI 相同的 `update` 命令。

For example:

例如:

```
$ kcadm.sh update groups/51204821-0580-46db-8f2d-27106c6b5ded -s 'attributes.email=[{"group@example.com"}]
```

Deleting a group

删除一个组

Use the `delete` command with the same endpoint URI that you use to get a specific group.

使用具有与用于获取特定组相同的端点 URI 的 `delete` 命令。

For example:

例如:

```
$ kcadm.sh delete groups/51204821-0580-46db-8f2d-27106c6b5ded -r demorealm
```

Creating a subgroup

创建子组

Find the ID of the parent group by listing groups. Use that ID to construct an endpoint URI, such as `groups/GROUP_ID/children`.

通过列出组来查找父组的 ID。使用该 ID 构造端点 URI，例如 `GROUP/GROUP_ID/children`。

For example:

例如:

```
$ kcadm.sh create groups/51204821-0580-46db-8f2d-27106c6b5ded/children -r demorealm -s name=SubGroup
```

Moving a group under another group

把一组人移到另一组人下面

1. Find the ID of an existing parent group and the ID of an existing child group.

查找现有父组的 ID 和现有子组的 ID。

2. Use the parent group's ID to construct an endpoint URI, such as

`groups/PARENT_GROUP_ID/children`.

使用父组的 ID 构造端点 URI，例如 `group/PARENT_GROUP_ID/children`。

3. Run the `create` command on this endpoint and pass the child group's ID as a JSON body.

在此端点上运行 `create` 命令，并将子组的 ID 作为 JSON 正文传递。

For example:

例如:

```
$ kcadm.sh create groups/51204821-0580-46db-8f2d-27106c6b5ded/children -r demorealm -s id=08d410
```

Get groups for a specific user

获取特定用户的组

Use a user's ID to determine a user's membership in groups to compose an endpoint URI, such as `users/USER_ID/groups`.

使用用户 ID 确定用户在组中的成员关系，以组成端点 URI，例如 `USER/USER_ID/groups`。

For example:

例如:

```
$ kcadm.sh get users/b544f379-5fc4-49e5-8a8d-5cfb71f46f53/groups -r demorealm
```

Adding a user to a group

向组添加用户

Use the `update` command with an endpoint URI composed of a user's ID and a group's ID, such as `users/USER_ID/groups/GROUP_ID`, to add a user to a group.

使用带有端点 URI (由用户 ID 和组 ID 组成) 的 `update` 命令(例如 `USER/USER_ID/groups/GROUP_ID`)将用户添加到组中。

For example:

例如:

```
$ kcadm.sh update users/b544f379-5fc4-49e5-8a8d-5cfb71f46f53/groups/ce01117a-7426-4670-a29a-5c11
```

Removing a user from a group

从组中删除用户

Use the `delete` command on the same endpoint URI you use for adding a user to a group, such as `users/USER_ID/groups/GROUP_ID`, to remove a user from a group.

在用于将用户添加到组(如 `users/USER_ID/groups/GROUP_ID`)的同一端点 URI 上使用 `delete` 命令从组中删除用户。

For example:

例如:

```
$ kcadm.sh delete users/b544f379-5fc4-49e5-8a8d-5cfb71f46f53/groups/ce01117a-7426-4670-a29a-5c11
```

Listing assigned, available, and effective realm roles for a group

列出为组分配的、可用的和有效的领域角色

Use a dedicated `get-roles` command to list assigned, available, and effective realm roles for a group.

使用专用的 `get-role` 命令列出为组分配的、可用的和有效的领域角色。

1. Specify the target group by name (`--gname` option), path (`--gpath` option), or ID (`--gid` option) to list **assigned** realm roles for the group.

通过名称(`--gname` 选项)、路径(`--gpath` 选项)或 ID (`--gid` 选项)指定目标组，以列出分配给该组的领域角色。

For example:

例如:

```
$ kcadm.sh get-roles -r demorealm --gname Group
```

2. Use the `--effective` option to list **effective** realm roles.

使用`--effective` 选项列出有效的领域角色。

For example:

例如:

```
$ kcadm.sh get-roles -r demorealm --gname Group --effective
```

3. Use the `--available` option to list realm roles that you can add to the group.

使用`--available` 选项列出可以添加到组中的领域角色。

For example:

例如:

```
$ kcadm.sh get-roles -r demorealm --gname Group --available
```

Listing assigned, available, and effective client roles for a group

列出为组分配的、可用的和有效的客户端角色

Use the `get-roles` command to list assigned, available, and effective client roles for a group.

使用 `get-role` 命令列出为组分配的、可用的和有效的客户端角色。

1. Specify the target group by name (`--gname` option) or ID (`--gid` option),

通过名称(— gname 选项)或 ID (— gid 选项)指定目标组,

2. Specify the client by the clientId attribute (`--cclientid` option) or ID (`--id` option) to list assigned client roles for the user.

通过 clientId 属性(— cclientid 选项)或 ID (— ID 选项)指定客户端, 以列出为用户分配的客户端角色。

For example:

例如:

```
$ kcadm.sh get-roles -r demorealm --gname Group --cclientid realm-management
```

3. Use the `--effective` option to list effective realm roles.

使用— effect 选项列出有效的领域角色。

For example:

例如:

```
$ kcadm.sh get-roles -r demorealm --gname Group --cclientid realm-management --effective
```

4. Use the `--available` option to list realm roles that you can still add to the group.

使用— able 选项列出仍然可以添加到组中的领域角色。

For example:

例如:

```
$ kcadm.sh get-roles -r demorealm --gname Group --cclientid realm-management --available
```

Identity provider operations 身份提供程序操作

Listing available identity providers

列出可用的身份提供程序

Use the `serverinfo` endpoint to list available identity providers.

使用 serverinfo 端点列出可用的标识提供程序。

For example:

例如:

```
$ kcadm.sh get serverinfo -r demorealm --fields 'identityProviders(*)'
```

 Keycloak processes the `serverinfo` endpoint similarly to the `realms` endpoint. Keycloak does not resolve the endpoint relative to a target realm because it exists outside any specific realm.

Keycloak 处理 serverinfo 端点的方式类似于领域端点。钥匙斗篷不解析相对于目标领域的端点，因为它存在于任何特定领域之外。

Listing configured identity providers

列出已配置的标识提供程序

Use the `identity-provider/instances` endpoint.

使用标识提供程序/实例端点。

For example:

例如:

```
$ kcadm.sh get identity-provider/instances -r demorealm --fields alias,providerId,enabled
```

Getting a specific configured identity provider

获取特定配置的标识提供程序

Use the identity provider's `alias` attribute to construct an endpoint URI, such as `identity-provider/instances/ALIAS`, to get a specific identity provider.

使用标识提供程序的 ALIAS 属性构造端点 URI，如标识提供程序/实例/ALIAS，以获取特定的标识提供程序。

For example:

例如:

```
$ kcadm.sh get identity-provider/instances/facebook -r demorealm
```

Removing a specific configured identity provider

删除特定配置的标识提供程序

Use the `delete` command with the same endpoint URI that you use to get a specific configured identity provider to remove a specific configured identity provider.

使用具有与获取特定配置的标识提供程序相同的端点 URI 的 `delete` 命令来删除特定配置的标识提供程序。

For example:

例如:

```
$ kcadm.sh delete identity-provider/instances/facebook -r demorealm
```

Configuring a Keycloak OpenID Connect identity provider

配置密钥斗篷 OpenID 连接标识提供程序

1. Use `keycloak-oidc` as the `providerId` when you create a new identity provider instance.

在创建新的标识提供程序实例时，使用 `keycloak-oidc` 作为 `ProviderId`。

2. Provide the `config` attributes: `authorizationUrl`, `tokenUrl`, `clientId`, and `clientSecret`.

提供配置属性: `authorizationUrl`、`tokenUrl`、`clientId` 和 `clientSecret`。

For example:

例如:

```
$ kcadm.sh create identity-provider/instances -r demorealm -s alias=keycloak-oidc -s provide
```

Configuring an OpenID Connect identity provider

配置 OpenID 连接标识提供程序

Configure the generic OpenID Connect provider the same way you configure the Keycloak OpenID Connect provider, except you set the `providerId` attribute value to `oidc`.

配置通用 OpenID Connect 提供程序的方式与配置 Key 斗篷 OpenID Connect 提供程序的方式相同，只不过提供程序 Id 属性值设置为 `oidc`。

Configuring a SAML 2 identity provider

配置 SAML2 标识提供程序

1. Use `saml` as the `providerId`.

使用 `saml` 作为 `ProviderId`。

2. Provide the `config` attributes: `singleSignOnServiceUrl`, `nameIDPolicyFormat`, and `signatureAlgorithm`.

提供配置属性: `singleSignOnServiceUrl`、`nameIDPolicyFormat` 和 `signatureAlgorithm`。

For example:

例如:

```
$ kcadm.sh create identity-provider/instances -r demorealm -s alias=saml -s providerId=saml -s e
```

Configuring a Facebook identity provider

配置 Facebook 身份提供程序

1. Use `facebook` as the `providerId`.

使用 `facebook` 作为提供者 ID。

2. Provide the `config` attributes: `clientId` and `clientSecret`. You can find these attributes in the Facebook Developers application configuration page for your application. See the Facebook identity broker page for more information.

提供配置属性: `clientId` 和 `clientSecret`。您可以在您的应用程序的 Facebook Developers 应用程序配置页面中找到这些属性。有关更多信息，请参见 Facebook 身份代理页面。

For example:

例如:

```
$ kcadm.sh create identity-provider/instances -r demorealm -s alias=facebook -s providerId=f
```

Configuring a Google identity provider

配置 Google 标识提供程序

1. Use `google` as the `providerId`.

使用 `google` 作为 ProviderId。

2. Provide the `config` attributes: `clientId` and `clientSecret`. You can find these attributes in the Google Developers application configuration page for your application. See the Google identity broker page for more information.

提供配置属性: `clientId` 和 `clientSecret`。您可以在您的应用程序的 Google Developers 应用程序配置页面中找到这些属性。有关更多信息，请参见谷歌身份代理页面。

For example:

例如:

```
$ kcadm.sh create identity-provider/instances -r demorealm -s alias=google -s providerId=goo
```

Configuring a Twitter identity provider

配置 Twitter 身份提供程序

1. Use `twitter` as the `providerId`.

使用 `twitter` 作为 `ProviderId`。

2. Provide the `config` attributes `clientId` and `clientSecret`. You can find these attributes in the Twitter Application Management application configuration page for your application. See the Twitter identity broker page for more information.

提供配置属性 `clientId` 和 `clientSecret`。您可以在应用程序的 Twitter Application Management 应用程序配置页面中找到这些属性。有关更多信息，请参见 Twitter 身份代理页面。

For example:

例如:

```
$ kcadm.sh create identity-provider/instances -r demorealm -s alias=google -s providerId=goo
```

Configuring a GitHub identity provider

配置 GitHub 标识提供程序

1. Use `github` as the `providerId`.

使用 `github` 作为 `ProviderId`。

2. Provide the `config` attributes `clientId` and `clientSecret`. You can find these attributes in the GitHub Developer Application Settings page for your application. See the GitHub identity broker page for more information.

提供配置属性 `clientId` 和 `clientSecret`。您可以在应用程序的 GitHub Developer Application Settings 页面中找到这些属性。有关更多信息，请参见 GitHub 身份代理页面。

For example:

例如:

```
$ kcadm.sh create identity-provider/instances -r demorealm -s alias=github -s providerId=git
```

Configuring a LinkedIn identity provider

配置 LinkedIn 标识提供程序

1. Use `linkedin` as the `providerId`.

使用 linkedin 作为 ProviderId。

2. Provide the config attributes `clientId` and `clientSecret`. You can find these attributes in the LinkedIn Developer Console application page for your application. See the LinkedIn identity broker page for more information.

提供配置属性 `clientId` 和 `clientSecret`。你可以在你的应用程序的 LinkedIn 开发者控制台应用中找到这些属性。有关更多信息，请参见 LinkedIn 身份代理页面。

For example:

例如:

```
$ kcadm.sh create identity-provider/instances -r demorealm -s alias=linkedin -s providerId=
```

Configuring a Microsoft Live identity provider

配置 MicrosoftLive 标识提供程序

1. Use `microsoft` as the `providerId`.

使用微软作为 ProviderId。

2. Provide the config attributes `clientId` and `clientSecret`. You can find these attributes in the Microsoft Application Registration Portal page for your application. See the Microsoft identity broker page for more information.

提供配置属性 `clientId` 和 `clientSecret`。您可以在应用程序的 Microsoft 应用程序注册门户页面中找到这些属性。有关更多信息，请参见 Microsoft 标识代理页。

For example:

例如:

```
$ kcadm.sh create identity-provider/instances -r demorealm -s alias=microsoft -s providerId=
```

Configuring a Stack Overflow identity provider

配置堆栈溢出标识提供程序

1. Use `stackoverflow` command as the `providerId`.

使用 `stackoverflow` 命令作为 ProviderId。

2. Provide the config attributes `clientId`, `clientSecret`, and `key`. You can find these attributes in the Stack Apps OAuth page for your application. See the Stack Overflow identity broker page for more information.

提供配置属性 `clientId`、`clientSecret` 和 `key`。您可以在应用程序的 StackAppsOAuth 页面中找到这些属性。有关更多信息，请参见 Stack Overflow 标识代理页。

For example:

例如:

```
$ kcadm.sh create identity-provider/instances -r demorealm -s alias=stackoverflow -s provide
```

Storage provider operations 存储提供程序操作

Configuring a Kerberos storage provider

配置 Kerberos 存储提供程序

1. Use the `create` command against the `components` endpoint.

对组件端点使用 `create` 命令。

2. Specify the realm id as a value of the `parentId` attribute.

将领域 id 指定为 `ParentId` 属性的值。

3. Specify `kerberos` as the value of the `providerId` attribute, and
`org.keycloak.storage.UserStorageProvider` as the value of the `providerType` attribute.

指定 `kerberos` 作为 `ProviderId` 属性的值，指定 `org.keycloak.Storage. UserStorageProvider` 作为 `ProviderType` 属性的值。

4. For example:

例如:

```
$ kcadm.sh create components -r demorealm -s parentId=demorealmId -s id=demokerberos -s name
```

Configuring an LDAP user storage provider

配置 LDAP 用户存储提供程序

1. Use the `create` command against the `components` endpoint.

对组件端点使用 `create` 命令。

2. Specify `ldap` as the value of the `providerId` attribute, and

`org.keycloak.storage.UserStorageProvider` as the value of the `providerType` attribute.

指定 `ldap` 作为 `ProviderId` 属性的值，指定 `org.keycloak.store.UserStorageProvider` 作为 `ProviderType` 属性的值。

3. Provide the realm ID as the value of the `parentId` attribute.

提供领域 ID 作为 `ParentId` 属性的值。

4. Use the following example to create a Kerberos-integrated LDAP provider.

使用以下示例创建 Kerberos 集成的 LDAP 提供程序。

```
$ kcadm.sh create components -r demorealm -s name=kerberos-ldap-provider -s providerId=ldap
```

Removing a user storage provider instance

移除用户存储提供程序实例

1. Use the storage provider instance's `id` attribute to compose an endpoint URI, such as `components/ID`.

使用存储提供程序实例的 ID 属性组成端点 URI，例如组件/ID。

2. Run the `delete` command against this endpoint.

对此端点运行 delete 命令。

For example:

例如:

```
$ kcadm.sh delete components/3d9c572b-8f33-483f-98a6-8bb421667867 -r demorealm
```

Triggering synchronization of all users for a specific user storage provider

触发特定用户存储提供程序的所有用户的同步

1. Use the storage provider's `id` attribute to compose an endpoint URI, such as `user-storage/ID_OF_USER_STORAGE_INSTANCE/sync`.

使用存储提供程序的 ID 属性组成端点 URI，例如 user-Storage/ID_OF_USER_STORAGE_INSTANCE/sync。

2. Add the `action=triggerFullSync` query parameter.

添加 action = triggerFullSync 查询参数。

3. Run the `create` command.

运行 create 命令。

For example:

例如:

```
$ kcadm.sh create user-storage/b7c63d02-b62a-4fc1-977c-947d6a09e1ea/sync?action=triggerFullSync
```

Triggering synchronization of changed users for a specific user storage provider

为特定用户存储提供程序触发已更改用户的同步

1. Use the storage provider's `id` attribute to compose an endpoint URI, such as `user-storage/ID_OF_USER_STORAGE_INSTANCE/sync`.

使用存储提供程序的 ID 属性组成端点 URI，例如 user-Storage/ID_OF_USER_STORAGE_INSTANCE-sync。

2. Add the `action=triggerChangedUsersSync` query parameter.

添加 `action = triggerChangedUsersSync` 查询参数。

3. Run the `create` command.

运行 `create` 命令。

For example:

例如:

```
$ kcadm.sh create user-storage/b7c63d02-b62a-4fc1-977c-947d6a09e1ea/sync?action=triggerChang
```

Test LDAP user storage connectivity

测试 LDAP 用户存储连通性

1. Run the `get` command on the `testLDAPConnection` endpoint.

在 `testLDAPConnection` 端点上运行 `get` 命令。

2. Provide query parameters `bindCredential`, `bindDn`, `connectionUrl`, and `useTruststoreSpi`.

提供查询参数 `bindCredential`、`bindDn`、`ConnectionUrl` 和 `useTruststoreSpi`。

3. Set the `action` query parameter to `testConnection`.

将操作查询参数设置为 `testConnection`。

For example:

例如:

```
$ kcadm.sh create testLDAPConnection -s action=testConnection -s bindCredential=secret -s bi
```

Test LDAP user storage authentication

测试 LDAP 用户存储身份验证

1. Run the `get` command on the `testLDAPConnection` endpoint.

在 `testLDAPConnection` 端点上运行 `get` 命令。

2. Provide the query parameters `bindCredential`, `bindDn`, `connectionUrl`, and `useTruststoreSpi`.

提供查询参数 `bindCredential`、`bindDn`、`ConnectionUrl` 和 `useTruststoreSpi`。

3. Set the `action` query parameter to `testAuthentication`.

将操作查询参数设置为 testAuthentication。

For example:

例如:

```
$ kcadm.sh create testLDAPConnection -s action=testAuthentication -s bindCredential=secret -
```

Adding mappers 添加映射器

Adding a hard-coded role LDAP mapper

添加硬编码的角色 LDAP 映射器

1. Run the `create` command on the `components` endpoint.

在组件端点上运行 create 命令。

2. Set the `providerType` attribute to

`org.keycloak.storage.ldap.mappers.LDAPStorageMapper`.

将 provisionerType 属性设置为 `org.keycloak.storage.ldap.mappers.LDAPStorageMapper`.

3. Set the `parentId` attribute to the ID of the LDAP provider instance.

将 ParentId 属性设置为 LDAP 提供程序实例的 ID。

4. Set the `providerId` attribute to `hardcoded-ldap-role-mapper`. Ensure you provide a value of `role` configuration parameter.

将 provisionerId 属性设置为 `hardcode-ldap-role-mapper`.

For example:

例如:

```
$ kcadm.sh create components -r demorealm -s name=hardcoded-ldap-role-mapper -s providerId=h
```

Adding an MS Active Directory mapper

添加 MSActiveDirectory 映射器

1. Run the `create` command on the `components` endpoint.

在组件端点上运行 create 命令。

2. Set the `providerType` attribute to

`org.keycloak.storage.ldap.mappers.LDAPStorageMapper`.

将 provisionerType 属性设置为 `org.keycloak.storage.ldap.mappers.LDAPStorageMapper`.

3. Set the `parentId` attribute to the ID of the LDAP provider instance.

将 ParentId 属性设置为 LDAP 提供程序实例的 ID。

4. Set the `providerId` attribute to `msad-user-account-control-mapper`.

将 provisionerId 属性设置为 msad-user-account-control-mapper。

For example:

例如:

```
$ kcadm.sh create components -r demorealm -s name=msad-user-account-control-mapper -s provid
```

Adding a user attribute LDAP mapper

添加用户属性 LDAP 映射器

1. Run the `create` command on the `components` endpoint.

在组件端点上运行 create 命令。

2. Set the `providerType` attribute to

```
org.keycloak.storage.ldap.mappers.LDAPStorageMapper .
```

将 provisionerType 属性设置为 org.keycloak.storage.ldap.mappers.LDAPStorageMapper。

3. Set the `parentId` attribute to the ID of the LDAP provider instance.

将 ParentId 属性设置为 LDAP 提供程序实例的 ID。

4. Set the `providerId` attribute to `user-attribute-ldap-mapper`.

将 provisionerId 属性设置为 user-tribute-ldap-mapper。

For example:

例如:

```
$ kcadm.sh create components -r demorealm -s name=user-attribute-ldap-mapper -s providerId=u
```

Adding a group LDAP mapper

添加组 LDAP 映射器

1. Run the `create` command on the `components` endpoint.

在组件端点上运行 create 命令。

2. Set the `providerType` attribute to

```
org.keycloak.storage.ldap.mappers.LDAPStorageMapper .
```

将 provisionerType 属性设置为 org.keycloak.storage.ldap.mappers.LDAPStorageMapper。

3. Set the `parentId` attribute to the ID of the LDAP provider instance.

将 ParentId 属性设置为 LDAP 提供程序实例的 ID。

4. Set the `providerId` attribute to `group-ldap-mapper`.

将 ProviderId 属性设置为 group-ldap-mapper。

For example:

例如:

```
$ kcadm.sh create components -r demorealm -s name=group-ldap-mapper -s providerId=group-ldap
```

Adding a full name LDAP mapper

添加完整名称的 LDAP 映射器

1. Run the `create` command on the `components` endpoint.

在组件端点上运行 create 命令。

2. Set the `providerType` attribute to

```
org.keycloak.storage.ldap.mappers.LDAPStorageMapper .
```

将 provisionerType 属性设置为 org.keycloak.storage.ldap.mappers.LDAPStorageMapper。

3. Set the `parentId` attribute to the ID of the LDAP provider instance.

将 ParentId 属性设置为 LDAP 提供程序实例的 ID。

4. Set the `providerId` attribute to `full-name-ldap-mapper`.

将 ProviderId 属性设置为 full-name-ldap-mapper。

For example:

例如:

```
$ kcadm.sh create components -r demorealm -s name=full-name-ldap-mapper -s providerId=full-n
```

Authentication operations 认证操作

Setting a password policy

设置密码策略

1. Set the realm's `passwordPolicy` attribute to an enumeration expression that includes the specific policy provider ID and optional configuration.

将领域的 passwordPolicy 属性设置为包含特定策略提供程序 ID 和可选配置的枚举表达式。

2. Use the following example to set a password policy to default values. The default values include:

使用下面的示例将密码策略设置为默认值。默认值包括:

- 27,500 hashing iterations
27500个哈希迭代
- at least one special character
至少有一个特殊的字符
- at least one uppercase character
至少一个大写字母
- at least one digit character
至少一个数字字符
- not be equal to a user's username
不等于用户的用户名
- be at least eight characters long
至少有八个字符长

```
$ kcadm.sh update realms/demorealm -s 'passwordPolicy="hashIterations and specialChars and uppercase and lowercase and digits and length"'
```



3. To use values different from defaults, pass the configuration in brackets.

若要使用不同于默认值的值，请在括号中传递配置。

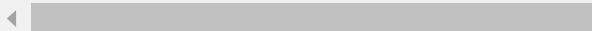
4. Use the following example to set a password policy to:

使用以下示例将密码策略设置为：

- 25,000 hash iterations
25,000个散列迭代
- at least two special characters
至少两个特殊字符
- at least two uppercase characters
至少两个大写字母
- at least two lowercase characters
至少两个小写字母
- at least two digits
至少两位数
- be at least nine characters long
至少有九个字符长

- not be equal to a user's `username`
不等于用户的用户名
- not repeat for at least four changes back
不要重复至少四次

```
$ kcadm.sh update realms/demorealm -s 'passwordPolicy="hashIterations(25000) and specialCh...
```



Obtaining the current password policy

获取当前密码策略

You can get the current realm configuration by filtering all output except for the `passwordPolicy` attribute.

您可以通过过滤除 `passwordPolicy` 属性之外的所有输出来获得当前领域配置。

For example, display `passwordPolicy` for `demorealm`.

例如，显示 `derealm` 的 `passwordPolicy`。

```
$ kcadm.sh get realms/demorealm --fields passwordPolicy
```

Listing authentication flows

列出身份验证流

Run the `get` command on the `authentication/flows` endpoint.

在身份验证/流端点上运行 `get` 命令。

For example:

例如：

```
$ kcadm.sh get authentication/flows -r demorealm
```

Getting a specific authentication flow

获取特定的身份验证流

Run the `get` command on the `authentication/flows/FLOW_ID` endpoint.

在身份验证/流/`FLOW_ID` 端点上运行 `get` 命令。

For example:

例如:

```
$ kcadm.sh get authentication/flows/febfd772-e1a1-42fb-b8ae-00c0566fafb8 -r demorealm
```

Listing executions for a flow

列出流的执行

Run the `get` command on the `authentication/flows/FLOW_ALIAS/executions` endpoint.

在身份验证/FLOW/FLOW_ALIAS/执行端点上运行 `get` 命令。

For example:

例如:

```
$ kcadm.sh get authentication/flows/Copy%20of%20browser/executions -r demorealm
```

Adding configuration to an execution

向执行添加配置

1. Get execution for a flow.

执行一个流程。

2. Note the ID of the flow.

请注意流的 ID。

3. Run the `create` command on the `authentication/executions/{executionId}/config` endpoint.

在身份验证/执行/{ ExecutionId }/配置端点上运行 `create` 命令。

For example:

例如:

```
$ kcadm create "authentication/executions/a3147129-c402-4760-86d9-3f2345e401c7/config" -r exemplar
```

Getting configuration for an execution

获取执行的配置

1. Get execution for a flow.

执行一个流程。

2. Note its `authenticationConfig` attribute, which contains the config ID.

请注意其 `enticationConfig` 属性，该属性包含配置 ID。

3. Run the `get` command on the `authentication/config/ID` endpoint.

在身份验证/config/ID 端点上运行 get 命令。

For example:

例如:

```
$ kcadm get "authentication/config/dd91611a-d25c-421a-87e2-227c18421833" -r exemplrealm
```

Updating configuration for an execution

更新执行的配置

1. Get the execution for the flow.

获取流的执行。

2. Get the flow's `authenticationConfig` attribute.

获取流的 `enticationConfig` 属性。

3. Note the config ID from the attribute.

注意属性中的配置 ID。

4. Run the `update` command on the `authentication/config/ID` endpoint.

在身份验证/config/ID 端点上运行 update 命令。

For example:

例如:

```
$ kcadm update "authentication/config/dd91611a-d25c-421a-87e2-227c18421833" -r exemplrealm -b '
```

Deleting configuration for an execution

正在删除执行的配置

1. Get execution for a flow.

执行一个流程。

2. Get the flows `authenticationConfig` attribute.

获取 flow `enticationConfig` 属性。

3. Note the config ID from the attribute.

注意属性中的配置 ID。

4. Run the `delete` command on the `authentication/config/ID` endpoint.

在身份验证/配置/ID 端点上运行 delete 命令。

For example:

例如:

```
$ kcadm delete "authentication/config/dd91611a-d25c-421a-87e2-227c18421833" -r exemplrealm
```