Red Hat build of Keycloak 24.0

升级指南

Last Updated: 2024-06-29

法律通告

摘要

本书是将红帽构建的 Keycloak 从 22.0.x 升级到 24.0.5。

目录

第 1 章 升级红帽构建的 KEYCLOAK

本指南论述了如何将红帽构建的 Keycloak 从版本 22.0.x 升级到 24.0.5。按照以下顺序使用以下流程：

1. 查看来自红帽构建的 Keycloak 版本的迁移更改。
2. 升级红帽构建的 Keycloak 服务器。
3. 升级红帽构建的 Keycloak 适配器。
4. 升级红帽构建的 Keycloak Admin 客户端。

对于 Red Hat Single Sign-On 7.6 客户，首先使用 迁移指南 升级到 Red Hat build of Keycloak 22.0。然后您可以使用此升级指南。

第 2 章 迁移更改

2.1. 用户配置文件更改

2.1.1. 默认启用用户配置集

用户配置集功能现在默认启用。**declarative-user-profile** 功能不再可用，因为假定启用了用户配置文件。因此，**User Profile Enabled** 交换机已从 **Realm Settings** 中删除，并由 **Unmanaged 属性** 替代。从之前的版本迁移时，行为如下：

- 对于用户配置文件 **已启用** 为 **ON** 的部署，在升级后，**Unmanaged 属性** 将设置为 **OFF**。因此，只允许用户配置集明确支持的用户属性。
- 对于用户配置文件 **Enabled** 为 **OFF** 的部署（这也是禁用 **declarative-user-profile** 功能的部署的默认设置，这是默认设置），在升级后，**Unmanaged 属性** 将设置为 **ON**。因此，行为应该与之前版本相同，并禁用了用户配置文件。**Attributes** 选项卡将保留在管理控制台的用户详细信息部分中。另外，只要特定的自定义主题支持，用户现在可以通过 UI 页面设置任意属性，如注册页面和更新配置文件页面。自定义主题也应在之前正常工作。但是，请考虑更新您的主题以使用 user-profile，并在需要添加自定义属性时删除您的自定义主题。请参阅主题上的后续部分。另外，请考虑将 **Unmanaged 属性** 切换为 **OFF**，或只为管理员启用这个开关，以便您可以主要依赖使用受管属性。

有关 **Unmanaged 属性的详情，请查看 用户配置文件文档**。

2.1.2. 默认验证

默认用户配置文件配置包括一组用于基本预定义字段的默认验证。默认情况下，当 **declarative-user-profile** 功能被禁用时，这些验证不会出现在之前的版本中。如果您因为向后兼容时遇到问题，您可以根据您的需要更改默认验证器。默认验证器如下：

- **用户名**、**电子邮件**、**firstName** 和 **lastName** 属性的最大长度为 255 个字符。这些验证在之前的版本中也间接存在，因为表 **USER_ENTITY** 上的数据库约束对于这些字段的最大长度为 255 个字符。但是，在使用用户存储提供程序时，可能会使用较长的值。
- **username** 属性的最短长度为三个字符。用户名默认具有 **username-prohibited-characters** 和 **up-username-not-idn-homograph** 验证器，这些验证器在之前的版本中不存在。有关这些属性的详细信息，请参阅用户配置文件文档中的 Validation 部分。请注意，除非您 **启用了 realm 切换 Edit username**，否则默认不可编辑用户名。这个更改意味着具有不正确的用户名的现有用户仍应该可以正常工作，且不会强制更新其用户名。但是，将强制新用户在注册或由 admin REST API 创建期间使用正确的用户名。
- **firstName** 和 **lastName** 属性具有 **person-name-prohibited-characters** 验证器，它们没有存在于之前的版本中。有关这些属性的详细信息，请参阅用户配置文件文档中的 Validation 部分。请注意，第一个名称和姓氏默认为可编辑，因此已在更新其用户配置集时，已有之前版本中错误的名/最后名名称的用户会被强制进行更新。

2.1.3. 用户属性名称，带有 strange 字符

在以前的版本中，您可以使用属性名称（如 **some:attribute** 或 **some/attribute** ）创建用户。用户配置文件特意不允许您在用户配置文件配置中创建具有此类奇数名称的属性。因此，您可能需要为域配置非受管 **属性**，并为管理员（最好）或最终用户启用非受管属性（如果需要）。虽然强烈建议避免使用这样的属性名称。

2.1.4. 验证默认启用 Profile 所需的操作

在新域中默认启用 **verify-profile** 所需的操作。但是，当您从现有版本迁移时，您的现有域将具有与之前相同的 **verify-profile** 操作状态，这通常意味着在以前的版本中默认禁用。有关此所需操作的详情，请查看 用户配置文件 文档。

2.1.5. 对 Freemarker 模板的更改，以根据用户配置集和域呈现页面

在这个发行版本中，以下模板已更新，以便可以根据用户配置集配置设置为 realm 来动态呈现属性：

- **login-update-profile.ftl**
- **register.ftl**
- **update-email.ftl**

这些模板负责渲染更新配置文件（当用户启用 **Update Profile** 所需的操作时）、注册以及更新电子邮件（当 **UPDATE_EMAIL** 功能被启用）页面。

如果您使用自定义主题更改这些模板，它们可以正常工作，因为只更新内容。但是，我们建议您了解如何配置 声明性用户配置文件，并可能避免使用此功能提供的所有功能来更改内置模板。

另外，在此发行版本中还需要删除 **declarative-user-profile** 功能用来呈现相同流的页面：

- **update-user-profile.ftl**
- **register-user-profile.ftl**

如果您在上一发行版本中使用 **declarative-user-profile** 功能，并自定义到上述模板，请相应地更新 **login-update-profile.ftl**，并相应地更新 **register.ftl**。

2.1.6. 首次通过代理登录时为更新配置集页面新的 Freemarker 模板

在本发行版本中，当用户首次使用 **idp-review-user-profile.ftl** 模板进行身份验证时，服务器会呈现更新配置集页面。

在以前的版本中，用于在第一个代理登录流期间更新配置集的模板是 **login-update-profile.ftl**，在用户向域进行身份验证时用于更新配置集。

通过为每个流使用单独的模板，存在更加明确的区别，因为实际使用模板而不是共享同一模板，并可能引入意外更改和行为，它们应该只影响特定流的页面。

如果您对 **login-update-profile.ftl** 模板进行了自定义，以便用户在通过代理进行身份验证时更新其配置文件，请确保将您的更改移到新模板。

2.2. 对主题的更改

2.2.1. 对 Welcome theme 的更改

"welcome"主题已更新为使用新布局，现在使用 PatternFly 5 而不是 PatternFly 3。如果您要扩展主题或自行提供，您可能需要按如下方式进行更新：

2.2.1.1. 从 PatternFly 3 迁移到 PatternFly 5

欢迎主题是红帽构建的 Keycloak 中的过期主题之一。它最初基于 PatternFly 3，但现已更新为使用 PatternFly 5，跳过过程中的主要版本。如果您的自定义主题扩展了内置主题，您需要对其进行更新以使用 PatternFly 5 语法。有关详细信息，请参阅 PatternFly 5 文档。

如果您仍然在您自己的自定义主题中使用 PatternFly 3 （没有扩展内置主题），您可以继续使用它，但 PatternFly 3 支持将在以后的版本中删除，因此您应该尽快迁移到 PatternFly 5。

2.2.1.2. 自动重定向到管理控制台

如果启用了 Admin 控制台，则欢迎页面将自动重定向到它（如果管理用户已存在）。通过在 **主题.properties** 文件中设置 **redirectToAdmin** 来修改此行为。默认情况下，属性设为 **false**，除非您扩展内置主题，在这种情况下，属性设为 **true**。

2.2.1.3. **documentationUrl** 和 **displayCommunityLinks** 属性已被删除。

这些属性以前用于现在不再存在的导航元素。如果要扩展内置主题，则需要从me.properties **文件中删除这些属性**，因为它们不再起作用。

2.2.1.4. 现在， 资产已从 'common' 资源加载

背景、徽标和 favicon 等镜像现在从 'common' 资源加载，而不是从主题资源加载。此更改意味着，如果您要扩展内置主题，并且覆盖这些镜像，则需要将它们移到主题的"通用"资源，并更新您的 **主题.properties** 文件使其包含新路径：

```
# This defaults to 'common/keycloak' if not set.
import=common/your-theme-name
```

2.2.2. 对帐户控制台主题自定义的更改

如果您之前扩展帐户控制台主题的已弃用版本 2，则需要更新您的主题以使用帐户控制台主题的新版本 3。新版本的帐户控制台主题会做一些关于它自定义方式的更改。要开始清理，您可以遵循新的自定义 快速入门。

要移动您的自定义主题，首先将 **父项** 改为新的主题：

```
# Before
parent=keycloak.v2

# After
parent=keycloak.v3
```

如果您有任何自定义 React 组件，您将直接导入 React，而不是使用相对路径：

```
// Before
import * as React from "../../../../common/keycloak/web_modules/react.js";

// After
import React from "react";
```

如果您使用 **content.json** 自定义主题，则对文件结构有一些更改，特别是：

- **content** 属性已重命名为 **children**。
- **id**、**icon** 和 **componentName** 属性已被删除，因为 **modulePath** 提供了相同的功能。

2.2.3. 用于它们的默认语言文件到 UTF-8 编码

此发行版本现在遵循 Java 及之后的版本的标准机制，它假定要在 UTF-8 中编码资源捆绑包文件。

之前的 Keycloak 版本支持在第一行中指定编码，它带有类似 **# encoding: UTF-8** 的注释，该注释不再被支持并会被忽略。

现在，sees 的消息属性文件在 UTF-8 编码中读取，自动回退到 ISO-8859-1 编码。如果您使用不同的编码，请将文件转换为 UTF-8。

2.3. OPERATOR 更改

2.3.1. Operator 引用的资源轮询

现在，通过 Keycloak CR 引用的 secret 和 ConfigMap 会被轮询以进行更改，而不是被 API 服务器监视。此轮询可能需要一分钟后才能检测到更改。

这是为了避免在这些资源上进行标签操作。在升级任何 Secret 仍然具有 operator.keycloak.org/component 标签后，可能会删除或忽略它。

2.3.2. Operator 自定义属性密钥

Operator 用于高级配置的属性键已从 **operator.keycloak** 改为 **kc.operator.keycloak**。

2.3.3. operator -secrets-store Secret

旧版本的 Operator 创建了一个 Secret，用于跟踪监视的 Secret。Operator 的较新版本不再使用 -secrets-store Secret，因此可以删除它。

2.4. KEYCLOAK CR 更改

2.4.1. Keycloak CR 资源选项

当在 Keycloak CR 和 KeycloakRealmImport CR 中没有指定资源选项时，会使用默认值。**Keycloak 部署的默认 请求内存 和域导入作业设置为 1700MiB，限值 内存设置为 2GiB。**

2.4.2. 默认 Keycloak CR 主机名

在 OpenShift 上运行时，启用了 ingress，并将 spec.ingress.classname 设置为 openshift-default，您可能会在 Keycloak CR 中保留 spec.hostname.hostname。Operator 将为 CR 的存储版本分配一个默认主机名，类似于没有显式主机的 OpenShift Route 创建的内容 - 为 **ingress-namespace.appsDomain 如果 appsDomain 更改，或者您需要不同的主机名来更新 Keycloak CR。**

2.5. 功能变化

在 --features 和 --features-disabled 列表中不再允许相同的功能。该功能应仅出现在一个列表中。

在 --features 列表中使用未指定版本的功能名称（如 docker ）将允许您启用最受支持的/最新功能版

本。如果您在版本之间需要更多的行为，请引用您想要的特定版本，如 docker:v1。

2.6. 其它更改

2.6.1. 在 Admin API 和帐户上下文中更改用户表示

org.keycloak.representations.idm.UserRepresentation s.keycloak.representations.account.UserRepresentation representation 类都已更改，以便 root 用户属性（如用户名、电子邮件、firstName、lastName 和 locale）在获取或将表示有效负载发送到 Admin 和 Account API 时具有一致的表示。

用户名、mail、firstName、lastName 和 locale 属性被移到一个新的 org.keycloak.representations.idm.AbstractUserRepresentation 基础类。

另外，getAttributes 方法只代表自定义属性，因此您不应该期望此方法返回的映射中的任何 root 属性。当为给定用户更新或获取任何自定义属性时，此方法主要针对客户端。

为了解析包括 root 属性的所有属性，添加了一个新的 getRawAttributes 方法，以便生成的映射也包含 root 属性。但是，此方法不能从表示有效负载中获得，它的目标是在管理用户配置文件时供服务器使用。

2.6.2. https-client-auth 是构建时间选项

选项 https-client-auth 已被视为运行时选项，但 Quarkus 不支持它。选项需要改为在构建时进行处理。

2.6.3. 后续载入离线会话和远程会话

从这个版本开始，红帽构建的 Keycloak 集群的第一个成员将按顺序加载远程会话，而不是并行加载。如果启用了离线会话预加载，它们也会按顺序加载。

之前的代码在启动时导致集群中资源消耗高，在生产环境中分析非常困难，并可能导致在加载过程中重启节点时出现复杂的故障场景。因此，它被改为后续会话加载。

对于离线会话，这个和之前版本的 Keycloak 默认是根据需要加载这些会话，与尝试并行加载它们相比，这个会话可以更好地扩展。使用此默认设置的设置不受加载策略更改离线会话的影响。启用离线会话预加载的设置应该迁移到禁用 offline-session 预加载的设置。

2.6.4. Infinispan 指标使用标签进行缓存管理器和缓存名称

当为 Keycloak 的嵌入式缓存启用指标时，指标现在使用缓存管理器的标签和缓存名称。

没有标签的旧指标示例

```
vendor_cache_manager_keycloak_cache_sessions_statistics_approximate_entries_in_memory{cache="sessions",node="..."}
```

带有标签的新指标示例

```
vendor_statistics_approximate_entries_in_memory{cache="sessions",cache_manager="keycloak",node="..."}
```

要恢复安装的更改，请使用自定义 Infinispan XML 配置并更改配置，如下所示：

```
<metrics names-as-tags="false" />
```

2.6.5. 用户属性值长度扩展

在这个版本中，Red Hat build of Keycloak 支持按用户属性值的存储和搜索，其比 255 个字符长，这之前是一个限制。

在允许用户更新属性（例如通过帐户控制台）设置中，请通过添加验证来防止拒绝服务攻击。确保不允许非受管属性，并且所有可编辑的属性都有限制输入长度的验证。

对于非受管属性，最大长度为 2048 个字符。对于受管属性，默认最大长度为 2048 个字符。管理员可以通过添加类型为 length 的验证器来更改。

警告

红帽构建的 Keycloak 会缓存其内部缓存中与用户相关的对象。使用较长的属性会增加缓存消耗的内存。因此，建议限制 length 属性的大小。考虑在红帽构建的 Keycloak 外部存储大型对象，并通过 ID 或 URL 引用它们。

此更改在表 USER_ATTRIBUTE 和 FED_USER_ATTRIBUTE 上添加新的索引。如果这些表包含 300000 项，Red Hat build of Keycloak 会在自动模式迁移过程中默认跳过索引创建，而是在迁移过程中在控制台上记录 SQL 语句，以便在红帽构建 Keycloak 的启动后手动应用。有关如何配置不同限制的详情，请参阅 升级指南。

2.6.5.1. LDAP 的其他迁移步骤

这适用于与以下所有条件匹配的安装：

- LDAP 目录中的用户属性大于 2048 个字符或大于 1500 字节的二进制属性。
- 这些属性由管理员或用户通过管理控制台、API 或帐户控制台更改。

要通过 UI 和 REST API 启用更改这些属性，请执行以下步骤：

1. 在域的用户配置文件中，声明上面识别的属性。
2. 为上一步中添加的每个属性定义一个 长度 验证器，指定属性值所需的最小和最大长度。对于二进制值，将 33% 添加到预期的二进制长度中，以计算红帽构建的 Keycloak 内部 base64 编码的开销。

2.6.5.2. 自定义用户存储供应商的其他迁移步骤

这适用于与以下所有条件匹配的安装：

- 将 MariaDB 或 MySQL 作为红帽构建的 Keycloak 的数据库运行。

- 表 FED_USER_ATTRIBUTE 中的条目，其内容在大于 2048 个字符的 VALUE 列中的内容。此表用于启用了联邦的自定义用户供应商。

- 管理员或用户通过管理控制台或帐户控制台更改长属性。

要通过 UI 和 REST API 启用更改这些属性，请执行以下步骤：

1. 在域的用户配置文件中，声明上面识别的属性。

2. 为上一步中添加的每个属性定义一个 长度 验证器，指定属性值所需的最小和最大长度。

2.6.6. Admin send-verify-email API 现在使用相同的电子邮件验证模板

```
PUT /admin/realms/{realm}/users/{id}/send-verify-email
```

在本发行版本中，API 将使用 email-verification.ftl 模板，而不是 executeActions.ftl。

在升级前

```
Perform the following action(s): Verify Email
```

升级后

```
Confirm validity of e-mail address email@example.org.
```

如果您自定义了 executeActions.ftl 模板，以修改用户如何使用此 API 验证其电子邮件，请确保将修改转移到新模板。

将引入一个名为 lifespan 的新参数，以允许覆盖默认的寿命值(12 小时)。

如果您更喜欢之前的行为，请使用 execute-actions-email API，如下所示：

```
PUT /admin/realms/{realm}/users/{id}/execute-actions-email

["VERIFY_EMAIL"]
```

2.6.7. 更改密码哈希

在本发行版本中，我们调整了密码散列默认为与密码 存储 的 OWASP 建议 匹配。

作为此更改的一部分，默认密码散列提供程序已从 pbkdf2-sha256 改为 pbkdf2-sha512。另外，基于 pbkdf2 的密码哈希算法的默认哈希迭代数量更改，如下所示：

供应商 ID	algorithm	旧迭代	新的迭代
pbkdf2	**PBKDF2WithHmacSHA1**	20.000	1.300.000
pbkdf2-sha256	**PBKDF2WithHmacSHA256**	27.500	600.000
pbkdf2-sha512	**PBKDF2WithHmacSHA512**	30.000	210.000

如果域没有使用 hashAlgorithm 和 hashIterations 明确配置密码策略，则新配置将根据登录或创建或更新用户密码时生效。

2.6.7.1. 新密码哈希配置的性能

在具有 Intel i9-8950HK CPU (12)@ 4.800GHz）的机器上测试，对于哈希 1000 密码(averages from 3 run)会产生以下 DESTINATION 时间差异。请注意，PBKDF2WithHmacSHA1 的平均持续时间是因为长时间运行时而计算较低密码的数量。

供应商 ID	algorithm	旧持续时间	新持续时间	区别
pbkdf2	**PBKDF2WithHmacSHA1**	122ms	3.114ms	+2.992ms
pbkdf2-sha256	**PBKDF2WithHmacSHA256**	20ms	451ms	+431ms
pbkdf2-sha512	**PBKDF2WithHmacSHA512**	33ms	224ms	+191ms

pbkdf2 供应商的用户可能需要显式减少哈希迭代的数量，才能重新获得可接受的性能。这可以通过在域的密码策略中明确配置哈希迭代来实现。

2.6.7.2. 预期增加整个 CPU 用量和临时增加的数据库活动

我们的测试中每个基于密码的登录的 CPU 使用率增加了五个因素，其中包括更改的密码哈希和未更改的 TLS 连接处理。由于红帽构建的 Keycloak 的其他活动（如刷新访问令牌和客户端凭证授予），整个 CPU 整体 CPU 的增加应达到两到三的因素。这仍然取决于安装的唯一工作负载。

升级后，在基于密码的登录时，用户的密码将使用新的哈希算法和哈希迭代作为一次性活动，并在数据库中更新。由于这会从红帽构建的 Keycloak 内部缓存中清除用户，因此您还会在数据库级别看到增加的读取活动。随着更多用户的密码被重新哈希，这会缩短数据库活动。

2.6.7.3. 如何继续使用旧的 pbkdf2-sha256 密码散列？

要为域保留旧的密码哈希，请在 realm 密码策略中明确指定 hashAlgorithm 和 hashIterations。

- **散列算法：pbkdf2-sha256**
- **哈希迭代：27500**

2.6.8. 重命名用于迁移的 JPA 供应商配置选项

删除 Map Store 后，会重命名以下配置选项：

- **spi-connections-jpa-legacy-initialize-empty to spi-connections-jpa-quarkus-initialize-empty**

- spi-connections-jpa-legacy-migration-export to spi-connections-jpa-quarkus-migration-export

- spi-connections-jpa-legacy-migration-strategy to spi-connections-jpa-quarkus-migration-strategy

2.6.9. 使用 event 替换临时锁定日志

现在，当用户被 brute 强制保护程序暂时锁定时，新的事件 USER_DISABLED_BY_TEMPORARY_LOCKOUT。ID KC-SERVICES0053 的日志已被删除，因为新事件以结构化的形式提供信息。

由于它是成功事件，新事件默认记录在 DEBUG 级别。使用设置 spi-events-listener-jboss-logging-success-level，如 服务器管理指南中的事件监听程序章节中所述，以更改所有成功事件的日志级别。

要触发自定义操作或自定义日志条目，请编写自定义事件监听程序，如 Server Developer Guide 中的 Event Listener SPI 所述。

2.6.10. Keycloak JS 导入可能需要更新

如果您要直接从红帽构建的 Keycloak JS 加载 Keycloak JS，则可以安全地忽略本节。如果您要从 NPM 软件包加载 Keycloak JS，并使用如 Webpack、Vite 等捆绑包程序，您可能需要对代码进行一些更改。Keycloak JS 软件包现在使用 package.json 文件中的 exports 字段。这意味着您可能需要更改导入：

```
// Before
import Keycloak from 'keycloak-js/dist/keycloak.js';
import AuthZ from 'keycloak-js/dist/keycloak-authz.js';

// After
import Keycloak from 'keycloak-js';
import AuthZ from 'keycloak-js/authz';
```

2.6.11. 内部算法从 HS256 改为 HS512

红帽构建的 Keycloak 用来为内部令牌签名（由红帽构建的 Keycloak 本身使用的 JWT，如刷新或操作令牌）已从 HS256 改为更安全的 HS512。现在为域添加了名为 hmac-generated-hs512 的新密钥提

供程序。请注意，在迁移的域中，旧的 hmac-generated 供应商和旧的 HS256 密钥会被维护，并仍然验证升级前发布的令牌。当 轮转密钥指南 后不存在更旧的令牌时，可以手动删除 HS256 供应商。

2.6.12. 在容器中运行时的不同 JVM 内存设置

在容器中运行时，JVM 选项 -Xms 和 -Xmx 被 -XX:InitialRAMPercentage 和 -XX:MaxRAMPercentage 替代。红帽构建的 Keycloak 而不是静态最大堆大小设置，指定容器内存总量的 70%。

由于堆大小根据容器内存总量动态计算，您应该始终为容器设置内存限值。

警告

如果没有设置内存限制，则内存消耗会快速增加，因为最大堆大小将容器内存总量增长到 70%。

如需了解更多详细信息，请参阅 指定不同的内存设置。

2.6.13. 在 OAuth 2.0/OpenID Connect Authentication Response 中添加是参数

RFC 9207 OAuth 2.0 Authorization Server Issuer Identification 规格添加 该参数 在 OAuth 2.0/OpenID Connect Authentication Response 中用于实现安全授权响应。

在过去的版本中，我们没有这个参数，但现在红帽构建的 Keycloak 会默认添加这个参数。

但是，一些 OpenID Connect / OAuth2 适配器，特别是较旧的红帽构建的 Keycloak 适配器，这个新参数可能会出现问题。

例如，参数在成功向客户端应用程序进行身份验证后始终存在于浏览器 URL 中。在这些情况下，禁用向身份验证响应添加 iss 参数可能很有用。这可以为红帽构建的 Keycloak 管理控制台中的特定客户端完成，在带有 OpenID Connect Compatibility Modes 的小节中的客户端详情中，如 第 4.1 节 “与旧适配器兼容” 所述。专用 Exclude Issuer From Authentication Response 开关存在，可以打开以防止将 iss 参数添加到身份验证响应中。

2.6.14. 通配符字符处理

JPA 在搜索时允许通配符 % 和 _，而 LDAP 等其他供应商只允许 X。因为 * 是 LDAP 中的自然通配符字符，它在所有位置工作，而 JPA 则仅在搜索字符串的开头和结尾工作。从这个版本开始，唯一的通配符字符是 *，它在搜索字符串中的所有位置都一致地工作。对于 JPA，特定供应商（如 % 和 _ ）中的所有特殊字符都会被转义。对于确切搜索，带有添加的引号，如 "w*ord"，行为与之前版本中的行为相同。

2.6.15. 由 realm 和 client 角色映射的声明的值格式更改

在此版本之前，域(User Realm Role)和客户端(User Client Role)协议映射器在禁用 Multivalued 设置时映射了一个字符串fied JSON 数组。

但是，Multivalued 设置指示声明应映射为列表，如果禁用，则只有来自相同值列表中的单个值。

在这个发行版本中，当角色和客户端映射器标记为单值时，角色和客户端映射器现在从用户的有效角色映射到单个值(多值 禁用)。

2.6.16. 在登录 UI 中更改密码字段

在这个版本中，我们需要引入隐藏/显示密码输入的切换。

受影响的页面：

- login.ftl
- login-password.ftl
- login-update-password.ftl
- register.ftl
- register-user-profile.ftl

一般情况下，所有 <input type="password" name="password" /& gt; 现在被封装在一个 div 中。

input 元素后跟一个按钮，用于切换密码输入的可见性。

旧代码示例：

```
<input type="password" id="password" name="password" autocomplete="current-password"
style="display:none;"/>
```

新代码示例：

```
<div class="${properties.kcInputGroup!}">
    <input type="password" id="password" name="password" autocomplete="current-password"
style="display:none;"/>
    <button class="pf-c-button pf-m-control" type="button" aria-label="${msg('showPassword')}"
            aria-controls="password" data-password-toggle
            data-label-show="${msg('showPassword')}" data-label-hide="${msg('hidePassword')}">
        <i class="fa fa-eye" aria-hidden="true"></i>
    </button>
</div>
```

2.6.17. kc.sh 和 shell metacharacters

kc.sh 不再使用额外的 shell 评估参数，环境变量 JAVA_OPTS_APPEND 和 JAVA_ADD_OPENS，因此继续使用双转义/仲裁将导致参数被误解。例如，而不是

```
bin/kc.sh start --db postgres --db-username keycloak --db-url
"\"jdbc:postgresql://localhost:5432/keycloak?ssl=false&connectTimeout=30\"" --db-password
keycloak --hostname localhost
```

使用单个转义：

```
bin/kc.sh start --db postgres --db-username keycloak --db-url
"jdbc:postgresql://localhost:5432/keycloak?ssl=false&connectTimeout=30" --db-password keycloak --
hostname localhost
```

此更改还意味着您无法使用带带所有参数的单个值来调用 kc.sh。例如，您无法再使用

```
bin/kc.sh "start --help"
```

它必须是单独的参数

```
bin/kc.sh start --help
```

同样地而不是，

```
bin/kc.sh build "--db postgres"
```

它必须是单独的参数

```
bin/kc.sh build --db postgres
```

Dockerfile 运行命令中还需要使用各个参数。

2.6.18. GroupProvider 更改

添加了一个新的方法，以允许通过顶级组搜索和分页。如果您实现这个接口，则需要使用以下方法：

```
Stream<GroupModel> getTopLevelGroupsStream(RealmModel realm,
                                           String search,
                                           Boolean exact,
                                           Integer firstResult,
                                           Integer maxResults)
```

2.6.19. GroupRepresentation 更改

- 添加了新字段 subGroupCount，以告知客户端在任何给定组中有多少个子组
- 现在，sub groups 列表只会填充在请求层次结构数据的查询中
- 此字段填充自"bottom up"，因此无法依赖它来获取组的所有子组。使用 GroupProvider 或从 GET {keycloak server}/realms/{realm}/groups/{group_id}/children中请求子组

2.6.20. Group Admin API 的新端点

端点 GET {keycloak server}/realms/{realm}/groups/{group_id}/children 添加为获取支持分页的特定组的子组的方法

2.6.21. RESTEeasy Reactive

依赖 RESTEasy Classic 不再是一个选项，因为它不再可用。依赖于 RESTEasy Classic 和相关软件包部分的 SPI 和相关软件包部分需要迁移。

2.6.22. 部分导出需要 manage-realm 权限

端点 POST {keycloak server}/realms/{realm}/partial-export 和 admin 控制台中的对应操作现在需要 manage-realm 权限才能执行，而不是 view-realm。此端点将 realm 配置导出到 JSON 文件中，新权限更为合适。参数 exportGroupsAndRoles 和 exportClients 分别在导出中包含 realm groups/roles 和 clients，继续管理相同的权限(query-groups 和 view-clients)。

2.6.23. 客户端的有效重定向 URI 始终与精确字符串匹配进行比较

在将重定向 URI 与客户端指定有效重定向进行比较时，1.8.0 为主机名和方案引入了小写。不幸的是，它并不在所有协议中完全工作，例如，主机对于 http 来说较低，但不适用于 https。因为 OAuth 2.0 安全最佳实践 建议使用准确字符串匹配比较 URI，Red Hat build of Keycloak 遵循建议，且在有效重定向中会按照具体情况进行比较，即使主机名和方案也是如此。

对于依赖于旧行为的域，其客户端的有效重定向 URI 现在应该保存服务器应识别的每个 URI 的独立条目。

虽然它在配置客户端时引入了更多步骤和详细程度，但新行为可以启用更安全的部署，因为基于模式的检查经常造成安全问题。不仅是由于它们的实施方式，也只是配置它们的方式。

2.6.24. 修复对 Groups.getSubGroups briefRepresentation 参数的处理

版本 23.0.0 在 Groups 资源上引入了一个新的端点 getSubGroups ("children")，其中参数 briefRepresentation 表示检索子组的完整表示。现在，这个含义已被修改来返回简短表示。

2.6.25. jboss-logging 事件消息中的更改

jboss-logging 消息值现已加引号（默认为 " 字符"），而 sanitized 以防止任何换行符。供应商中有两个新选项(spi-events-listener-jboss-logging-sanitize 和 spi-events-listener-jboss-logging-quotes)，允许您自定义新行为。例如，为了避免清理和引用，可以以这种方式启动服务器：

```
./kc.sh start --spi-events-listener-jboss-logging-sanitize=false --spi-events-listener-jboss-logging-quotes=none ...
```

有关选项的更多信息，请参阅 所有供应商配置。

2.7. 弃用和删除的功能

2.7.1. truststore 弃用

spi-truststore-file jpeg 选项和信任存储相关的选项 https-trust-store114 已被弃用。因此，对信任存储资料使用新的默认位置 conf/truststores，或使用 truststore-paths 选项指定您需要的路径。详情请参阅 为传出请求配置可信证书。

应该使用 tls-hostname-verifier 属性，而不是 spi-truststore-file-hostname-verification-policy 属性。

更改的资料效果是，信任存储供应商总是使用一些证书（至少存在默认的 Java 可信证书）。这个新行为可能会影响红帽构建的 Keycloak 的其他部分。

例如，如果 测试 机构被配置为 Direct 验证，webauthn 注册可能会失败。在以前的版本中，如果 truststore 供应商没有配置传入的证书，则不会验证传入的证书。但是现在，这个验证总是被执行。注册失败，并显示 无效的证书路径 错误，因为 dongle 发送的证书链不被红帽构建的 Keycloak 信任。验证器的证书颁发机构需要存在于信任存储提供程序中，才能正确执行 attestation。

2.7.2. 弃用的 --proxy 选项

--proxy 选项已弃用，并将在以后的发行版本中删除。下表说明了已弃用的选项如何映射到支持的选项。

弃用的用法	新用法
kc.sh （没有 代理 选项集）	kc.sh
kc.sh --proxy none	kc.sh
kc.sh --proxy edge	kc.sh --proxy-headers forwarded|xforwarded --http-enabled true
kc.sh --proxy passthrough	kc.sh --hostname-port 80|443 （如果使用 HTTPS）
kc.sh --proxy reencrypt	kc.sh --proxy-headers forwarded|xforwarded

注意

为提高安全性，--proxy-headers 选项不允许同时选择 正向和 x forwarded 值（因为它是 --proxy edge 和 --proxy reencrypt之前的情况）。

警告

使用 proxy 标头选项时，请确保您的反向代理正确设置并分别覆盖 Forwarded 或 X-ForwardedGalaxy 标头。要设置这些标头，请参阅您的反向代理文档。错误配置会将红帽构建的 Keycloak 暴露给安全漏洞。

您还可以使用 Operator 时设置代理标头：

```
apiVersion: k8s.keycloak.org/v2alpha1
kind: Keycloak
metadata:
  name: example-kc
spec:
  ...
  proxy:
    headers: forwarded|xforwarded
```

注意

如果没有指定 proxy.headers 字段，Operator 会默认隐式设置 proxy=passthrough 来回退到之前的行为。这会在服务器日志中产生弃用警告。此回退将在以后的发行版本中被删除。

2.7.3. 弃用的离线会话预加载

红帽构建的 Keycloak 的默认行为是按需加载离线会话。在启动时预加载它们的旧行为现已弃用，因为在启动时预加载它们无法很好地扩展，并增加红帽构建的 Keycloak 内存用量。旧的行为将在以后的发行版本中被删除。

要在已弃用且还没有删除时重新启用旧行为，请使用功能标记和 SPI 选项，如下所示：

```
bin/kc.[sh|bat] start --features-enabled offline-session-preloading --spi-user-sessions-infinispan-preload-offline-sessions-from-database=true
```

UserSessionProvider 的 API 弃用了方法 getOfflineUserSessionByBrokerSessionId (RealmModel realm, String brokerSessionId)。除了此方法外，请使用 getOfflineUserSessionByBrokerUserIdStream (RealmModel, String brokerUserId) 来首先获取用户的会话，然后根据需要过滤代理会话 ID。

2.7.4. 弃用了数据供应商和模型的方法

- RealmModel#getTopLevelGroupsStream （） 和超载方法现已弃用

2.7.5. Cookie 的弃用和删除

作为红帽构建的 Keycloak 中重构 Cookie 处理的一部分，有一些对 Cookie 的设置方式的更改：

- 现在，如果请求通过安全上下文，则所有 Cookie 都会设置 secure 属性
- WELCOME_STATE_CHECKER cookies 现在设置 SameSite=Strict

对于自定义扩展，可能需要进行一些更改：

- LocaleSelectorProvider.KEYCLOAK_LOCALE 已被弃用，因为 Cookie 现在通过 CookieProvider 管理
- HttpResponse.setWriteCookiesOnTransactionComplete has been removed
- HttpCookie 已被弃用，请使用 NewCookie.Builder
- ServerCookie 已被弃用，请使用 NewCookie.Builder

2.7.6. 为 SAML 加密删除已弃用的模式

现在，在版本 21 中引入的 SAML 加密的兼容性模式已被删除。系统属性

keycloak.saml.deprecated.encryption 不再由服务器管理。仍使用旧签名密钥进行加密的客户端应该从新的 IDP 配置元数据进行更新。

2.7.7. 重命名模型模块

删除 Map Store 后，以下模块被重命名：

- org.keycloak:keycloak-model-legacy-private to org.keycloak:keycloak-model-storage-private
- org.keycloak:keycloak-model-legacy-services to org.keycloak:keycloak-model-storage-services

和 org.keycloak:keycloak-model-legacy 模块已被弃用，并将在 org.keycloak:keycloak-model-storage 模块的下一发行版本中删除。

2.7.8. 删除了 RegistrationProfile 表单操作

表单操作 RegistrationProfile （在身份验证流的 UI 中显示为 Profile Validation）已从代码库中删除，也可以从所有验证流中删除。默认情况下，它位于每个域的内置注册流中。验证用户属性以及创建用户，包括所有用户的属性都由 RegistrationUserCreation 表单操作处理，因此不再需要 RegistrationProfile。与这个更改相关的通常不需要进一步的操作，除非您在您自己的供应商中使用 RegistrationProfile 类。

2.8. 不再支持通过 ADMIN USER API 更新用户时对用户属性进行部分更新

通过 Admin User API 更新用户属性时，在更新用户属性时，您无法执行部分更新，包括 用户名、电子邮件、firstName 和 lastName 等 root 属性。

2.8.1. 弃用的 auto-build CLI 选项已被删除

auto-build CLI 选项在很长时间内被标记为弃用。在这个发行版本中，它已被完全删除，它不再被支持。

在执行 start 命令时，服务器会根据配置自动构建。要防止此行为，请设置 --optimized 标志。

2.8.2. 删除选项以修剪事件的详情长度

从这个版本开始，Keycloak 支持 EventEntity details 列的长值。因此，它不再支持修剪事件详情长度 --spi-events-store-jpa-max-detail-length 和 --spi-events-store-jpa-max-field-length 的选项。

2.8.3. 从转换中删除命名空间

如果您编写了自己的翻译或扩展管理员 ui，则需要将它们迁移到此新格式时，我们将所有翻译移到 admin-ui 中的一个文件中。另外，如果您的数据库中有"overrides"，则必须从密钥中删除命名空间。有些键只在不同的命名空间中相同，这最明显有助于。在这些情况下，我们使用 帮助 来记录密钥。

如果要使用这个节点脚本来帮助迁移。它将获取所有单个文件并将其放入一个新文件，并负责一些映射：

```
import { readFileSync, writeFileSync, appendFileSync } from "node:fs";

const ns = [
  "common",
  "common-help",
  "dashboard",
  "clients",
  "clients-help",
  "client-scopes",
  "client-scopes-help",
  "groups",
  "realm",
  "roles",
  "users",
  "users-help",
  "sessions",
  "events",
  "realm-settings",
  "realm-settings-help",
  "authentication",
  "authentication-help",
  "user-federation",
  "user-federation-help",
  "identity-providers",
  "identity-providers-help",
  "dynamic",
];

const map = new Map();
const dup = [];

ns.forEach((n) => {
  const rawData = readFileSync(n + ".json");
  const translation = JSON.parse(rawData);
  Object.entries(translation).map((e) => {
    const name = e[0];
    const value = e[1];
```

```
    if (map.has(name) && map.get(name) !== value) {
      if (n.includes("help")) {
        map.set(name + "Help", value);
      } else {
        map.set(name, value);
        dup.push({
          name: name,
          value: map.get(name),
          dup: { ns: n, value: value },
        });
      }
    } else {
      map.set(name, value);
    }
  });
});

writeFileSync(
  "translation.json",
  JSON.stringify(Object.fromEntries(map.entries()), undefined, 2),
);

const mapping = [
  ["common:clientScope", "clientScopeType"],
  ["identity-providers:createSuccess", "createIdentityProviderSuccess"],
  ["identity-providers:createError", "createIdentityProviderError"],
  ["clients:createError", "createClientError"],
  ["clients:createSuccess", "createClientSuccess"],
  ["user-federation:createSuccess", "createUserProviderSuccess"],
  ["user-federation:createError", "createUserProviderError"],
  ["authentication-help:name", "flowNameHelp"],
  ["authentication-help:description", "flowDescriptionHelp"],
  ["clientScopes:noRoles", "noRoles-clientScope"],
  ["clientScopes:noRolesInstructions", "noRolesInstructions-clientScope"],
  ["users:noRoles", "noRoles-user"],
  ["users:noRolesInstructions", "noRolesInstructions-user"],
  ["clients:noRoles", "noRoles-client"],
  ["clients:noRolesInstructions", "noRolesInstructions-client"],
  ["groups:noRoles", "noRoles-group"],
  ["groups:noRolesInstructions", "noRolesInstructions-group"],
  ["roles:noRoles", "noRoles-roles"],
  ["roles:noRolesInstructions", "noRolesInstructions-roles"],
  ["realm:realmName:", "realmNameField"],
  ["client-scopes:searchFor", "searchForClientScope"],
  ["roles:searchFor", "searchForRoles"],
  ["authentication:title", "titleAuthentication"],
  ["events:title", "titleEvents"],
  ["roles:title", "titleRoles"],
  ["users:title", "titleUsers"],
  ["sessions:title", "titleSessions"],
  ["client-scopes:deleteConfirm", "deleteConfirmClientScopes"],
  ["users:deleteConfirm", "deleteConfirmUsers"],
  ["groups:deleteConfirm_one", "deleteConfirmGroup_one"],
  ["groups:deleteConfirm_other", "deleteConfirmGroup_other"],
  ["identity-providers:deleteConfirm", "deleteConfirmIdentityProvider"],
  ["realm-settings:deleteConfirm", "deleteConfirmRealmSetting"],
```

```
  ["roles:whoWillAppearLinkText", "whoWillAppearLinkTextRoles"],
  ["users:whoWillAppearLinkText", "whoWillAppearLinkTextUsers"],
  ["roles:whoWillAppearPopoverText", "whoWillAppearPopoverTextRoles"],
  ["users:whoWillAppearPopoverText", "whoWillAppearPopoverTextUsers"],
  ["client-scopes:deletedSuccess", "deletedSuccessClientScope"],
  ["identity-providers:deletedSuccess", "deletedSuccessIdentityProvider"],
  ["realm-settings:deleteSuccess", "deletedSuccessRealmSetting"],
  ["client-scopes:deleteError", "deletedErrorClientScope"],
  ["identity-providers:deleteError", "deletedErrorIdentityProvider"],
  ["realm-settings:deleteError", "deletedErrorRealmSetting"],
  ["realm-settings:saveSuccess", "realmSaveSuccess"],
  ["user-federation:saveSuccess", "userProviderSaveSuccess"],
  ["realm-settings:saveError", "realmSaveError"],
  ["user-federation:saveError", "userProviderSaveError"],
  ["realm-settings:validateName", "validateAttributeName"],
  ["identity-providers:disableConfirm", "disableConfirmIdentityProvider"],
  ["realm-settings:disableConfirm", "disableConfirmRealm"],
  ["client-scopes:updateSuccess", "updateSuccessClientScope"],
  ["client-scopes:updateError", "updateErrorClientScope"],
  ["identity-providers:updateSuccess", "updateSuccessIdentityProvider"],
  ["identity-providers:updateError", "updateErrorIdentityProvider"],
  ["user-federation:orderChangeSuccess", "orderChangeSuccessUserFed"],
  ["user-federation:orderChangeError", "orderChangeErrorUserFed"],
  ["authentication-help:alias", "authenticationAliasHelp"],
  ["authentication-help:flowType", "authenticationFlowTypeHelp"],
  ["authentication:createFlow", "authenticationCreateFlowHelp"],
  ["client-scopes-help:rolesScope", "clientScopesRolesScope"],
  ["client-scopes-help:name", "scopeNameHelp"],
  ["client-scopes-help:description", "scopeDescriptionHelp"],
  ["client-scopes-help:type", "scopeTypeHelp"],
  ["clients-help:description", "clientDescriptionHelp"],
  ["clients-help:clientType", "clientsClientTypeHelp"],
  ["clients-help:scopes", "clientsClientScopesHelp"],
  ["common:clientScope", "clientScopeTypes"],
  ["dashboard:realmName", "realmNameTitle"],
  ["common:description", "description"],
];

mapping.forEach((m) => {
  const key = m[0].split(":");
  try {
    const data = readFileSync(key[0] + ".json");
    const translation = JSON.parse(data);
    const value = translation[key[1]];
    if (value) {
      appendFileSync(
        "translation.json",
        '"' + m[1] + '": ' + JSON.stringify(value) + ',\n',
      );
    }
  } catch (error) {
    console.error("skipping namespace key: " + key);
  }
});
```

将此保存到 public/locale/<language > 文件夹中的名为 transform.mjs 的文件，并使用以下内容运行它：

```
node ./transform.mjs
```

注意

这可能不会进行完整的转换，但非常接近。

第 3 章 升级红帽构建的 KEYCLOAK 服务器

您可以在升级适配器前升级服务器。

3.1. 准备升级

在升级服务器前执行以下步骤。

流程

1. 备份旧安装，如配置、主题等。

2. 处理任何打开的事务并删除 data/tx-object-store/ transaction 目录。

3. 按照相关数据库文档中的说明备份数据库。

   升级服务器后，数据库将不再与旧服务器兼容。如果您需要恢复升级，首先恢复旧的安装，然后从备份副本中恢复数据库。

警告

在升级红帽构建的 Keycloak 后，除了离线用户会话外，用户会话将会丢失。用户必须再次登录。

3.2. 下载红帽构建的 KEYCLOAK 服务器

准备升级后，您可以下载服务器。

流程

1. 从红帽构建的 Keycloak 网站下载并提取 rhbk-24.0.5.zip。

提取此文件后，您应该有一个名为 rhbk-24.0.5 的目录。

2. 将此目录移到所需位置。

3. 将 conf /、provider / 和es/ 从上一安装复制到新安装。

3.3. 迁移数据库

红帽构建的 Keycloak 可以自动迁移数据库架构，也可以选择手动迁移它。默认情况下，在第一次启动新安装时，数据库会自动迁移。

3.3.1. 自动关系数据库迁移

要执行自动迁移，请启动连接到所需数据库的服务器。如果新服务器版本更改了数据库架构，则迁移会自动启动，除非数据库有太多记录。

例如，在具有数百万记录的表上创建索引可能非常耗时，并导致出现重大服务中断。因此，自动迁移存在 300000 记录阈值。如果记录数量超过这个阈值，则不会创建索引。反之，您可以使用 SQL 命令在服务器日志中找到一个警告，您可以手动应用。

要更改阈值，为默认的 connection- liquibase 供应商设置 index-creation- threshold 属性：

```
kc.[sh|bat] start --spi-connections-liquibase-default-index-creation-threshold=300000
```

3.3.2. 手动关系数据库迁移

要启用对数据库模式的手动升级，请将默认 connections-jpa 供应商的 migration-strategy 属性值设置为 "manual" ：

```
kc.[sh|bat] start --spi-connections-jpa-quarkus-migration-strategy=manual
```

当您使用此配置启动服务器时，服务器会检查是否需要迁移数据库。所需的更改会被写入 bin/keycloak-database-update.sql SQL 文件，您可以检查并手动针对数据库运行。

要更改导出的 SQL 文件的路径和名称，请为默认 connection- jpa 供应商设置 migration- export 属

性：

```
kc.[sh|bat] start --spi-connections-jpa-quarkus-migration-export=<path>/<file.sql>
```

有关如何将此文件应用到数据库的详情，请查看您的相关数据库文档。将更改写入文件后，服务器将退出。

3.4. 迁移主题

如果您创建了自定义主题，则这些主题必须迁移到新的服务器。另外，对内置主题的任何更改可能需要反映在自定义主题中，具体取决于您自定义的各个方面。

流程

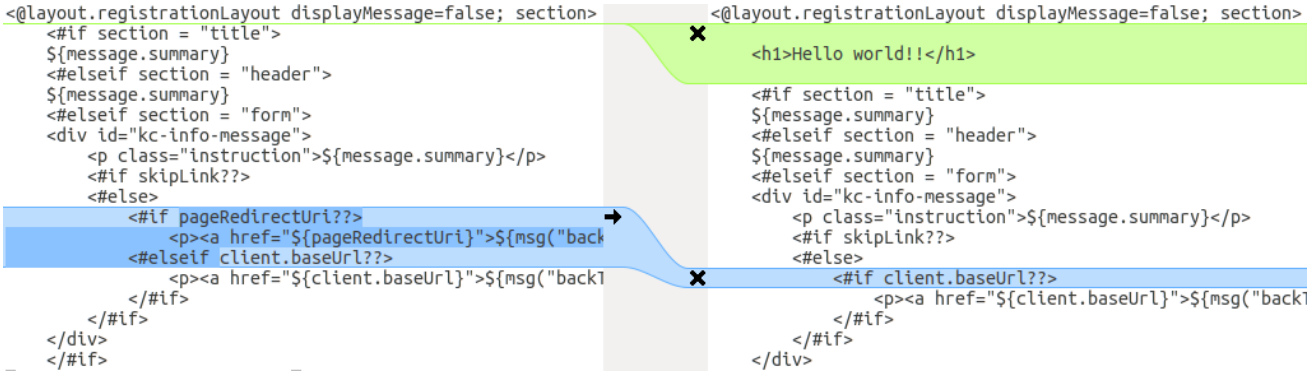
1. 将自定义主题从旧服务器复制到新的服务器 `themes` 目录中。
2. 使用以下部分迁移模板、消息和样式。
   - 如果您自定义了 Migration Changes 中列出的任何更新模板，请比较基础主题中的模板，以检查您需要应用的任何更改。
   - 如果自定义消息，可能需要更改键或值或添加其他消息。
   - 如果您自定义任何样式且您要扩展红帽构建的 Keycloak 主题，请查看对样式的更改。如果要扩展基础主题，您可以跳过这一步。

3.4.1. 迁移模板

如果自定义模板，请查看新版本以决定更新自定义模板。如果您进行次要更改，您可以将更新的模板与自定义模板进行比较。但是，如果您进行了许多更改，请考虑将新模板与自定义模板进行比较。此比较将向您展示您需要进行哪些更改。

您可以使用 diff 工具比较模板。以下屏幕截图比较了登录主题和示例自定义主题中的 info.ftl 模板：

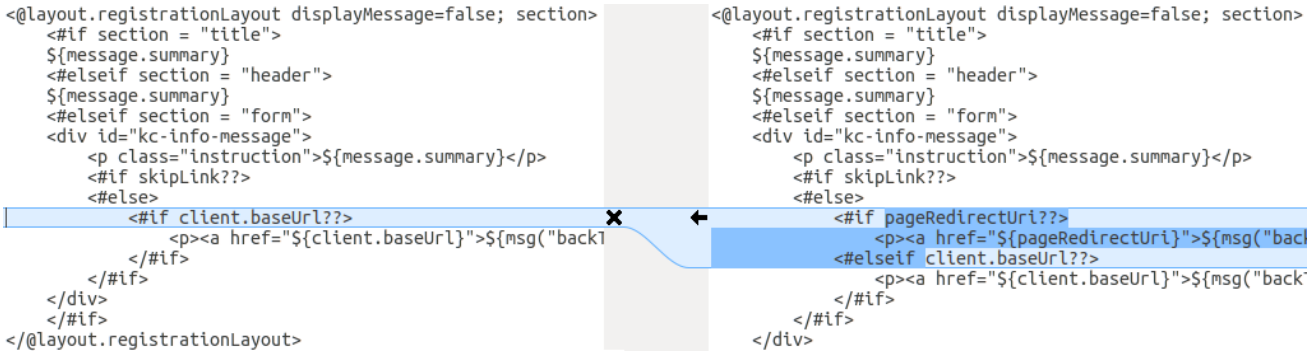
更新了登录主题模板和自定义登录主题模板的版本



这一比较显示，第一次更改(Hello world!!!)是自定义，而第二次更改(如果 pageRedirectUri)是基础主题的更改。通过将第二次更改复制到自定义模板，您已成功更新自定义模板。

在替代方法中，以下屏幕截图将旧安装中的 info.ftl 模板与新安装中的 info.ftl 模板进行比较：

从旧安装和更新的登录主题模板登录主题模板



此比较显示了基础模板中已更改的内容。然后您可以手动对修改后的模板进行相同的更改。由于这种方法更为复杂，因此仅在第一种方法不可行时才使用此方法。

3.4.2. 迁移消息

如果您添加了对其他语言的支持，则需要应用上面列出的所有更改。如果您还没有添加对其他语言的支持，则可能不需要更改任何内容。只有在您更改了主题中的受影响消息时，才需要进行更改。

流程

1. 对于添加的值，请查看基本主题中消息的值，以确定是否需要自定义该消息。

2. 对于重命名的密钥，请重命名自定义主题中的密钥。

3. 对于更改的值，请检查基础主题中的值，以确定是否需要对自定义主题进行更改。

3.4.3. 迁移风格

您可能需要更新您的自定义样式，以反映对内置主题对样式所做的更改。考虑使用 diff 工具将旧服务器安装和新服务器安装之间的风格表的更改进行比较。

例如：

```
$ diff RHSSO_HOME_OLD/themes/keycloak/login/resources/css/login.css \
RHSSO_HOME_NEW/themes/keycloak/login/resources/css/login.css
```

检查更改，并确定它们是否影响您的自定义策略。

第 4 章 升级红帽构建的 KEYCLOAK 适配器

升级红帽构建的 Keycloak 服务器后，您可以升级适配器。适配器的早期版本可能适用于红帽构建的 Keycloak 服务器的后续版本，但红帽构建的 Keycloak 服务器的早期版本可能不适用于以后的适配器版本。

4.1. 与旧适配器兼容

红帽构建的 Keycloak 服务器的较新版本可能用于较旧版本的适配器。但是，红帽构建的 Keycloak 服务器的一些修复可能会破坏与较旧版本的适配器的兼容性。例如，OpenID Connect 规格的新实现可能与旧的客户端适配器版本不匹配。

在这种情况下，您可以使用兼容性模式。对于 OpenID Connect 客户端，管理控制台在页面中包含 OpenID Connect 兼容性模式，以及客户端详情。使用此选项，您可以禁用红帽构建的 Keycloak 服务器的一些新方面，以保持与旧客户端适配器的兼容性。详情请查看单个交换机的工具提示。

4.2. 升级 EAP 适配器

要升级 JBoss EAP 适配器，请完成以下步骤：

流程

1. 下载新的适配器存档。
2. 通过删除 EAP_HOME/modules/system/add-ons/keycloak/ 目录来删除前面的适配器模块。
3. 将下载的存档解压缩到 EAP_HOME 中。

4.3. 升级 JAVASCRIPT 适配器

要升级已复制到 web 应用程序的 JavaScript 适配器，请执行以下步骤。

流程

1. 下载新的适配器存档。

2. 使用下载的存档中的 keycloak.js 文件覆盖应用程序中的 keycloak.js 文件。

4.4. 升级 NODE.JS 适配器

要升级已复制到 web 应用程序的 Node.js 适配器，请执行以下步骤。

流程

1. 下载新的适配器存档。

2. 删除现有 Node.js 适配器目录

3. 将更新的文件解压缩到其位置

4. 在应用程序的 package.json 中更改 keycloak-connect 的依赖项

第 5 章 升级红帽构建的 KEYCLOAK ADMIN 客户端

在升级 admin-client 前，请确保升级红帽构建的 Keycloak 服务器。admin-client 的早期版本可能会与 Red Hat build of Keycloak 服务器的后续版本一起工作，但红帽构建的 Keycloak 服务器的早期版本可能不适用于 admin-client 的后续版本。因此，使用与当前红帽构建的 Keycloak 服务器版本匹配的 admin-client 版本。