Red Hat build of Keycloak 24.0

Upgrading Guide

Last Updated: 2024-06-28

Legal Notice

Abstract

This book is a guide to upgrading Red Hat build of Keycloak from 22.0.x to 24.0.5.

Table of Contents

CHAPTER 1. UPGRADING RED HAT BUILD OF KEYCLOAK

This guide describes how to upgrade Red Hat build of Keycloak from version 22.0.x to version 24.0.5. Use the following procedures in this order:

1. Review the migration changes from the previous version of Red Hat build of Keycloak.
2. Upgrade the Red Hat build of Keycloak server.
3. Upgrade the Red Hat build of Keycloak adapters.
4. Upgrade the Red Hat build of Keycloak Admin Client.

For Red Hat Single Sign-On 7.6 customers, use the Migration Guide instead of this guide.

CHAPTER 2. MIGRATION CHANGES

2.1. USER PROFILE CHANGES

2.1.1. User profile enabled by default

The user profile feature is now enabled by default. The **declarative-user-profile** feature is no longer available, because the user profile is assumed to be enabled. Therefore, the **User Profile Enabled** switch is removed from the **Realm Settings** and replaced by **Unmanaged attributes**. When migrating from previous version, the behavior is as follows:

- For the deployments where **User Profile Enabled** was set to **ON**, **Unmanaged attributes** will be set to **OFF** after the upgrade. As a result, only user attributes supported explicitly by user profile will be allowed.

- For the deployments where the **User Profile Enabled** was set to **OFF** (which was also the default settings for the deployments with **declarative-user-profile** feature disabled, which was the default), **Unmanaged attributes** will be set to **ON** after the upgrade. As a result, the behavior should be basically the same as in previous versions with the user profile disabled. The **Attributes** tab will remain in the user details part of the Admin Console. Also users can now set arbitrary attributes through UI pages such as the registration page and update profile page as long as a particular custom theme supports it. The custom themes should work as before as well. However, consider updating your themes to use the user-profile and maybe even remove your custom themes if those themes were need to add custom attributes. See the subsequent section on themes. Also, consider toggling **Unmanaged attributes** to **OFF** or enable this switch only for administrators so that you can rely mainly on using managed attributes.

See the User Profile Documentation for the details about the **Unmanaged attributes**.

2.1.2. Default validations

Default user profile configuration comes with a set of default validations for the basic predefined fields. Those validations were not present in previous versions when the **declarative-user-profile** feature was disabled by default. If you have issues due to backward compatibility, you can change the default validators according to your needs. The default validators are as follows:

- The **username**, **email**, **firstName** and **lastName** attributes have a maximum length of 255 characters. These validations were indirectly present in previous versions as well because of the database constraint on the table **USER_ENTITY** for those fields with a maximum length of 255 characters. However, when using user storage providers, it might be possible before to use longer values.

- The **username** attribute has a minimum length of three characters. Username has also **username-prohibited-characters** and **up-username-not-idn-homograph** validator by default, which were not present in previous versions. See the Validation section of the User Profile Documentation for the details about those attributes. Note that username is not editable by default unless you have the realm switch **Edit username enabled**. This change means that existing users with incorrect usernames should still work and they will not be enforced to update their usernames. But new users will be enforced to use correct usernames during their registration or creation by the admin REST API.

- The **firstName** and **lastName** attributes have the **person-name-prohibited-characters** validator on them, which were not present in previous versions. See the Validation section of the User Profile Documentation for the details about those attributes. Note that both first name

and last name are editable by default, so users, who already have such incorrect first/last name from a previous version will be forced to update them when updating their user profiles.

2.1.3. User attribute names with strange characters

In previous versions, you could create a user with attribute names such as **some:attribute** or **some/attribute**. The user profile intentionally does not allow you to create attributes with such strange names in the user profile configuration. So you may need to configure **Unmanaged attributes** for your realm and enable unmanaged attributes for administrators (ideally) or for end users (if really needed). Although it is strongly preferred to avoid using such attribute names.

2.1.4. Verify Profile required action enabled by default

The **verify-profile** required action is enabled by default for new realms. However, when you migrate from a previous version, your existing realms will have the same state of this **verify-profile** action as before, which usually means disabled as it was disabled by default in previous versions. For the details about this required action, see the User Profile Documentation .

2.1.5. Changes to Freemarker templates to render pages based on the user profile and realm

In this release, the following templates were updated to make it possible to dynamically render attributes based on the user profile configuration set to a realm:

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

These templates are responsible for rendering the update profile (when the **Update Profile** required action is enabled for a user), the registration, and the update email (when the **UPDATE_EMAIL** feature is enabled) pages, respectively.

If you use a custom theme to change these templates, they will function as expected because only the content is updated. However, we recommend you to take a look at how to configure a declarative user profile and possibly avoid changing the built-in templates by using all the capabilities provided by this feature.

Also, the templates used by the **declarative-user-profile** feature to render the pages for the same flows are longer necessary and removed in this release:

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

If you were using the **declarative-user-profile** feature in a previous release with customizations to the above templates, update the **login-update-profile.ftl** and **register.ftl** accordingly.

2.1.6. New Freemarker template for the update profile page at first login through a broker

In this release, the server will render the update profile page when the user is authenticating through a broker for the first time using the **idp-review-user-profile.ftl** template.

In previous releases, the template used to update the profile during the first broker login flow was the **login-update-profile.ftl**, the same used to update the profile when users are authenticating to a realm.

By using separate templates for each flow, a more clear distinction exist as to which flow a template is actually used rather than sharing a same template, and potentially introduce unexpected changes and behavior that should only affect pages for a specific flow.

If you have customizations to the **login-update-profile.ftl** template to customize how users update their profiles when authenticating through a broker, make sure to move your changes to the new template.

2.2. CHANGES TO THEMES

2.2.1. Changes to the Welcome theme

The 'welcome' theme has been updated to use a new layout and now uses PatternFly 5, rather than PatternFly 3. If you are extending the theme, or providing your own, you may need to update it as follows:

2.2.1.1. Migrate from PatternFly 3 to PatternFly 5

The welcome theme was one of the more outdated themes in Red Hat build of Keycloak. It was originally based on PatternFly 3, but has now been updated to use PatternFly 5, skipping a major version in the process. If your custom theme extends the built-in theme, you will need to update it to use PatternFly 5 syntax. Consult the PatternFly 5 documentation for details.

If you are still using PatternFly 3 in your own custom theme (not extending the built-in one), you can continue to use it, but PatternFly 3 support will be removed in a future release, so you should consider migrating to PatternFly 5 as soon as possible.

2.2.1.2. Automatic redirect to the Admin Console

If the Admin Console is enabled, the welcome page will automatically redirect to it if the administrative user already exists. This behavior can be modified by setting the **redirectToAdmin** in your **theme.properties** file. By default, the property is set to **false**, unless you are extending the built-in theme, in which case, the property is set to **true**.

2.2.1.3. The documentationUrl and displayCommunityLinks properties have been removed.

These properties were previously used for navigational elements that are now no longer present. If you are extending the built-in theme, you will need to remove these properties from your **theme.properties** file, as they no longer have any effect.

2.2.1.4. Assets are now loaded from 'common' resources

Images such as the background, logo and favicon are now loaded from the 'common' resources, rather than the theme resources. This change means that if you are extending the built-in theme, and are overwriting these images, you will need to move them to the 'common' resources of your theme, and update your **theme.properties** file to include the new paths:

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

2.2.2. Changes to the Account Console theme customization

If you were previously extending the now deprecated version 2 of the Account Console theme, you will need to update your theme to use the new version 3 of the Account Console theme. The new version of the Account Console theme comes with some changes in regards to how it is customized. To start with a clean slate, you can follow the new customization quickstart.

To move your custom theme start by changing the **parent** to the new theme:

```
# Before
parent=keycloak.v2

# After
parent=keycloak.v3
```

If you have any custom React components, you will import React directly, rather than using a relative path:

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

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

If you are using **content.json** to customize the theme there are some changes to the structure of the file, specifically:

- The **content** property has been renamed to **children**.
- The **id**, **icon** and **componentName** properties have been removed, as **modulePath** provides the same functionality.

2.2.3. Language files for themes default to UTF-8 encoding

This release now follows the standard mechanisms of Java and later, which assumes resource bundle files to be encoded in UTF-8.

Previous versions of Keycloak supported specifying the encoding in the first line with a comment like **# encoding: UTF-8**, which is no longer supported and is ignored.

Message properties files for themes are now read in UTF-8 encoding, with an automatic fallback to ISO-8859-1 encoding. If you are using a different encoding, convert the files to UTF-8.

2.3. OPERATOR CHANGES

2.3.1. Operator Referenced Resource Polling

Secrets and ConfigMaps referenced via the Keycloak CR will now be polled for changes, rather than watched by the API server. This polling may take one minute for changes to be detected.

This was done to avoid requiring label manipulation on those resources. After upgrading if any Secret still has the operator.keycloak.org/component label, it may be removed or ignored.

2.3.2. Operator Customization Property Keys

The property keys used by the operator for advanced configuration have changed from **operator.keycloak** to **kc.operator.keycloak**.

2.3.3. Operator -secrets-store Secret

Older versions of the operator created a Secret to track watched Secrets. Newer versions of the operator no longer use the -secrets-store Secret, so it may be deleted.

2.4. KEYCLOAK CR CHANGES

2.4.1. Keycloak CR resources options

When no **resources** options are specified in the Keycloak CR and KeycloakRealmImport CR, default values are used. The default **requests** memory for Keycloak deployment and the realm import Job is set to **1700MiB**, and the **limits** memory is set to **2GiB**.

2.4.2. Default Keycloak CR Hostname

When running on OpenShift, with ingress enabled, and with the spec.ingress.classname set to openshift-default, you may leave the spec.hostname.hostname unpopulated in the Keycloak CR. The operator will assign a default hostname to the stored version of the CR similar to what would be created by an OpenShift Route without an explicit host - that is ingress-namespace.appsDomain If the appsDomain changes, or should you need a different hostname for any reason, then update the Keycloak CR.

2.5. FEATURES CHANGES

It is no longer allowed to have the same feature in both the **--features** and **--features-disabled** list. The feature should appear in only one list.

The usage of unversioned feature names, such as **docker**, in the **--features** list will allow for the most supported / latest feature version to be enabled for you. If you need more predictable behavior across releases, reference the particular version you want instead, such as **docker:v1**.

2.6. MISCELLANEOUS CHANGES

2.6.1. Changes to the user representation in both Admin API and Account contexts

Both **org.keycloak.representations.idm.UserRepresentation** and **org.keycloak.representations.account.UserRepresentation** representation classes have changed so that the root user attributes (such as **username**, **email**, **firstName**, **lastName**, and **locale**) have a consistent representation when fetching or sending the representation payload to the Admin and Account APIs, respectively.

The **username**, **email**, **firstName**, **lastName**, and **locale** attributes were moved to a new **org.keycloak.representations.idm.AbstractUserRepresentation** base class.

Also the **getAttributes** method is targeted for representing only custom attributes, so you should not expect any root attribute in the map returned by this method. This method is mainly targeted for clients when updating or fetching any custom attribute for a given user.

In order to resolve all the attributes including the root attributes, a new **getRawAttributes** method was added so that the resulting map also includes the root attributes. However, this method is not available from the representation payload and it is targeted to be used by the server when managing user

profiles.

2.6.2. **https-client-auth** is a build time option

Option **https-client-auth** had been treated as a run time option, however this is not supported by Quarkus. The option needs to be handled at build time instead.

2.6.3. Sequential loading of offline sessions and remote sessions

Starting with this release, the first member of a Red Hat build of Keycloak cluster will load remote sessions sequentially instead of in parallel. If offline session preloading is enabled, those will be loaded sequentially as well.

The previous code led to high resource-consumption across the cluster at startup and was challenging to analyze in production environments and could lead to complex failure scenarios if a node was restarted during loading. Therefore, it was changed to sequential session loading.

For offline sessions, the default in this and previous versions of Red Hat build of Keycloak is to load those sessions on demand, which scales better with a lot of offline sessions than the attempt to preload them in parallel. Setups that use this default setup are not affected by the change of the loading strategy for offline sessions. Setups that have offline session preloading enabled should migrate to a setup where offline-session preloading is disabled.

2.6.4. Infinispan metrics use labels for cache manager and cache names

When enabling metrics for Red Hat build of Keycloak's embedded caches, the metrics now use labels for the cache manager and the cache names.

Old metric example without labels

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

New metric example with labels

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

To revert the change for an installation, use a custom Infinispan XML configuration and change the configuration as follows:

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

2.6.5. User attribute value length extension

As of this release, Red Hat build of Keycloak supports storing and searching by user attribute values longer than 255 characters, which was previously a limitation.

In setups where users are allowed to update attributes, for example, via the account console, prevent denial of service attacks by adding validations. Ensure that no unmanaged attributes are allowed and all editable attributes have a validation that limits the input length.

For unmanaged attributes, the maximum length is 2048 characters. For managed attributes, the default maximum length is 2048 characters. Administrator can change this by adding a validator of type **length**.

WARNING

Red Hat build of Keycloak caches user-related objects in its internal caches. Using longer attributes increases the memory that is consumed by the cache. Therefore, limiting the size of the length attributes is recommended. Consider storing large objects outside Red Hat build of Keycloak and reference them by ID or URL.

This change adds new indexes on the tables **USER_ATTRIBUTE** and **FED_USER_ATTRIBUTE**. If those tables contain more than 300000 entries, Red Hat build of Keycloak will skip the index creation by default during the automatic schema migration and instead log the SQL statement on the console during migration to be applied manually after Red Hat build of Keycloak's startup. See the Upgrading Guide for details on how to configure a different limit.

2.6.5.1. Additional migration steps for LDAP

This is for installations that match all the following criteria:

- User attributes in the LDAP directory are larger than 2048 characters or binary attributes that are larger than 1500 bytes.
- The attributes are changed by admins or users via the admin console, the APIs or the account console.

To be able to enable changing those attributes via UI and REST APIs, perform the following steps:

1. Declare the attributes identified above as managed attributes in the user profile of the realm.
2. Define a **length** validator for each attribute added in the previous step specifying the desired minimum and maximum length of the attribute value. For binary values, add 33 percent to the expected binary length to count in the overhead for Red Hat build of Keycloak's internal base64 encoding of binary values.

2.6.5.2. Additional migration steps for custom user storage providers

This is for installations that match all the following criteria:

- Running MariaDB or MySQL as a database for Red Hat build of Keycloak.
- Entries in table **FED_USER_ATTRIBUTE** with contents in the **VALUE** column that are larger than 2048 characters. This table is used for custom user providers which have federation enabled.
- The long attributes are changed by admins or users via the admin console or the account console.

To be able to enable changing those attributes via UI and REST APIs, perform the following steps:

1. Declare the attributes identified above as managed attributes in the user profile of the realm.

2. Define a **length** validator for each attribute added in the previous step specifying the desired minimum and maximum length of the attribute value.

2.6.6. The Admin send-verify-email API now uses the same email verification template

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

In this release, the API will use the **email-verification.ftl** template instead of **executeActions.ftl**.

Before upgrading

> Perform the following action(s): Verify Email

After upgrading

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

If you have customized the **executeActions.ftl** template to modify how users verify their email using this API, ensure that you transfer your modifications to the new template.

A new parameter called **lifespan** will be introduced to allow overriding of the default lifespan value (12 hours).

If you prefer the previous behavior, use the **execute-actions-email** API as follows.

> PUT /admin/realms/{realm}/users/{id}/execute-actions-email
>
> ["VERIFY_EMAIL"]

2.6.7. Changes to Password Hashing

In this release, we adapted the password hashing defaults to match the OWASP recommendations for Password Storage.

As part of this change, the default password hashing provider has changed from **pbkdf2-sha256** to **pbkdf2-sha512**. Also, the number of default hash iterations for **pbkdf2** based password hashing algorithms changed as follows:

Provider ID	Algorithm	Old Iterations	New Iterations
pbkdf2	**PBKDF2WithHmacSHA1**	20.000	1.300.000
pbkdf2-sha256	**PBKDF2WithHmacSHA256**	27.500	600.000
pbkdf2-sha512	**PBKDF2WithHmacSHA512**	30.000	210.000

If a realm does not explicitly configure a password policy with **hashAlgorithm** and **hashIterations**, then the new configuration will take effect on the next password based login, or when a user password is created or updated.

2.6.7.1. Performance of new password hashing configuration

Tests on a machine with an Intel i9-8950HK CPU (12) @ 4.800GHz yielded the following ø time differences for hashing 1000 passwords (averages from 3 runs). Note that the average duration for the **PBKDF2WithHmacSHA1** was computed with a lower number of passwords due to the long runtime.

Provider ID	Algorithm	Old duration	New duration	Difference
pbkdf2	**PBKDF2WithHmacSHA1**	122ms	3.114ms	+2.992ms
pbkdf2-sha256	**PBKDF2WithHmacSHA256**	20ms	451ms	+431ms
pbkdf2-sha512	**PBKDF2WithHmacSHA512**	33ms	224ms	+191ms

Users of the **pbkdf2** provider might need to explicitly reduce the number of hash iterations to regain acceptable performance. This can be done by configuring the hash iterations explicitly in the password policy of the realm.

2.6.7.2. Expected increased overall CPU usage and temporary increased database activity

The CPU usage per password-based login in our tests increased by the factor of five, which includes both the changed password hashing and unchanged TLS connection handling. The overall CPU increase should be around the factor of two to three due to the averaging effect of Red Hat build of Keycloak's other activities like refreshing access tokens and client credential grants. Still, this depends on the unique workload of an installation.

After the upgrade, during a password-based login, the user's passwords will be re-hashed with the new hash algorithm and hash iterations as a one-off activity and updated in the database. As this clears the user from Red Hat build of Keycloak's internal cache, you will also see an increased read activity on the database level. This increased database activity will decrease over time as more and more user's passwords have been re-hashed.

2.6.7.3. How to keep using the old pbkdf2-sha256 password hashing?

To keep the old password hashing for a realm, specify **hashAlgorithm** and **hashIterations** explicitly in the realm password policy.

- **Hashing Algorithm: pbkdf2-sha256**
- **Hashing Iterations: 27500**

2.6.8. Renaming JPA provider configuration options for migration

After removal of the Map Store the following configuration options were renamed:

- **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. Temporary lockout log replaced with event

There is now a new event **USER_DISABLED_BY_TEMPORARY_LOCKOUT** when a user is temporarily locked out by the brute force protector. The log with ID **KC-SERVICES0053** has been removed as the new event offers the information in a structured form.

As it is a success event, the new event is logged by default at the **DEBUG** level. Use the setting **spi-events-listener-jboss-logging-success-level** as described in the Event listener chapter in the Server Administration Guide to change the log level of all success events.

To trigger custom actions or custom log entries, write a custom event listener as described in the Event Listener SPI in the Server Developer Guide.

2.6.10. Keycloak JS imports might need to be updated

If you are loading Keycloak JS directly from the Red Hat build of Keycloak server, this section can be safely ignored. If you are loading Keycloak JS from the NPM package and are using a bundler like Webpack, Vite, and so on, you might need to make some changes to your code. The Keycloak JS package now uses the **exports** field in the package.json file. This means that you might have to change your imports:

```
// 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. Internal algorithm changed from HS256 to HS512

The algorithm that Red Hat build of Keycloak uses to sign internal tokens (a JWT which is consumed by Red Hat build of Keycloak itself, for example a refresh or action token) is being changed from **HS256** to the more secure **HS512**. A new key provider named **hmac-generated-hs512** is now added for realms. Note that in migrated realms the old **hmac-generated** provider and the old **HS256** key are maintained and still validate tokens issued before the upgrade. The **HS256** provider can be manually deleted when no more old tokens exist following the rotating keys guidelines.

2.6.12. Different JVM memory settings when running in a container

The JVM options **-Xms** and **-Xmx** were replaced by **-XX:InitialRAMPercentage** and **-XX:MaxRAMPercentage** when running in a container. Instead of the static maximum heap size settings, Red Hat build of Keycloak specifies the maximum as 70% of the total container memory.

As the heap size is dynamically calculated based on the total container memory, you should **always set the memory limit** for the container.

WARNING

If the memory limit is not set, the memory consumption rapidly increases as the maximum heap size grows up to 70% of the total container memory.

For more details, see Specifying different memory settings.

2.6.13. Added iss parameter to OAuth 2.0/OpenID Connect Authentication Response

RFC 9207 OAuth 2.0 Authorization Server Issuer Identification specification adds the parameter **iss** in the OAuth 2.0/OpenID Connect Authentication Response for realizing secure authorization responses.

In past releases, we did not have this parameter, but now Red Hat build of Keycloak adds this parameter by default, as required by the specification.

However, some OpenID Connect / OAuth2 adapters, and especially older Red Hat build of Keycloak adapters, may have issues with this new parameter.

For example, the parameter will be always present in the browser URL after successful authentication to the client application. In these cases, it may be useful to disable adding the **iss** parameter to the authentication response. This can be done for the particular client in the Red Hat build of Keycloak Admin console, in client details in the section with **OpenID Connect Compatibility Modes**, described in Section 4.1, "Compatibility with older adapters". Dedicated **Exclude Issuer From Authentication Response** switch exists, which can be turned on to prevent adding the **iss** parameter to the authentication response.

2.6.14. Wildcard characters handling

JPA allows wildcards **%** and _ when searching, while other providers like LDAP allow only *****. As ***** is a natural wildcard character in LDAP, it works in all places, while with JPA it only worked at the beginning and the end of the search string. Starting with this release the only wildcard character is ***** which work consistently across all providers in all places in the search string. All special characters in a specific provider like **%** and _ for JPA are escaped. For exact search, with added quotes e.g. **"w*ord"**, the behavior remains the same as in previous releases.

2.6.15. Changes to the value format of claims mapped by the realm and client role mappers

Before this release, both realm (**User Realm Role**) and client (**User Client Role**) protocol mappers were mapping a stringfied JSON array when the **Multivalued** setting was disabled.

However, the **Multivalued** setting indicates whether the claim should be mapped as a list or, if disabled, only a single value from the same list of values.

In this release, the role and client mappers now map to a single value from the effective roles of a user when they are marked as single-valued (**Multivalued** disabled).

2.6.16. Changes to password fields in Login UI

In this version we want to introduce a toggle to hide/show password inputs.

Affected pages:

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

In general all **<input type="password" name="password" />** are encapsulated within a div now. The input element is followed by a button which toggles the visibility of the password input.

Old code example:

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

New code example:

```
<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 and shell metacharacters

The kc.sh no longer uses an additional shell eval on parameters and the environment variables JAVA_OPTS_APPEND and JAVA_ADD_OPENS, thus the continued use of double escaping/quoting will result in the parameter being misunderstood. For example instead of

```
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
```

Use a single escape:

```
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
```

This change also means you cannot invoke kc.sh using a single quoted value of all arguments. For example you can no longer use

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

-

it must instead be individual arguments

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

Similarly instead of

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

it must instead be individual arguments

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

The usage of individual arguments is also required in Dockerfile run commands.

2.6.18. GroupProvider changes

A new method has been added to allow for searching and paging through top level groups. If you implement this interface you will need to implement the following method:

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

2.6.19. GroupRepresentation changes

- new field **subGroupCount** added to inform client how many subgroups are on any given group
- **subGroups** list is now only populated on queries that request hierarchy data
- This field is populated from the "bottom up" so can't be relied on for getting all subgroups for a group. Use a **GroupProvider** or request the subgroups from **GET {keycloak server}/realms/{realm}/groups/{group_id}/children**

2.6.20. New endpoint for Group Admin API

Endpoint **GET {keycloak server}/realms/{realm}/groups/{group_id}/children** added as a way to get subgroups of specific groups that support pagination

2.6.21. RESTEeasy Reactive

Relying on RESTEasy Classic is not longer an option because it is not available anymore. Migration will be needed for SPI's and code that is relying on RESTEasy Classic and related packages part of **org.jboss.resteasy.spi.***.

2.6.22. Partial export requires manage-realm permission

The endpoint **POST {keycloak server}/realms/{realm}/partial-export** and the corresponding action in the admin console now require **manage-realm** permission for execution instead of **view-realm**. This endpoint exports the realm configuration into a JSON file and the new permission is more appropriate.

The parameters **exportGroupsAndRoles** and **exportClients**, which include the realm groups/roles and clients in the export respectively, continue managing the same permissions (**query-groups** and **view-clients**).

2.6.23. Valid redirect URIs for clients are always compared with exact string matching

Version 1.8.0 introduced a lower-case for the hostname and scheme when comparing a redirect URI with the specified valid redirects for a client. Unfortunately it did not fully work in all the protocols, and, for example, the host was lower-cased for **http** but not for **https**. As OAuth 2.0 Security Best Current Practice advises to compare URIs using exact string matching, Red Hat build of Keycloak will follow the recommendation and for now on valid redirects are compared with exact case even for the hostname and scheme.

For realms relying on the old behavior, the valid redirect URIs for their clients should now hold separate entries for each URI that should be recognized by the server.

Although it introduces more steps and verbosity when configuring clients, the new behavior enables more secure deployments as pattern-based checks are frequently the cause of security issues. Not only due to how they are implemented but also how they are configured.

2.6.24. Fix handling of Groups.getSubGroups briefRepresentation parameter

Version 23.0.0 introduced a new endpoint getSubGroups ("children") on the Groups resource, where the meaning of the parameter briefRepresentation meant the retrieval of full representations of the sub groups. The meaning is now changed to return the brief representation.

2.6.25. Changes in jboss-logging event messages

The **jboss-logging** message values are now quoted (character **"** by default) and sanitized to prevent any line break. There are two new options in the provider (**spi-events-listener-jboss-logging-sanitize** and **spi-events-listener-jboss-logging-quotes**) that allow you to customize the new behavior. For example, to avoid both sanitization and quoting, the server can be started in this manner:

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

For more information about the options, see All provider configuration.

2.7. DEPRECATED AND REMOVED FEATURES

2.7.1. Truststore deprecations

The **spi-truststore-file-*** options and the truststore related options **https-trust-store-*** are deprecated. Therefore, use the new default location for truststore material, **conf/truststores**, or specify your desired paths by using the **truststore-paths** option. For details, see Configuring trusted certificates for outgoing requests.

The **tls-hostname-verifier** property should be used instead of the **spi-truststore-file-hostname-verification-policy** property.

A collateral effect of the changes is that now the truststore provider is always configured with some certificates (at least the default Java trusted certificates are present). This new behavior can affect other parts of Red Hat build of Keycloak.

For example, **webauthn** registration can fail if **attestation conveyance** was configured to **Direct** validation. Previously, if the truststore provider was not configured the incoming certificate was not validated. But now this validation is always performed. The registration fails with **invalid cert path** error as the certificate chain sent by the dongle is not trusted by Red Hat build of Keycloak. The Certificate Authorities of the authenticator need to be present in the truststore provider to correctly perform the attestation.

2.7.2. Deprecated --proxy option

The **--proxy** option has been deprecated and will be removed in a future release. The following table explains how the deprecated option maps to supported options.

Deprecated usage	New usage
kc.sh (no **proxy** option set)	**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** (depending if HTTPS is used)
kc.sh --proxy reencrypt	**kc.sh --proxy-headers forwarded|xforwarded**

NOTE

For hardened security, the **--proxy-headers** option does not allow selecting both **forwarded** and **xforwarded** values at the same time (as it was the case before for **--proxy edge** and **--proxy reencrypt**).

WARNING

When using the proxy headers option, make sure your reverse proxy properly sets and overwrites the **Forwarded** or **X-Forwarded-*** headers respectively. To set these headers, consult the documentation for your reverse proxy. Misconfiguration will leave Red Hat build of Keycloak exposed to security vulnerabilities.

You can also set the proxy headers when using the Operator:

```
apiVersion: k8s.keycloak.org/v2alpha1
kind: Keycloak
metadata:
  name: example-kc
spec:
```

```
...
proxy:
  headers: forwarded|xforwarded
```

NOTE

If the **proxy.headers** field is not specified, the Operator falls back to the previous behavior by implicitly setting **proxy=passthrough** by default. This results in deprecation warnings in the server log. This fallback will be removed in a future release.

2.7.3. Deprecated offline session preloading

The default behavior of Red Hat build of Keycloak is to load offline sessions on demand. The old behavior to preload them at startup is now deprecated, as preloading them at startup does not scale well with a growing number of sessions, and increases Red Hat build of Keycloak memory usage. The old behavior will be removed in a future release.

To re-enable old behavior while it is deprecated and not removed yet, use the feature flag and the SPI option as shown below:

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

The API of **UserSessionProvider** deprecated the method **getOfflineUserSessionByBrokerSessionId(RealmModel realm, String brokerSessionId)**. Instead of this method, use **getOfflineUserSessionByBrokerUserIdStream(RealmModel, String brokerUserId)** to first get the sessions of a user, and then filter by the broker session ID as needed.

2.7.4. Deprecated methods from data providers and models

- **RealmModel#getTopLevelGroupsStream()** and overloaded methods are now deprecated

2.7.5. Deprecations and removals for cookies

As part of refactoring cookie handling in Red Hat build of Keycloak there are some changes to how cookies are set:

- All cookies will now have the secure attribute set if the request is through a secure context
- **WELCOME_STATE_CHECKER** cookies now set **SameSite=Strict**

For custom extensions there may be some changes needed:

- **LocaleSelectorProvider.KEYCLOAK_LOCALE** is deprecated as cookies are now managed through the CookieProvider
- **HttpResponse.setWriteCookiesOnTransactionComplete** has been removed
- **HttpCookie** is deprecated, please use **NewCookie.Builder** instead
- **ServerCookie** is deprecated, please use **NewCookie.Builder** instead

2.7.6. Removal of the deprecated mode for SAML encryption

The compatibility mode for SAML encryption introduced in version 21 is now removed. The system property **keycloak.saml.deprecated.encryption** is not managed anymore by the server. The clients which still used the old signing key for encryption should update it from the new IDP configuration metadata.

2.7.7. Renaming model modules

After removal of the Map Store the following modules were renamed:

- **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**

and **org.keycloak:keycloak-model-legacy** module was deprecated and will be removed in the next release in favour of **org.keycloak:keycloak-model-storage** module.

2.7.8. Removed RegistrationProfile form action

The form action **RegistrationProfile** (displayed in the UI of authentication flows as **Profile Validation**) was removed from the codebase and also from all authentication flows. By default, it was in the built-in registration flow of every realm. The validation of user attributes as well as creation of the user including all that user's attributes is handled by **RegistrationUserCreation** form action and hence **RegistrationProfile** is not needed anymore. There is usually no further action needed in relation to this change, unless you used **RegistrationProfile** class in your own providers.

2.8. PARTIAL UPDATE TO USER ATTRIBUTES WHEN UPDATING USERS THROUGH THE ADMIN USER API IS NO LONGER SUPPORTED

When updating user attributes through the Admin User API, you cannot execute partial updates when updating the user attributes, including the root attributes such as **username**, **email**, **firstName**, and **lastName**.

2.8.1. The deprecated auto-build CLI option was removed

The **auto-build** CLI option has been marked as deprecated for a long time. In this release, it was completely removed, and it is no longer supported.

When executing the **start** command, the server is automatically built based on the configuration. In order to prevent this behavior, set the **--optimized** flag.

2.8.2. Removal of the options to trim the event's details length

Since this release, Keycloak supports long value for **EventEntity** details column. Therefore, it no longer supports options for trimming event detail length **--spi-events-store-jpa-max-detail-length** and **--spi-events-store-jpa-max-field-length**.

2.8.3. Removed namespaces from our translations

We moved all translations into one file for the admin-ui, if you have made your own translations or extended the admin ui you will need to migrate them to this new format. Also if you have "overrides" in your database you'll have to remove the namespace from the keys. Some keys are the same only in different namespaces, this is most obvious to help. In these cases we have postfix the key with **Help**.

If you want you can use this node script to help with the migration. It will take all the single files and put them into a new one and also take care of some of the mapping:

```
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);
  }
});
```

Save this into a file called **transform.mjs** in your **public/locale/<language>** folder and run it with:

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

NOTE

This might not do a complete transform, but very close to it.

CHAPTER 3. UPGRADING THE RED HAT BUILD OF KEYCLOAK SERVER

You upgrade the server before you upgrade the adapters.

3.1. PREPARING FOR UPGRADING

Perform the following steps before you upgrade the server.

Procedure

1. Back up the old installation, such as configuration, themes, and so on.
2. Handle any open transactions and delete the **data/tx-object-store/** transaction directory.
3. Back up the database using instructions in the documentation for your relational database. The database will no longer be compatible with the old server after you upgrade the server. If you need to revert the upgrade, first restore the old installation, and then restore the database from the backup copy.

WARNING

After upgrade of Red Hat build of Keycloak, except for offline user sessions, user sessions are lost. Users will have to log in again.

3.2. DOWNLOADING THE RED HAT BUILD OF KEYCLOAK SERVER

Once you have prepared for the upgrade, you can download the server.

Procedure

1. Download and extract rhbk-24.0.5.zip from the Red Hat build of Keycloak website. After extracting this file, you should have a directory that is named **rhbk-24.0.5**.
2. Move this directory to the desired location.
3. Copy **conf/**, **providers/** and **themes/** from the previous installation to the new installation.

3.3. MIGRATING THE DATABASE

Red Hat build of Keycloak can automatically migrate the database schema, or you can choose to do it manually. By default the database is automatically migrated when you start the new installation for the first time.

3.3.1. Automatic relational database migration

To perform an automatic migration, start the server connected to the desired database. If the database schema has changed for the new server version, the migration starts automatically unless the database has too many records.

For example, creating an index on tables with millions of records can be time-consuming and cause a major service disruption. Therefore, a threshold of **300000** records exists for automatic migration. If the number of records exceeds this threshold, the index is not created. Instead, you find a warning in the server logs with the SQL commands that you can apply manually.

To change the threshold, set the **index-creation-threshold** property, value for the default **connections-liquibase** provider:

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

3.3.2. Manual relational database migration

To enable manual upgrading of the database schema, set the **migration-strategy** property value to "manual" for the default **connections-jpa** provider:

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

When you start the server with this configuration, the server checks if the database needs to be migrated. The required changes are written to the **bin/keycloak-database-update.sql** SQL file that you can review and manually run against the database.

To change the path and name of the exported SQL file, set the **migration-export** property for the default **connections-jpa** provider:

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

For further details on how to apply this file to the database, see the documentation for your relational database. After the changes have been written to the file, the server exits.

3.4. MIGRATING THEMES

If you created custom themes, those themes must be migrated to the new server. Also, any changes to the built-in themes might need to be reflected in your custom themes, depending on which aspects you customized.

Procedure

1. Copy your custom themes from the old server **themes** directory to the new server **themes** directory.
2. Use the following sections to migrate templates, messages, and styles.
   - If you customized any of the updated templates listed in Migration Changes, compare the template from the base theme to check for any changes you need to apply.
   - If you customized messages, you might need to change the key or value or to add additional messages.

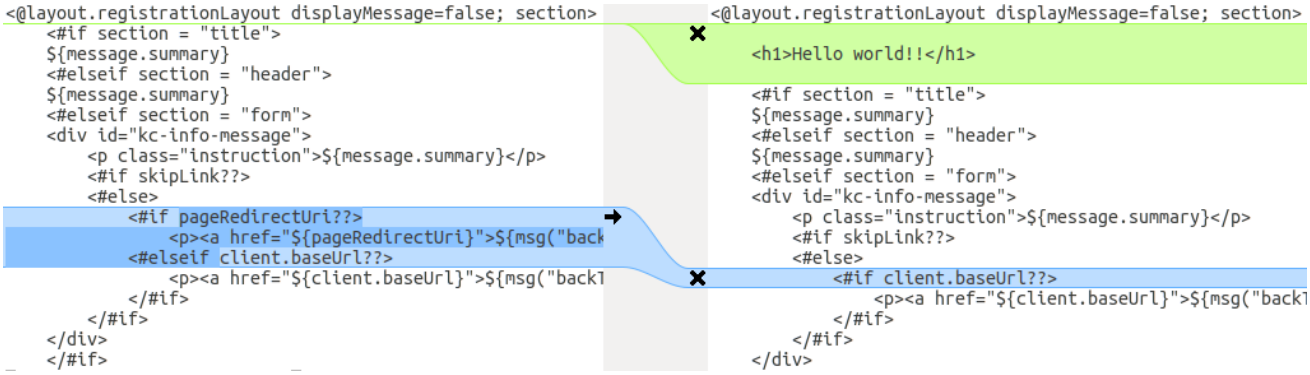
- If you customized any styles and you are extending the Red Hat build of Keycloak themes, review the changes to the styles. If you are extending the base theme, you can skip this step.

3.4.1. Migrating templates

If you customized any template, review the new version to decide about updating your customized template. If you made minor changes, you could compare the updated template to your customized template. However, if you made many changes, consider comparing the new template to your customized template. This comparison will show you what changes you need to make.

You can use a diff tool to compare the templates. The following screenshot compares the **info.ftl** template from the Login theme and an example custom theme:

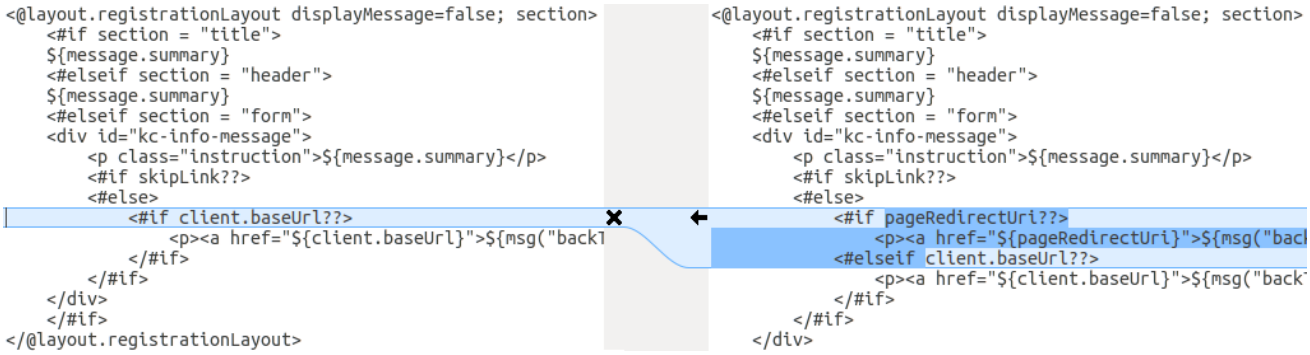
Updated version of a Login theme template versus a custom Login theme template



This comparison shows that the first change (**Hello world!!**) is a customization, while the second change (**if pageRedirectUri**) is a change to the base theme. By copying the second change to your custom template, you have successfully updated your customized template.

In an alternative approach, the following screenshot compares the **info.ftl** template from the old installation with the updated **info.ftl** template from the new installation:

Login theme template from the old installation versus the updated Login theme template



This comparison shows what has been changed in the base template. You can then manually make the same changes to your modified template. Since this approach is more complex, use this approach only if the first approach is not feasible.

3.4.2. Migrating messages

If you added support for another language, you need to apply all the changes listed above. If you have not added support for another language, you might not need to change anything. You need to make changes only if you have changed an affected message in your theme.

Procedure

1. For added values, review the value of the message in the base theme to determine if you need to customize that message.
2. For renamed keys, rename the key in your custom theme.
3. For changed values, check the value in the base theme to determine if you need to make changes to your custom theme.

3.4.3. Migrating styles

You might need to update your custom styles to reflect changes made to the styles from the built-in themes. Consider using a diff tool to compare the changes to stylesheets between the old server installation and the new server installation.

For example:

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

Review the changes and determine if they affect your custom styling.

CHAPTER 4. UPGRADING RED HAT BUILD OF KEYCLOAK ADAPTERS

After you upgrade the Red Hat build of Keycloak server, you can upgrade the adapters. Earlier versions of the adapter might work with later versions of the Red Hat build of Keycloak server, but earlier versions of the Red Hat build of Keycloak server might not work with later versions of the adapter.

4.1. COMPATIBILITY WITH OLDER ADAPTERS

Newer versions of the Red Hat build of Keycloak server potentially work with older versions of the adapters. However, some fixes of the Red Hat build of Keycloak server may break compatibility with older versions of the adapters. For example, a new implementation of the OpenID Connect specification may not match older client adapter versions.

For this situation, you can use Compatibility modes. For OpenID Connect clients, the Admin Console includes **OpenID Connect Compatibility Modes** on the page with client details. With this option, you can disable some new aspects of the Red Hat build of Keycloak server to preserve compatibility with older client adapters. For more details, see the tool tips of individual switches.

4.2. UPGRADING THE EAP ADAPTER

To upgrade the JBoss EAP adapter, complete the following steps:

Procedure

1. Download the new adapter archive.
2. Remove the previous adapter modules by deleting the **EAP_HOME/modules/system/add-ons/keycloak/** directory.
3. Unzip the downloaded archive into **EAP_HOME**.

4.3. UPGRADING THE JAVASCRIPT ADAPTER

To upgrade a JavaScript adapter that has been copied to your web application, perform the following procedure.

Procedure

1. Download the new adapter archive.
2. Overwrite the **keycloak.js** file in your application with the **keycloak.js** file from the downloaded archive.

4.4. UPGRADING THE NODE.JS ADAPTER

To upgrade a **Node.js** adapter that has been copied to your web application, perform the following procedure.

Procedure

1. Download the new adapter archive.

2. Remove the existing **Node.js** adapter directory
3. Unzip the updated file into its place
4. Change the dependency for keycloak-connect in the **package.json** of your application

CHAPTER 5. UPGRADING THE RED HAT BUILD OF KEYCLOAK ADMIN CLIENT

Be sure that you upgrade the Red Hat build of Keycloak server before you upgrade the admin-client. Earlier versions of the admin-client might work with later versions of Red Hat build of Keycloak server, but earlier versions of Red Hat build of Keycloak server might not work with later versions of the admin-client. Therefore, use the admin-client version that matches the current Red Hat build of Keycloak server version.