Red Hat build of Keycloak 26.0

Upgrading Guide

Last Updated: 2025-11-05

Legal Notice

Abstract

This book is a guide to upgrading Red Hat build of Keycloak from 24.0.x to 26.0.17.

Table of Contents

CHAPTER 1. UPGRADING RED HAT BUILD OF KEYCLOAK

This guide describes how to upgrade Red Hat build of Keycloak from version 24.0.x to version 26.0.17. Use the following procedures in this order:

1. Review the release-specific 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.

If you want to upgrade from Red Hat build of Keycloak 22.x, review all the changes in the version 24.0 Upgrading Guide. Then, you can perform the upgrade procedures described in this guide.

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

CHAPTER 2. RELEASE-SPECIFIC CHANGES

2.1. SERVER CONFIGURATION CHANGES

2.1.1. New Hostname options

Hostname v2 options are supported by default, as the old hostname options were removed.

List of necessary migrations:

Old options	New options
hostname <hostname> **hostname-url <url>** **hostname-path <path>** **hostname-port <port>**	**hostname <hostname/url>**
hostname-admin <hostname> **hostname-admin-url <url>**	**hostname-admin <url>**
hostname-strict-backchannel <true/false>	**hostname-backchannel-dynamic <true/false>**

As you can see, the ***-url** suffixes were removed for **hostname** and **hostname-admin** options. Option **hostname** accepts both hostname and URL, but **hostname-admin** accepts only full URL now.

Additionally, there is no way to set **path** or **port** separately. You can achieve it by providing the full URL for the **hostname** and **hostname-admin** options.

If the port is not part of the URL, it is dynamically resolved from the incoming request headers.

HTTPS is no longer enforced unless it is part of **hostname** and **hostname-admin** URLs. If not specified, the used protocol (**http/https**) is dynamically resolved from the incoming request. The **hostname-strict-https** option is removed.

Removed options
hostname-url
hostname-admin-url
hostname-path
hostname-port
hostname-strict-backchannel
hostname-strict-https

2.1.1.1. Examples

Simplified notation

```
# Hostname v1
bin/kc.[sh|bat] start --hostname=mykeycloak.org --https-port=8543 --hostname-path=/auth --
hostname-strict-https=true

# Hostname v2
bin/kc.[sh|bat] start --hostname=https://mykeycloak.org:8543/auth
```

As you can see in the example, all the parts of a URL can be now specified by using a single **hostname** option, which simplifies the hostname setup process. Notice that HTTPS is not enforced by the **hostname-strict-https** option, but by specifying it in the hostname URL.

Backchannel setting

```
# Hostname v1
bin/kc.[sh|bat] start --hostname=mykeycloak.org --hostname-strict-backchannel=true

# Hostname v2
bin/kc.[sh|bat] start --hostname=mykeycloak.org --hostname-backchannel-dynamic=false
```

Be aware that there is a change in behavior if the same URL is to be used for both backend and frontend endpoints. Previously, in hostname v1, the backchannel URL was dynamically resolved from request headers. Therefore, to achieve the required results, you had to specify the **hostname-strict-backchannel=true**.

For hostname v2, the backchannel URLs are already the same as the frontend ones. In order to dynamically resolve it from request headers, you need to set the **hostname-backchannel-dynamic=true** and provide a full URL for the **hostname** option.

For more details and more comprehensive scenarios, see Configuring the hostname (v2).

2.1.2. kcadm and kcreg changes

How kcadm and kcreg parse and handle options and parameters has changed. Error messages from usage errors, the wrong option or parameter, may be slightly different than previous versions. Also usage errors will have an exit code of 2 instead of 1.

2.1.3. Escaping slashes in group paths

Red Hat build of Keycloak has never escaped slashes in the group paths. Because of that, a group named **group/slash** child of **top** uses the full path **/top/group/slash**, which is clearly misleading. Starting with this version, the server can be started to perform escaping of those slashes in the name:

```
bin/kc.[sh|bat] start --spi-group-jpa-escape-slashes-in-group-path=true
```

The escape char is the tilde character ~. The previous example results in the path **/top/group~/slash**. The escape marks the last slash as part of the name and not a hierarchy separator.

The escaping is currently disabled by default because it represents a change in behavior. Nevertheless enabling escaping is recommended and it can be the default in future versions.

2.1.4. --import-realm option can import the master realm

When running a **start** or **start-dev** command with the **--import-realm** option before the master realm exists, it will be imported if it exists in the import material. The previous behavior was that the master realm was created first, then its import skipped.

2.1.5. Additional validations on the --optimized startup option

The **--optimized** startup option now requires the optimized server image to be built first. This can be achieved either by running **kc.sh|bat build** first or by any other server commands (such as **start**, **export**, **import**) without the **--optimized** flag.

2.1.6. Specify cache options at runtime

Options **cache**, **cache-stack**, and **cache-config-file** are no longer build options, and they can be specified only during runtime. This eliminates the need to execute the build phase and rebuild your image due to them. Be aware that they will not be recognized during the **build** phase, so you need to remove them from the **build** phase and add them to the **runtime** phase. If you do not add your current caching options to the **runtime** phase, Red Hat build of Keycloak will fall back to the default caching settings.

2.1.7. Limiting memory usage when consuming HTTP responses

In some scenarios, such as brokering, Red Hat build of Keycloak uses HTTP to talk to external servers. To avoid a denial of service when those providers send too much data, Red Hat build of Keycloak now restricts responses to 10 MB by default.

Users can configure this limit by setting the provider configuration option **spi-connections-http-client-default-max-consumed-response-size**:

Restricting the consumed responses to 1 MB

```
bin/kc.[sh|bat] --spi-connections-http-client-default-max-consumed-response-size=1000000
```

2.1.8. kc.sh/bat import placeholder replacement

The **kc.[sh|bat] import** command now has placeholder replacement enabled. Previously placeholder replacement was only enabled for realm import at startup.

If you wish to disable placeholder replacement for the **import** command, add the system property **-Dkeycloak.migration.replace-placeholders=false**

2.2. HOSTNAME VERIFICATION POLICY

The default for **spi-truststore-file-hostname-verification-policy** and the new **tls-hostname-verifier** option is now DEFAULT, rather than WILDCARD. The WILDCARD and STRICT option values have been deprecated. You should simply rely upon DEFAULT instead.

Behavior supported by WILDCARD, that is not supported by DEFAULT: * allows wildcards in subdomain names (for example, *.foo.com) to match anything, including multiple levels (for example, a.b.foo.com). * allows matching against well known public suffixes – for example, foo.co.gl may match *.co.gl

Behavior supported by STRICT, that is not supported by DEFAULT: * STRICT uses a small exclusion list for 2 or 3 letter domain names ending in a 2 letter top level (*.XXX.YY) when determining if a wildcard matches. Instead DEFAULT uses a more complete list of public suffix rules and exclusions from https://publicsuffix.org/list/

It is not expected that you should be relying upon these behaviors from the WILDCARD or STRICT options.

2.3. PERSISTENT USER SESSIONS

The new feature, **persistent-user-sessions**, stores online user sessions and online client sessions in the database. This change allows a user to stay logged in even if all instances of Red Hat build of Keycloak are restarted or upgraded.

Previous versions of Red Hat build of Keycloak stored only offline user and offline client sessions in the databases. This behavior is identical to previous versions of Red Hat build of Keycloak.

NOTE

When migrating to this version, all existing online user sessions and online client sessions are cleared and the users are logged out. Offline user sessions and offline client sessions are not affected.

2.3.1. Enabling persistent user sessions

In Red Hat build of Keycloak 26, all user sessions are persisted in the database by default. It is possible to revert this behavior to the previous state by disabling the feature. Use the *Volatile user sessions* procedure in the Configuring distributed caches guide.

With persistent sessions enabled, the in-memory caches for online user sessions, offline user sessions, online client sessions and offline client sessions are limited to 10000 entries per node by default, which will reduce the overall memory usage of Keycloak for larger installations. Items which are evicted from memory will be loaded on-demand from the database when needed. Once this feature is enabled, expect a reduced memory usage and an increased database utilization on each login, logout and refresh token request.

To configure the cache size in an external Data Grid in a Red Hat build of Keycloak multi-site setup, see Deploy Data Grid for HA with the Data Grid Operator .

With this feature enabled, the options **spi-user-sessions-infinispan-offline-session-cache-entry-lifespan-override** and **spi-user-sessions-infinispan-offline-client-session-cache-entry-lifespan-override** are no longer available, as they were previously used to override the time offline sessions were kept in-memory.

2.3.2. Signing out existing users

To sign out all online users sessions of a realm when **persistent-user-sessions** is enabled, perform these steps:

1. Log in to the Admin Console.
2. Select the menu entry **Sessions**.
3. Select the action **Sign out all active sessions**.

2.3.3. Restricting the size of session caches

Since the database is now the source of truth for user sessions, it is possible to restrict the size of the session caches to reduce memory usage. If you use the default **conf/cache-ispn.xml** file, the caches for storing user and client sessions are by default configured to store only 10000 sessions and one owner for each entry.

Update the size of the caches using the options **cache-embedded-sessions-max-count**, **cache-embedded-client-sessions-max-count**, **cache-embedded-offline-sessions-max-count** and **cache-embedded-offline-client-sessions-max-count**.

For details about the updated resource requirements, see Concepts for sizing CPU and memory resources.

2.4. METRICS AND HEALTH ENDPOINTS

2.4.1. Metrics for embedded caches enabled by default

Metrics for the embedded caches are now enabled by default. To enable histograms for latencies, set the option **cache-metrics-histograms-enabled** to **true**.

2.4.2. Metrics for HTTP endpoints enabled by default

The metrics provided by Red Hat build of Keycloak now include HTTP server metrics starting with **http_server**. See below for some examples.

```
http_server_active_requests 1.0
http_server_requests_seconds_count{method="GET",outcome="SUCCESS",status="200",uri="/realms/
{realm}/protocol/{protocol}/auth"} 1.0
http_server_requests_seconds_sum{method="GET",outcome="SUCCESS",status="200",uri="/realms/{r
ealm}/protocol/{protocol}/auth"} 0.048717142
```

Use the new options **http-metrics-histograms-enabled** and **http-metrics-slos** to enable default histogram buckets or specific buckets for service level objectives (SLOs). Read more about histograms in the Prometheus documentation about histograms on how to use the additional metrics series provided in **http_server_requests_seconds_bucket**.

2.4.3. Management port for metrics and health endpoints

The /**health** and /**metrics** endpoints are accessible on the management port **9000**, which is turned on by default. That means these endpoints are no longer exposed to the standard Red Hat build of Keycloak ports **8080** and **8443**.

In order to reflect the old behavior, use the property **--legacy-observability-interface=true**, which will not expose these endpoints on the management port. However, this property is deprecated and will be removed in future releases, so it is recommended not to use it.

The management interface uses a different HTTP server than the default Red Hat build of Keycloak HTTP server, and it is possible to configure them separately. Beware, if no values are supplied for the management interface properties, they are inherited from the default Red Hat build of Keycloak HTTP server.

For more details, see Configuring the Management Interface.

2.5. XA CHANGES

2.5.1. XA Transaction Changes

- The option **transaction-xa-enabled** will default to false, rather than true. If you want XA transaction support you will now need to explicitly set this option to true.
- XA Transaction recovery support is enabled by default if **transaction-xa-enabled** is true. Transaction logs will be stored at KEYCLOAK_HOME/data/transaction-logs.

2.5.2. Additional datasources now require using XA

Red Hat build of Keycloak by default does not use XA datasources. However, this is considered unsafe if more than one datasource is used. Starting with this release, you need to use XA datasources if you are adding additional datasources to Red Hat build of Keycloak. If the default datasource supports XA, you can do this by setting the **--transaction-xa-enabled=true** option. For additional datasources, you need to use the **quarkus.datasource.<your-datasource-name>.jdbc.transactions=xa** option in your **quarkus.properties** file. At most one datasource can be non-XA. Recovery isn't supported when you don't have persistent storage for the transaction store.

2.6. OPERATOR CHANGES

2.6.1. Operator no longer defaults to proxy=passthrough

The proxy option has been removed from the server.

2.6.2. Operator scheduling defaults

Red Hat build of Keycloak Pods will now have default affinities to prevent multiple instances from the same CR from being deployed on the same node, and all Pods from the same CR will prefer to be in the same zone to prevent stretch cache clusters.

2.6.3. Operator's default CPU and memory limits/requests

In order to follow the best practices, the default CPU and memory limits/requests for the Operator were introduced. It affects both non-OLM and OLM installs. To override the default values for the OLM install, edit the **resources** section in the operator's subscription.

2.7. API CHANGES

2.7.1. New method in ClusterProvider API

The following method was added to **org.keycloak.cluster.ClusterProvider**:

- **void notify(String taskKey, Collection<? extends ClusterEvent> events, boolean ignoreSender, DCNotify dcNotify)**

When multiple events are sent to the same **taskKey**, this method batches events and just perform a single network call. This is an optimization to reduce traffic and network related resources.

In Red Hat build of Keycloak 26, the new method has a default implementation to keep backward compatibility with custom implementation. The default implementation performs a single network call per an event, and it will be removed in a future version of Red Hat build of Keycloak.

2.7.2. New Java API to search realms by name

The **RealmProvider** Java API now contains a new method **Stream<RealmModel> getRealmsStream(String search)** which allows searching for a realm by name. While there is a default implementation which filters the stream after loading it from the provider, implementations are encouraged to provide this with more efficient implementation.

2.8. EVENT CHANGES

2.8.1. Group-related events no longer fired when removing a realm

With the goal of improving the scalability of groups, they are now removed directly from the database when removing a realm. As a consequence, group-related events such as the **GroupRemovedEvent** are no longer fired when removing a realm.

If you have extensions handling any group-related event when a realm is removed, make sure to use the **RealmRemovedEvent** instead to perform any cleanup or custom processing when a realm, and their groups, are removed.

The **GroupProvider** interface is also updated with a new **preRemove(RealmModel)** method to force implementations to properly handle the removal of groups when a realm is removed.

2.8.2. Changed userId for events related to refresh token

The **userId** in the **REFRESH_TOKEN** event is now always taken from the user session instead of **sub** claim in the refresh token. The **userId** in the **REFRESH_TOKEN_ERROR** event is now always null. The reason for this change is that the value of the **sub** claim in the refresh token may be null with the introduction of the optional **sub** claim or even different from the real user id when using pairwise subject identifiers or other ways to override the **sub** claim.

However a **refresh_token_sub** detail is now added as backwards compatibility to have info about the user in the case of missing userId in the **REFRESH_TOKEN_ERROR** event.

2.9. KEYCLOAK JS

This release includes several changes to Keycloak JS library that should be taken into account. The main motivation for these changes is to de-couple the library from the Red Hat build of Keycloak server, so that it can be refactored independently, simplifing the code and making it easier to maintain in the future. The changes are as follows:

2.9.1. The library is no longer served statically from the server

The Keycloak JS library is no longer served statically from the Red Hat build of Keycloak server. This means that the following URLs are no longer available:

- **/js/keycloak-authz.js**
- **/js/keycloak-authz.min.js**

- **/js/keycloak.js**
- **/js/keycloak.min.js**
- **/js/{version}/keycloak-authz.js**
- **/js/{version}/keycloak-authz.min.js**
- **/js/{version}/keycloak.js**
- **/js/{version}/keycloak.min.js**

Additionally, the **keycloakJsUrl** property that linked to the library on these URLs has been removed from the Admin Console theme. If your custom theme was using this property to include the library, you should update your theme to include the library using a different method.

You should now include the library in your project using a package manager such as NPM. The library is available on the NPM registry as **keycloak-js**. You can install it using the following command:

```
npm install keycloak-js
```

Alternatively, the distribution of the server includes a copy of the library in the **keycloak-js-26.0.0.tgz** archive. You can copy the library from there into your project. If you are using the library directly in the browser without a build, you'll need to host the library yourself. A package manager is still the recommended way to include the library in your project, as it will make it easier to update the library in the future.

2.9.2. Keycloak instance configuration is now required

Previously it was possible to construct a **Keycloak** instance without passing any configuration. The configuration would then automatically be loaded from the server from a **keycloak.json** file based on the path of the included **keycloak.js** script. Since the library is no longer statically served from the server this feature has been removed. You now need to pass the configuration explicitly when constructing a **Keycloak** instance:

```
// Before
const keycloak = new Keycloak();

// After
const keycloak = new Keycloak({
    url: "http://keycloak-server",
    realm: "my-realm",
    clientId: "my-app"
});

// Alternatively, you can pass a URL to a `keycloak.json` file.
// Note this is not reccomended as it creates additional network requests, and is prone to change in
the future.
const keycloak = new Keycloak('http://keycloak-server/path/to/keycloak.json');
```

2.9.3. Methods for login are now async

Keycloak JS now utilizes the Web Crypto API to calculate the SHA-256 digests needed to support PKCE. Due to the asynchronous nature of this API the following public methods will now always return a **Promise**:

- **login()**
- **createLoginUrl()**
- **createRegisterUrl()**

Make sure to update your code to **await** these methods:

```
// Before
keycloak.login();
const loginUrl = keycloak.createLoginUrl();
const registerUrl = keycloak.createRegisterUrl();

// After
await keycloak.login();
const loginUrl = await keycloak.createLoginUrl();
const registerUrl = await keycloak.createRegisterUrl();
```

Make sure to update your code to **await** these methods.

2.9.4. Stricter startup behavior for build-time options

When the provided build-time options differ at startup from the values persisted in the server image during the last optimized Red Hat build of Keycloak build, Red Hat build of Keycloak will now fail to start. Previously, a warning message was displayed in such cases.

2.9.5. New default client scope **basic**

The new client scope named **basic** is added as a realm "default" client scope and hence will be added to all newly created clients. The client scope is also automatically added to all existing clients during migration.

This scope contains preconfigured protocol mappers for the following claims:

- **sub** (See the details below in the dedicated section)
- **auth_time**

This provides additional help to reduce the number of claims in a lightweight access token, but also gives the chance to configure claims that were always added automatically.

2.9.5.1. **sub** claim is added to access token via protocol mapper

The **sub** claim, which was always added to the access token, is now added by default but using a new **Subject (sub)** protocol mapper.

The **Subject (sub)** mapper is configured by default in the **basic** client scope. Therefore, no extra configuration is required after upgrading to this version.

If you are using the **Pairwise subject identifier** mapper to map a **sub** claim for an access token, you can consider disabling or removing the **Subject (sub)** mapper, however it is not strictly needed as the **Subject (sub)** protocol mapper is executed before the **Pairwise subject identifier** mapper and hence the **pairwise** value will override the value added by **Subject (sub)** mapper. This may apply also to other custom protocol mapper implementations, which override the **sub** claim, as the **Subject (sub)** mapper is currently executed as the first protocol mapper.

You can use the **Subject (sub)** mapper to configure the **sub** claim only for access token, lightweight access token, and introspection response. IDToken and Userinfo always contain **sub** claim.

The mapper has no effects for service accounts, because no user session exists, and the **sub** claim is always added to the access token.

2.9.5.2. Nonce claim is only added to the ID token

The nonce claim is now only added to the ID token strictly following the OpenID Connect Core 1.0 specification. As indicated in the specification, the claim is compulsory inside the ID token when the same parameter was sent in the authorization request. The specification also recommends against adding the **nonce** after a refresh request. Previously, the claim was set to all the tokens (Access, Refresh and ID) in all the responses (refresh included).

A new **Nonce backwards compatible** mapper is also included in the software that can be assigned to client scopes to revert to the old behavior. For example, the JS adapter checked the returned **nonce** claim in all the tokens before fixing issue #26651 in version 24.0.0. Therefore, if an old version of the JS adapter is used, the mapper should be added to the required clients by using client scopes.

2.9.5.3. Using older javascript adapter

If you use the latest Red Hat build of Keycloak server with older versions of the javascript adapter in your applications, you may be affected by the token changes mentioned above as previous versions of javascript adapter rely on the claims, which were added by Red Hat build of Keycloak, but not supported by the OIDC specification. This includes:

- Adding the **Session State (session_state)** mapper in case of using the Red Hat build of Keycloak Javascript adapter 24.0.3 or older
- Adding the **Nonce backwards compatible** mapper in case of using a Red Hat build of Keycloak Javascript adapter that is older than Red Hat build of Keycloak 24

You can add the protocol mappers directly to the corresponding client or to some client scope, which can be used by your client applications relying on older versions of the Red Hat build of Keycloak Javascript adapter. Some more details are in the previous sections dedicated to **session_state** and **nonce** claims.

2.10. IDENTITY PROVIDERS CHANGES

2.10.1. Identity Providers no longer available from the realm representation

As part of the improvements around the scalability of realms and organizations when they have many identity providers, the realm representation no longer holds the list of identity providers. However, they are still available from the realm representation when exporting a realm.

To obtain the query the identity providers in a realm, prefer using the **/realms/{realm}/identity-provider/instances** endpoint. This endpoint supports filters and pagination.

2.10.2. Improving performance for selection of identity providers

New indexes were added to the **IDENTITY_PROVIDER** table to improve the performance of queries that fetch the IDPs associated with an organization, and fetch IDPs that are available for login (those that are **enabled**, not **link_only**, not marked as **hide_on_login**).

If the table currently contains more than 300,000 entries, Red Hat build of Keycloak will skip the creation of the indexes by default during the automatic schema migration, and will instead log the SQL statements on the console during migration. In this case, the statements must be run manually in the DB after Red Hat build of Keycloak's startup.

Also, the **kc.org** and **hideOnLoginPage** configuration attributes were migrated to the identity provider itself, to allow for more efficient queries when searching for providers. As such, API clients should use the **getOrganizationId**/**setOrganizationId** and **isHideOnLogin**/**setHideOnLogin** methods in the **IdentityProviderRepresentation**, and avoid setting these properties using the legacy config attributes that are now deprecated.

2.11. OTHER CHANGES

2.11.1. Argon2 password hashing

Argon2 is now the default password hashing algorithm used by Red Hat build of Keycloak in a non-FIPS environment.

Argon2 was the winner of the 2015 password hashing competition and is the recommended hashing algorithm by OWASP.

In Red Hat build of Keycloak 24 the default hashing iterations for PBKDF2 were increased from 27.5K to 210K, resulting in a more than 10 times increase in the amount of CPU time required to generate a password hash. With Argon2, you can achieve better security, with almost the same CPU time as previous releases of Red Hat build of Keycloak. One downside is Argon2 requires more memory, which is a requirement to be resistant against GPU attacks. The defaults for Argon2 in Red Hat build of Keycloak requires 7MB per-hashing request.

To prevent excessive memory and CPU usage, the parallel computation of hashes by Argon2 is by default limited to the number of cores available to the JVM. To support the memory intensive nature of Argon2, we have updated the default GC from ParallelGC to G1GC for a better heap utilization.

Note that Argon2 is not compliant with FIPS 140-2. So if you are in the FIPS environment, the default algorithm will be still PBKDF2. Also note that if you are on non-FIPS environment and you plan to migrate to the FIPS environment, consider changing the password policy to a FIPS compliant algorithm such as **pbkdf2-sha512** at the outset. Otherwise, users will not be able to log in after they switch to the FIPS environment.

2.11.2. Default **http-pool-max-threads** reduced

http-pool-max-threads if left unset will default to the greater of 50 or 4 x (available processors). Previously it defaulted to the greater of 200 or 8 x (available processors). Reducing the number or task threads for most usage scenarios will result in slightly higher performance due to less context switching among active threads.

2.11.3. Improved performance of **findGrantedResources** and **findGrantedOwnerResources** queries

These queries performed poorly when the **RESOURCE_SERVER_RESOURCE** and **RESOURCE_SERVER_PERM_TICKET** tables had over 100k entries and users were granted access to over 1k resources. The queries were simplified and new indexes for the **requester** and **owner** columns were introduced.

The new indexes are both applied to the **RESOURCE_SERVER_PERM_TICKET** table. If the table currently contains more than 300,000 entries, Red Hat build of Keycloak will skip the creation of the indexes by default during the automatic schema migration, and will instead log the SQL statements on the console during migration. In this case, the statements must be run manually in the DB after Red Hat build of Keycloak's startup.

2.11.4. Method getExp added to SingleUseObjectKeyModel

As a consequence of the removal of deprecated methods from **AccessToken**, **IDToken**, and **JsonWebToken**, the **SingleUseObjectKeyModel** also changed to keep consistency with the method names related to expiration values.

The previous **getExpiration** method is now deprecated and you should prefer using new newly introduced **getExp** method to avoid overflow after 2038.

2.11.5. Concurrent login requests are blocked by default when brute force is enabled

If an attacker launched many login attempts in parallel then the attacker could have more guesses at a password than the brute force protection configuration permits. This was due to the brute force check occurring before the brute force protector has locked the user. To prevent this race the Brute Force Protector now rejects all login attempts that occur while another login is in progress in the same server.

If you prefer to disable this feature, use this command:

```
bin/kc.[sh|bat] start --spi-brute-force-protector-default-brute-force-detector-allow-concurrent-
requests=true
```

2.11.6. Changes in redirect URI verification when using wildcards

Because of security concerns, the redirect URI verification now performs an exact string matching (no wildcard involved) if the passed redirect uri contains a **userinfo** part or its **path** accesses the parent directory (**/../**).

The full wildcard ***** can still be used as a valid redirect in development for http(s) URIs with those characteristics. In production environments, configure an exact valid redirect URI without wildcard needs for any URI of that type.

Note that wildcard valid redirect URIs are not recommended for production and not covered by the OAuth 2.0 specification.

2.11.7. Infinispan marshalling changes

Marshalling is the process of converting Java objects into bytes to send them across the network between Red Hat build of Keycloak servers. With Red Hat build of Keycloak 26, the marshalling library has changed from JBoss Marshalling to Infinispan Protostream. The libraries are not compatible between each other and, it requires some steps to ensure the session data is not lost.

WARNING

JBoss Marshalling and Infinispan Protostream are not compatible with each other and incorrect usage may lead to data loss. Consequently, all caches are cleared when upgrading to this version. All existing online user and client sessions are cleared. Offline user and client sessions are not affected.

2.11.8. Automatic redirect from root to relative path

The user is automatically redirected to the path where Red Hat build of Keycloak is hosted when the **http-relative-path** property is specified. It means when the relative path is set to **/auth**, and the user accesses **localhost:8080/**, the page is redirected to **localhost:8080/auth**.

The same change applies to the management interface when the **http-management-relative-path** or **http-relative-path** property is specified. This change improves user experience. Users no longer need to set the relative path to the URL explicitly.

2.11.9. Consistent usage of UTF-8 charset for URL encoding

org.keycloak.common.util.Encode now always uses the **UTF-8** charset for URL encoding instead relying implicitly on the **file.encoding** system property.

2.11.10. Configuring the LDAP Connection Pool

In this release, the LDAP connection pool configuration relies solely on system properties. The main reason is that the LDAP connection pool configuration is a JVM-level configuration rather than specific to an individual realm or LDAP provider instance.

Compared to previous releases, any realm configuration related to the LDAP connection pool will be ignored. If you are migrating from previous versions where any of the following settings are set to your LDAP provider(s), consider using system properties instead:

- **connectionPoolingAuthentication**
- **connectionPoolingInitSize**
- **connectionPoolingMaxSize**
- **connectionPoolingPrefSize**
- **connectionPoolingTimeout**
- **connectionPoolingProtocol**
- **connectionPoolingDebug**

For more details, see Configuring the connection pool.

2.11.11. Persisting revoked access tokens across restarts

In this release, revoked access tokens are written to the database and reloaded when the cluster is restarted by default when using the embedded caches.

To disable this behavior, use the SPI option **spi-single-use-object-infinispan-persist-revoked-tokens** as outlined in All provider configuration.

The SPI behavior of **SingleUseObjectProvider** has changed that for revoked tokens only the methods **put** and **contains** must be used. This is enforced by default, and can be disabled using the SPI option **spi-single-use-object-infinispan-persist-revoked-tokens**.

2.11.12. Highly available multi-site deployments

Red Hat build of Keycloak 26 introduces significant improvements to the recommended high availability multi-site architecture, most notably:

- Red Hat build of Keycloak deployments are now able to handle user requests simultaneously in both sites. Previous load balancer configurations handling requests only in one site at a time will continue to work.
- Active monitoring of the connectivity between the sites is now required to re-configure the replication between the sites in case of a failure. The blueprints describe a setup with Alertmanager and AWS Lambda.
- The loadbalancer blueprint has been updated to use the AWS Global Accelerator as this avoids prolonged fail-over times caused by DNS caching by clients.
- Persistent user sessions are now a requirement of the architecture. Consequently, user sessions will be kept on Red Hat build of Keycloak or Data Grid upgrades.
- External Data Grid request handling has been improved to reduce memory usage and request latency.

As a consequence of the above changes, the following changes are required to your existing Red Hat build of Keycloak deployments.

1. **distributed-cache** definitions provided by a cache configuration file are ignored when the **multi-site** feature is enabled, so you must configure the connection to the external Data Grid deployment via the **cache-remote-*** command line arguments or Keycloak CR as outlined in the blueprints. All **remote-store** configurations must be removed from the cache configuration file.
2. Review your current cache configurations in the external Data Grid and update them with those outlined in the latest version of the Red Hat build of Keycloak's documentation. While previous versions of the cache configurations only logged warnings when the backup replication between sites failed, the new configurations ensure that the state in both sites stays in sync: When the transfer between the two sites fails, the caller will see an error. Due to that, you need to set up monitoring to disconnect the two sites in case of a site failure. The High Availability Guide contains a blueprint on how to set this up.
3. While previous load balancer configurations will continue to work with Red Hat build of Keycloak, consider upgrading an existing Route53 configuration to avoid prolonged failover times due to client side DNS caching.
4. If you have updated your cache configuration XML file with remote-store configurations, those will no longer work. Instead, enable the **multi-site** feature and use the **cache-remove-*** options.

2.11.13. Required actions improvements

The required action provider name is now returned via the **kc_action** parameter when redirecting back from an application initiated required action execution. This eases the detection of which required action was executed for a client. The outcome of the execution can be determined via the **kc_action_status** parameter.

Note: This feature required changes to the Keycloak JS adapter, therefore it is recommended to upgrade to the latest version of the adapter if you want to make use of this feature.

2.11.14. Keystore and trust store default format change

Red Hat build of Keycloak now determines the format of the keystore and trust store based on the file extension. If the file extension is **.p12**, **.pkcs12** or **.pfx**, the format is PKCS12. If the file extension is **.jks**, **.keystore** or **.truststore**, the format is JKS. If the file extension is **.pem**, **.crt** or **.key**, the format is PEM.

You can still override automatic detection by specifying the **https-key-store-type** and **https-trust-store-type** explicitly. The same applies to the management interface and its **https-management-key-store-type**. Restrictions for the FIPS strict mode stay unchanged.

NOTE

The **spi-truststore-file-*** options and the truststore related options **https-trust-store-*** are deprecated, we strongly recommend to use System Truststore. For more details refer to the relevant guide.

2.11.15. Paths for common theme resources have changed

Some of the paths for the **common** resources of the **keycloak** theme have changed, specifically the resources for third-party libraries. Make sure to update your custom themes accordingly:

- **node_modules/patternfly/dist** is now **vendor/patternfly-v3**
- **node_modules/@patternfly/patternfly** is now **vendor/patternfly-v4**
- **node_modules/@patternfly-v5/patternfly** is now **vendor/patternfly-v5**
- **node_modules/rfc4648/lib** is now **vendor/rfc4648**

Additionally, the following resources have been removed from the **common** theme:

- **node_modules/alpinejs**
- **node_modules/jquery**

If you previously used any of the removed resources in your theme, make sure to add them to your own theme resources instead.

2.11.16. BouncyCastle FIPS updated

Our FIPS 140-2 integration is now tested and supported with version 2 of BouncyCastle FIPS libraries. This version is certified with Java 21. If you use FIPS 140-2 integration, it is recommended to upgrade BouncyCastle FIPS library to the versions mentioned in the latest documentation.

The BouncyCastle FIPS version 2 is certified with FIPS 140-3. So Red Hat build of Keycloak can be FIPS 140-3 compliant as long as it is used on the FIPS 140-3 compliant system. This might be the RHEL 9 based system, which itself is compliant with the FIPS 140-3. But note that RHEL 8 based system is only certified for the FIPS 140-2.

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. Shutdown Red Hat build of Keycloak.
2. Back up the old installation, such as configuration, themes, and so on.
3. If XA transaction is enabled, handle any open transactions and delete the **data/transaction-logs/** transaction directory.
4. 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.

NOTE

In case the feature **persistent-user-sessions** is disabled in your current setup and the server is upgraded, all user sessions will be lost except for offline user sessions. Users owning these sessions will have to log in again. Note the feature **persistent-user-sessions** is disabled by default in the Red Hat build of Keycloak server releases prior to 26.0.0.

WARNING

Information about failed logins for the brute force detection and currently ongoing authentication flows is only stored in the internal caches that are cleared when Red Hat build of Keycloak is shut down. Users currently authenticating, changing their passwords or resetting their password will need to restart the authentication flow once Red Hat build of Keycloak is up and running 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-26.0.17.zip from the Red Hat build of Keycloak website.
   After extracting this file, you should have a directory that is named **rhbk-26.0.17**.

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 **connections-liquibase** provider:

```
kc.[sh|bat] start --spi-connections-liquibase-quarkus-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

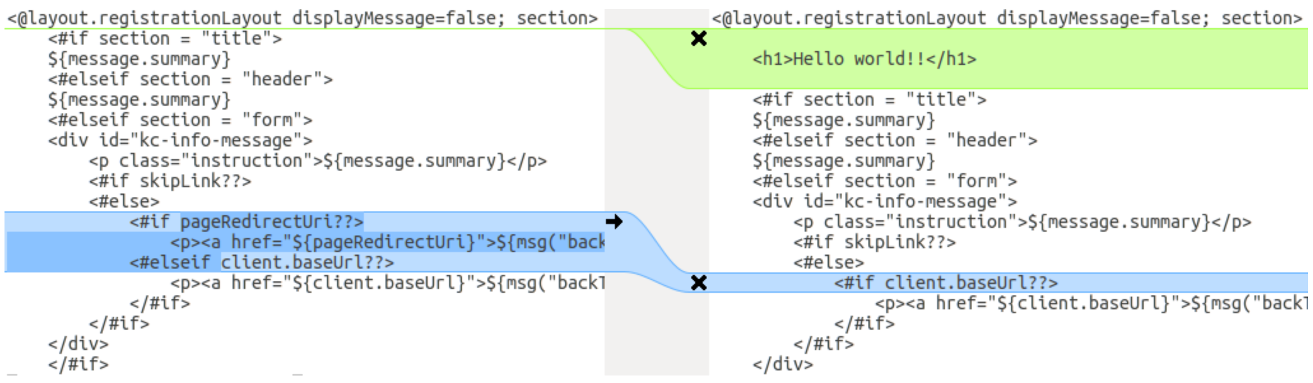
1. Copy your custom themes from the old server **themes** directory to the new server **themes** directory.
2. Use the following sections to migrate templates, messages, and styles.
   - If you customized any of the updated templates listed in Migration Changes, compare the template from the base theme 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



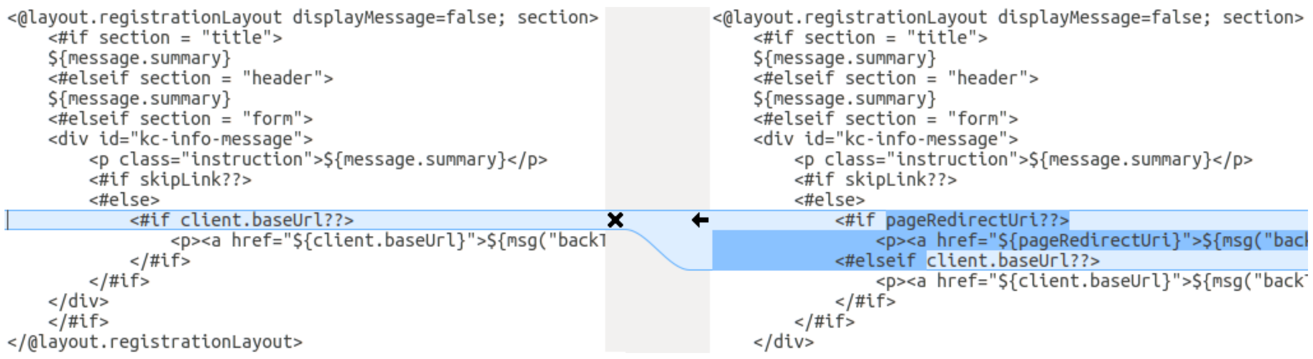
```
<@layout.registrationLayout displayMessage=false; section>
    <#if section = "title">
    ${message.summary}
    <#elseif section = "header">
    ${message.summary}
    <#elseif section = "form">
    <div id="kc-info-message">
        <p class="instruction">${message.summary}</p>
        <#if skipLink??>
        <#else>
            <#if pageRedirectUri??>
                <p><a href="${pageRedirectUri}">${msg("back
            <#elseif client.baseUrl??>
                <p><a href="${client.baseUrl}">${msg("backT
            </#if>
        </#if>
    </div>
    </#if>
```

```
<@layout.registrationLayout displayMessage=false; section>

    <h1>Hello world!!</h1>

    <#if section = "title">
    ${message.summary}
    <#elseif section = "header">
    ${message.summary}
    <#elseif section = "form">
    <div id="kc-info-message">
        <p class="instruction">${message.summary}</p>
        <#if skipLink??>
        <#else>
            <#if client.baseUrl??>
                <p><a href="${client.baseUrl}">${msg("backT
            </#if>
        </#if>
    </div>
```

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



```
<@layout.registrationLayout displayMessage=false; section>
    <#if section = "title">
    ${message.summary}
    <#elseif section = "header">
    ${message.summary}
    <#elseif section = "form">
    <div id="kc-info-message">
        <p class="instruction">${message.summary}</p>
        <#if skipLink??>
        <#else>
            <#if client.baseUrl??>
                <p><a href="${client.baseUrl}">${msg("backT
            </#if>
        </#if>
    </div>
    </#if>
</@layout.registrationLayout>
```

```
<@layout.registrationLayout displayMessage=false; section>
    <#if section = "title">
    ${message.summary}
    <#elseif section = "header">
    ${message.summary}
    <#elseif section = "form">
    <div id="kc-info-message">
        <p class="instruction">${message.summary}</p>
        <#if skipLink??>
        <#else>
            <#if pageRedirectUri??>
                <p><a href="${pageRedirectUri}">${msg("bac
            <#elseif client.baseUrl??>
                <p><a href="${client.baseUrl}">${msg("back
            </#if>
        </#if>
    </div>
```

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 upgrading the Red Hat build of Keycloak server, you upgrade the adapters. Versions of adapters and Red Hat build of Keycloak are now decoupled, meaning that they are released on different schedules. Therefore, use these rules to determine which adapters you upgrade:

- Earlier versions of an adapter might work with later versions of the Red Hat build of Keycloak server.
- Earlier versions of the Red Hat build of Keycloak server might not work with later versions of an adapter.

Each adapter upgrade section provides details on supported adapter versions.

4.1. UPGRADING THE JBOSS EAP SAML ADAPTER

As of Red Hat build of Keycloak 26.0, the JBoss EAP SAML adapter is no longer released with Red Hat build of Keycloak. If you deployed an application with version 6.x or 7.x of that adapter, it is not supported by Red Hat build of Keycloak. Those versions of the adapter are only supported to be used in combination with Red Hat Single Sign-On 7.6.

The fully supported adapter for SAML is the Keycloak SAML Adapter feature pack or RPM for JBoss EAP 8.0.

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

Procedure

1. Remove the previous adapter modules by deleting the **EAP_HOME/modules/system/add-ons/keycloak/** directory.
2. Install the new version of the adapter. For full details, see Installing JBoss EAP by using the RPM installation method.

4.2. UPGRADING THE JBOSS EAP OPENID CONNECT ADAPTER

As of Red Hat build of Keycloak 26.0, the JBoss EAP OpenID connect (OIDC) adapter is no longer released with Red Hat build of Keycloak. This adapter has reached end of life and it is only supported to be used in combination with Red Hat Single Sign-On 7.6. The supported adapter for OIDC is supplied by JBoss EAP 8.0.

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

Procedure

1. Remove the previous adapter modules by deleting the **EAP_HOME/modules/system/add-ons/keycloak/** directory.
2. Install the OIDC client supplied by JBoss EAP 8.0. For details, see Securing Applications with OIDC.

4.3. UPGRADING THE JAVASCRIPT ADAPTER

For this release of Red Hat build of Keycloak, the supported version of this adapter is 26.0.12.

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

Procedure

1. Remove the previous version of the JavaScript adapter.
2. Use these NPM commands to install the 26.0.12 version of the adapter:

```
npm config set @redhat:registry https://npm.registry.redhat.com
install: npm install @redhat/keycloak-js@latest
```

4.4. UPGRADING THE NODE.JS ADAPTER

For this release of Red Hat build of Keycloak, the supported version of this adapter is 26.0.11.

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

Procedure

1. Remove the previous version of the Node.js adapter.
2. Use these NPM commands to install the 26.0.11 version of the Node.js adapter:

```
npm config set @redhat:registry https://npm.registry.redhat.com
npm install @redhat/keycloak-connect@latest
```

3. Change the dependency for keycloak-connect in the **package.json** of your application.

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

The client libraries are those artifacts:

- Java admin client - Maven artifact **org.keycloak:keycloak-admin-client**
- Java authorization client - Maven artifact **org.keycloak:keycloak-authz-client**
- Java policy enforcer - Maven artifact **org.keycloak:keycloak-policy-enforcer**

The client libraries are supported with all the supported Red Hat build of Keycloak server versions. The fact that client libraries are supported with more server versions makes the update easier, so you may not need to update the server at the same time when you update client libraries of your application.

It is possible that client libraries may work even with the older releases of the Red Hat build of Keycloak server, but it is not guaranteed and officially supported.

It may be needed to consult the javadoc of the client libraries like Java admin-client to see what endpoints and parameters are supported with which Red Hat build of Keycloak server version.