Red Hat Single Sign-On Continuous Delivery 7.3.0.cd03

Release Notes

For Use with Red Hat Single Sign-On Continuous Delivery 7.3.0.cd03

Last Updated: 2018-08-30

Red Hat Single Sign-On Continuous Delivery 7.3.0.cd03 Release Notes

For Use with Red Hat Single Sign-On Continuous Delivery 7.3.0.cd03

Legal Notice

Abstract

This guide consists of release notes for Red Hat Single Sign-On Continuous Delivery

Table of Contents

CHAPTER 1. RED HAT SINGLE SIGN-ON CONTINUOUS DELIVERY 7.3.CD03

1.1. HOSTNAME SPI

The hostname SPI introduces a more flexible way to configure the hostname for Red Hat Single Sign-On. There are two built-in providers. The first is request, which uses the request headers to determine the hostname. The second is fixed, which allows configuring a fixed hostname. The latter makes sure that only valid hostnames can be used and also allows internal applications to invoke Red Hat Single Sign-On through an alternative URL.

For more details refer to the threat mitigation section in the Server Administration Guide.

1.2. X509 CLIENT AUTHENTICATOR

The newly added Client Authenticator uses X509 Client Certificates and Mutual TLS to secure a connection from the client. In addition to that the Keycloak Server validates Subject DN field of the client's certificate.

1.3. PERFORMANCE IMPROVEMENTS TO AUTHORIZATION SERVICES

For this release, we improved policy evaluation performance across the board, increasing reliability and throughput. The main changes we did were related with trying to optimize the policy evaluation path by avoiding unnecessary flows and collect decisions as soon as they happen. We also introduced a policy decision cache on a per request basis, avoiding redundant decisions from policies previously evaluated.

We are also working on other layers of cache which should give a much better experience. See KEYCLOAK-7952.

1.4. CHOOSING THE RESPONSE MODE WHEN OBTAINING PERMISSIONS FROM THE SERVER

In previous versions, permissions were always returned from the server using standard OAuth2 response, containing the access and refresh tokens. In this release, clients can use a **response_mode** parameter to specify how the server should respond to an authorization request. This parameter accepts two values:

- **decision**
  Indicating that responses should only contain a flag indicating whether or not permissions were granted by the server. Otherwise a **403** HTTP status code is returned.

- **permissions**
  Indicating that a response should contain every single permission granted by the server using a JSON format.

1.5. NODEJS POLICY ENFORCER

The keycloak-nodejs-connect, an adapter for NodeJS, now supports constructs to protect resources based on decisions taken from the server. The new construct allows users to protect their resources using fine-grained permissions as follows:

```
app.get('/protected/resource', keycloak.enforcer('resource:view'),
function (req, res) {
  res.json({message: 'access granted'});
});
```

1.6. SUPPORT HOSTED DOMAIN FOR GOOGLE LOGINS

Login with Google now supports the **hd** parameter to restrict Google logins to a specific hosted domain at Google. When this is specified in the identity provider any login from a different domain is rejected.

Thanks to brushmate for the contribution.

1.7. ESCAPE UNSAFE TAGS IN HTML OUTPUT

Most HTML output is already escaped for HTML tags, but there are some places where HTML tags are permitted. These are only where admin access is needed to update the value. Even though it would require admin access to update such fields we have added an extra layer of defence and are now escaping unsafe elements like `<script>`.

1.8. BROWSER TAB SUPPORT FOR CORDOVA

We now have support for using browser tab and universal links in the JavaScript adapter for Cordova. This enables SSO between multiple applications as well as increases security.

Thanks to gtudan for the contribution.

1.9. SAML ADAPTER MULTITENANCY SUPPORT

The SAML adapter can support multi-tenancy now just like the built in adapter for OpenID Connect.

1.10. AN OPTION TO CREATE CLAIMS WITH DOTS (.) IN THEM

In previous versions, it was not possible to create claims in the token using a claim name containing a dot (.) character. Now it is possible to escape the dot character in the configuration, so a claim name with the dot character can be used.

1.11. MAKING SPRING BOOT 2 THE DEFAULT STARTER

Starting with release 4.1, the Spring Boot starter will be based on the Spring Boot 2 adapter. If you are using an older Spring Boot version, the keycloak-legacy-spring-boot-starter is available.

CHAPTER 2. RED HAT SINGLE SIGN-ON CONTINUOUS DELIVERY 7.3.CD02

2.1. CLIENT SCOPES AND SUPPORT FOR OAUTH 2 SCOPE PARAMETER

We added support for Client Scopes, which replaces Client Templates. Client Scopes are a more flexible approach and also provides better support for the OAuth **scope** parameter.

There are changes related to Client Scopes to the consent screen. The list on the consent screen is now linked to client scopes instead of protocol mappers and roles.

See the documentation and migration guide for more details.

2.2. OAUTH 2 CERTIFICATE BOUND ACCESS TOKENS

We now have a partial implementation of the specification OAuth 2.0 Mutual TLS Client Authentication and Certificate Bound Access Tokens . More accurately we have support for the Certificate Bound Access Tokens. If your confidential client is able to use 2-way SSL, Red Hat Single Sign-On will be able to add the hash of the client certificate into the tokens issued for the client. At this moment, it's just the Red Hat Single Sign-On itself, which verifies the token hashes (for example during `refresh token` requests). We plan to add support to adapters as well. We also plan to add support for Mutual TLS Client Authentication.

Thanks to tnorimat for the contribution.

2.3. AUTHORIZATION SERVICES

2.3.1. UMA 2.0 Support

UMA 2.0 is now supported for Authorization Services. Check the documentation for more details if you are coming from previous versions of Red Hat Single Sign-On.

2.3.1.1. User-Managed Access through the Red Hat Single Sign-On Account Service

Now end-users are able to manage their resources and the permissions associated with them through the Red Hat Single Sign-On Account Service. From there, resource owners can now check their resources, share resources with another users as well approve requests from other users.

2.3.1.2. Asynchronous Authorization Flow

When using UMA, client applications can now choose whether or not an authorization request should start an authorization flow to ask for the resource owner approval. This functionality allows applications to ask for resource owner approval when trying to access one of his resources on behalf of another user.

2.3.1.3. User-Managed Permission API

Resource servers are now capable of associating additional policies to resources owned by a particular user. The new API provides operations to manage these permissions using different policy types such as role, group, user, client or a condition using JavaScript.

2.3.2. Pushed Claims

Clients applications are now able to send arbitrary claims to Red Hat Single Sign-On along with an authorization request in order to evaluate permissions based on these claims. This is a very handy addition when access should be granted (or denied) in the scope of a specific transaction or based on information about the runtime.

2.3.3. Resource Attributes

It is now possible to associated attributes with resources protected by Red Hat Single Sign-On and use these same attributes to evaluate permissions from your policies.

2.3.4. Policy enforcer now accepts regular access tokens

In some situations, you may want to just send regular access tokens to a resource server but still be able to enforce policies on these resources.

One of the main changes introduced by this release is that you are no longer required to exchange access tokens with RPTs in order to access resources protected by a resource server (when not using UMA). Depending on how the policy enforcer is configured on the resource server side, you can just send regular access tokens as a bearer token and permissions will still be enforced.

2.3.5. Policy enforcer can now load resources from the server on-demand

Until now, when deploying an application configured with a `policy-enforcer`, the policy enforcer would either load all protected paths from the server or just map these paths from the adapter configuration. Users can now decide to load paths on-demand from the server and avoid map these resources in the adapter configuration. Depending on how many protected resources you have this functionality can also improve the time to deploy an application.

2.3.6. Policy enforcer now supports configuring the resource cache

In order to avoid unnecessary hits to the server, the policy enforcer caches the mapping between protected resources and their corresponding paths in your application. Users can now configure the behaviour of the cache or even completely disable it.

2.3.7. Claim Information Points

The `policy-enforcer` definition on the adapters (`keycloak.json`) was also updated to support the concept of pushed claims. There you have the concept of a `claim-information-point` which can be set to push claims from different sources such as the HTTP request or even from an external HTTP service.

2.3.8. Improvements to the Evaluation API

The Evaluation API used to implement policies in Red Hat Single Sign-On, especially JavaScript and Drools policies, provides now methods to:

- Access information from the current realm such as check for user roles, groups and attributes
- Push back arbitrary claims to the resource server in order to provide additional information on how a specific permissions should be enforced

2.4. AUTHORIZATION SERVICES

2.4.1. UMA 2.0

UMA 2.0 is now supported for Authorization Services, including support for users to manage user access through the account management console. There are also other additions and improvements to authorization services.

2.4.2. Pushed Claims

Clients can now push additional claims and have them used by policies when evaluating permissions.

2.4.3. Resource Attributes

It is now possible to define attributes on resources in order to have them used by policies when evaluating permissions.

2.5. THEMES AND THEME RESOURCES

It is now possible to hot-deploy themes to Keycloak through a regular provider deployment. We have also added support for theme resources, which allows adding additional templates and resources without creating a theme. This is useful for custom authenticators that require additional pages to be added to the authentication flow.

We have also added support to override the theme for specific clients. If that is not adequate for your needs, then there is also a new Theme Selector SPI that allows you to implement custom logic to select the theme.

2.6. INSTAGRAM IDENTITY PROVIDER

We have added support to login with Instagram. Thanks to hguerrero for the contribution.

2.7. SEARCH BY USER ID IN ADMIN CONSOLE

To search for a user by id in the admin console you previously had to edit the URL. It is now possible to search directly in the user search field.

2.8. ADAPTERS

2.8.1. Spring Boot 2

We now have support for Spring Boot 2.

2.8.2. Fuse 7

We now have support for Fuse 7.

2.8.3. JavaScript - Native Promise Support

The JavaScript adapter now supports native promises. It retains support for the old style promises as well. Both can be used interchangeably.

2.8.4. JavaScript - Cordova Options

It is now possible to pass Cordova-specific options to login and other methods in the JavaScript adapter. Thanks to loorent for the contribution.