

Session Connect Authenticator

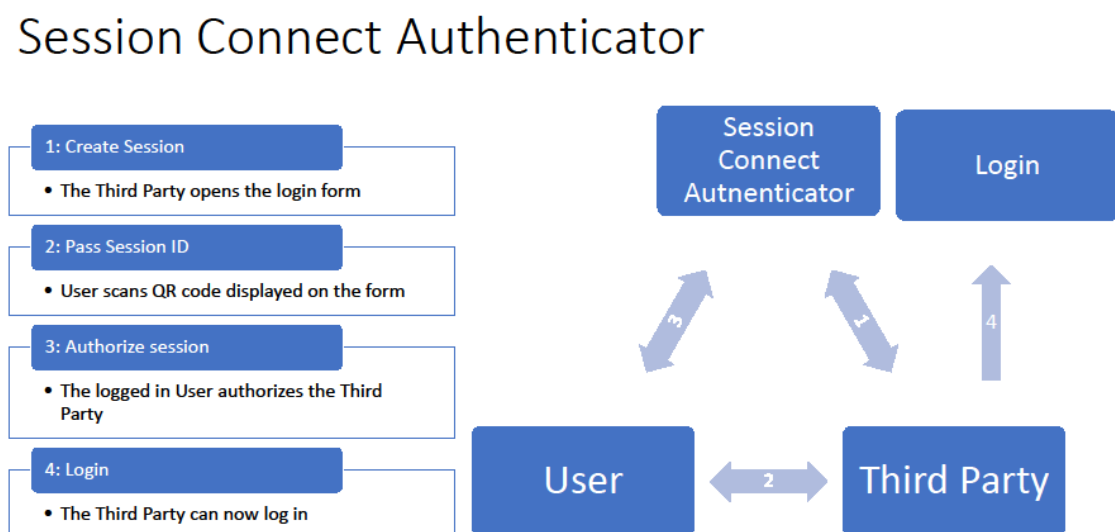
1 GENERAL

The Session Connect Authenticator consists of two JBoss modules, which can be added to a Keycloak server.

The authenticator enables a third party to log in with a user's account, without knowing his username and password. Therefore, the third party calls the authentication website with the query parameter „use_sessionconnect“, e.g. http://mykeycloak.com/auth/realms/master/protocol/openid-connect/auth?client_id=...&redirect_uri=...&response_mode=fragment&response_type=code&scope=openid&use_sessionconnect.

The returned website contains a QR code. The user can scan this code using a special app, whereby he authorizes the third party to log in as him. Hereby, the user's app performs a PUT operation on the REST endpoint <http://mykeycloak.com/auth/realms/master/sessionconnect/<QR-code-content>>. When the third party then clicks the login button, he will be logged in as the user.

The following figure describes, how the Session Connect Authenticator works:



2 USING A CUSTOM FORM

The form, which will be displayed to the Third Party can be configured. We recommend to use your own one, specifically tailored to your application's needs. Therefore, you can create your own Freemaker file named „sessionconnect-form.ftl“. To use this form, you can replace the file „sessionconnect-form.ftl“ in the „ftl“ directory of the Authenticator project with your own one.

The Session Id will be passed to the form under the attribute name „session_id“. Make sure the Session Id will be displayed in a format that can be processed by the user's application.

When submitting the form, the Keycloak server needs to know, which Session Id should be used to log in. Therefore, there must be an input element named „session_id“ containing the Session Id.

The Third Party has to submit this form, after the user connected the displayed Session Id with his account. If you are displaying the form in a web browser, your form should therefore have a submit element. Alternatively, you can embed the form into your own user interface. In this case, you should submit the form programmatically. If you are doing so, do not forget to add the Session Id as an input parameter to the submission.

3 DEPLOYING THE KEYCLOAK SERVER

There are several possibilities to make the Session Connect Authenticator usable on a Keycloak server.

3.1 DEPLOYMENT ON IBM CLOUD

Execute `deploy.cmd` in the Keycloak project.

Make sure, that the version numbers of the authenticator and the webservice in the beginning of the script are equal to the version numbers defined in the `build.gradle` files in the corresponding projects. Furthermore, make sure that the version number in `deployment.yml` is equal to the one defined in `deploy.cmd`.

3.2 DOCKER IMAGE

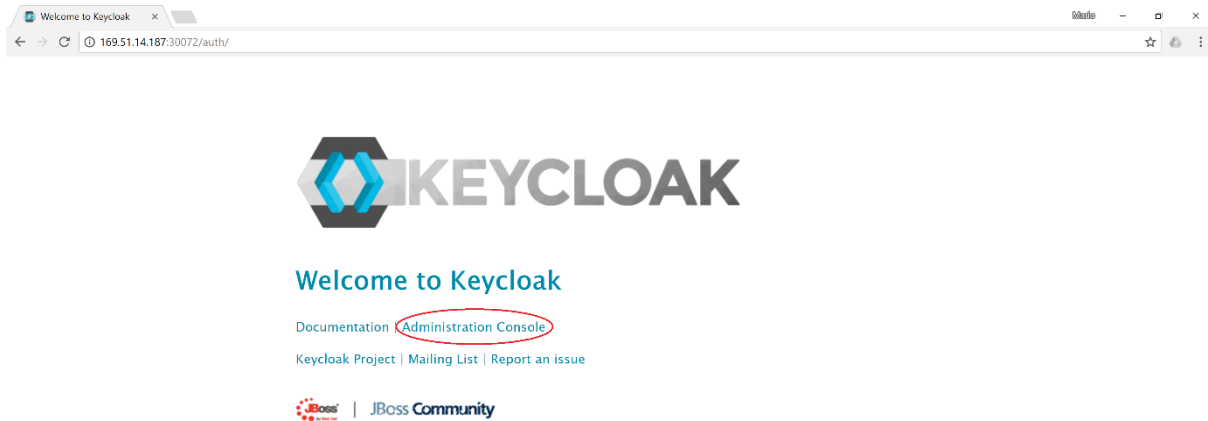
If you want to use a Keycloak server using the Session Connect Authenticator without a Kubernetes cluster on IBM Cloud, you can create and run a Docker image. Make sure, that the version numbers of the authenticator and the webservice in the beginning of the script are equal to the version numbers defined in the `build.gradle` files in the corresponding projects. Then you can build the Authenticator and the Webservice project separately by calling „`gradle build`“ in the corresponding project directories. Now, you have to create a new directory „`module`“ in the Keycloak project and copy the `.jar` files of the builds into this directory, using the names „`Authenticator.jar`“ and „`Webservice.jar`“ respectively. After that, copy the `ftl` directory of the Authenticator into the Keycloak project. If you are using a custom form, replace the `.ftl` files in this directory with your own ones. Now you can create a Docker image using the `Dockerfile` in the Keycloak project.

3.3 EXISTING KEYCLOAK SERVER

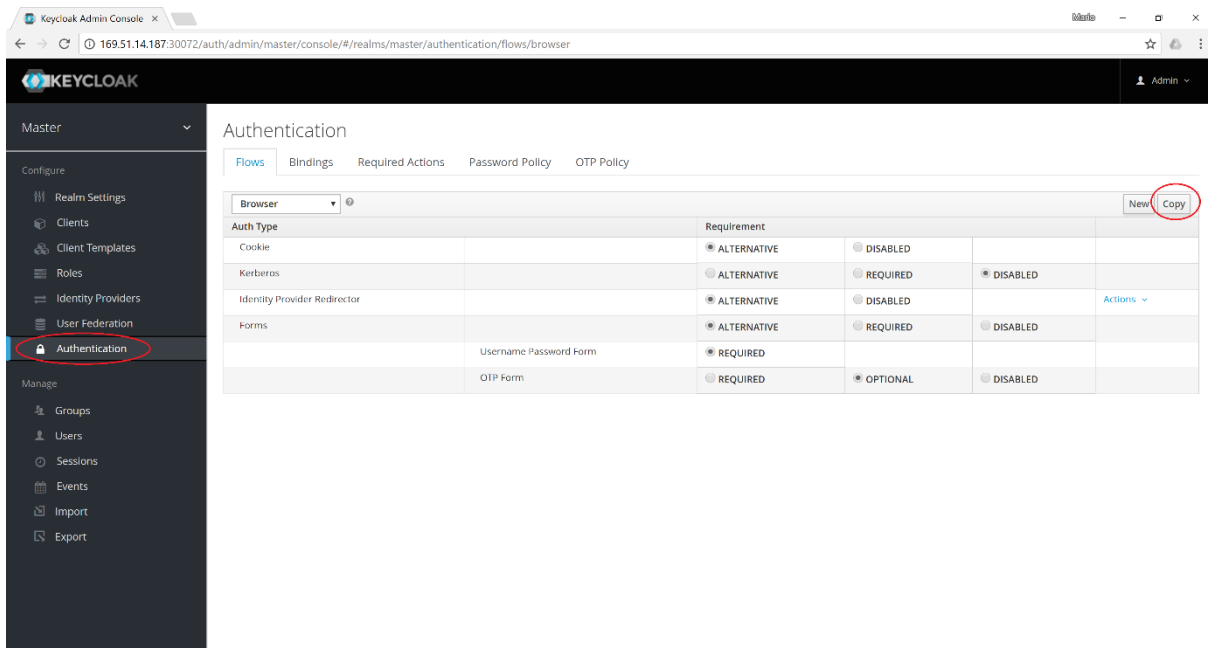
If you already have an existing Keycloak server, or you do not want to use Docker, you can add the Session Connect Authenticator as two JBoss modules to your Keycloak server. Therefore, build the Authenticator and the Webservice separately by calling „`gradle build`“ in their project directories. For each of the two, create a new JBoss module for each of the resulting `.jar` files by using the JBoss CLI. Add the modules to the `standalone.xml` of your JBoss Server. You can find the exact commands in the `Dockerfile` in the Keycloak project.

4 CONFIGURING THE KEYCLOAK SERVER

Open the administration console of your Keycloak server. If you created the server using the deploy script or the Dockerfile, you can log in with „admin“ as username and password:



Now choose the entry „Authentication“ in the left context menu and copy the existing „Browser“ flow. Alternatively, you can define your own flow:



Now add the Session Connect Authenticator as a new execution.

The screenshot shows the Keycloak Admin Console interface. The left sidebar contains a navigation menu with sections for 'Configure' (Realm Settings, Clients, Client Templates, Roles, Identity Providers, User Federation) and 'Authentication' (Groups, Users, Sessions, Events, Import, Export). The main content area is titled 'Authentication' and has tabs for 'Flows', 'Bindings', 'Required Actions', 'Password Policy', and 'OTP Policy'. The 'Flows' tab is active, showing a table of authentication flows. The flow 'Copy Of Browser' is selected, and its details are shown in a table. The 'Add execution' button is circled in red.

Auth Type	Requirement	Actions
Cookie	ALTERNATIVE, DISABLED	Actions
Kerberos	ALTERNATIVE, REQUIRED, DISABLED	Actions
Identity Provider Redirector	ALTERNATIVE, DISABLED	Actions
Copy Of Browser Forms	ALTERNATIVE, REQUIRED, DISABLED	Actions
Username Password Form	REQUIRED	Actions
OTP Form	REQUIRED, OPTIONAL, DISABLED	Actions

The screenshot shows the 'Create Authenticator Execution' page in the Keycloak Admin Console. The left sidebar is the same as the previous screenshot. The main content area has tabs for 'Flows', 'Bindings', 'Required Actions', 'Password Policy', and 'OTP Policy'. The 'Flows' tab is active, and the 'Provider' dropdown menu is open, showing 'Session Connect Authenticator' selected. The 'Save' and 'Cancel' buttons are visible below the dropdown.

Now mark the Session Connect Authenticator as an alternative. This is necessary, because there has to be another authenticator besides this one, which will be used by the user:

The screenshot shows the Keycloak Admin Console interface. On the left is a sidebar with navigation options: Master, Configure, and Manage. The 'Authentication' section is selected under 'Configure'. The main panel displays the 'Authentication Flows' configuration for the 'master' realm. A table lists various authenticators and their requirements. The 'Session Connect Authenticator' is highlighted with a red circle, showing its requirement type as 'ALTERNATIVE'.

Auth Type	Requirement	Requirement Type	Requirement Status	Requirement Priority	Requirement Order	Actions
Cookie	ALTERNATIVE	DISABLED				Actions
Kerberos	ALTERNATIVE	REQUIRED	DISABLED			Actions
Identity Provider Redirector	ALTERNATIVE	DISABLED				Actions
Copy Of Browser Forms	ALTERNATIVE	REQUIRED	DISABLED			Actions
Username Password Form	REQUIRED					Actions
OTP Form	REQUIRED	OPTIONAL	DISABLED			Actions
Session Connect Authenticator	ALTERNATIVE	REQUIRED	OPTIONAL	DISABLED		Actions

Now move the Session Connect Authenticator up in the priority list, so that it is above all alternative executions using a form. If you copied the „Browser“ flow, you have to move it above the execution „Copy of Browser Forms“. This is necessary, because in contrast to the Session Connect Authenticator, other form based authenticators cannot be skipped, so they would block the Session Connect Authenticator.

A user can now log into the Keycloak server using the following username and password: http://<keycloak-server>/auth/realms/master/protocol/openid-connect/auth?client_id=...&redirect_uri=...&response_mode=fragment&response_type=code&scope=openid:

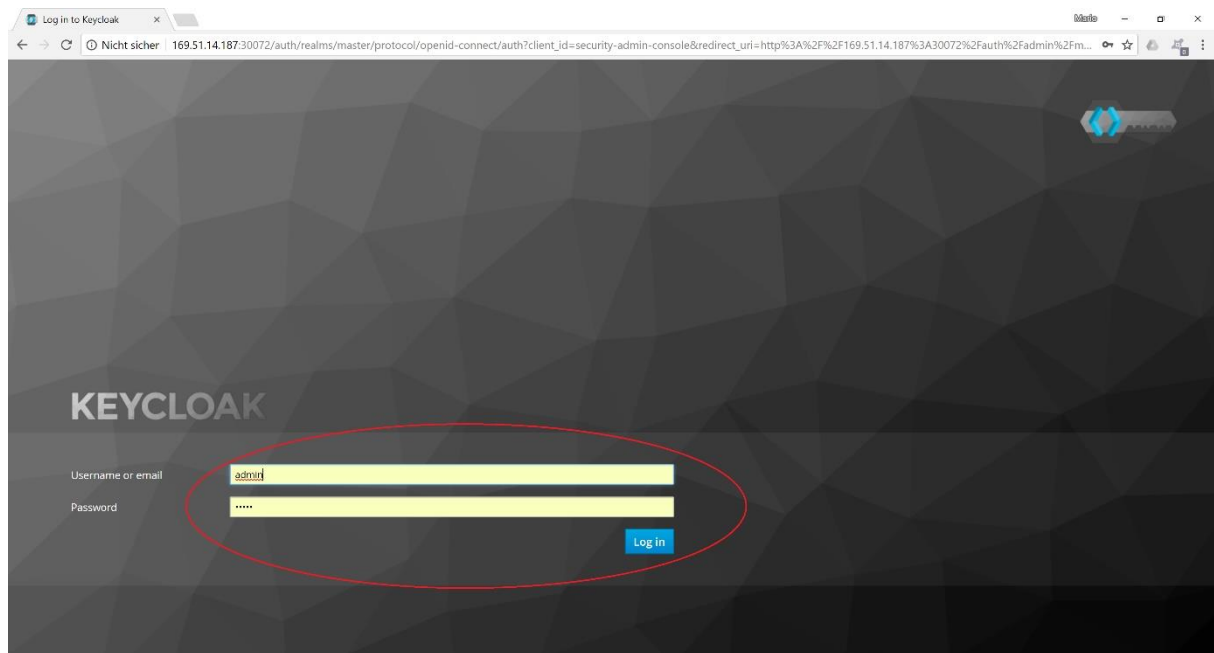


Welcome to Keycloak

[Documentation](#) [Administration Console](#)

[Keycloak Project](#) | [Mailing List](#) | [Report an issue](#)





If a third party wants to log in using the Session Connect Authenticator, he has to add the query parameter „use_sessionconnect“ to the url:

