Delegating the chores of authenticating users to Keycloak

Alexander Schwartz, Principal Software Engineer @ Red Hat

FOSDEM Identity and Access Management Devroom | Brussels, BE| 2025-02-02

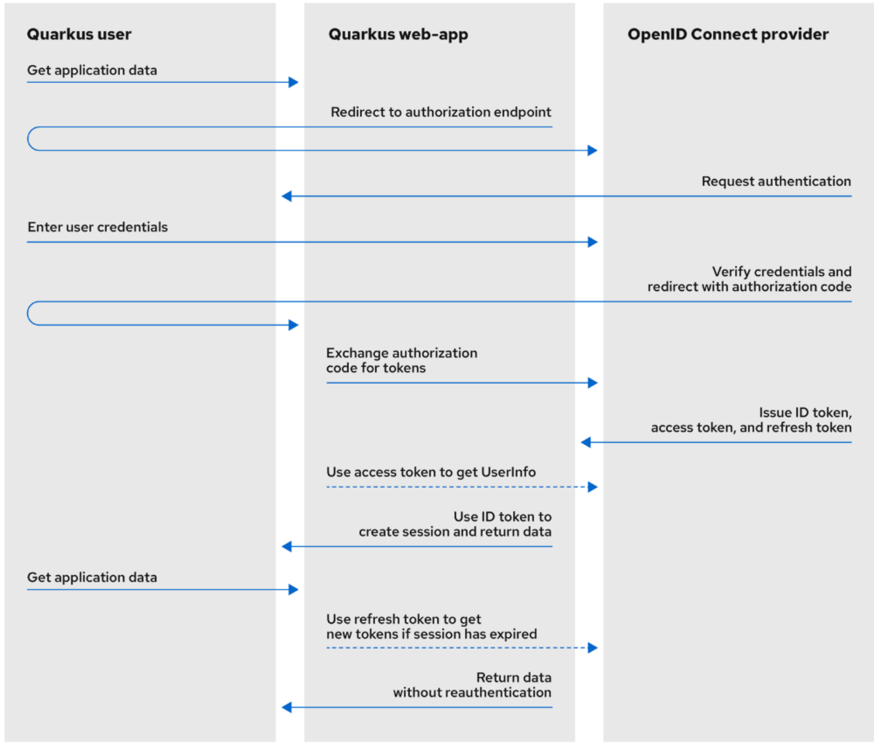
Delegating the chores of authenticating users to Keycloak

Delegating the chores of authenticating users to Keycloak

Motivation

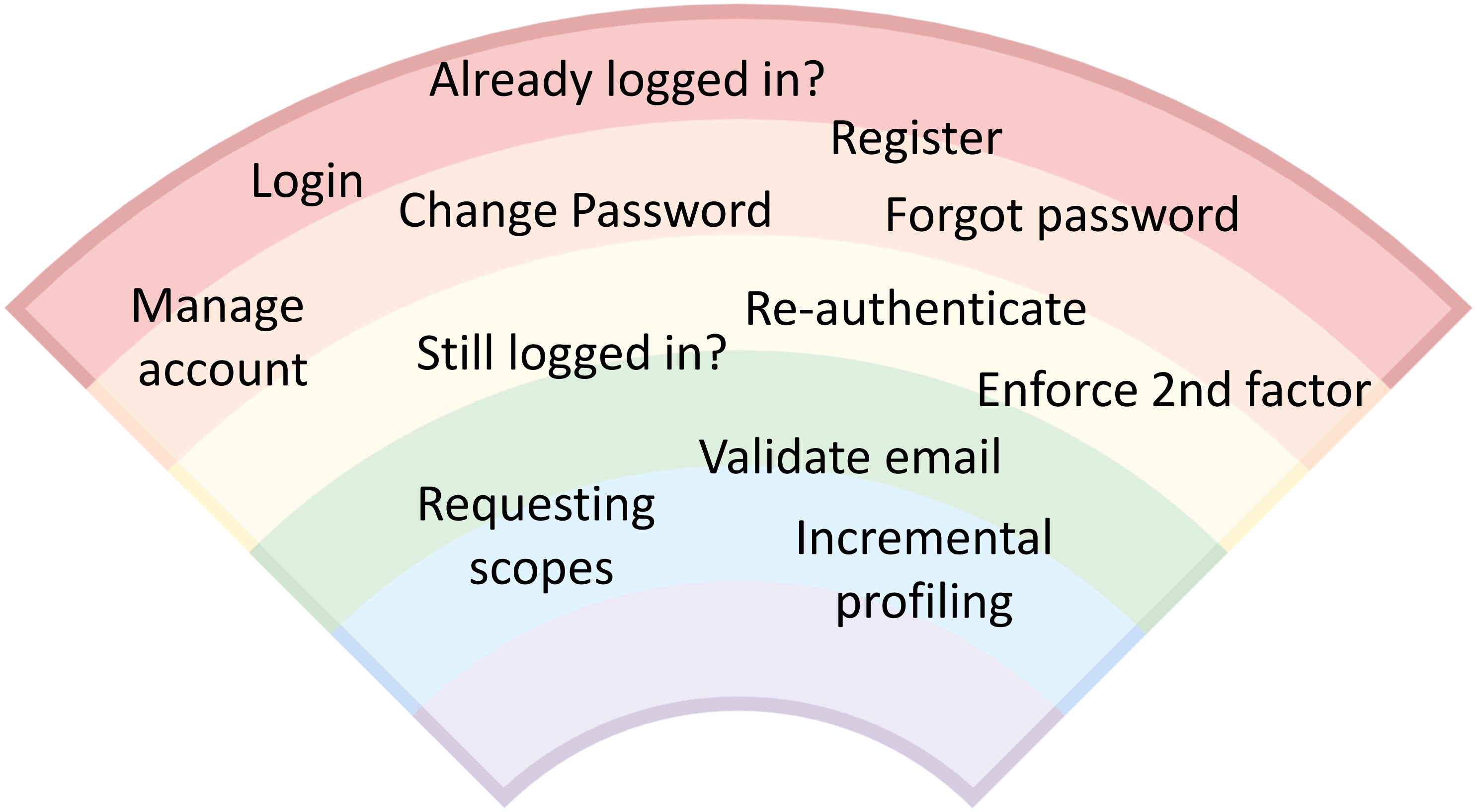
Authentication is answering the question “Who are you?”

- You want users to log in …
  … but it starts earlier as you want to know
  if they are already logged in
- You have seen the diagram of the Authentication Code Flow …
  … but how to I put it to use?
- How to benefit of the features in Keycloak
  … with spending a minimal of work?


Quarkus user
Quarkus web-app
OpenID Connect provider
Get application data
Redirect to authorization endpoint
Request authentication
Enter user credentials
Verify credentials and redirect with authorization code
Exchange authorization code for tokens
Issue ID token, access token, and refresh token
Use access token to get UserInfo
Use ID token to create session and return data
Get application data
Use refresh token to get new tokens if session has expired
Return data without reauthentication

https://quarkus.io/guides/security-oidc-code-flow-authentication

Know the things it can do!



Delegating the chores of authenticating users to Keycloak

1. Motivation
2. **Practical authentication by example**
3. The other things you will also need
4. Standards everywhere!

The actors in this play

End user

- Has Credentials
- Operates a web browser

Relying Party (RP)

aka Client application

- Shows a web application
- Interact with an OpenID Provider and other Relying Parties
- Want a user to authenticate

OpenID Provider (OP)

aka Identity Provider

- Shows the login screen
- Validate credentials
- Issue and validate tokens

Keycloak is an Open Source Identity and Access Management Solution

- 🎂 Initial commit 2013-07-02
- 🏆 Cloud Native Computing Foundation Incubating project since April 2023
- 📜 Apache License, Version 2.0
- ⭐ 26k GitHub stars

Know your OpenID Provider

```
GET issuer + "/.well-known/openid-configuration"
```

```
{
  "issuer": "http://localhost:8080/realms/test",
  "authorization_endpoint": "http://localhost:8080/realms/test/protocol/openid-connect/auth",
  "token_endpoint": "http://localhost:8080/realms/test/protocol/openid-connect/token",
  "introspection_endpoint": "http://localhost:8080/realms/test/protocol/openid-connect/token/introspect",
  "userinfo_endpoint": "http://localhost:8080/realms/test/protocol/openid-connect/userinfo",
  "end_session_endpoint": "http://localhost:8080/realms/test/protocol/openid-connect/logout",
  "frontchannel_logout_session_supported": true,
  "frontchannel_logout_supported": true,
  "jwks_uri": "http://localhost:8080/realms/test/protocol/openid-connect/certs",
  "check_session_iframe": "http://localhost:8080/realms/test/protocol/openid-connect/login-status-iframe.ht
  "grant_types_supported": [
    "authorization_code",
    "implicit",
    "refresh_token",
    "password",
```

Is the user already logged in?

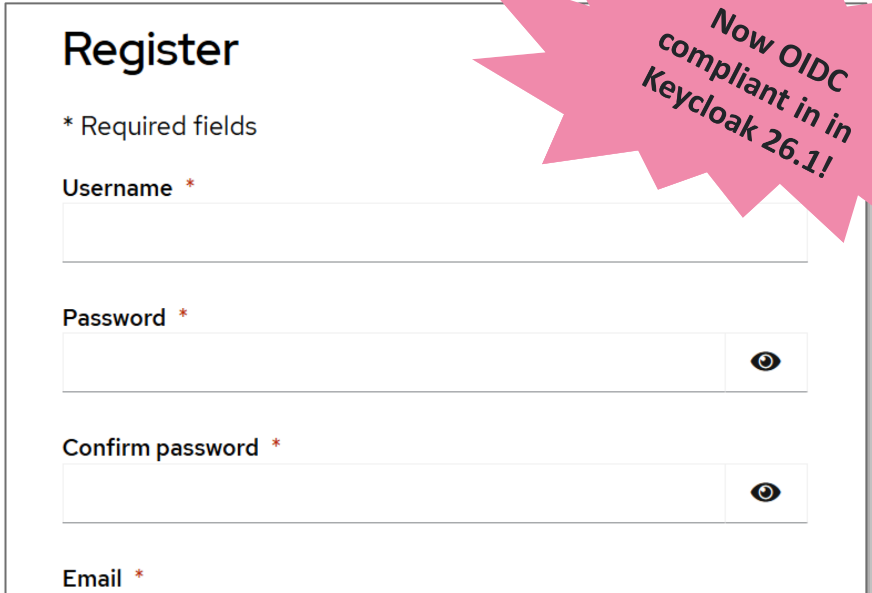
```
REDIRECT TO authorization_endpoint + "?redirect_uri=...&prompt=none..."
```

```
GET ON redirect_uri "?error=login_required..."
```

Register as a new user!

```
REDIRECT TO authorization_endpoint + "?redirect_uri=...&prompt=create..."
```

(continue with a regular login)



Register

* Required fields

Username *

Password *

Confirm password *

Email *

Now OIDC compliant in in Keycloak 26.1!

https://openid.net/specs/openid-connect-prompt-create-1_0.html

Log in the user!

```
REDIRECT TO authorization_endpoint + "?redirect_uri=...&prompt=login..."
```

```
GET ON redirect_uri "?...session_state=...code=..."
```

```
POST code and other parameters to token_endpoint
```

```
RESPONSE with ID token, access token, refresh token, ...
```

Is the user still logged in?

```
IFRAME with check_session_iframe + session_state + JavaScript sendMessage()
```

```
JavaScript receiveMessage() with information if session_state is valid
```

https://openid.net/specs/openid-connect-session-1_0.html

Refresh the access token!

```
POST refresh_token to token endpoint
```

```
RESPONSE with ID token, access token, refresh token, ...
```

Get some information about the user

```
GET userinfo_endpoint with access token as authorization bearer header
```

```
RESPONSE with user information as JSON
```

Log out user from all applications

```
GET end_session_endpoint + "post_logout_redirect_uri=...&id_token_hint=...&client_id=..."
```

```
REDIRECT to post_logout_redirect_uri
```

https://openid.net/specs/openid-connect-rpinitiated-1_0.html

Delegating the chores of authenticating users to Keycloak

Enforce second factor authentication

[Configure a new flow in Keycloak]

```
REDIRECT TO authorization_endpoint + "?...acr_values=2..."
```

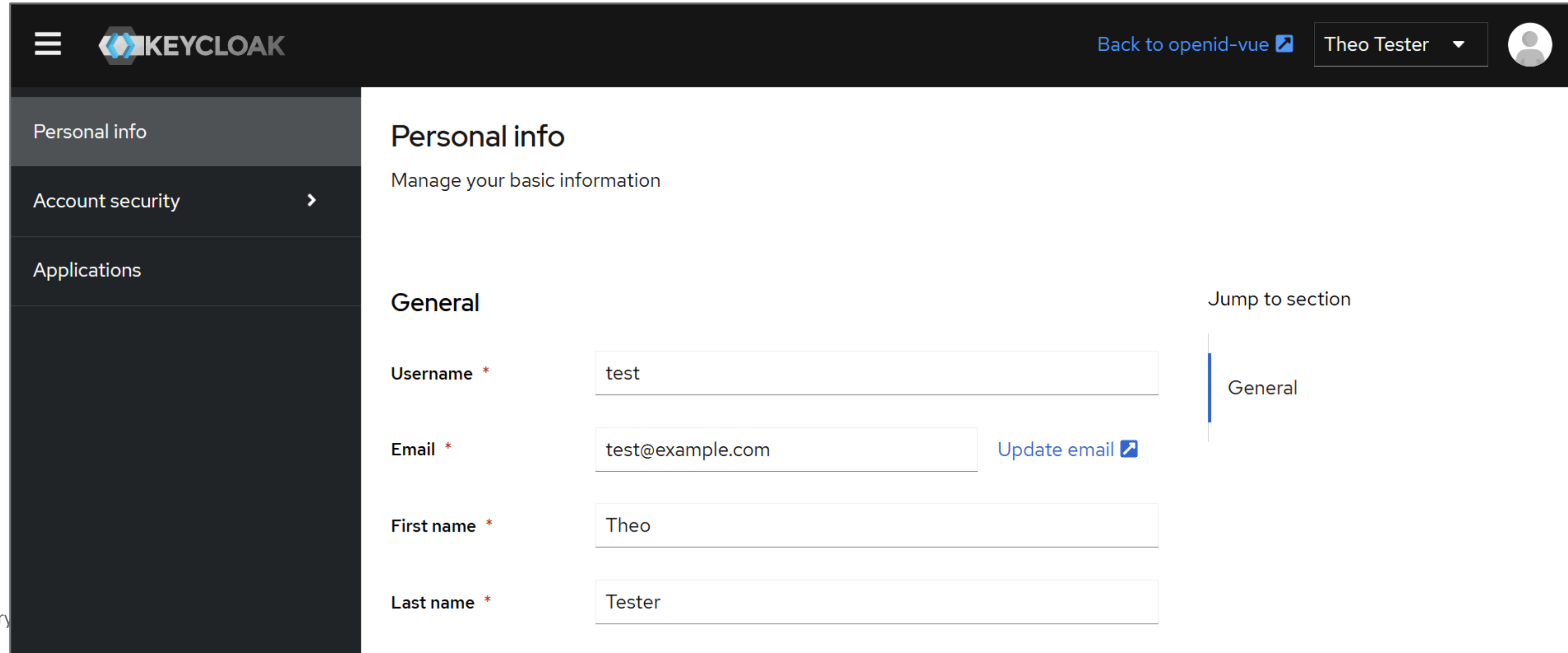
Minimum ACR value in Keycloak Client configuration in 26.1 as an alternative

(continue as with a login)

https://www.keycloak.org/docs/latest/server_admin/#_step-up-flow

Let users manage their data and credentials in Keycloak's account console

```
REDIRECT TO .../account?referrer=...&referrer_uri=...
```



Personal info

Manage your basic information

General

Username *	test
Email *	test@example.com
First name *	Theo
Last name *	Tester

Update your password, add Passkeys or other IDM tasks (Keycloak custom)

```
REDIRECT TO authorization_endpoint + "?kc_action=UPDATE_PROFILE..."
```

```
REDIRECT TO authorization_endpoint + "?kc_action=UPDATE_PASSWORD..."
```

```
REDIRECT TO authorization_endpoint + "?kc_action=delete_account..."
```

```
REDIRECT TO authorization_endpoint + "?kc_action=CONFIGURE_TOTP..."
```

```
REDIRECT TO authorization_endpoint + "?kc_action=webauthn-register..."
```

```
REDIRECT TO authorization_endpoint + "?kc_action=webauthn-register-passwordless..."
```

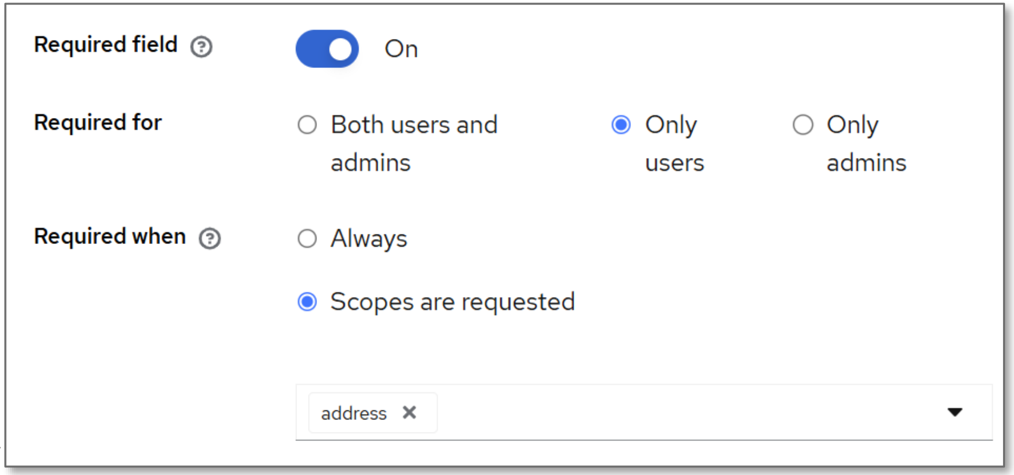
See "Application Initiated Actions" in the Keycloak documentation

https://www.keycloak.org/docs/latest/server_admin/#con-aia_server_administration_guide

Use scopes to acquire additional data

Manage the user profile and make fields profile specific and required

```
REDIRECT TO authorization_endpoint +  &scope=openid+email+address...
```



Required field	On
Required for	Both users and admins [✓ Only users] Only admins
Required when	Always [✓ Scopes are requested]
	address ✕

Delegating the chores of authenticating users to Keycloak

Standards everywhere!

- A lot of authentication and user management functionality is just a redirect away.
- Use an OpenID Connect library to do the heavy lifting.
- Read the standards especially around “prompt”, and leverage modular Keycloak functionality using “kc_action”.
- Use scopes to incrementally acquire user data.
- Try out Keycloak’s preview features and provide feedback, so they can mature and be eventually supported.

KEYCLOAK

https://www.keycloak.org

OpenID® CERTIFIED

Links

Keycloak

https://www.keycloak.org

https://www.keycloak.org/server/features

OpenID Connect Core

https://openid.net/specs/openid-connect-core-1_0.html

Demo Code

https://github.com/ahus1/authentication-demo

JavaScript library used in the demo

https://github.com/panva/openid-client

Slides:

@ahus1@fosstodon.org

@ahus1.de

Contact

Alexander Schwartz
Principal Software Engineer

alexander.schwartz@gmx.net
https://www.ahus1.de

@ahus1.de

@ahus1@fosstodon.org

Reauthenticate when the user is already logged in

```
REDIRECT TO authorization_endpoint + "?redirect_uri=...&prompt=login..."
```

(continue as with a login)

Pushed Autorization Request for the PARanoid!

`POST redirect_uri, prompt` and other information to the `pushed_authorization_request_endpoint`

`RECEIVE a request_uri`

`REDIRECT TO authorization_endpoint + "?request_uri..."`

(continue as before)