Integrasjonsguide UIDP

Innhold

Pålogging via Open ID Connect

Påloggingsapplikasjonens prefererte påloggingsmekanisme er Open ID Connect (OIDC). For dotnet core applikasjoner anbefaler vi å bruke Microsoft.AspNetCore.Authentication

For .NET-applikasjoner anbefaler vi å benytte Microsoft.Owin.Security.OpenIdConnect -biblioteket. Her finner du en øvrig liste over rammeverk og produkter på andre plattformer.

Påkrevd informasjon

Hver applikasjon som skal benytte tjenesten må registreres i påloggingsapplikasjonen, sammen med en del data om applikasjonen. Her er en oversikt

- **ClientId**: denne må registreres både hos påloggingsapplikasjonen og den enkelte applikasjon. Denne identifiserer applikasjonen (client = applikasjon). Påloggingsapplikasjonen har en navnekonvensjon for clientid: "udir.<app>-<miljø>".
- **Redirect URIs**: dette er en liste over URL-er som er gyldige å sende brukeren tilbake til etter autentisering (pålogging). En applikasjon kan i en pålo ggingsforespørsel sette hvilken URL brukeren skal sendes tilbake til etter at autentiseringen er ferdig, men denne URL-en må være forhåndsregistrert i påloggingsapplikasjonen. En applikasjon kan ha flere slike URLer forhåndsregistrert.
- **Post logout Redirect URIs (valgfri):** etter at en bruker har logget ut i påloggingsapplikasjonen, kan applikasjonen be om at brukeren blir sendt tilbake til applikasjonen. Dette gjøres i utloggingsforespørselen, men URLen må være forhåndsregistrert på samme måte som Redirect URIs.
  - **Flow**: OIDC/OAuth2 definerer flere flyter for pålogging. Påloggingsapplikasjonen anbefaler å bruke Authorization code flow med PKCE. Hvis man får problemer med at id_tokenet blir for stort, eller at man ønsker å overføre sensitiv informasjon (som ikke brukeren selv potensielt kan se), kan man bruke authorization code grant
- **LogoutUri**: dette er et endepunkt i applikasjonen som påloggingsapplikasjonen vil påkalle hvis utlogging har blitt igangsatt fra en annen applikasjon. Funksjonaliteten bak dette endepunktet skal være å logge brukeren ut av applikasjonen. Se definisjon her. Påloggingsapplikasjonen vil sende med `?sid=<sid>` i forespørselen, og det anbefales sterkt at applikasjonen sjekker innholdet i denne parameteren opp mot `sid` -claimet som den fikk ved pålogging. Dette for å beskytte brukeren mot clickjacking som forårsaker at brukeren utforvarende blir logget ut.

Gotchas:

- For å kunne kjøre en single sign out er vi avhengige av å kunne kjøre siden din i en iframe.
- Applikasjonen din må kjøre på et 'udir.no"-domene for at du skal kunne gjøre single sign out.
- Dersom du ikke skal ha en Post logout redirect uri er det viktig å sørge for at post_logout_redirect_uri parameteren til 'endsession'-endepunktet er null. I Microsoft.AspNetCore.Authentication blir denne defaultet til 'signout-callback-oidc'. Dette ordner du med å eksplisitt sette SignedOutCallbackPath til null

Scopes

OAuth-protokollen definerer ressurser innenfor ulike scopes. I påloggingsøyemed benyttes disse blant annet for å bestemme hvilken informasjon (hvilke claims) applikasjonen skal motta fra påloggingsapplikasjonen. Hvilke claims og scopes som påloggingsapplikasjonen kjenner til kan du finne i metadata-endepunktet. Ved pålogging må klienten minst etterspørre `openid` -scopet, som ligger til grunn for å utstede et `id_token`. Hvis applikasjonen skal ha tilgang til standard kontaktinformasjon for brukeren, benyttes scopet `profile` (her er en oversikt over hvilke claims som ligger i dette scopet). Det er bare de egenskapene som er markert med "sendes over i ID-token". Resten av brukerinformasjonen må hentes ved å kalle userinfo-endepunktet med bearer token. her kan man lese mer om userinfo-endepunktet. userinfo-endepunktet finnes på "/connect/userinfo". F.eks "https://id.udir.no/connect/userinfo". Hvis man vil ha tilgang til roller må man få opprettet et applikasjonsspesifikt scope

Ressurser, verktøy

- Verktøy for å deserialisere Java Web Tokens (også som Chrome-plugin)

API-tilgang via OAuth

Påloggingsapplikasjonen kan operere som en *Authorization server* ihht. OAuth2-protokollen. Dette kan brukes til å beskytte tilgang til API-er. Dette gjøres ved at man kan få `access_tokens` fra påloggingsapplikasjonen som man benytter for å aksessere et API. Dette kan f.eks. benyttes for å få tilgang til påloggingsapplikasjonens eget API.

For slike scenarioer benyttes en såkalt client credentials grant.

Påkrevd informasjon

Hver applikasjon som skal benytte tjenesten må registreres i påloggingsapplikasjonen, sammen med en del data om applikasjonen. Her er en oversikt

- **ClientId**: denne må registreres både hos påloggingsapplikasjonen og den enkelte applikasjon. Denne identifiserer applikasjonen (client = applikasjon). Påloggingsapplikasjonen har en navnekonvensjon for clientid: "udir.<app>-machine2machine-<miljø>".
- **ClientSecret**: Et "passord" som klienten må bruke for å autentisere seg for påloggingstjenesten. Passordet må være tilstrekkelig sterkt. Applikasjonsteamet kan generere et passord selv, f.eks. med denne tjensten, lage en SHA-256-hash av denne og sende til teamet for påloggingsapplikasjonen.
- **Scopes**: dette er ressurser/API-er man ønsker å få tilgang til. En oversikt over scopes som påloggingsapplikasjonen kjenner til, finner du i metadata-dokumentet. De scopene som er tilgjengelig for API-tilgang (teknisk sett i access_tokens), er de som er prefikset med `urn:udir:` i listen i metadatadokumentet.

Miljøer

DEV: https://uidp-dev.udir.no/

TST: https://uidp-tst.udir.no/

QA: https://uidp-qa.udir.no/

PROD: https://uidp.udir.no/