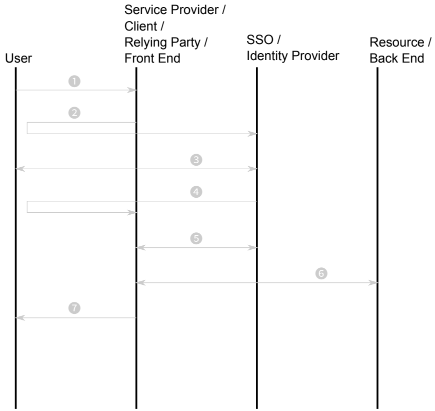
Link ICAT with a Keycloak SSO

Rolf Krahl

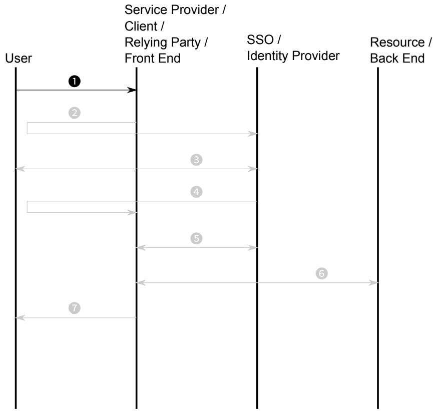
ICAT F2F Meeting, 03 May 2023, Berlin

OpenID Connect Authorization Code Flow



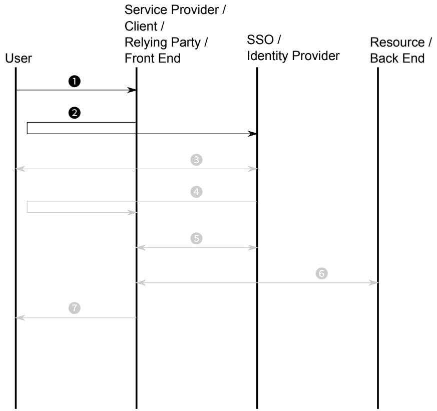
OpenID Connect Authorization Code Flow

1. User make a request to Service



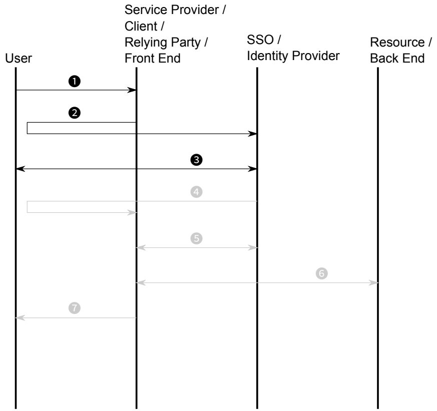
OpenID Connect Authorization Code Flow

1. User make a request to Service
2. Service redirects to SSO



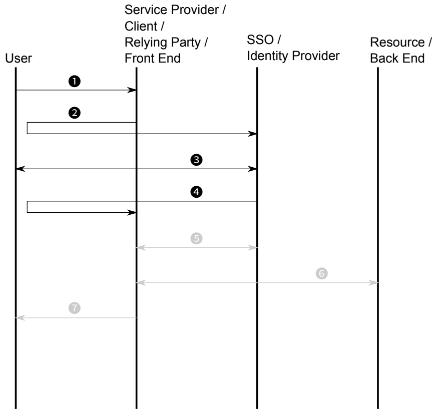
OpenID Connect Authorization Code Flow

1. User make a request to Service
2. Service redirects to SSO
3. User authenticates with SSO



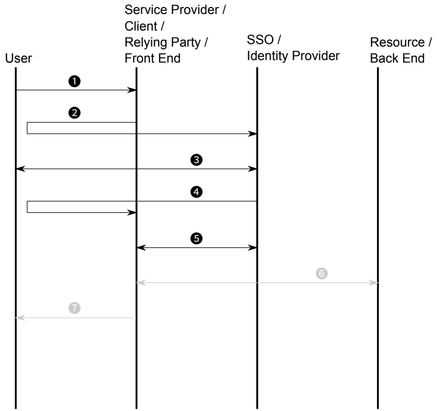
OpenID Connect Authorization Code Flow

1. User make a request to Service
2. Service redirects to SSO
3. User authenticates with SSO
4. SSO redirects back to Service



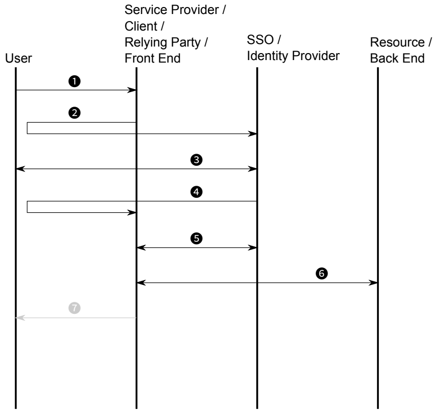
OpenID Connect Authorization Code Flow

1. User make a request to Service
2. Service redirects to SSO
3. User authenticates with SSO
4. SSO redirects back to Service
5. Service gets access token and optionally user info from SSO



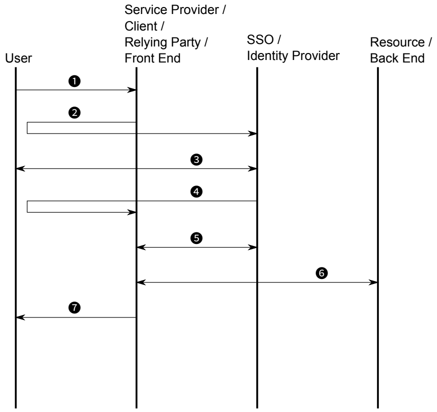
OpenID Connect Authorization Code Flow

1. User make a request to Service
2. Service redirects to SSO
3. User authenticates with SSO
4. SSO redirects back to Service
5. Service gets access token and optionally user info from SSO
6. Service accesses Resource on behalf of User



OpenID Connect Authorization Code Flow

1. User make a request to Service
2. Service redirects to SSO
3. User authenticates with SSO
4. SSO redirects back to Service
5. Service gets access token and optionally user info from SSO
6. Service accesses Resource on behalf of User
7. Service yields some result



Ingredients for an Implementation

We need the following ingredients:

- A running Keycloak: not covered in this talk
- An implementation for the OpenID Connect client:
    - General advice: do not code that yourself! Use existing third party solutions (preferably open source) for that!
    - Here (at HZB): we take advantage that we have ICAT behind an Apache HTTP Server acting as reverse proxy anyway. We may use the `mod_auth_openidc` Apache module.
- Login to ICAT:
    - ICAT provides a specialized OpenID Connect authentication plugin: `authn.oidc`.
    - It takes one single credential key: `token`. The value must be an OpenID Connect access token.
    - As a result you may login to ICAT with:
      `client.login('oidc', {'token': token})`
- A tiny web service script to do the ICAT login for TopCAT

Keycloak Configuration

- Create a Client in Keycloak, set access type to `Confidential`
- Configure the `Valid Redirect URIs`
- Keycloak will generate a random secret for the client that you'll need to copy over to the Apache configuration
- Optional: add a mapper that maps a user attribute `icat_user` to a corresponding token claim

Apache Configuration

- Make sure the module `mod_auth_openidc` gets loaded
- Add some configuration:

```
OIDCProviderMetadataURL \
    http://keycloak:8080/auth/realms/HZB/.well-known/openid-configuration
OIDCClientID icat
OIDCRemoteUserClaim email
OIDCScope "openid␣email"
OIDCClientSecret <secret-generated-by-keycloak>
OIDCCryptoPassphrase <some-random-secret>
```

- This will enable a new AuthType `openid-connect`, so you can do things like:

```
    <Location /auth/>
        ProxyPass !
        AuthType openid-connect
        Require valid-user
    </Location>
```

The module will care for all the Authorization Code Flow (Steps 1–5) and leave the access token in the environment of the request.

ICAT Configuration

- Install and configure `authn.oidc`, follow the install instructions

TopCAT Login Script

We need a web service script that does the ICAT login and puts the session info into the browser storage:

```
@contextlib.contextmanager
def get_icat_client(token):
    preset = dict(configSection='oidc', cred_token=token)
    config = icat.config.Config(ids=False, preset=preset)
    client, conf = config.getconfig()
    try:
        client.login(conf.auth, conf.credentials)
        client.autoLogout = False
        yield client, conf
    finally:
        client.cleanup()

@app.route('/')
def topcat_login():
    try:
        token = request.environ['OIDC_access_token']
    except KeyError:
        abort(403)
    try:
        with get_icat_client(token) as (client, conf):
            session = ICATSession(conf, client)
            return render_template("topcat-login.html",
                                   sessionInfo=session.sessionInfo)
    except icat.ICATSessionError:
        abort(403)
```

Hands-on Demo

. . .

Note

What can you do if you don't use Apache HTTP Server?

- There are plenty of OpenID Connect clients and libraries around
- Install one that fits into your environment
- Make sure it is open source, well maintained and supports Keycloak
- It should run the Authorization Code Flow and yield an access token as a result
- Using the access token, you can login to ICAT with `authn.oidc`