Single-Sign-On with DIKSHA

Trusted, Passwordless authentication with DIKSHA

Overview

DIKSHA allows signed on users on integrated systems to seamlessly navigate to DIKSHA. By design, we keep the systems loosely-coupled, and achieve the seamless navigation by sharing trusted login status. **This keeps the technical complexity for integrating systems simple and the systems do not need to change their own sign-on protocols.**

The philosophy is built around a premise of trust between DIKSHA and the integrated systems. When an authenticated user on integrated system navigates to DIKSHA, the integrated system redirects the browser and sends the details of the user to DIKSHA. This data is signed using a private key of the sender, to allow DIKSHA to trust the incoming data and allow the user to sign in. This ensures that for integrating systems, the change is only how they redirect users to DIKSHA, and not on their internal authentication implementation.

This document describes a protocol to create new users in the DIKSHA and to log them in via the signed token.

Protocol

The authentication protocol provides:

1. Registration of trusted state systems
2. A secure API endpoint to auto-login a user into DIKSHA

The **Integrating State System** (IS) can register itself as a trusted partner with DIKSHA system using the *Registration* process. Later the IS can direct a User to login to DIKSHA and create a new session. To authenticate the User, the IS must send a signed token which contains the user identity and profile information.

Registration

Single sign-on will only be available to registered DIKSHA clients -- a new client will need to register before being able to access this endpoint. In order to register a new client which will authenticate users through a link, the client will need to provide the following information:

- Identifier for partner system (iss), example: *apekx*
- RSA public key

To create a pair of public/private keys, the following command can be used:

```
openssl genrsa -out private.pem 2048
```

This private key must be kept secret. To generate a public key corresponding to the private key, execute:

```
openssl rsa -in private.pem -outform PEM -pubout -out public.pem
```

When the above commands are run, the public key will be written in public.pem, while the private key will be written in private.pem. The private key should be kept with the client and never shared. The public key should be given to DIKSHA integration team to verify the client's signature.

Auto-login

The **Integrating State System** (IS) must redirect users to DIKSHA system by sending a signed JWT token which will be verified by DIKSHA system against the registered public key and allows users into the system.

JWT Tokens

We use JWT (pronounced "jot") in our protocol to generate signed tokens which can be verified by the **Receiving Party** (RP). A JWT is a signed token containing claims from the sender to the recipient. Because the token is signed by the sender, the recipient can be sure that the token has not been tampered. For a more detailed overview of JWTs see: https://jwt.io/introduction/

JWTs are represented in JSON. To authenticate with DIKSHA, a JWT provided to the auto-login endpoint must contain the following fields (also known as claims) in its payload. Unless noted, the fields are

- `jti`: a unique id of the token, can be any string generated by the sender which is unique
- `iss`: the issuer of the JWT, this must be the id of the registered client (see Registration below)
- `sub`: the subject of the token, this must be the userid of the person in the state system who will be logged in to DIKSHA
- `aud`: the consumer of the token, for now this must be <base_url>
- `nbf`: not before, the earliest time when the token can be used (expressed as the number of seconds since epoch in GMT). The `nbf` time cannot be in the future

- `exp`: expires, the timestamp at which the token expires (expressed as the number of seconds since epoch in GMT). The `exp` time cannot be more than 600 seconds after the `nbf` time
- `name`: the name of the person whose userid is in sub
- `state_id`: channel value of this state's rootOrg in DIKSHA system
- `school_id`: the id of the school to which the user belongs to, this must be the id of the school in the state system and the school should be pre-created in DIKSHA system with this id as the external id: *optional*
- `redirect_uri`: the url of the page where the user should be directed after login

Example Payload

```
{
  "jti": "261263cd-3a0e-4aee-8faf-6d9d9eb14bb1",
  "iss": "<replace with id provided by at registration time>",
  "sub": "user_external_id",
  "aud": "<base_url>",
  "iat": 1498556656,
  "exp": 1498560256,
  "name": "Some User",
  "state_id": "state",
  "school_id": "pre_created_school_external_id",
  "redirect_uri": "<base_url>/resources"
}
```

Base Url

The base url for DIKSHA staging will be `https://staging.ntp.net.in`

The base url for DIKSHA production will be `https://diksha.gov.in`

Signature

The payload described above must be signed before sending it the the authentication endpoint. The JWT specification permits signing via HMAC, RSA or ECDSA algorithms. For the purpose of this authentication end-point, the only permitted algorithm will be RSA signing via a private key. The corresponding public key will need to be provided during registration of the client. (for an example of creating a RSA-signed JWT in Java see: https://connect2id.com/products/nimbus-jose-jwt/examples/jwt-with-rsa-signature)

Auto-Login Users

To login a user, the client must provide a link (which makes a GET request) to the auto-login endpoint `/v2/user/session/create?token=<jwt_token>` with the JWT token in the query parameter of the request. Here's an example:

```
<base_url>/v2/user/session/create?token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWV9.TJVA95OrM7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ
```

If a user already exists in DIKSHA system with the given *external id* and within the specified *state*, the user will be automatically logged in. If the user is not found, user will be prompted to provide their *phone number*, which will be used to create the user account in DIKSHA system and then will be logged in.