Department Service Integration with e-Pramaan

OIDC Integration document

Centre for Development of Advanced Computing

Gulmohar Cross Road No. 9, Juhu, Mumbai, 400049,
Telephone: +91 22 2620 1606, +91 22 2620 1574,
Fax: +91 22 2621 0139, +91 22 2623 2195
Website: www.cdac.in

Content

Contents

Revision History

Version	Date	Reason for Change
1.0 (Draft)	10-05-22	
1.1	14-06-22	Updated URLs to epstg.meripehchaan.gov.in
1.2	25-07-22	Updated process flow and parameter descriptions

Abbreviations

OIDC	Open ID Connect
SP	Service Provider (Department)
SSO	Single Sign On
AES	Advanced Encryption Standard
JWT	JSON Web Token

Intended Audience

The recommended audience for this document is the technical personnel responsible for e-Pramaan integration at Department end. This document may be useful for the Project manager/Department Head to assess the effort required for integration with e-Pramaan.

Prerequisite

The integrating person at Department end should be well versed in web application development and familiar with the technology and work flow of the Department Service.

1. Introduction

This document explains the steps involved in integrating Department services with e-Pramaan. It explains the workflow and the step-by-step process for integrating application with e-Pramaan.

2. Process Flow for SSO

Single Sign-On (SSO) is an access control mechanism across multiple independent software systems. This allows user to log in once and gain access to all related services, without being prompted for log in again at each of them. e-Pramaan allows the user to initiate SSO either from Department Service or from e-Pramaan portal. The OIDC protocol is used for SSO implementation.

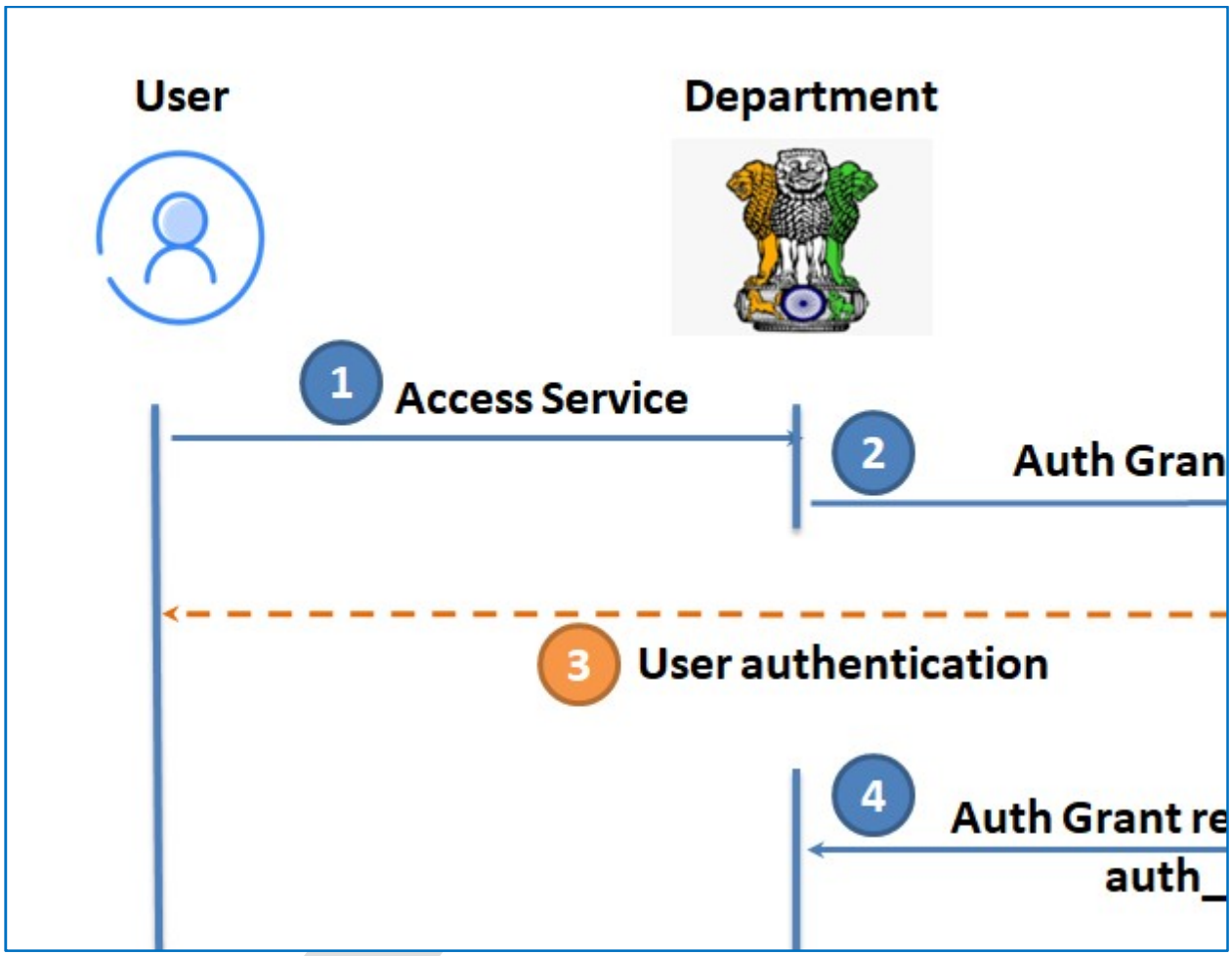

Figure 1: SSO initiated by Department Service

Steps involved in SSO initiated by Department Service are depicted in Figure 1:

- User at Department Service initiates SSO by clicking the option to "Login Using e-Pramaan MeriPehchaan".
- Department Service then sends an *auth grant request* to e-Pramaan and forwards the user to e-Pramaan for authentication.
- User is authenticated by e-Pramaan using Challenge-Response mechanism.
- Once user is authenticated successfully on e-Pramaan, e-Pramaan sends the *auth grant response along with auth_code* to the service.
- Service sends a *token request* to e-Pramaan along with the *auth_code*.
- e-Pramaan checks for the *auth_code* and sends the *token response* to the service.
- Department Service consumes the token and allows user to login.
- If the user fails to authenticate himself / herself on e-Pramaan, the OIDC response returns failure in the auth grant response step.

3. Required steps for integration

The steps involved in integrating an application with e-Pramaan are listed below. Some sample JAVA code snippets are also provided in this document.

- Register the Department Service on e-Pramaan's Department portal.
- Provide link *"Login using e-Pramaan MeriPehchaan"* on the login page of Department Service.
- Modify *onClick event* of *"Login using e-Pramaan MeriPehchaan"* link to authenticate using e-Pramaan.
- Modify/implement logic to *consume SSO Token* sent by e-Pramaan.

The steps are explained in detail below.

Step 1: Register the Department service

- Whitelist your application's public IP and register On e-Pramaan's department portal - https://sp.epramaan.in:4003/
- Once the Department is registered, contact e-Pramaan team for activation of Department.
- Register the services as required. Step 2: Add a link – "Login using e-Pramaan MeriPehchaan"
- In your application, add a button/link named 'Login using e-Pramaan MeriPehchaan'.
- onClick event of "Login using e-Pramaan MeriPehchaan" link, send Auth Grant request to e-Pramaan.

Step 2: Add a link – "*Login using e-Pramaan MeriPehchaan*"

- In your application, add a button/link named *'Login using e-Pramaan MeriPehchaan'.*
- *onClick event* of *"Login using e-Pramaan MeriPehchaan"* link, send Auth Grant request to e-Pramaan.

Step 3: Send Auth Grant request to e-Pramaan (HTTP Redirect POST)

Url: *https://epstg.meripehchaan.gov.in/openid/jwt/processJwtAuthGrantRequest.do*

Following parameters required to be set while creating Authentication Request:

Parameter Name	Description
client_id	Service id sent to department email id at time of registration of service
scope	Must always be constant value openid
state	UUID (Must be unique for every request)
redirect_uri	Callback URL on which service wants to receive auth grant response.
request_uri	The url from which the request originates
response_type	Must always be constant value code
nonce	Create new randomly generated 16 characters string for every request
code_challenge	HMAC SHA256 of code_verifier (code_verifier is randomly generated string of 43 to 128 characters which needs to be created for each call, stored and sent during the token request call)
code-challenge_method	S256
apiHmac	HMAC SHA256 of queryString (serviceId+aesKey+stateID+nonce+redirectionURI+scope+codeChallenge) using Service aes_key.

Sample auth grant request: Method POST

```
https://epstg.meripehchaan.gov.in/openid/jwt/processJwtAuthGrantRequest.do?
&scope=openid
&response_type=code
&redirect_uri=https://epstg.meripehchaan.gov.in/OIDCClient/UDemo
&state=343fb7f4-b3dc-47b3-8f01-613a72eb022e
&code_challenge_method=S256
&nonce=W03PmTz97lpqMnsv43Kl1d5UzZLjJ55kNuh148t2Prs
&client_id=100000909
&code_challenge=DodV_hT3r-TVoNTbKY4rRN4xeMNIfbkVGmXnX3CMhrc
&request_uri=https://epstg.meripehchaan.gov.in/opened/jwt/processJwtAuthGrantRequest.do
&apiHmac=rvSj7XibSYgb1Xzyi5PzP3eBDuR_O0e0i3J9oMfl55E=
```

Step 4: Create API for receiving the auth_code and state

Department Service needs to create an API which will receive the auth_code and state from e-Pramaan. In case of error, e-Pramaan will send the response with 3 error parameters i.e error, error_description and errorUri in the request parameters.

Step 5: Send token request (REST call method - POST)

After receiving the auth_code and state from e-Pramaan, Department Service will then send a Token grant request to e-Pramaan.

Url: *https://epstg.meripehchaan.gov.in/openid/jwt/processJwtTokenRequest.do*

Following parameters required to be set while creating Token Request:

Parameter name	Description
code	Authorization code received in the auth grant request call
grant_type	Must always be constant value authorization_code
scope	Must always be constant value openid
redirect_uri	https://epstg.meripehchaan.gov.in/openid/jwt/processJwtTokenRequest.do
request_uri	Request originating url (Service URL requesting the token)
code_verifier	UUID

Sample Token grant request (REST call method – POST):

```
{"code":["a2906a46-2315-4836-9df4-375afb1ee9b4"],
"grant_type":["authorization_code"],
"scope":["openid"],
"redirect_uri":["https://epstg.meripehchaan.gov.in/openid/jwt/processJwtTokenRequest.do"],
"code_verifier":["t2Hvc0I1An57kT5BoZu60Uvzv5VTf6kFE3cgjl-M5sY"],
"request_uri" : ["https://epstg.meripehchaan.gov.in/UDemo"],
"client_id":["1000XXXX"]}
```

Step 7: Consume Token response

After successful authentication at e-Pramaan the user is redirected to the Department service for which OIDC request was initiated. The user demographic information will be provided by e-Pramaan in the Token grant response. The service has to consume the response and decide the further logic for allowing the user to access desired service. The encrypted JSON Web *Token* (*JWT*) will have to be decrypted and the signature should be verified with the public key shared by e-Pramaan. For decryption and signature verification, appropriate library will have to be used (can be downloaded from https://jwt.io/libraries).

Snapshot of the possible values of JSON Web *Token* (*JWT*):

Sr. no.	Property Name	Description	Datatype	
1	sub	unique user Id	String	Mandatory
2	iat	Token Issued Time	String (datetime in long format)	Mandatory
3	exp	Token Expiry Time	String (datetime in long format)	Mandatory
4	jti	Token Identifier	String	Mandatory
5	name		String	Optional
6	email		String	Optional
7	mobile_number		String	Optional
8	dob	date of birth	String in dd/MM/yyyy	Optional
9	gender		String	Optional
10	house		String	Optional
11	locality		String	Optional
12	pincode		String	Optional
13	district		String	Optional
14	state		String	Optional
15	aadhaar_ref_no	Reference Number	String	Optional
16	sso_id	Same as sub	String	Mandatory
17	session_id		String	Optional

4. Appendix I: Pseudo code

4.1. Creation of Code verifier, code challenge and nonce

As per the technology of the Department Service, a suitable library needs to be used from the https://jwt.io site for OIDC request and response.

```
CodeVerifier codeVerifier= new CodeVerifier();
Nonce nonce= new Nonce();

//Create Code Challenge with the code Verifier
CodeChallenge codeChallenge =
        CodeChallenge.compute(CodeChallengeMethod.S256,codeVerifier);
```

4.2. Pseudo code for creation of API HMAC

```
public static String hashHMACHex(String hMACKey, String inputValue) {

  byte[] keyByte = hMACKey.getBytes(StandardCharsets.US_ASCII);
  byte[] messageBytes = inputValue.getBytes(StandardCharsets.US_ASCII);

  Mac sha256_HMAC = null;
                    sha256_HMAC = Mac.getInstance("HmacSHA256");

  SecretKeySpec secret_key = new SecretKeySpec(keyByte, "HmacSHA256");
                    sha256_HMAC.init(secret_key);
  return Base64.getUrlEncoder()
                    .encodeToString(sha256_HMAC.doFinal(messageBytes));
}
```

Note: HMAC key will be AES key shared by e-Pramaan at the time of service registration and input value will be string of "serviceId+aesKey+stateID+nonce+redirectionURI+scope+codeChallenge"

5. Appendix II: Dot Net code snippets

5.1. Creation of Code verifier, code challenge, nonce and state id

- `Install NuGet package "IdentityModel"`
- Use `System.Security.Cryptography for creation of nonce and verifier`

```
string stateID = Guid.NewGuid().ToString();
string nonce = CryptoRandom.CreateUniqueId(16);
string codeVerifier = CryptoRandom.CreateUniqueId(64);

//Create Code Challenge with the code Verifier
string code_challenge;
        using (var sha256 = SHA256.Create())
        {
           var challengeBytes = sha256.ComputeHash(Encoding.UTF8.GetBytes(codeVerifier));
           code_challenge = IdentityModel.Base64Url.Encode(challengeBytes);
        }
```

5.2. Creation of API HMAC

```
//client_id and aesKey will be service id and aes key respectively
//servce id and aes key will be shared by ePramaan on email after registering the service
public static string client_id = "1XXXXXXXXX9";
public static string aesKey = "cXXXXXXX-XXXX-XXXX-XXXX-XXXXXb3";

//redirect_uri is the success uri which will be entered at the time of registering the service
//redirect_uri  must be same as where auth code will be received
public static string redirect_uri =
"https://localhost:44355/Epramaan/ProcessAuthCodeAndGetToken";

//call this method to create API HMAC
string apiHmac = hashHMACHex(inputvalue, aeskey);

private string hashHMACHex(string message, string secret)
     {
        secret = secret ?? "";
        var encoding = new System.Text.ASCIIEncoding();
        byte[] keyByte = encoding.GetBytes(secret);
        byte[] messageBytes = encoding.GetBytes(message);
        using (var hmacsha256 = new HMACSHA256(keyByte))
        {
           byte[] hashmessage = hmacsha256.ComputeHash(messageBytes);
           return Convert.ToBase64String(hashmessage);
        }
     }
```

Note: aeskey will be AES key and client_id is service id shared by e-Pramaan at the time of service registration, input value will be string of
"client_id + aeskey + stateID + nonce + redirect_uri + scope + code_challenge"

5.3. Creation of final uri for auth code request

```
ViewBag.finalUrl = auth_grant_request_uri + "?&scope=" + scope + "&response_type=" +
response_type + "&redirect_uri=" + redirect_uri + "&state=" + stateID +
"&code_challenge_method=" + code_challenge_method + "&nonce=" + nonce + "&client_id=" +
client_id + "&code_challenge=" + code_challenge + "&request_uri=" + auth_grant_request_uri +
"&apiHmac=" + apiHmac;
```

5.4. Code for creation of Token Request

- Install NuGet package "RestSharp" (version must be <=106.0.0)
- Install NuGet package "jose-jwt"

```
public static readonly string scope = "openid";
public static readonly string response_type = "code";
public static readonly string code_challenge_method = "S256";
public static readonly string grant_type = "authorization_code";
public static string auth_grant_request_uri =
"https://epstg.meripehchaan.gov.in/openid/jwt/processJwtAuthGrantRequest.do";
public static string Certificate =
"G:/Integration/DotNet/OidcDotNetProduction/epramaanprod2016.cer"; // path of the certificate
according to your certificate location

var client = new RestClient(token_request_uri);
var request = new RestSharp.RestRequest(Method.POST);
request.AddHeader("Content-Type", "application/json");

//client_id is shared by e-Pramaan after Service Registration
//authCode is sent back after authentication on the redirect_uri of service
//codeVerifier is the same as created in Auth Grant Request
var body = "{\"code\":[\"" + authCode + "\"],\"grant_type\":[\"" + grant_type +
        "\"],\"scope\":[\"" + scope + "\"],\"redirect_uri\":[\"" + auth_grant_request_uri +
        "\"],\"request_uri\":[\"" + redirect_uri + "\"],\"code_verifier\":[\"" + codeVerifier + "
        \"],\"client_id\":[\"" + client_id + "\"]}";

request.AddParameter("application/json", body, ParameterType.RequestBody);
IRestResponse response = client.Execute(request);
string jwtToken = response.Content;
byte[] secretKey_byte = generateAES256Key(nonce);
var decryptedToken = Jose.JWT.Decode(jwtToken, secretKey_byte);
X509Certificate2 cert = new X509Certificate2(Certificate);
RSACryptoServiceProvider csp = (RSACryptoServiceProvider)cert.PublicKey.Key;
string json = Jose.JWT.Decode(decryptedToken, csp);
```

```
public byte[] generateAES256Key(string seed)
{
        SHA256 sha256 = SHA256CryptoServiceProvider.Create();
        return sha256.ComputeHash(Encoding.UTF8.GetBytes(seed));
}
```