New Saudi Smart ID (SSID) - OIDC Integration Guide

May 17, 2024

Change History

Version	Author	Summary of Changes	Date
V1	Emroualqais Abdelrahim	Initial Draft	17/05/2024

Table of Contents

Introduction

This guide offers comprehensive instructions for integrating OpenID Connect (OIDC) authentication with the New SSID. OIDC is an identity layer built on top of the OAuth 2.0 protocol, enabling clients to verify the end-user's identity based on authentication conducted by an authorization server.

Environment

Environment	Name	URL
PROD	KC_URL	https://sso.visitsaudi.com/idp
UAT	KC_URL	https://rv-uat-ssid.visitsaudi.com/idp
PROD	BE_URL	https://sso.visitsaudi.com/api
UAT	BE_URL	https://rv-uat-ssid.visitsaudi.com/api

Implementing SSID in Your Application

To integrate the New Saudi Smart ID (SSID) with your application, you will need to use the following details provided by SSID:

- Client ID
- Client Secret
- Token Endpoint
- User-Info Endpoint
- Authorization Endpoint

- Logout Endpoint
- Introspection Endpoint
- Revocation Endpoint

Token Endpoint

Item	Details
Description	This endpoint is used to exchange an authorization code for an access token.
HTTP Method	POST
Endpoint URL	{{KC_URL}}/realms/{{realm}}/protocol/openid-connect/token
Request Body	- grant_type: Authorization code, password, client_credentials, or refresh_token. - client_id - client_secret - code - redirect_uri - code_verifier

Headers	- Cookie: This request includes session cookies to maintain the user's authentication session. Example cookies: - AUTH_SESSION_ID: A session identifier. - AUTH_SESSION_ID_LEGACY: A legacy session identifier. - KC_RESTART: A Keycloak-specific cookie used to restart the authentication process.

User-Info Endpoint

Item	Details
Description	This endpoint fetches the user's information.
HTTP Method	GET

Endpoint URL	{{KC_URL}}/realms/{{realm}}/protocol/openid-connect/userinfo
Headers	Authorization: Bearer {access_token}

Authorization Endpoint

Item	Details
Description	This endpoint initiates the OpenID Connect authentication process by redirecting the user to the identity provider's authentication page, where they can log in and grant access to the requesting application. It includes parameters for client identification, redirection, response type, scope, nonce, and code challenge, ensuring secure authentication and authorization.
HTTP Method	GET
Endpoint URL	{{KC_URL}}/realms/ssid-realm/protocol/openid-connect/auth

Query Parameters	- client_id : The client identifier assigned to the application. - redirect_uri: The URL to which the identity provider will redirect the user after authentication. - response_type: Specifies the type of response desired. For this flow, it is set to `code`. - scope: The scope of the access request. In this example, it includes `openid profile email`. nonce`, `state`, `code_challenge`, and `code_challenge_method`: For PKCE. - nonce: A unique value to associate with the authentication request to mitigate replay attacks. - code_challenge: A code challenge for PKCE (Proof Key for Code Exchange). - code_challenge_method: The method used to derive the code challenge. Typically, `S256` for SHA-256. - state: An opaque value used to maintain state between the request and callback.
Headers	- Cookie: This request includes session cookies to maintain the user's authentication session. Example cookies: - AUTH_SESSION_ID: A session identifier. - AUTH_SESSION_ID_LEGACY: A legacy session identifier. - KC_RESTART: A Keycloak-specific cookie used to restart the authentication process.

Logout Endpoint

Item	Details
Description	This endpoint logs the user out and terminates the session.
HTTP Method	GET
Endpoint URL	{{KC_URL}}/realms/{{realm}}/protocol/openid-connect/logout
Query Parameters	Query Parameters: - post_logout_redirect_uri: URL to redirect after logout. - id_token_hint: ID token to identify the session to logout. - client_id: The client identifier assigned to the application.

Revocation Endpoint

Item	Details
Description	This endpoint revokes a token.
HTTP Method	POST
Endpoint URL	{{KC_URL}}/realms/{{realm}}/protocol/openid-connect/revoke
Query Parameters	- token - client_id - client_secret

Introspection Endpoint

Item	Details
Description	This endpoint verifies the validity of a token.
HTTP Method	POST
Endpoint URL	{{KC_URL}}/realms/{{realm}}/protocol/openid-connect/token/introspect
Query Parameters	- token - client_id - client_secret

Update-Mobile-Number Endpoint

Item	Details
Description	This endpoint updates the mobile number.
HTTP Method	POST
Endpoint URL	{{BE_URL}}/updateMobileNumber
Request Body	-countryCode -mobileNumber
Headers	Authorization: Bearer {access_token}

Validate-Mobile-OTP Endpoint

Item	Details
Description	This endpoint to validate mobile number
HTTP Method	PATCH
Endpoint URL	{{BE_URL}}/validateMobileOTP

Request Body	oneTimePass
Headers	Authorization: Bearer {access_token}

Integration Steps

Step 1: Implement Authorization Code Flow with PKCE

- Generate a code verifier and code challenge.
- Redirect the user to the Keycloak authorization endpoint with the code challenge.
- Exchange the authorization code for an access token at the token endpoint.

Step 2: Use the Access Token

Include the access token in the Authorization header for API requests to the UserInfo endpoint.

Step 3: Token Introspection and Revocation

Implement endpoints to introspect and revoke tokens as needed for security.

Step 4: Implement Logout

Redirect the user to the Keycloak logout endpoint to end the session.

Sample API Requests:

1. Authorization Request
   GET {{KC_URL}}/realms/ssid-realm/protocol/openid-connect/auth
   ?client_id=YOUR_CLIENT_ID
   &redirect_uri=https://yourapp.com/callback

&response_type=code
&scope=openid profile email smartPass
&nonce=RANDOM_NONCE
&state=RANDOM_STATE
&code_challenge=CODE_CHALLENGE
&code_challenge_method=S256

2. Token Request
   POST {{KC_URL}}/realms/ssid-realm/protocol/openid-connect/token
   Content-Type: application/x-www-form-urlencoded
   -d 'grant_type=authorization_code'
   -d 'client_id=YOUR_CLIENT_ID'
   -d 'client_secret=YOUR_CLIENT_SECRET'
   -d 'code=AUTHORIZATION_CODE'
   -d 'redirect_uri=https://yourapp.com/callback'
   -d 'code_verifier=CODE_VERIFIER'
3. UserInfo Request
   GET {{KC_URL}}/realms/ssid-realm/protocol/openid-connect/userinfo
   Authorization: Bearer ACCESS_TOKEN
4. Token Introspection Request
   POST {{KC_URL}}/realms/ssid-realm/protocol/openid-connect/token/introspect
   Content-Type: application/x-www-form-urlencoded
   -d 'token=ACCESS_TOKEN'
   -d 'client_id=YOUR_CLIENT_ID'
   -d 'client_secret=YOUR_CLIENT_SECRET'
5. Token Revocation Request
   POST {{KC_URL}}/realms/ssid-realm/protocol/openid-connect/revoke
   Content-Type: application/x-www-form-urlencoded
   -d 'token=ACCESS_TOKEN'
   -d 'client_id=YOUR_CLIENT_ID'
   -d 'client_secret=YOUR_CLIENT_SECRET'
6. Logout Request
   GET {{KC_URL}}/realms/ssid-realm/protocol/openid-connect/logout

```
?post_logout_redirect_uri=https://yourapp.com
&id_token_hint=ID_TOKEN
&client_id=YOUR_CLIENT_ID
```

Best Practices and Security Considerations

- PKCE (Proof Key for Code Exchange): Always use PKCE for public clients to enhance security.

- Token Management: Implement proper token lifecycle management including introspection and revocation.

- Logging and Monitoring: Monitor and log authentication requests and responses for auditing and troubleshooting.

Sample Token Response

```
{

"exp": 1715947814, // Expiration time of the token in Unix epoch time

  "iat": 1715947514, // Issued at time in Unix epoch time

  "jti": "98f61430-77f8-4494-82b5-8c807e5db7ff", // JWT ID, a unique identifier for the token

  "iss": "https://rv-uat-ssid.visitsaudi.com/idp/realms/ssid-realm", // Issuer of the token

  "aud": [“SSID_B2C","account"],// Audience(s) that the token is intended for

  "sub": "ca6f48cd-77dc-405b-a0dc-cf9e003cbc67", // Subject of the token, usually the user ID

  "typ": "Bearer", // Type of token

  "azp": "SSID_B2C", // Authorized party, the party to which the token was issued

  "session_state": "02cc3da6-dc45-4c90-bf58-993ca6b8573e", // Session state identifier
```

```
"acr": "1", // Authentication context class reference

"allowed-origins": [ // Allowed origins for CORS"*"],

"realm_access": { // Roles assigned to the user within the realm

"roles": ["default-roles-ssid-realm","offline_access","uma_authorization"]},

"resource_access": { // Roles assigned to the user for specific resources

"account": {"roles": ["manage-account","manage-account-links","view-profile"]} },

"scope": "openid email profile", // Scopes that the token grants access to

"sid": "02cc3da6-dc45-4c90-bf58-993ca6b8573e", // Session ID

"loyaltyId": "GO0100098", // User's loyalty ID

"lastName": "Abdelrahim", // User's last name

"gender": "male", // User's gender

"mobileNumber": "791412727", // User's mobile number

"pictureUrl": "", // URL to the user's picture

"given_name": "Emroualqais", // User's given name

"ssid": "SID-100805-2024-2024-a93e34-ba28-4b29-9a6e-0f60cdb5d", // User's Saudi Smart ID (SSID)

"birthDate": "1996-12-01", // User's birth date

"isEmailVerified": Y, // Whether the user's email is verified

"firstName": "Emroualqais", // User's first name

"countryCode": "+962", // User's country code
```

```
  "name": "Emroualqais Abdelrahim", // User's full name

  "isMobileVerified": "Y", // Whether the user's mobile number is verified

  "interests": "", // User's interests

  "email": "emroualqais.abdelrahim@pwc.com", // User's email address

  "travelPartner": "", // User's travel partner

  "username": "emroualqais.abdelrahim@pwc.com" // User's username

}
```

Conclusion

By following this guide, you have successfully integrated OIDC authentication with the New SSID. You can now leverage SSID robust authentication and authorization capabilities in your application.