OpenID Connect in Liferay DXP

Authentication, SSO, Configuration, Token Flow, User Mapping, and Security Concepts

Liferay DXP Technical Notes

September 28, 2026

Contents

1 Introduction **3**

2 OAuth 2.0 vs OpenID Connect **3**

3 Important Components **3**
- 3.1 End User . . . 4
- 3.2 Liferay DXP . . . 4
- 3.3 OpenID Provider . . . 4
- 3.4 Authorization Endpoint . . . 4
- 3.5 Token Endpoint . . . 4
- 3.6 User Information Endpoint . . . 4
- 3.7 JWKS URI . . . 4

4 High-Level Authentication Architecture **4**

5 Typical Authentication Flow **5**
- 5.1 Step 1: User Accesses Liferay . . . 5
- 5.2 Step 2: User Selects OpenID Connect . . . 5
- 5.3 Step 3: Redirect to Identity Provider . . . 5
- 5.4 Step 4: Provider Authenticates the User . . . 5
- 5.5 Step 5: Authorization Response . . . 6
- 5.6 Step 6: Token Processing . . . 6
- 5.7 Step 7: Identity Verification . . . 6
- 5.8 Step 8: Liferay Session . . . 6

6 ID Token **6**

7 Registering Liferay with the Provider **7**

8 Discovery Endpoint **7**

9 Configuring OpenID Connect in Liferay **8**

10 Enabling OpenID Connect **8**

11 User Information Mapping **8**

12 Role Mapping **9**

13 Authentication vs Authorization **9**
13.1 Authentication . . . 9
13.2 Authorization . . . 9

14 Security Considerations **10**
14.1 Use HTTPS . . . 10
14.2 Protect the Client Secret . . . 10
14.3 Validate Token Signatures . . . 10
14.4 Validate Issuer . . . 10
14.5 Validate Audience . . . 10
14.6 Validate Expiration . . . 10
14.7 Use Appropriate Scopes . . . 11

15 PKCE **11**

16 Single Sign-On **11**

17 OpenID Connect vs SAML **12**

18 Troubleshooting **12**
18.1 Provider Does Not Appear . . . 12
18.2 Redirect URI Error . . . 12
18.3 Invalid Client . . . 12
18.4 User Attributes Missing . . . 13
18.5 Discovery Endpoint Failure . . . 13

19 Example Enterprise Architecture **13**

20 Single Logout **14**

21 Interview and Revision Notes **14**

22 Multiple Choice Questions **15**

23 Quick Reference **17**

24 Key Takeaway **17**

25 Official Reference **17**

1 Introduction

OpenID Connect (OIDC) is an identity and authentication protocol built on top of OAuth 2.0.

OAuth 2.0 primarily addresses **authorization**, while OpenID Connect adds an **authentication and identity layer**.

In a typical Liferay deployment:

$$\textbf{User} \rightarrow \textbf{Liferay} \rightarrow \textbf{Identity Provider}$$

Examples of identity providers include:

- Keycloak
- Okta
- Microsoft Entra ID
- Google Identity
- Auth0
- Other standards-compliant OpenID Connect providers

Liferay acts as an OpenID Connect **Relying Party (RP)**, while the external authentication system acts as the OpenID Connect **Provider (OP)**.

Key Idea

Instead of Liferay independently authenticating the user's password, authentication can be delegated to a trusted OpenID Connect provider. After successful authentication, Liferay receives identity information that can be used to establish the user's authenticated session.

2 OAuth 2.0 vs OpenID Connect

A common source of confusion is the difference between OAuth 2.0 and OpenID Connect.

Feature	OAuth 2.0	OpenID Connect
Primary purpose	Authorization	Authentication and identity
Main question	What can the application access?	Who is the user?
Access Token	Yes	Yes
ID Token	No	Yes
User identity	Not its primary purpose	Core functionality
Built on	OAuth framework	OAuth 2.0

The important relationship is:

$$\boxed{\text{OpenID Connect} = \text{OAuth 2.0} + \text{Identity Layer}}$$

3 Important Components

The major participants in an OpenID Connect authentication flow are:

3.1 End User

The person attempting to authenticate to the Liferay portal.

3.2 Liferay DXP

Liferay acts as the **Relying Party (RP)**.

It trusts identity information provided by a configured OpenID Connect provider.

3.3 OpenID Provider

The OpenID Provider performs authentication.

Examples include Keycloak, Okta, Auth0, Microsoft Entra ID, and other compatible providers.

3.4 Authorization Endpoint

The user's browser is redirected to this endpoint so that the provider can authenticate the user.

3.5 Token Endpoint

After authorization, the client communicates with the token endpoint to obtain the appropriate tokens.

3.6 User Information Endpoint

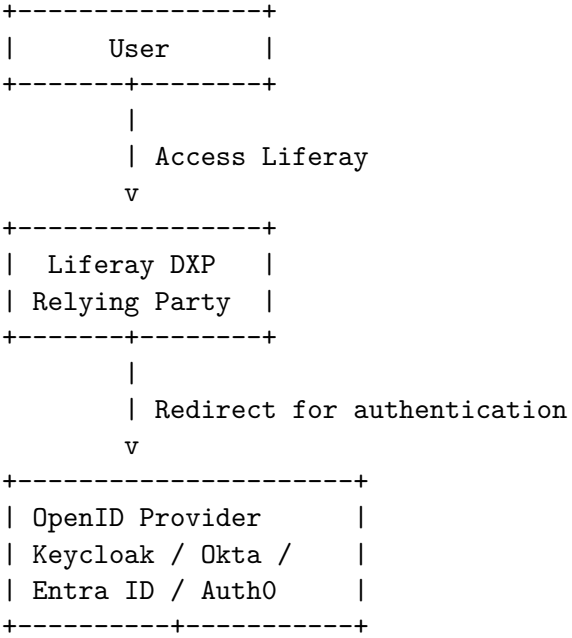
This endpoint can provide additional information about the authenticated user.

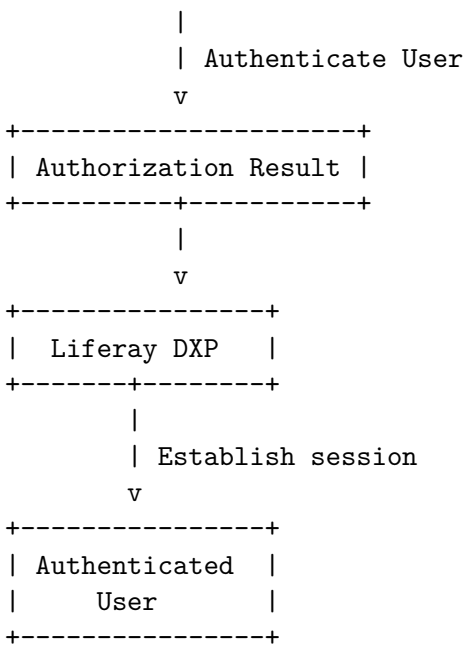
3.7 JWKS URI

The JSON Web Key Set (JWKS) endpoint publishes public cryptographic keys that can be used to verify signed tokens.

4 High-Level Authentication Architecture

A simplified architecture is:





5 Typical Authentication Flow

The authentication sequence can be understood step-by-step.

5.1 Step 1: User Accesses Liferay

The user opens the Liferay portal.

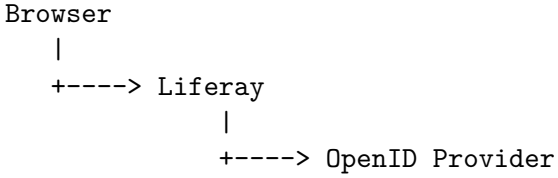
5.2 Step 2: User Selects OpenID Connect

The user chooses OpenID Connect authentication from the Liferay sign-in interface.

5.3 Step 3: Redirect to Identity Provider

Liferay redirects the browser to the configured provider's authorization endpoint.

Conceptually:



5.4 Step 4: Provider Authenticates the User

The provider may authenticate the user using mechanisms such as:

- username and password,
- multi-factor authentication,
- passwordless authentication,
- smart cards,

- enterprise directory credentials,
- biometric-backed authentication.

The authentication mechanism is primarily controlled by the identity provider.

5.5 Step 5: Authorization Response

After successful authentication, the provider redirects the user back to Liferay.

5.6 Step 6: Token Processing

Liferay processes the authorization result and obtains the necessary tokens from the provider.

5.7 Step 7: Identity Verification

The ID Token contains identity-related claims.

Typical claims can include:

`sub`
`name`
`given_name`
`family_name`
`email`
`preferred_username`

5.8 Step 8: Liferay Session

After successful validation and user mapping, Liferay establishes an authenticated portal session.

6 ID Token

One of the most important additions provided by OpenID Connect is the **ID Token**.

An ID Token is typically represented as a JSON Web Token (JWT):

$$\text{JWT} = \text{Header.Payload.Signature}$$

A conceptual payload could look like:

```
{
  "iss": "https://identity.example.com",
  "sub": "248289761001",
  "aud": "liferay-client",
  "email": "user@example.com",
  "given_name": "Rajesh",
  "family_name": "Chaurasia"
}
```

Important claims include:

`iss` Issuer that created the token.

`sub` Subject identifier representing the user.

`aud` Audience for which the token was issued.

`exp` Expiration time.

`iat` Time at which the token was issued.

7 Registering Liferay with the Provider

Before Liferay can authenticate users through an external provider, Liferay must normally be registered as a client.

The provider typically supplies:

- Client ID
- Client Secret
- Issuer URL
- Discovery Endpoint
- Authorization Endpoint
- Token Endpoint
- User Information Endpoint
- JWKS URI

The standard Liferay redirect URL is generally:

```
https://[server.domain]/c/portal/login/openidconnect
```

For local development it could conceptually be:

```
http://localhost:8080/c/portal/login/openidconnect
```

The redirect URI registered with the identity provider must match the Liferay configuration expected by the provider.

8 Discovery Endpoint

Modern OpenID Connect providers usually expose a discovery document.

A common pattern is:

```
https://provider.example.com/.well-known/openid-configuration
```

The discovery document can provide metadata such as:

- issuer,
- authorization endpoint,
- token endpoint,
- user information endpoint,
- JWKS URI,
- supported scopes,
- supported signing algorithms.

This greatly simplifies configuration because the client does not need every endpoint to be manually entered.

9 Configuring OpenID Connect in Liferay

For the standard provider configuration, navigate approximately to:

Global Menu → Control Panel → Instance Settings → Security → SSO

Then select:

OpenID Connect Provider

and add the provider connection.
Typical configuration properties include:

Property	Purpose
Provider Name	Human-readable identity provider name.
Scopes	Defines the identity information requested from the provider.
Discovery Endpoint	Provides OpenID Connect metadata and endpoint discovery.
Authorization Endpoint	Used to redirect users for authentication.
Issuer URL	Identifies the token issuer.
JWKS URI	Provides public keys for signature verification.
Token Endpoint	Used for requesting tokens.
User Information Endpoint	Provides additional authenticated user information.
Client ID	Identifies the Liferay client application.
Client Secret	Authenticates the client to the provider where required.
Custom Claims	Allows additional provider claims to be mapped into Liferay.

10 Enabling OpenID Connect

Creating a provider configuration alone does not necessarily enable OpenID Connect authentication.

Navigate to:

Control Panel → Instance Settings → Security → SSO → OpenID Connect

Enable OpenID Connect authentication and save the configuration.
Users can then use the configured provider through the Liferay sign-in experience.

11 User Information Mapping

Identity providers and Liferay may use different names for the same user attributes.

For example:

Liferay Attribute	OIDC Claim
firstName	given_name
lastName	family_name
emailAddress	email
screenName	preferred_username

Conceptually:

$$\text{OIDC Claim} \longrightarrow \text{Liferay User Attribute}$$

For example:

$$\texttt{given_name} \longrightarrow \texttt{firstName}$$

Liferay also supports mapping appropriate OpenID Connect claims to Liferay custom user fields.

12 Role Mapping

OpenID Connect identity information may also be used for role-related mapping.

Conceptually:

```
OIDC Provider
     |
     | roles / claims
     v
Liferay User
     |
     v
Liferay Roles
```

For automatic role assignment, the corresponding role must exist in Liferay and satisfy the requirements expected by Liferay's mapping mechanism.

This enables centralized identity systems to participate in application-level authorization.

13 Authentication vs Authorization

It is important to distinguish these concepts.

13.1 Authentication

Authentication answers:

Who are you?

OpenID Connect primarily addresses authentication and identity.

13.2 Authorization

Authorization answers:

What are you allowed to do?

In Liferay, authorization is typically governed using:

- roles,
- permissions,
- organizations,

- sites,
- user groups,
- resource permissions.

Therefore:

$$\boxed{\text{OIDC Authentication} \neq \text{Liferay Authorization}}$$

They work together, but solve different problems.

14 Security Considerations

14.1 Use HTTPS

Production systems should use HTTPS.

Without TLS, authentication traffic can become vulnerable to interception or manipulation.

14.2 Protect the Client Secret

Never expose a client secret in:

- JavaScript,
- public repositories,
- JSP pages,
- browser-visible configuration,
- log files.

14.3 Validate Token Signatures

Tokens must be cryptographically verified.

The provider's JWKS endpoint supplies the public keys needed for signature verification.

14.4 Validate Issuer

The token issuer should match the configured trusted provider.

$$\texttt{token.iss} = \texttt{configured issuer}$$

14.5 Validate Audience

The token should have been issued for the expected client.

$$\texttt{token.aud} \supseteq \texttt{Liferay Client ID}$$

14.6 Validate Expiration

Expired identity tokens must not be accepted.

$$t_{\text{current}} < t_{\text{expiration}}$$

14.7 Use Appropriate Scopes

Applications should request only the identity information they actually require.

Typical OIDC scopes include:

```
openid
profile
email
```

15 PKCE

Liferay supports Proof Key for Code Exchange (PKCE) as an OpenID Connect client.

PKCE introduces a dynamically generated verifier and challenge into the authorization flow.

Conceptually:

$$\text{code_verifier} \longrightarrow H(\text{code_verifier}) \longrightarrow \text{code_challenge}$$

The authorization request sends the challenge.

Later, the token request supplies the verifier.

The authorization server verifies:

$$H(\text{received verifier}) = \text{original challenge}$$

This strengthens authorization-code flows against certain forms of authorization-code interception.

16 Single Sign-On

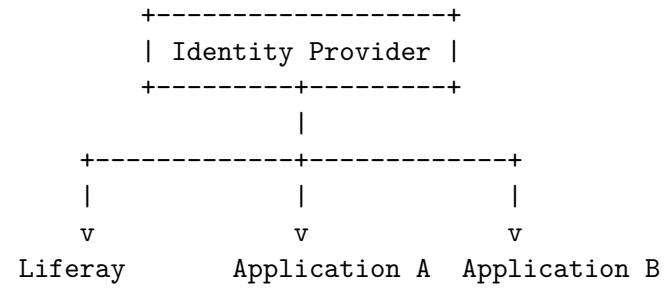
OpenID Connect is useful for Single Sign-On (SSO).

Suppose an organization has:

- Employee Portal
- Liferay DXP
- Learning Platform
- Support Portal
- Internal Applications

All applications can delegate authentication to the same identity provider.

Conceptually:



The identity provider becomes the centralized authentication authority.

17 OpenID Connect vs SAML

Both technologies are commonly used for enterprise SSO.

Feature	OpenID Connect	SAML
Foundation	OAuth 2.0	XML-based SAML standard
Typical token format	JWT / JSON	XML assertion
Common usage	Modern web/mobile/API ecosystems	Enterprise SSO and legacy integrations
Developer experience	Often REST/JSON friendly	More XML-oriented
Identity federation	Yes	Yes

Neither protocol is universally "better." The appropriate choice depends on the identity infrastructure, application architecture, security requirements, and organizational standards.

18 Troubleshooting

18.1 Provider Does Not Appear

Check that:

- OpenID Connect is enabled.
- The provider configuration is saved.
- The discovery endpoint is reachable from the Liferay server.

18.2 Redirect URI Error

A common identity-provider error is:

```
redirect_uri_mismatch
```

Verify that the redirect URI configured at the provider matches the URI used by Liferay.

18.3 Invalid Client

An error such as:

```
invalid_client
```

usually indicates a problem with:

- Client ID
- Client Secret
- provider configuration

18.4 User Attributes Missing

Check:

- requested scopes,
- provider claims,
- UserInfo response,
- Liferay user mapping configuration.

18.5 Discovery Endpoint Failure

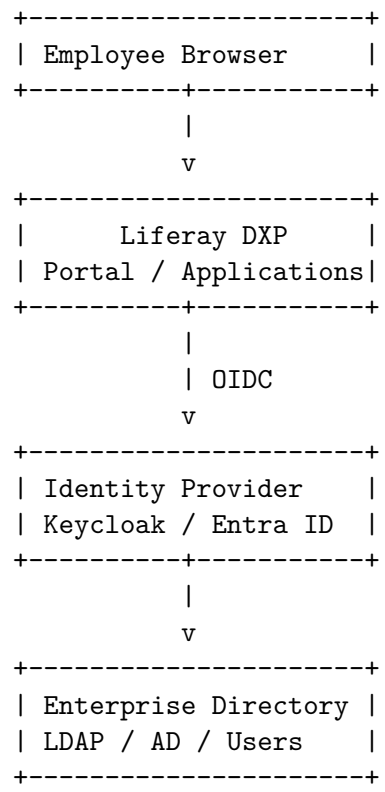
From the Liferay server, test whether the provider metadata can actually be reached:

```
curl https://provider.example.com/.well-known/openid-configuration
```

Firewall rules, proxies, DNS problems, or TLS issues may prevent Liferay from reaching the provider.

19 Example Enterprise Architecture

Consider the following architecture:



The responsibilities can be separated as follows:

Identity Provider: Authentication and identity federation.

Liferay: Portal session, roles, permissions, sites, applications, and business functionality.

Enterprise Directory: Authoritative organizational identity information where applicable.

20 Single Logout

Modern Liferay DXP releases also support OpenID Connect **Back-Channel Logout**.

The provider can notify Liferay through a server-to-server request that an OpenID Connect session has ended.

The Liferay back-channel endpoint follows the pattern:

```
https://your-liferay-domain/o/open_id_connect/backchannel_logout
```

The important architectural difference is:

```
Normal Login:

Browser -> Liferay -> OIDC Provider


Back-Channel Logout:

OIDC Provider ---------> Liferay
               server-to-server
```

This mechanism does not depend on the user's browser delivering the logout notification. Feature availability depends on the specific Liferay DXP release.

21 Interview and Revision Notes

What is OpenID Connect?

OpenID Connect is an authentication and identity layer built on top of OAuth 2.0.

What role does Liferay play?

Liferay acts as an OpenID Connect Relying Party (RP) and delegates authentication to an external OpenID Provider.

OAuth 2.0 vs OIDC

OAuth 2.0 primarily handles authorization. OpenID Connect adds authentication and standardized identity information.

What is an ID Token?

An ID Token is a signed token containing claims about the authenticated user and authentication event.

What is the Discovery Endpoint?

It provides metadata describing the provider's OIDC configuration, including important endpoints and supported capabilities.

What is JWKS?

JWKS is a JSON Web Key Set containing public cryptographic keys used to verify signed tokens.

22 Multiple Choice Questions

Question 1

OpenID Connect is primarily used for:

A. Database replication

B. Authentication and identity

C. Java dependency management

D. Liferay indexing

Answer: B

Question 2

OpenID Connect is built on top of:

A. LDAP

B. Kerberos

C. OAuth 2.0

D. JDBC

Answer: C

Question 3

In a standard external OIDC integration, Liferay primarily acts as:

A. OpenID Provider

B. Relying Party

C. Certificate Authority

D. LDAP server

Answer: B

Question 4

Which token is specifically introduced by OpenID Connect for identity information?

A. Refresh Token

B. Access Token

C. ID Token

D. CSRF Token

Answer: C

Question 5

Which endpoint normally publishes the provider's public signing keys?

A. Token Endpoint

B. JWKS URI

C. Redirect URI

D. Logout Page

Answer: B

Question 6

What is the purpose of the `sub` claim?

A. Store the user's password

B. Identify the token encryption algorithm

C. Uniquely identify the subject

D. Specify the HTTP method

Answer: C

Question 7

Which claim identifies the intended recipient of a token?

A. aud

B. exp

C. iss

D. iat

Answer: A

Question 8

What does PKCE primarily strengthen?

A. Database encryption

B. Authorization-code exchange

C. Liferay indexing

D. LDAP replication

Answer: B

23 Quick Reference

Concept	Meaning
OIDC	Authentication and identity protocol
OAuth 2.0	Authorization framework
RP	Relying Party, such as Liferay
OP	OpenID Provider
ID Token	Identity claims about authenticated user
Access Token	Authorization for protected resources
JWKS	Public keys for token verification
Discovery	Provider metadata discovery
PKCE	Protection for authorization-code exchange
SSO	Single Sign-On

24 Key Takeaway

The central idea behind OpenID Connect integration in Liferay is:

External Identity Provider → Authenticate User → OIDC Identity → Liferay Session

OpenID Connect allows Liferay to delegate authentication to a central identity platform while Liferay continues to control portal-level roles, permissions, sites, applications, and other authorization concerns.

25 Official Reference

For current configuration details and version-specific behavior, refer to the official Liferay documentation:

https://learn.liferay.com/w/dxp/security-and-administration/security/configuring-sso/using-openid-connect