

OAuth 2.0, OpenID Connect, PKCE & Authorization Flows

This document explains all major concepts students typically ask about in OAuth 2.0, OpenID Connect (OIDC), PKCE, and common flows. It concludes with a section specific to Okta.

1. What is OAuth 2.0?

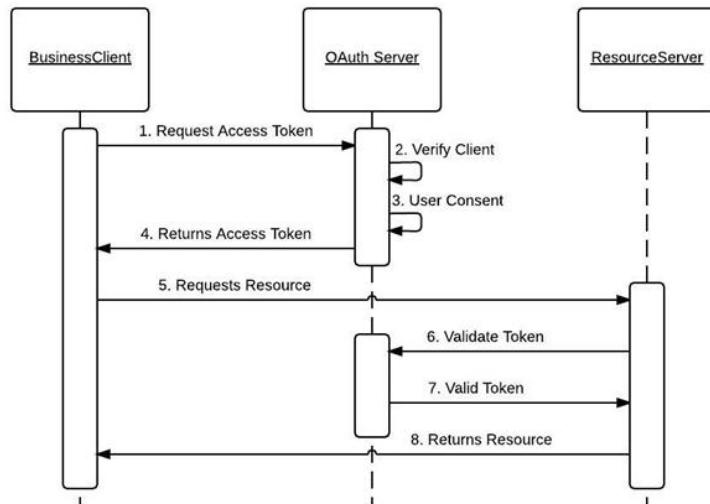
OAuth 2.0 is an **authorization framework** that allows one application to get limited access to a user's resources on another service without exposing the user's password.

Key Roles

- **Resource Owner** – Usually the end user.
- **Client** – App requesting access (web, mobile, API service).
- **Authorization Server** – Issues tokens after authenticating user.
- **Resource Server** – API that validates tokens.

Token Types

- **Access Token** – Used to access APIs.
- **Refresh Token** – Used to get new access tokens.
- **ID Token** – Provided by OpenID Connect.



2. What is OpenID Connect (OIDC)?

OIDC is an **identity layer** on top of OAuth 2.0. - Provides **ID Token** (JWT) that contains user identity claims. - Adds features like /userinfo endpoint. - Ideal for user authentication (login).

Common OIDC Claims

- `sub` – Unique user identifier.
 - `name`, `email`, `preferred_username`.
 - `iat`, `exp` – Token issuance & expiry.
-

3. What is PKCE?

PKCE (Proof Key for Code Exchange) is an extension to OAuth 2.0 that adds extra security, mainly for mobile & SPA clients.

Why PKCE?

Prevents **authorization code interception**.

PKCE Flow Steps

1. Client generates a `code_verifier`.
2. Creates a `code_challenge` (hashed version).
3. Sends `code_challenge` during /authorize.
4. Sends `code_verifier` during /token.
5. Server verifies match → issues tokens.

PKCE is mandatory for SPAs and public clients.

4. OAuth 2.0 / OIDC Flows

Below are the major flows students commonly ask about.

4.1 Authorization Code Flow

Best for **web apps with backend server**. - Secure because tokens are exchanged on the server. - Uses code + client secret. - Optionally uses PKCE.

4.2 Authorization Code Flow with PKCE

Best for **SPAs, mobile, native apps**. - No client secret. - Uses PKCE to prevent code interception.

4.3 Implicit Flow (Deprecated)

- Earlier method for SPAs.
- Delivered tokens via browser URL.
- Not recommended anymore.

4.4 Client Credentials Flow

Used for **server-to-server** (no user). - App gets token using client ID + secret. - Useful for background jobs, microservices.

4.5 Resource Owner Password Credential (ROPC) Flow (Not recommended)

- User gives username/password directly to client.
- High security risk.
- Deprecated.

4.6 Device Authorization Flow

Used when device can't show login UI (TVs, IoT). - User logs in separately using a code. - Device polls token endpoint.

5. Access Token vs ID Token vs Refresh Token

Access Token

- Used for calling APIs.
- Short-lived.
- Should not be interpreted by client.

ID Token

- Identity information about user.
- JWT format.
- Used by client to log user into app.

Refresh Token

- Used to obtain new access tokens.
 - Long-lived.
 - Should be kept secure.
-

6. Token Validation

Access Token (JWT or opaque)

- Validate signature.
- Validate expiry.
- Validate issuer (iss) and audience (aud).

ID Token

- Must validate nonce.
 - Must validate signature and aud.
-

7. Scopes & Claims

Scopes

Define what your app can access.

- openid – Required for OIDC.
- profile, email – Additional claims.
- Custom scopes for APIs.

Claims

Data returned in tokens. - Standard (email, name) - Custom claims (roles, group membership)

8. Authorization vs Authentication

Authorization

- Granting access to a resource.
- Done by OAuth 2.0.

Authentication

- Verifying identity.
 - Done by OIDC.
-

9. API Security Using OAuth

Typical backend API security workflow: 1. Client receives access token. 2. Sends it in Authorization: Bearer <token> header. 3. API validates token signature + claims. 4. API allows or denies access.

What is SAML?

SAML (Security Assertion Markup Language) is an XML-based open standard for transferring identity (authentication) data between two parties: an Identity Provider (IdP) and a Service Provider (SP).

How it works (high-level)

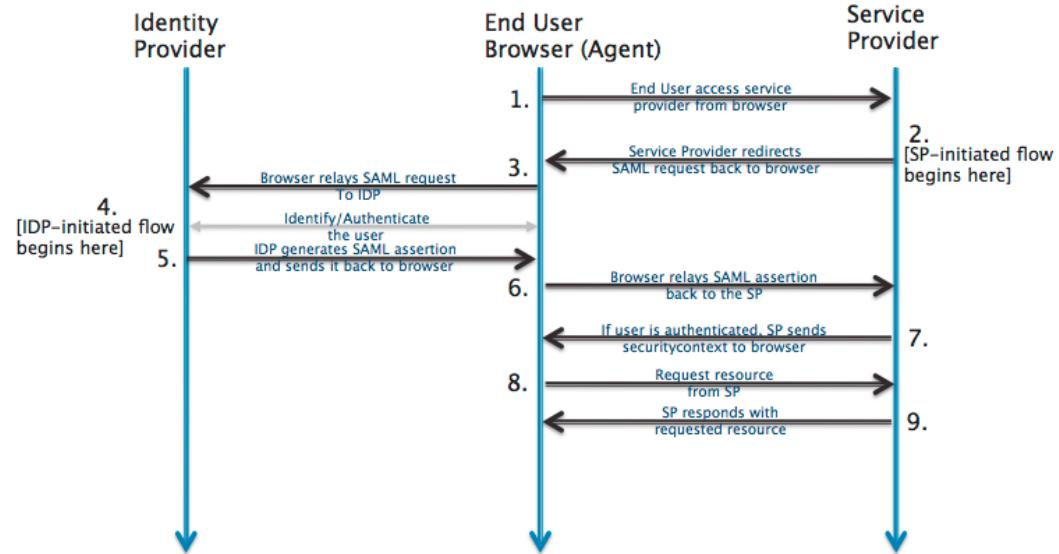
- User attempts to access a resource at a Service Provider (SP).
- SP sends an AuthnRequest to the Identity Provider (IdP) via browser redirect or POST.
- User authenticates at IdP.
- IdP sends a SAML assertion (an XML document) back to SP via browser.
- SP validates the assertion and grants access.

Scenarios where SAML is used

- Enterprise Single Sign-On (SSO) for web applications (internal corporate apps).
- Federated identity between organizations/partners.
- Large legacy web-apps where SOAP, XML and older infrastructure exist.

Comparison to OIDC/OAuth

- Token Format: SAML uses XML Assertions; OIDC uses JWT/JSON.
- SAML is geared toward web browser SSO in enterprise context; OIDC is more API/mobile friendly.
- Use cases: if you have large enterprise apps, old infrastructure and need SSO across domains, SAML may be appropriate.



10. Okta – Overview

Okta is a **cloud identity provider** that supports OAuth 2.0, OIDC, SAML, and other standards.

10.1 Key Okta Components

- **Okta Authorization Server** – Issues access & ID tokens.
- **App Integration** – Web, SPA, API service apps.
- **User Directory** – Users, groups, MFA policies.
- **Custom Authorization Server** – For custom API access policies.

10.2 Okta Flows Supported

- Authorization Code (with/without PKCE)
- Device Flow
- Client Credentials
- Token Refresh

10.3 Okta Policies

- Access policies define which apps can request which scopes.
- Resource access governed through API Access Management.
- Common error: no_matching_policy – app not allowed under any policy.

10.4 Okta Tokens

- ID tokens include Okta-specific claims like `preferred_username`.
- Access tokens may include custom claims based on groups or attributes.

10.5 Integrating Okta with Applications

For Web Apps

- Use Authorization Code Flow.
- Configure callback URL.

For SPAs

- Use PKCE flow.
- Configure trusted origins.

For APIs

- Use Okta Authorization Server.
 - Validate tokens using JWKS.
-

11. Summary

- OAuth 2.0 handles authorization.
- OIDC adds authentication.
- PKCE improves security for public clients.
- Choose flow based on client type.
- Okta provides enterprise-grade identity tools that fully support all these standards.