Understanding Authentication With OpenID Connect

Kevin Dockx
Architect

@KevinDockx https://www.kevindockx.com

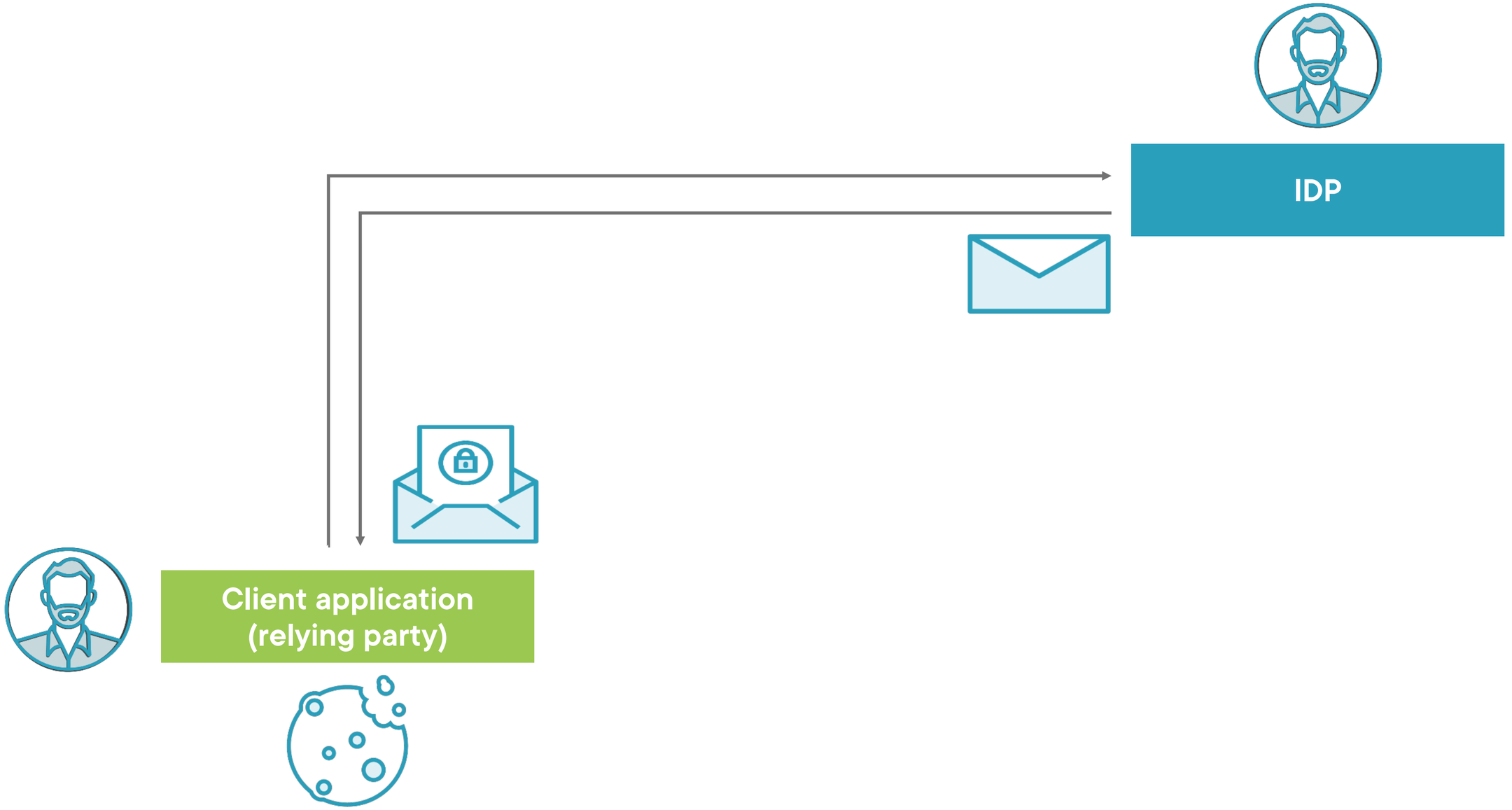
Coming Up

Learning how OpenID Connect works

Learning about clients, endpoints and flows

Setting up an identity provider: Duende.IdentityServer v6

Learning How OpenID Connect Works



Public and Confidential Clients

Confidential clients

Capable of maintaining the confidentiality of their credentials (e.g.: clientid, clientsecret)

Live on the server

These client applications can safely authenticate

E.g.: server-side web apps

Public clients

Incapable of maintaining the confidentiality of their credentials (e.g.: clientid, clientsecret)

Live on the device

These client applications cannot safely authenticate

E.g.: JavaScript apps, Blazor WASM apps *(and mobile apps)*

OIDC flow

A set of (HTTP) requests and responses that determine how code(s) and/or tokens are safely delivered to clients

OpenID Connect Flows and Endpoints

Different client types and/or requirements lead to different (variations of) flows

Flows use endpoints (at level of the IDP and at level of the client)

- These replaces old-style homegrown endpoints

Authorization endpoint (IDP level)

- Used by the client application to obtain authentication and/or authorization, via redirection

TLS

TLS is a requirement for OIDC. Out of the box, tokens are not encrypted. OIDC relies on the transport layer to take care of encryption.

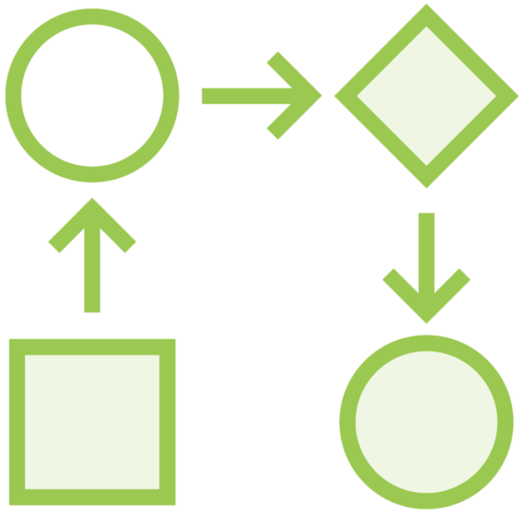
Redirection endpoint (client level)

- Used by the IDP to return code & token(s) to the client application

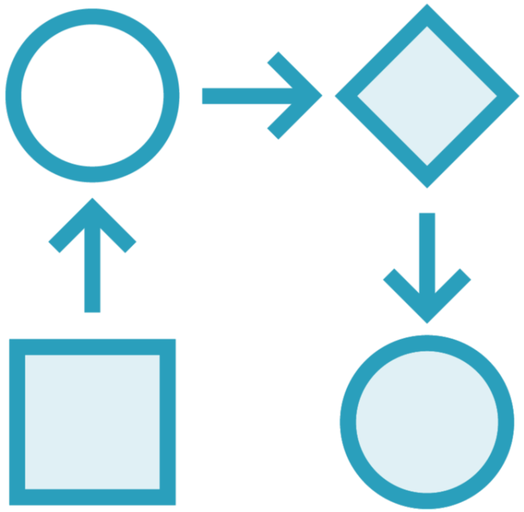
Token endpoint (IDP level)

- Used by the client application to request tokens (without redirection) from the IDP
- Requests can be authenticated (confidential clients) or not (public clients)

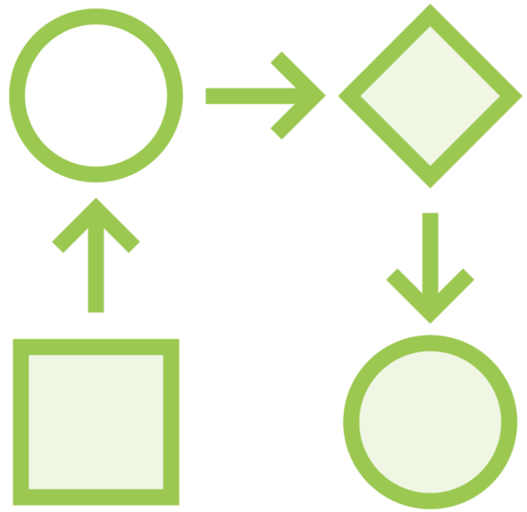
OpenID Connect Flows



Authorization code

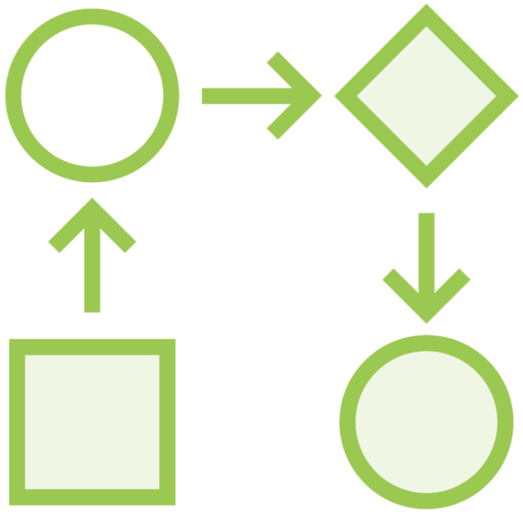


Implicit



Hybrid

OpenID Connect Flows

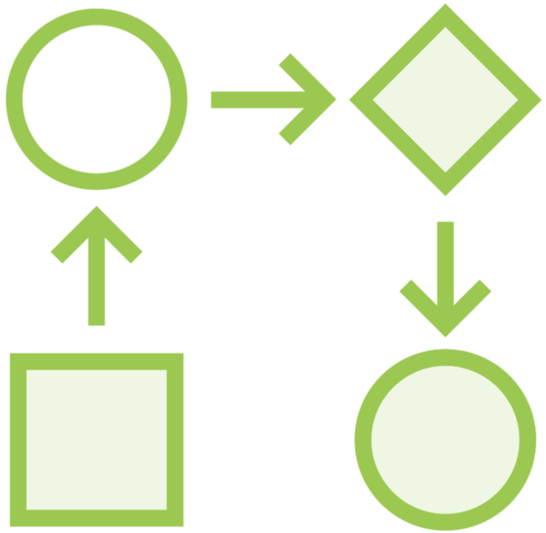


Authorization code

Implicit

Hybrid

OpenID Connect Flows

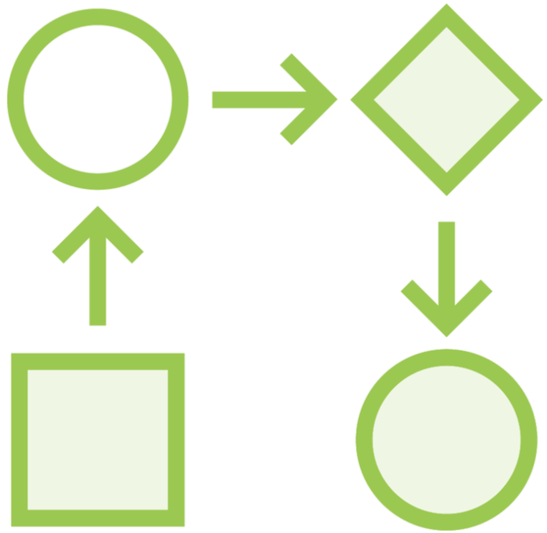


Confidential clients

Authorization code flow + PKCE

Code returned from authorization endpoint

Authenticated call to token endpoint



Public clients

Authorization code flow + PKCE

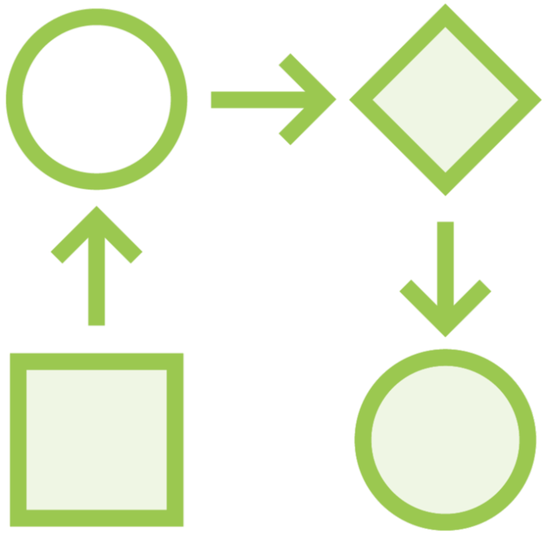
Code returned from authorization endpoint

Unauthenticated call to token endpoint

Authorization code

A short-lived, single use credential, used to verify that the user who logged in at level of the IDP is the same one who started the flow at level of the client application

OpenID Connect Flows

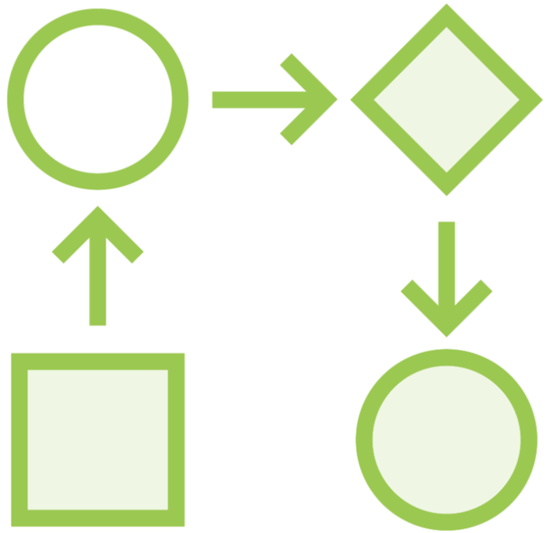


Confidential clients

Authorization code flow + PKCE

Code returned from authorization endpoint

Authenticated call to token endpoint



Public clients

Authorization code flow + PKCE

Code returned from authorization endpoint

Unauthenticated call to token endpoint

OpenID Connect for Public Clients

Angular, React, Vue, Blazor WASM, …

- Consensus towards moving away from handling security at level of the client in favor of the server

Handle flows at level of the hosting application (e.g.: an ASP.NET Core 6 application)

- Potentially use the BFF pattern

OpenID Connect Flows

Choosing the wrong (variation of a) flow may open you up to security vulnerabilities

- Technically, nothing is stopping us from doing this

The thing with security is that a lot of approaches will work, but most of them are not a good idea

The most important statement of the entire course

What IS a good idea changes over time

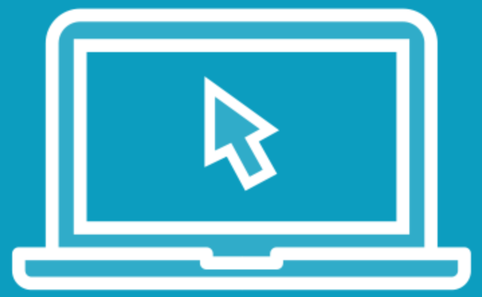
The second most important statement of the entire course

Introducing Duende IdentityServer

OpenID Connect and OAuth 2 framework for ASP.NET Core, developed by Duende Software

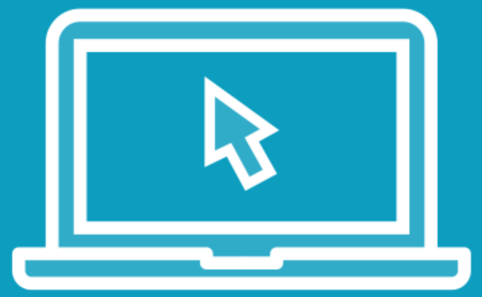
- https://duendesoftware.com
- https://github.com/DuendeSoftware

Demo



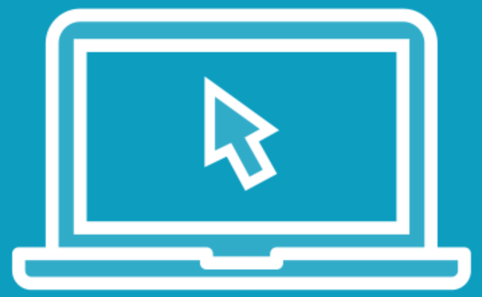
Setting up IdentityServer

Demo



Adding a user interface

Demo



Adding users to test with

Standardized Scopes and Claims

Scope: openid (required scope for OIDC)
Claims: sub (user identifier)

Scope: profile
Claims: name, family_name, given_name, middle_name, nickname, preferred_username, profile, picture, website, gender, birthdate, zoneinfo, locale, updated_at

Scope: email
Claims: email, email_verified

Standardized Scopes and Claims

Scope: address
Claims: address

Scope: phone
Claims: phone_number, phone_number_verified

Scope: offline_access
Claims: / (used for long-lived access)

Summary

A confidential client can safely store secrets, a public client can't

A flow can be seen as a set of requests and responses via which a client can safely achieve authentication (and authorization)

- For ASP.NET Core web applications, the authorization code flow + PKCE, with client authentication, is advised

Summary

Authorization endpoint (IDP)

- Used by the client application to obtain authentication and/or authorization

Token endpoint (IDP)

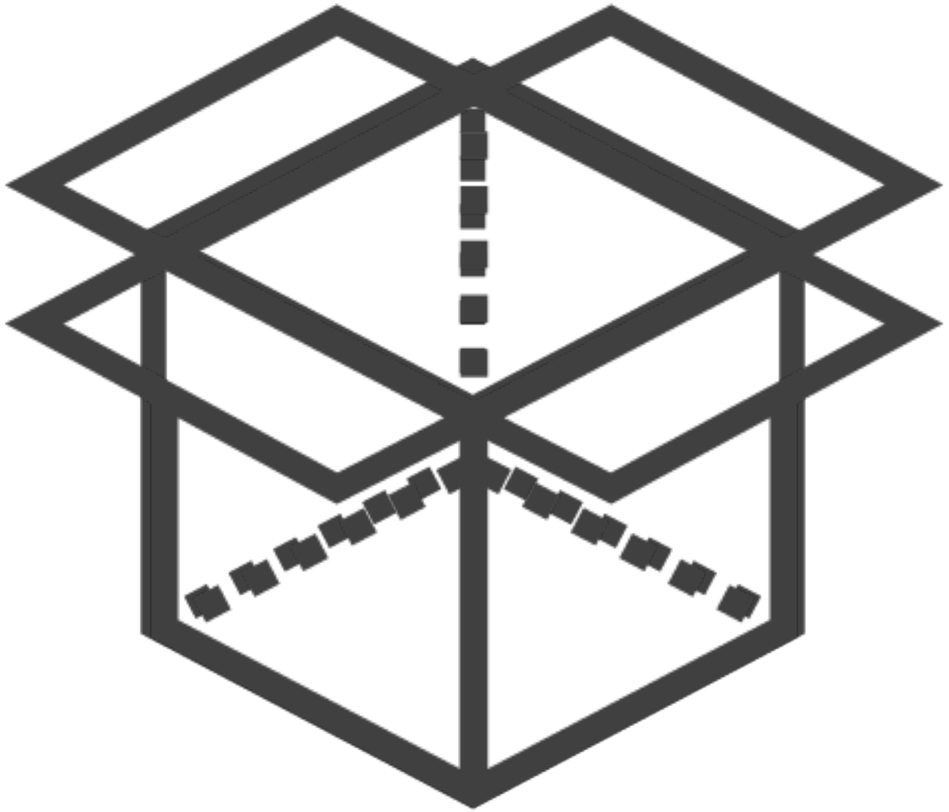
- Used by an application to programmatically request tokens

Summary

Redirection endpoint (client application)

- Where the tokens are delivered to from the authorization endpoint

TLS is a requirement



It's our responsibility to keep the holes in this box as small as possible

Up Next:

Securing Your User Authentication Processes