

Choosing the Right OIDC Application Type

Overview

When registering an application in ID Everywhere, you must choose the appropriate OIDC application type.

The two available types are:

- **Public Application (PKCE)**
- **Web / Server Application (Confidential)**

Both use OpenID Connect. They are not separate authentication protocols.

The correct selection depends primarily on whether the application can securely protect a client secret.

Public Application (PKCE)

A public application cannot reliably keep a client secret confidential.

Examples include:

- Native mobile applications.
- Desktop applications distributed to users.
- Browser-only applications.
- Single-page applications running entirely in the user's browser.

Because users can inspect application code or distributed binaries, storing a permanent client secret inside these applications would be insecure.

How public applications authenticate

Public applications use the Authorization Code flow with **Proof Key for Code Exchange (PKCE)**.

PKCE helps protect the authorization-code exchange.

The application generates a temporary random value called a *code verifier*. It derives a *code challenge* using the S256 method and sends that challenge during authorization.

When exchanging the authorization code for tokens, the application provides the original verifier.

ID Everywhere can then verify that the client exchanging the code possesses the matching verifier.

Public application configuration

Setting	Value
Application type	Public Application (PKCE)
Authorization grant	Authorization Code
PKCE	Required
Code challenge method	S256
Client ID	Required
Client Secret	Not used
Redirect URI	Required

Web / Server Application (Confidential)

A confidential application has a trusted server-side environment capable of protecting a client secret.

Examples include:

- Traditional server-rendered web applications.
- Backend-driven business applications.
- Applications with a secure server responsible for exchanging authorization codes.

How confidential applications authenticate

1. The user starts sign-in from the application.

2. The browser redirects to ID Everywhere.
3. The user authenticates.
4. ID Everywhere redirects the browser back with an authorization code.
5. The application's backend exchanges that code for tokens.
6. The backend authenticates itself using its registered client credentials.

The client secret must remain on the trusted server.

Confidential application configuration

Setting	Value
Application type	Web / Server Application (Confidential)
Authorization grant	Authorization Code
Client ID	Required
Client Secret	Required
Client authentication	Client Secret Basic, where supported
Redirect URI	Required
PKCE	May be used additionally when supported

Important: Confidential applications can also use PKCE. PKCE and confidential client authentication provide different protections and are not mutually exclusive.

Which application type should I select?

Ask this question:

Can the application securely store a secret that ordinary users cannot retrieve?

If the answer is **no**, choose Public Application (PKCE).

If the answer is **yes**, and the application performs the token exchange on a trusted backend, choose Web / Server Application (Confidential).

Examples

Example 1: A native mobile application

The application runs on customer-owned phones.

Recommended type: **Public Application (PKCE)**.

Example 2: A PHP web application

The application runs on a secured server. Its backend performs the token exchange and stores the client secret outside the public web directory.

Recommended type: **Web / Server Application (Confidential)**.

Example 3: A browser-only JavaScript application

The application code is delivered to the browser, where users can inspect it.

Recommended type: **Public Application (PKCE)**.

Example 4: A web application with both frontend and backend components

The correct type depends on which component performs the authorization-code exchange and whether the backend securely manages the client secret.

A trusted backend may use a confidential client. A browser-only token exchange should use a public client.

Security recommendations

- Never place a confidential client secret in browser JavaScript.
- Never embed a confidential client secret in a mobile or desktop application.
- Register only the redirect URLs your application needs.
- Use HTTPS for production redirect URLs.
- Use PKCE S256 for public clients.
- Protect client secrets in server-side configuration or a secret manager.
- Rotate a client secret if it may have been exposed.
- Validate received tokens according to OIDC requirements.

Frequently asked questions

Is PKCE a different login protocol?

No. PKCE is a security mechanism used with the OAuth 2.0 Authorization Code flow.

Does a public application have a Client ID?

Yes. Public applications still use a Client ID to identify their registration.

Does a public application need a Client Secret?

No. Public applications cannot securely protect a permanent client secret.

Should every website use Confidential?

Not necessarily. A browser-only website generally cannot protect a secret. The architecture of the application determines the correct client type.

Related articles

- [Understanding Authentication Methods in ID Everywhere](#)
- [Connecting an Application Using OIDC](#)
- [Understanding OIDC Settings and Security Terms](#)

Revision #1

Created 2026-10-10 22:10:32 UTC by Administrator

Updated 2026-10-10 22:10:55 UTC by Administrator