


Implicit Flow with Form Post

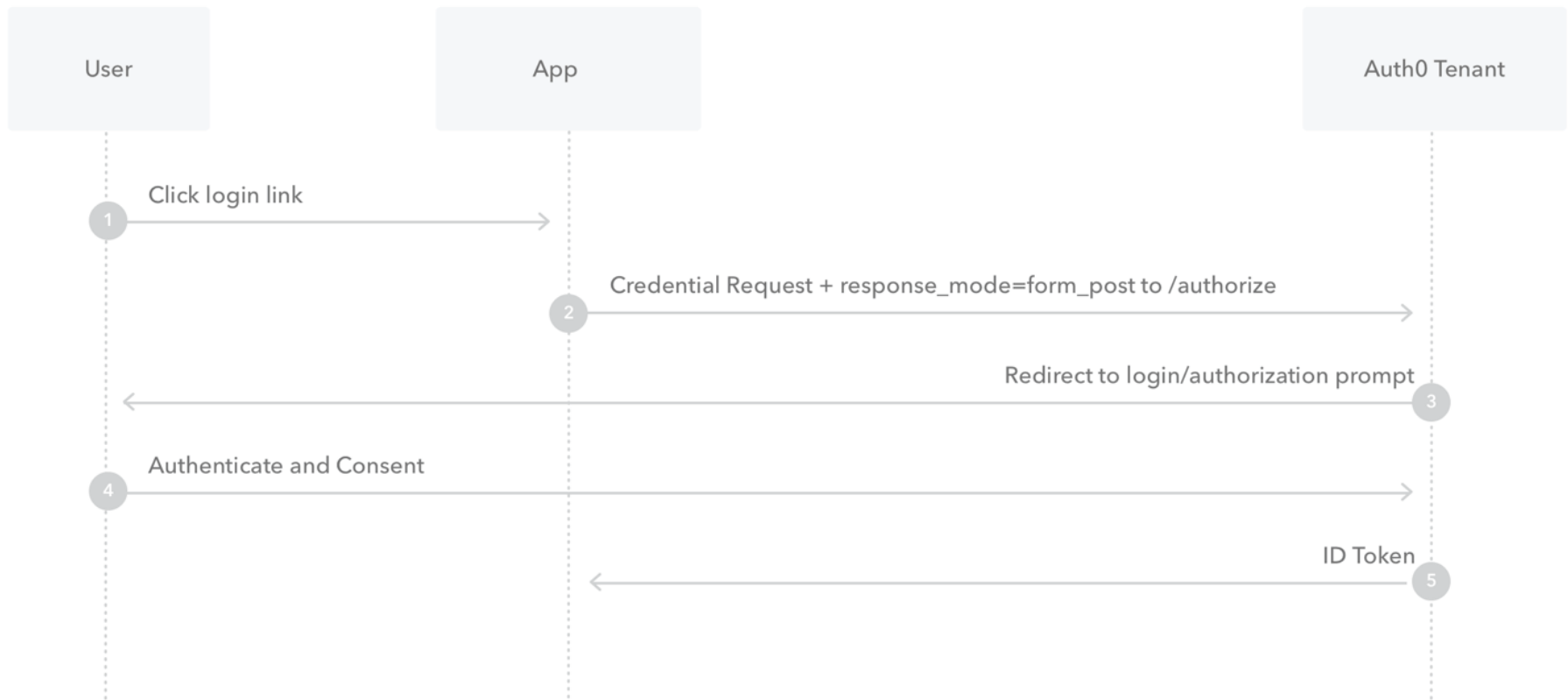
⚠ Don't let the term "implicit" mislead you! Although OAuth now discourages the use of the implicit grant for obtaining access tokens in SPAs, the scenario addressed by Implicit Flow with Form Post is completely different and is **unaffected by the security issues** that led to discouraging use with SPAs. Specifically, Implicit Flow with Form Post applies to traditional web apps as opposed to SPAs. You obtain ID tokens as opposed to access tokens, which have a completely different intended use. The flow uses POST as opposed to placing tokens in URL fragments (as with SPAs) which can expose token bits to browser history attacks, redirect headers, and so on.

You can use OpenID Connect (OIDC) with many different flows to achieve web sign-in for a traditional web app. In one common flow, you obtain an ID token using authorization code flow performed by the app backend. This method is effective and robust, however, it requires your web app to obtain and manage a secret. You can avoid that burden if all you want to do is implement sign-in and you don't need to obtain access tokens for invoking APIs.

Implicit Flow with Form Post flow uses OIDC to implement web sign-in that is very similar to the way SAML and WS-Federation operates. The web app requests and obtains tokens through the front channel, without the need for secrets or extra backend calls. With this method, you don't need to obtain, maintain, use, and protect a secret in your application.

How it works

 You should use this flow for login-only use cases; if you need to request Access Tokens while logging the user in so you can call an API, use the [Authorization Code Flow with PKCE](#) or the [Hybrid Flow](#).




1. The user clicks **Login** in the app.
2. Auth0's SDK redirects the user to the Auth0 Authorization Server (`/authorize` endpoint) passing along a `response_type` parameter of `id_token` that indicates the type of requested credential. It also passes along a `response_mode` parameter of `form_post` to ensure security.
3. Your Auth0 Authorization Server redirects the user to the login and authorization prompt.
4. The user authenticates using one of the configured login options and may see a consent page listing the permissions Auth0 will give to the app.

5. Your Auth0 Authorization Server redirects the user back to the app with an ID Token.

How to implement it

You can use our [Express OpenID Connect SDK](#) to securely implement the Implicit Flow with Form Post.

 The [Auth0 Single-Page App SDK](#) and [Single-Page Quickstarts](#) adhere to the new recommendations and use the [Authorization Code Flow with PKCE](#).

Finally, you can follow our tutorials to use our API endpoints to [Add Login Using the Implicit Flow with Form Post](#).

Keep reading

- [Auth0 Rules](#)
- [Auth0 Hooks](#)
- [Tokens](#)
- [Token Best Practices](#)
- [Which OAuth 2.0 Flow Should I Use?](#)
- [Mitigate Replay Attacks When Using the Implicit Flow](#)

Was this article helpful?



YES



NO



PRODUCT

[Pricing](#)

[Why Auth0](#)

[How It Works](#)

[Lock](#)

COMPANY

[About Us](#)

[Blog](#)

[Jobs](#)

Press

LEARN

Availability & Trust

Security

White Hat

API Explorer

MORE

Help & Support

Professional Services

Documentation

Open Source

WordPress

CONTACT

10800 NE 8th Street

Suite 600

Bellevue, WA 98004

+1 (888) 235-2699

+1 (425) 312-6521

+44 (0) 33-3234-1966

Follow 14 086

Follow 5 412

Like 14 395

[Privacy Policy](#) [Terms of Service](#) © 2013-2021 Auth0®, Inc. All Rights Reserved.