[IdentityServer4 vs AspNet.Security.OpenIdConnect.Server vs OpenIddict](https://stackoverflow.com/questions/46608920/identityserver4-vs-aspnet-security-openidconnect-server-vs-openiddict)

In an attempt to understand what I may use for OpenId Connect Server implementation, I have looked into what each of them is:

* [IdentityServer4](https://github.com/IdentityServer/IdentityServer4):

an OpenID Connect and OAuth 2.0 framework for ASP.NET Core 2.

* [AspNet.Security.OpenIdConnect.Server](https://github.com/aspnet-contrib/AspNet.Security.OpenIdConnect.Server):

is an advanced OAuth2/OpenID Connect server framework for both ASP.NET Core 1.x/2.x and OWIN/Katana 3.x/4.x, designed to offer a low-level, protocol-first approach.

* [OpenIddict](https://github.com/openiddict/openiddict-core):

OpenIddict aims at providing a simple and easy-to-use solution to implement an OpenID Connect server in any ASP.NET Core 1.x or 2.x application.

OpenIddict is based on AspNet.Security.OpenIdConnect.Server to control the OpenID Connect authentication flow and can be used with any membership stack, including ASP.NET Core Identity.

* Also have checked that all of them use well the [ASP.NET Core Identity](https://github.com/aspnet/Identity) as a membership system.

And so my current understanding is that IdentityServer4 and OpenIdConnect.Server are two alternative frameworks that solve the same problem. The main difference is the list of supported ASP.NET Core versions.

Regarding Openiddict - it is a kind of extension to simplify server creation based on AspNet.Security.OpenIdConnect.Server.

Actually, I believe the most important difference is that **these two libs don't share the same objective**. ASOS' only mission is to help you deal with the raw OAuth2/OIDC protocol details: everything else is totally out of scope. Concretely, this means that concepts like users, applications or stores - that you can find in OpenIddict and IdentityServer - are completely inexistant in ASOS (which means you're free to bring your own implementation... and your own abstraction).

While IdentityServer will expose many abstractions and services allowing to configure specific features, ASOS - that was forked from Katana's OAuthAuthorizationServerMiddleware - has a centralized low-level events-based API (named OpenIdConnectServerProvider) that behaves exactly the same way as the ASP.NET Core security middleware developed by MSFT.

When working with ASOS, you have to deal with raw OpenID Connect requests and implement potentially sensitive things like client authentication (e.g using a database containing the client credentials) and that's why ASOS' core target is people who understand how the OAuth2/OIDC protocol work. OpenIddict and IdentityServer, on the other hand, will implement these things for you.

Regarding Openiddict - it is a kind of extension to simplify server creation based on AspNet.Security.OpenIdConnect.Server.

Initially, that's indeed how I was asked to design it. OpenIddict was created for non-experts who don't feel super comfortable with the protocol details of OAuth2 and OpenID Connect.

While it will give you full flexibility to implement the user authentication part (e.g in your own authorization controller, using ASP.NET Core Identity or your own authentication method), it will handle the complex request validation process and make it transparent for your app, without drowning you with tons of configuration options.

Unlike ASOS (that tries to be as flexible and as close to the specifications as possible), OpenIddict generally comes with more restrictive validation routines that I personally consider as best practices. For instance, it will automatically reject authorization requests that contain response\_type=token if the client is a confidential application, even if that's not prohibited by the OpenID Connect specification.