[Robin van der Knaap](https://dev.to/robinvanderknaap)

[Robin van der Knaap](https://dev.to/robinvanderknaap)

Posted on Dec 11, 2020 • Updated on Dec 15, 2020

**Setting up an Authorization Server with OpenIddict - Part V - OpenID Connect**

[#dotnet](https://dev.to/t/dotnet)[#oauth2](https://dev.to/t/oauth2)[#openiddict](https://dev.to/t/openiddict)[#aspnet](https://dev.to/t/aspnet)

This article is part of a series called **Setting up an Authorization Server with OpenIddict**. The articles in this series will guide you through the process of setting up an OAuth2 + OpenID Connect authorization server on the the [ASPNET Core](https://dotnet.microsoft.com/apps/aspnet) platform using [OpenIddict](https://github.com/openiddict/openiddict-core).

* [Part I: Introduction](https://dev.to/robinvanderknaap/setting-up-an-authorization-server-with-openiddict-part-i-introduction-4jid)
* [Part II: Create ASPNET project](https://dev.to/robinvanderknaap/setting-up-an-authorization-server-with-openiddict-part-ii-create-aspnet-project-4949)
* [Part III: Client Credentials Flow](https://dev.to/robinvanderknaap/setting-up-an-authorization-server-with-openiddict-part-iii-client-credentials-flow-55lp)
* [Part IV: Authorization Code Flow](https://dev.to/robinvanderknaap/setting-up-an-authorization-server-with-openiddict-part-iv-authorization-code-flow-3eh8)
* **Part V: OpenID Connect**
* [Part VI: Refresh tokens](https://dev.to/robinvanderknaap/setting-up-an-authorization-server-with-openiddict-part-vi-refresh-tokens-5669)

GitHub logo [robinvanderknaap](https://github.com/robinvanderknaap)/ [**authorization-server-openiddict**](https://github.com/robinvanderknaap/authorization-server-openiddict)

Authorization Server implemented with OpenIddict.

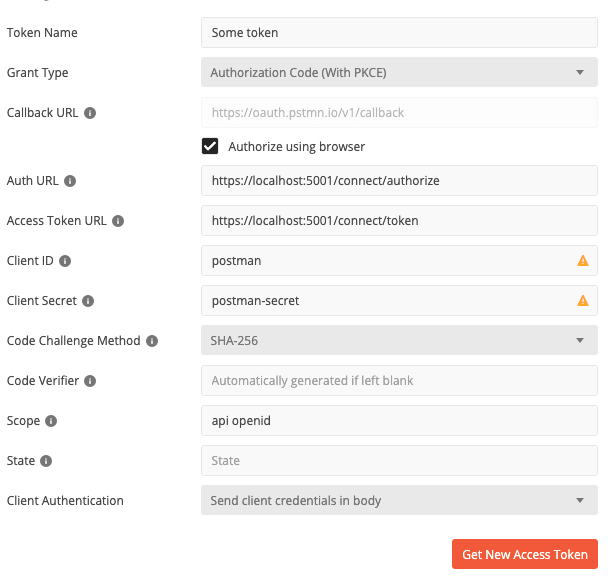
OpenIddict implements the OpenID Connect protocol, which is an identity layer on top of the OAuth2 protocol.  
The implementation of the OpenID Connect protocol issues an extra token to the client application, called the identity token. This token contains user profile information which can be used by client applications to identify the end-user.

[It's wise to keep your tokens small](https://leastprivilege.com/2016/12/14/optimizing-identity-tokens-for-size/). Therefore, the OpenID Connect protocol offers the possibility to expose an userinfo endpoint from which clients can retrieve extra information about the end-user which is not stored in the identity token.

In this part we will see how to leverage the OpenID Connect protocol and how to add an endpoint for querying extra user information.

**Enable OpenID Connect**

As said, OpenIddict has implemented the OpenID Connect protocol. You can test this by requesting an extra scope called openid with Postman:

[](https://res.cloudinary.com/practicaldev/image/fetch/s--g8IllWeK--/c_limit%2Cf_auto%2Cfl_progressive%2Cq_auto%2Cw_880/https:/dev-to-uploads.s3.amazonaws.com/i/j51cb90e7fft92k38xvh.png)

In the response you will find an identity token together with the access token.

To store extra profile information in the identity token, you can add claims in the Authorize method in the AuthorizationController when using the authorization code flow or add claims in the token endpoint if you are using the client credentials flow. Make sure to set the destination to IdentityToken. Here's an example using a default claim called email:

public async Task<IActionResult> Authorize()

{

...

var claims = new List<Claim>

{

// 'subject' claim which is required

new Claim(OpenIddictConstants.Claims.Subject, result.Principal.Identity.Name),

new Claim("some claim", "some value").SetDestinations(OpenIddictConstants.Destinations.AccessToken),

new Claim(OpenIddictConstants.Claims.Email, "some@email").SetDestinations(OpenIddictConstants.Destinations.IdentityToken)

};

...

}

It is also possible for a claim to have multiple destinations, so we can add a claim to the access token and the identity token if we want.

You can view the content of the identity token on [jwt.io](https://jwt.io/):

{

"sub": "Robin van der Knaap",

"email": "some@email",

"oi\_au\_id": "e5e12b78-ddea-42df-8350-5b7bda36b07b",

"azp": "postman",

"at\_hash": "iuzdujyas1p\_1XjUuxtw9A",

"oi\_tkn\_id": "7f0ae652-c997-4425-bda5-6d024ea3acb3",

"aud": "postman",

"exp": 1607684956,

"iss": "https://localhost:5001/",

"iat": 1607683756

}

**Userinfo endpoint**

Besides the identity token, the OpenID Connect protocols also allows for an url to get information about the user. To leverage the userinfo endpoint, we need to enable the endpoint and set the url in Startup.cs first:

options

.SetAuthorizationEndpointUris("/connect/authorize")

.SetTokenEndpointUris("/connect/token")

.SetUserinfoEndpointUris("/connect/userinfo");

options

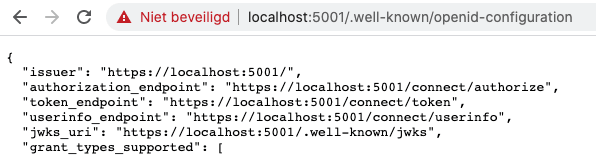
.UseAspNetCore()

.EnableTokenEndpointPassthrough()

.EnableAuthorizationEndpointPassthrough()

.EnableUserinfoEndpointPassthrough();

Now if you restart the authorization server and navigate to <https://localhost:5001/.well-known/openid-configuration>. You will see that the userinfo endpoint is added to public configuration file:

[](https://res.cloudinary.com/practicaldev/image/fetch/s--rpUQuo8B--/c_limit%2Cf_auto%2Cfl_progressive%2Cq_auto%2Cw_880/https:/dev-to-uploads.s3.amazonaws.com/i/owh28639ry3i7mni27oj.png)

Next step is to implement the userinfo endpoint. We can decide for ourselves which user info we want to share with clients. We will add the method to the AuthorizationController:

[Authorize(AuthenticationSchemes = OpenIddictServerAspNetCoreDefaults.AuthenticationScheme)]

[HttpGet("~/connect/userinfo")]

public async Task<IActionResult> Userinfo()

{

var claimsPrincipal = (await HttpContext.AuthenticateAsync(OpenIddictServerAspNetCoreDefaults.AuthenticationScheme)).Principal;

return Ok(new

{

Name = claimsPrincipal.GetClaim(OpenIddictConstants.Claims.Subject),

Occupation = "Developer",

Age = 43

});

}

The user info endpoint can only be accessed when the client has a valid access token. This is why we added the Authorize attribute. For this to work, we have to enable authorization in Startup.cs:

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

...

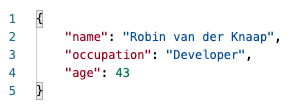
app.UseAuthentication();

app.UseAuthorization();

...

}

Now if we do a GET request to /connect/userinfo and use a valid access token, the response looks something like this:

[](https://res.cloudinary.com/practicaldev/image/fetch/s--pGvUzzMC--/c_limit%2Cf_auto%2Cfl_progressive%2Cq_auto%2Cw_880/https:/dev-to-uploads.s3.amazonaws.com/i/6w7e51in6s6ib8cozty3.png)

**Next**

[Next up](https://dev.to/robinvanderknaap/setting-up-an-authorization-server-with-openiddict-part-vi-refresh-tokens-5669), we will enable the usage of refresh tokens.

**Discussion (8)**

Subscribe

Top of Form





Bottom of Form

[rohans profile image](https://dev.to/rohans)

Rohan Singh

•[Apr 7 '21](https://dev.to/rohans/comment/1d74c)

I could not get authorization to work without:

1) Enable validation (note: this snippet only works for authorizing requests within the same endpoint as the authorization server, such as the userinfo API)

services.AddOpenIddict()

.AddServer(() =>

{

// ...

})

.AddValidation(options =>

{

// Import the configuration from the local OpenIddict server instance.

options.UseLocalServer();

// Register the ASP.NET Core host.

options.UseAspNetCore();

});

2) Use the validation authentication scheme to authorize instead of the server's scheme

// Replace this:

//[Authorize(AuthenticationSchemes = OpenIddictServerAspNetCoreDefaults.AuthenticationScheme)]

// With this:

[Authorize(AuthenticationSchemes = OpenIddictValidationAspNetCoreDefaults.AuthenticationScheme)]

3 likes[Reply](https://dev.to/robinvanderknaap/setting-up-an-authorization-server-with-openiddict-part-v-openid-connect-a8j#/robinvanderknaap/setting-up-an-authorization-server-with-openiddict-part-v-openid-connect-a8j/comments/new/1d74c)

[joepb profile image](https://dev.to/joepb)

Joep Beusenberg

•[May 14 '21](https://dev.to/joepb/comment/1ef46)

Awesome catch. I think I'd never figured this out if you hadn't added this.  
Also in the line below, you will need to use OpenIddictValidationAspNetCoreDefaults. Then it works flawlessly.

2 likes[Reply](https://dev.to/robinvanderknaap/setting-up-an-authorization-server-with-openiddict-part-v-openid-connect-a8j#/robinvanderknaap/setting-up-an-authorization-server-with-openiddict-part-v-openid-connect-a8j/comments/new/1ef46)

[keithn profile image](https://dev.to/keithn)

Keith Nicholas

•[Jan 29 '21](https://dev.to/keithn/comment/1b0p7)

Part VI - Integrating External Authorization Providers ... google/azure/facebook/twitter.... etc ? :)

6 likes[Reply](https://dev.to/robinvanderknaap/setting-up-an-authorization-server-with-openiddict-part-v-openid-connect-a8j#/robinvanderknaap/setting-up-an-authorization-server-with-openiddict-part-v-openid-connect-a8j/comments/new/1b0p7)

[hdsoftware profile image](https://dev.to/hdsoftware)

HD Software

•[Aug 23 '21](https://dev.to/hdsoftware/comment/1hdka)

Really nice article :D

Can anyone please give an example on how to implement OpenIddict with the following situation:

3 apps, where app1 is the ID server, app2 as the API server and app3 as a client application.

Meaning no Postman, no WEB browser involvement.

I have tried to understand the "Zirku" example shipped with OpenIddict, but - I have no idea how to read the Zirku.Client

What I nead is plain and simple:  
Client app asking the IdServer for a token, based on Username&password  
Client app asking the ApiServer for something, passing along the token.  
  
As simple as absolutly possible.

I tried to implement something like the "Hollastin" example, but didnt manage to get it running correctly.

So, an example in plain C# would be extreemly nice to see.

2 likes[Reply](https://dev.to/robinvanderknaap/setting-up-an-authorization-server-with-openiddict-part-v-openid-connect-a8j#/robinvanderknaap/setting-up-an-authorization-server-with-openiddict-part-v-openid-connect-a8j/comments/new/1hdka)

[nonsenseless profile image](https://dev.to/nonsenseless)

nonsenseless

•[Oct 27 '21](https://dev.to/nonsenseless/comment/1j8c4)

This is something I've been looking for and I think the confusion is in expecting open id to work with username/password. Hopefully [@robinvanderknaap](https://dev.to/robinvanderknaap) can clarify, but the oauth flows are built around supporting communication between servers or between clients and the server via third-party login. Logging in with Username/Password is basically what you set up in part 2 and can use the more normal AspNetCore individual authentication.

I'd definitely be interested in more tips on how to combine OpenIddict and the standard motely array of AspNetUsers, AspNetRoles, etc.

1 like[Reply](https://dev.to/robinvanderknaap/setting-up-an-authorization-server-with-openiddict-part-v-openid-connect-a8j#/robinvanderknaap/setting-up-an-authorization-server-with-openiddict-part-v-openid-connect-a8j/comments/new/1j8c4)

[zizounetgit profile image](https://dev.to/zizounetgit)

zizounetgit

•[Dec 30 '21](https://dev.to/zizounetgit/comment/1ko3c)

I still got this even if i Use the validation authentication scheme  
The userinfo request was rejected because the mandatory 'access\_token' parameter was missing.  
info: OpenIddict.Server.OpenIddictServerDispatcher[0]  
The response was successfully returned as a challenge response: {  
"error": "missing\_token",  
"error\_description": "The mandatory 'access\_token' parameter is missing.",  
"error\_uri": "[documentation.openiddict.com/error...](https://documentation.openiddict.com/errors/ID2029)"  
}.

1 like[Reply](https://dev.to/robinvanderknaap/setting-up-an-authorization-server-with-openiddict-part-v-openid-connect-a8j#/robinvanderknaap/setting-up-an-authorization-server-with-openiddict-part-v-openid-connect-a8j/comments/new/1ko3c)

[svenni profile image](https://dev.to/svenni)

svenniuwe

•[Jun 22 '21](https://dev.to/svenni/comment/1fia6)

For me it is working without any change.

Like[Reply](https://dev.to/robinvanderknaap/setting-up-an-authorization-server-with-openiddict-part-v-openid-connect-a8j#/robinvanderknaap/setting-up-an-authorization-server-with-openiddict-part-v-openid-connect-a8j/comments/new/1fia6)

[xargon180 profile image](https://dev.to/xargon180)

xargon180

•[Dec 14 '21](https://dev.to/xargon180/comment/1kfec)

Same here. Worked without any change. Realy great article.

2 likes[Reply](https://dev.to/robinvanderknaap/setting-up-an-authorization-server-with-openiddict-part-v-openid-connect-a8j#/robinvanderknaap/setting-up-an-authorization-server-with-openiddict-part-v-openid-connect-a8j/comments/new/1kfec)

[Code of Conduct](https://dev.to/code-of-conduct) • [Report abuse](https://dev.to/report-abuse)