# Welcome to SimpleIdServer

SimpleIdServer is a refactored version of the SimpleIdentityServer project (<https://github.com/thabart/SimpleIdentityServer>). It keeps the OPENID and OAUTH2.0 logic to respect the security standards but offers less dependencies with third parties like Entity Framework and less Nuget packages. Instead of installing a bunch of Nuget packages to create a OAUTH2.0 API only two are now needed.

SimpleIdServer aims to be an open source identity and authorization manager.

It is developed in DOTNET CORE version 2.2.

## OAUTH2.0 server

### Getting started

An OAUTH2.0 server can be deployed into an ASP.NET CORE project by installing the Nuget package

“SimpleIdServer.OAuth”. As a developer you can follow the steps below to deploy a local version:

1. Create an empty ASP.NET CORE project.
2. Install the Nuget package “SimpleIdServer.OAuth”.
3. Open the Startup.cs file and register the dependencies by inserting the instruction “services.AddSIDOAuth()”.
4. Always in the “Startup.cs” file configure the application builder by inserting the instruction “services.UseSID()”.
5. Launch the solution.
6. Check if JSON are returned by the relative URL “/.well-known/oauth-authorization-server”.

Default OAUTH2.0 clients, OAUTH2.0 scopes, users and JSON Web Keys are stored in memory and they can be overridden by using the operations returned by the instruction “services.AddSIDOAuth()”

* AddClients(List<OAuthClient>) : add list of OAUTH2.0 clients.
* AddScopes(List<OAuthScope>) : add list of OAUTH2.0 scopes.
* AddUsers(List<OAuthUser>) : add list of users.
* AddJsonWebKeys(List<JsonWebKey>) : add list of JSON Web keys.

OAUTH2.0 default settings can be modified by manipulating the option “OAuthHostOptions”.

The following chapter explains how to do it and list the properties.

### OAuth2.0 options

The OAUTH2.0 framework takes its configuration from the “OAuthHostOptions” option class.

The properties can be changed in the Startup class like this:

**public void ConfigureServices(IServiceCollection services)**

**{**

**services.AddSIDOAuth(o =>**

**{**

**o.ClientSecretExpirationInSeconds = 2;**

**})**

**}**

The OAuthHostOptions class contains the following properties:

**DefaultScopes** : Default scopes assigned to a OAUTH2.0 client during the registration process.

**DefaultTokenProfile** : Default token profile assigned to a OAUTH2.0 client during the registration process. The possible values are “mac” or “bearer”.

**DefaultOAuthScopes** : Default OAUTH2.0 scopes. Refer to the chapter “OAuth2.0 scope’s properties” for more information about the scope’s properties.

**SoftwareStatementTrustedParties** : If a “software\_statement” parameter is passed to the client registration request then the property “SoftwareStatementTrustedParties” is used by the authorization process to check the parameter. The validation process is made of two steps :

* Fetch the “iss” parameter from the JWS header and get the corresponding JWKS URL from “SoftwareStatementTrustedParties”.
* Fetch the JSON Web Key from the URL and check the signature of the “software\_statement” parameter.

**DefaultCulture** : Default culture used by the UI if there is no “ui\_locales” parameter specified in the authorization request.

### OAuth2.0 client’s properties

**ClientId** : client identifier.

**Secrets** : one or more client secrets.

**ClientNames** : one or more human readable client name.

**LogoUris** : one or more URL that references a logo for the client.

**ClientUris** : one or more URL of a web page providing information about the client.

**PolicyUris** : one or more URL that points to a human-readable policy document for the client.

**TosUris** : one or more URL that points to a human-readable terms of service document for the client.

**TokenSignedResponseAlg** : Cryptographic algorithm used to secure the JWS access token.

**TokenEncryptedResponseAlg** :Cryptographic algorithm used to encrypt the JWS access token.

**TokenEncryptedResponseEnc** : Content encryption algorithm used perform authenticated encryption on the JWS access token.

**TokenEndpointAuthMethod** : Requested authentication method for the token endpoint. The possible values are :

* “client\_secret\_post”
* “client\_secret\_basic”
* “private\_key\_jwt”
* “client\_secret\_jwt”

**GrantTypes** : Array of OAUTH2.0 grant type strings that the client can use at the token endpoint. The possible values are :

* “authorization\_code”
* “implicit”
* “password”
* “client\_credentials”
* “refresh\_token”

**ResponseTypes** : Array of the OAUTH2.0 response type strings that the client can use at the authorization endpoint. The possible values are :

* “code”
* “token”

**AllowedScopes** : Scope values that the client can use when requesting access tokens.

**RedirectionUrls** : Array of redirection URIS for use in redirect-based flows.

**JwksUri** : URI string referencing the client’s JSON Web Key (JWK) Set document, which contains the client’s public keys.

**JsonWebKeys** : Client’s JSON Web Key Set document value, which contains the client’s public keys.

**TokenExpirationTimeInSeconds** : Token expiration time in seconds.

**RefreshTokenExpirationTimeInSeconds** : Refresh token expiration time in seconds.

**PreferredTokenProfile** : preferred token profile, possible values are : “bearer” or “mac”.

**Contacts** : Array of strings representing was to contact people responsible for the client, typically email addresses.

**SoftwareId** : A unique identifier assigned by the client developer or software publisher used by registration endpoints to identify the client software to be dynamically registered.

**SoftwareVersion** : A version identifier string for the client software identified by “software\_id”.

### OAuth2.0 scope’s properties

**Name** : name of the OAUTH2.0 scope.

**IsExposedInConfigurationEdp** : If the value is true then the scope is returned by the configuration endpoint “*.well-known/oauth-authorization-server*”.

### OAuth2.0 user’s properties

**Id** :identifier of the user.

**Claims** : list of information that identify an end user.

**Consents** : list of user’s consents.

**Credentials** : list of user’s credentials.

### OAuth2.0 Json Web Key (JWK)

A FAIRE.

TODOLIST :

* Tester tous les paramètres pour l’enregistrement d’un client OAUTH2.0.
* Ajouter l’authentification de type “tls\_client\_auth” (<https://tools.ietf.org/html/draft-ietf-oauth-mtls-08#section-2.1>)
* Ajouter PKCE (<https://tools.ietf.org/html/rfc7636>).
* Ajouter session management (<https://openid.net/specs/openid-connect-session-1_0.html>).