OAuthClientExample

README

Last Updated August 6th, 2019

Contents

[Overview 3](#_Toc15984343)

[Registering the Client with Auth0 4](#_Toc15984344)

[Persistent, Encrypted Storage of Sensitive Data 4](#_Toc15984345)

[Equipment 5](#_Toc15984346)

[Devices 5](#_Toc15984347)

[Software / Firmware 5](#_Toc15984348)

[Important Notes 5](#_Toc15984349)

[Before Loading the Program 5](#_Toc15984350)

[Loading the Program and Project Files 5](#_Toc15984351)

# Overview

This OAuthClientExample project was created to exercise the HttpCwsServer, HttpsClient, CrestronDataStoreStatic, and related types from the SIMPL# Pro library. The program implements an OAuth 2.0 client in SIMPL# Pro to demonstrate the application of these types.

The OAuth2 client serves a website rooted at https://<control-system-hostname-or-IP>/cws/, where <control-system-hostname-or-IP> represents the control system’s hostname or IP address. The user navigating to this URL in a standard web browser will find an HTML web interface guiding them through the process of registering the OAuth client with an “Authorization Server” (which is provided by [Auth0](https://auth0.com/)), permitting the client to access a protected resource via an OAuth Access Token, testing the request to the protected resource, and disposing of the OAuth tokens. Specifically, the web interface provides the following buttons:

* Register Client with Auth0
* Request Access/Refresh Tokens
* Test
* Forget Access Token
* Forget Refresh Token

By clicking “Test,” the user may observe how the attempt to retrieve the protected resource fails when the client possesses neither a valid Access nor Refresh Token, while it succeeds once the user has authorized the client to access this resource. The protected resource itself consists of user profile information: the user authorizes the client to read the name, email address, email verification flag, and profile picture of the user that authenticates to Auth0’s server and accepts the client’s authorization request. The user who authenticates to Auth0 and subsequently chooses to accept or deny the client’s authorization request is therefore the “Resource Owner” in OAuth’s vocabulary.

All SIMPL# Pro source code defining the program is provided in the following files:

* ControlSystem.cs
* RequestHandler.cs
* EncryptionLayer.cs

These source files are thoroughly commented and are intended to provide an illustrative application of various SIMPL# Pro APIs.

## Registering the Client with Auth0

All OAuth clients must undergo a registration process before they can perform an authorization flow. Registration entails associating the client with an authorization server; the server generates a Client ID and Client Secret for the client and provides the Authorization and Token endpoints that the client will later navigate (or direct the user agent) to. Registration, then, is simply the implementation-defined process of communicating the Client ID and Client Secret to the client itself.

In this example project, Auth0 fills the role of the authorization server; it also allows the application owner to add user accounts whose information acts as the protected resource that users may authorize a client to work with. Readers intending to demo this example project must first make an Auth0 account, access the Settings page of an Auth0 application in the dashboard, plug the client’s callback URL into the “Allowed Callback URLs” field, and copy and paste the auto-generated Domain, Client ID, Client Secret into a submission form on the client’s website. Both this submission form and a step-by-step explanation of the registration process can be found when the user clicks “Register Client with Auth0” with an unregistered client.

## Persistent, Encrypted Storage of Sensitive Data

Once authorized, the OAuth client can continue to retrieve the protected resource even after the Access Token expires, is forgotten via the web interface button, or lost due to a reboot. This is accomplished through a Refresh Token, a sensitive client resource that is encrypted with an RSA public key, then encoded and persisted as ciphertext in the Crestron DataStore. The Client Secret is also a sensitive credential, akin to a password, and it therefore is also encrypted before being stored in the DataStore. The Client ID, the Auth0 Domain name, and two Boolean flags indicating whether the client is registered or whether it has a Refresh Token are also persisted as cleartext, since they are not sensitive pieces of information.

If the Refresh Token is forgotten via the client web interface, revoked via the User & Roles -> Users -> (current user) -> Authorized Applications menu in the Auth0 Dashboard, or invalidated by any other means, the client will have no means of accessing the protected resource without requesting re-authorization from the user.

**Note:** The RSA public and private key pair used to encrypt and decrypt the Client Secret and Refresh Token are persisted in a Windows Key Container, accessed in SIMPL# Pro via the CspParameters class, an instance of which is passed to the RSACryptoServiceProvider constructor. Since this key container is only available for use with [asymmetric](https://docs.microsoft.com/en-us/dotnet/standard/security/how-to-store-asymmetric-keys-in-a-key-container?view=netframework-4.8) keys, a more efficient symmetric key could not be used to implement the cryptography.

# Equipment

This program is designed to work with the following hardware/software:

## Devices

* 3-Series Control System
* Any device with a web browser

## Software / Firmware

|  |  |
| --- | --- |
| Device | Firmware Version |
| 3-Series Control System | 1.601.xxx or later |

|  |  |
| --- | --- |
| Software | Version |
| Toolbox | 3.03.xxx or later |

# Important Notes

## Before Loading the Program

* Turn on SSL so that the CWS server uses HTTPS
* Turn on Authentication so that users must be logged into the control system to navigate the website. This prevents unauthorized users from using/forgetting the tokens through the website.
* Optionally, configure your web browser to trust the certificate from the control system, if such trust has not already been established. Otherwise, it may be necessary to click through security warnings from your browser about an untrusted certificate to access the website via HTTPS.

# Loading the Program and Project Files

1. Open the top-level solution file, OAuthClientExample.sln, in Visual Studio 2008.
2. Press F6 to Build All. This will place the .cpz file in the OAuthClientExample\bin\Debug folder.
3. Transfer the .cpz file to one of the control system’s program slots.
4. On the control system, invoke **progload -p:#**, where **#** is the number of the slot into which you placed the .cpz file.