

PowerBuilder

Code for nerds, stuff that matters

How to Add Multiple Authentication Providers to an Optimizely CMS 12 Site (Entra ID, Google, Facebook, and Local Identity)

ON 22 OCTOBER, 2025 / BY FRANCISCO QUINTANILLA / IN OPTIMIZEZY

Modern websites often need to let users sign in with their **corporate account (Entra ID)**, their **social identity (Google, Facebook)**, or a simple **email/password** for internal users.

If you're building on **Optimizely CMS 12**, you can support *all of them at once* — without breaking the built-in CMS login system.

In this post, we'll walk through how to:

- Keep **Optimizely CMS 12 local Identity** (email/password) at **/util/login**
- Add **Entra ID (Azure AD)** at **/login**
- Add **Google** at **/login/google**
- Add **Facebook** at **/login/facebook**
- Use **one shared cookie** for all external providers
- Automatically **synchronize users/roles** into Optimizely

NuGet packages to install

```
1 dotnet add package EPiServer.CMS.UI.AspNetIden
2 dotnet add package Microsoft.AspNetCore.Authen
3 dotnet add package Microsoft.IdentityModel.Pro
4 dotnet add package Microsoft.AspNetCore.Authen
5 dotnet add package Microsoft.AspNetCore.Authen
```

You already have Optimizely CMS 12 packages in your project; the first package above brings in the Identity helpers used by the CMS UI.

Files to add

```
1 // File: Infrastructure/Security/AuthenticationExtensions.cs
2 using System.Security.Claims;
3 using System.Text;
4 using EPiServer.Cms.UI.AspNetIdentity;
5 using EPiServer.ServiceLocation;
6 using EPiServer.Shell.Security;
7 using Microsoft.AspNetCore.Authentication;
8 using Microsoft.AspNetCore.Authentication.Cookies;
9 using Microsoft.AspNetCore.Authentication.Facebook;
10 using Microsoft.AspNetCore.Authentication.Google;
11 using Microsoft.AspNetCore.Authentication.OpenIdConnect;
12 using Microsoft.AspNetCore.Http;
13 using Microsoft.Extensions.Configuration;
14 using Microsoft.Extensions.DependencyInjection;
15 using Microsoft.IdentityModel.Protocols.OpenIdConnect;
16 using Microsoft.IdentityModel.Tokens;
17
18 namespace YourNamespace.Infrastructure.Security
19 {
20     public static class AuthenticationExtensions
21     {
22         private const string ExternalAppCookieName = "ExternalAppCookie";
23
24         /// <summary>
25         /// Keep Optimizely CMS local Identity
26         /// </summary>
27         public static IServiceCollection UseOptimizelyCMS(
28             this IServiceCollection services,
29             IConfiguration configuration)
30         {
31             services.AddCmsAspNetIdentity<User>(
32                 configuration,
33                 var conn = configuration.GetConnectionString(
34                     if (!string.IsNullOrEmpty(
35                         o.ConnectionStringOption
36                         {
37                             Name = "EPiServerDB"
38                             ConnectionString = configuration
39                         }
40                     });
41             });
42
43             return services;
44         }
45     }
46
47     /// <summary>
48     /// Adds Entra ID (Azure AD) via OpenIdConnect
49     /// Required config keys:
50     /// Authentication:AzureClientID, Authentication:AzureTenantID
51     /// </summary>
```

```

52 public static IServiceCollection Use
53 {
54     Microsoft.IdentityModel.Logging.
55
56     var clientId      = configurati
57     var clientSecret  = configurati
58     var azureAuthority = configurati
59     var callbackPath  = configurati
60
61     services.AddAuthentication()
62         .AddCookie(ExternalAppCookie
63         {
64             // Any external login th
65             options.Events = new Co
66             {
67                 OnSignedIn = async c
68                 {
69                     if (ctx.Principa
70                     {
71                         var sync = c
72                         await sync.S
73                     }
74                 }
75             };
76         })
77         .AddOpenIdConnect("azure", o
78         {
79             options.SignInScheme = E
80             options.ResponseType = O
81             options.UsePkce = true;
82
83             options.ClientId = clien
84             options.ClientSecret = c
85             options.Authority = azur
86             options.CallbackPath = n
87
88             // Ensure standard profi
89             options.Scope.Clear();
90             options.Scope.Add(OpenId
91             options.Scope.Add(OpenId
92             options.Scope.Add(OpenId
93
94             options.MapInboundClaims
95             options.GetClaimsFromUse
96
97             options.ClaimActions.Map
98             options.ClaimActions.Map
99             options.ClaimActions.Map
100
101             options.TokenValidationP
102             {
103                 RoleClaimType = "rol
104                 NameClaimType = "pre
105                 ValidateIssuer = fal
106             };
107
108             options.Events = new Ope
109             {
110                 OnRedirectToIdentity

```

```

111         {
112             if (ctx.Response
113             {
114                 ctx.HandleRe
115             }
116             return Task.Comp
117         },
118         OnAuthenticationFail
119         {
120             context.HandleRe
121             context.Response
122             return Task.Comp
123         },
124         OnTokenValidated = c
125         {
126             // Safe redirect
127             var redirect = c
128             if (!string.IsNu
129                 Uri.TryCreat
130                 uri.IsAbsolu
131             {
132                 ctx.Properti
133             }
134
135             // Tag provider
136             if (ctx.Principa
137             {
138                 id.AddClaim(
139             }
140
141             // Fire-and-forg
142             ServiceLocator.C
143                 .GetInstance
144                 .Synchronize
145
146             return Task.Comp
147         }
148     };
149 });
150
151     return services;
152 }
153
154 /// <summary>
155 /// Adds Google OAuth. Signs into th
156 /// </summary>
157 public static IServiceCollection Use
158 {
159     var clientId      = configuration
160     var clientSecret = configuration
161
162     services.AddAuthentication()
163         .AddGoogle("google", options
164         {
165             options.SignInScheme = E
166             options.ClientId      = c
167             options.ClientSecret = c
168
169             options.ClaimActions.Map

```

```

170         options.ClaimActions.Map
171         options.ClaimActions.Map
172
173         options.Events = new OAuth
174         {
175             OnCreatingTicket = c
176             {
177                 if (ctx.Principa
178                     id.AddClaim(
179                     return Task.Comp
180             }
181         });
182     });
183
184     return services;
185 }
186
187 /// <summary>
188 /// Adds Facebook OAuth. Signs into
189 /// Requires Authentication:Facebook
190 /// </summary>
191 public static IServiceCollection Use
192 {
193     var appId = configuration["A
194     var appSecret = configuration["A
195
196     services.AddAuthentication()
197         .AddFacebook("facebook", opt
198         {
199             options.SignInScheme = E
200
201             options.AppId = appI
202             options.AppSecret = appS
203
204             options.Scope.Clear();
205             options.Scope.Add("email
206             options.Scope.Add("publi
207
208             options.Fields.Add("emai
209             options.Fields.Add("firs
210             options.Fields.Add("last
211             options.Fields.Add("name
212
213             options.ClaimActions.Map
214             options.ClaimActions.Map
215             options.ClaimActions.Map
216
217             options.Events = new OAuth
218             {
219                 OnCreatingTicket = c
220                 {
221                     if (ctx.Principa
222                         id.AddClaim(
223                         return Task.Comp
224                 }
225             });
226     });
227
228     return services;

```

229 | }
230 | }
231 | }

```

1 // File: Infrastructure/Security/MultiAuthExt
2 using Microsoft.AspNetCore.Authentication;
3 using Microsoft.AspNetCore.Identity;
4 using Microsoft.Extensions.DependencyInjection
5
6 namespace YourNamespace.Infrastructure.Security
7 {
8     public static class MultiAuthExtensions
9     {
10         public static IServiceCollection UseM
11         {
12             const string AppCookie = "azure-c
13             const string IdentityCookie = Ide
14
15             services.AddAuthentication()
16                 // Authenticate: choose which
17                 .AddPolicyScheme("smart-auth"
18                 {
19                     options.ForwardDefaultSel
20                     {
21                         var cookies = ctx.Req
22                         if (cookies.ContainsK
23                         if (cookies.ContainsK
24                         return AppCookie; //
25                     };
26                 })
27                 // Challenge: route by path
28                 .AddPolicyScheme("smart-chall
29                 {
30                     options.ForwardDefaultSel
31                     {
32                         var path = (ctx.Reque
33
34                         if (path.StartsWith("
35                         if (path.StartsWith("
36                         if (path.StartsWith("
37                         if (path.StartsWith("
38
39                         return "azure"; // de
40                     };
41                 });
42
43                 // Make our policy schemes the de
44                 services.PostConfigure<Authentica
45                 {
46                     o.DefaultScheme =
47                     o.DefaultAuthenticateScheme =
48                     o.DefaultChallengeScheme =
49                     o.DefaultSignInScheme =
50                 });
51
52                 return services;
53             }
54         }
55     }

```



```

1 // File: Controllers/LoginController.cs
2 using Microsoft.AspNetCore.Authentication;
3 using Microsoft.AspNetCore.Mvc;
4
5 namespace YourNamespace.Controllers
6 {
7     [Route("login")]
8     public class LoginController : Controller
9     {
10         [HttpGet("")]
11         public IActionResult Microsoft([FromQuery] string? provider)
12             => Challenge(new AuthenticationProperties { RedirectUri = "/" }, provider);
13
14         [HttpGet("google")]
15         public IActionResult Google([FromQuery] string? provider)
16             => Challenge(new AuthenticationProperties { RedirectUri = "/" }, provider);
17
18         [HttpGet("facebook")]
19         public IActionResult Facebook([FromQuery] string? provider)
20             => Challenge(new AuthenticationProperties { RedirectUri = "/" }, provider);
21
22         public async Task Logout()
23         {
24             var authProvider = User.FindFirst(AuthenticationSchemes.GoogleOAuth2) ??
25                 User.FindFirst(AuthenticationSchemes.FacebookOAuth2);
26
27             if (string.IsNullOrEmpty(authProvider?.Scheme))
28             {
29                 await signInManager.SignOutAsync();
30             }
31             else
32             {
33                 await ControllerContext.HttpContext.Response.Cookies.Delete(authProvider.Scheme);
34             }
35
36             return Redirect(urlResolver.GetUrl(""));
37         }
38     }
39 }

```

In ConfigureServices :

```

1 services.UseOptimizelyCmsIdentity<YourNamespace>(configuration);
2 services.UseEntraIdForCms(configuration);
3 services.UseGoogleForCms(configuration);
4 services.UseFacebookForCms(configuration);
5 services.UseMultiAuthGateway();

```

In appsettings.json:

```

1  {
2    "ConnectionStrings": {
3      "EpiServerDB": "Server=.;Database=YourCms
4    },
5    "Authentication": {
6      "AzureClientID": "YOUR-ENTRA-CLIENT-ID",
7      "AzureClientSecret": "YOUR-ENTRA-CLIENT-S
8      "AzureAuthority": "https://login.microsof
9      "CallbackPath": "/signin-oidc",
10     "Google": {
11       "ClientId": "YOUR-GOOGLE-CLIENT-ID",
12       "ClientSecret": "YOUR-GOOGLE-CLIENT-SEC
13     },
14     "Facebook": {
15       "AppId": "YOUR-FACEBOOK-APP-ID",
16       "AppSecret": "YOUR-FACEBOOK-APP-SECRET"
17     }
18   }
19 }

```

Facebook: ensure your “Valid OAuth Redirect URI” in the Facebook app matches your site’s `https://your-host/signin-facebook` .

Google: allow `https://your-host/signin-google` .

Entra: set redirect to `https://your-host/signin-oidc` .

How it works

- `/util/login` uses **Optimizely local Identity** (email/password) — unchanged.
- `/login`, `/login/google`, `/login/facebook` route to **Entra**, **Google**, **Facebook** respectively.
- All external providers sign into the **same cookie** (`azure-cookie`).
- On cookie sign-in, we call `ISynchronizingUserService.SynchronizeAsync(...)` so users/roles appear in the CMS.
- We add an `"auth_provider"` claim so you can tell who logged the user in.

◀ [AZURE](#) ◀ [CLOUD](#) ◀ [MICROSOFT](#) ◀ [OPTIMIZELY](#) ◀ [SECURITY](#)

CREATE A FREE WEBSITE OR BLOG AT WORDPRESS.COM. DO NOT SELL OR SHARE MY PERSONAL INFORMATION