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 OPTIMIZELY

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

```
dotnet add package EPiServer.CMS.UI.AspNetIden
dotnet add package Microsoft.AspNetCore.Authen
dotnet add package Microsoft.IdentityModel.Pro
dotnet add package Microsoft.AspNetCore.Authen
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

```
// File: Infrastructure/Security/Authenticat
using System.Security.Claims;
using System.Text;
using EPiServer.Cms.UI.AspNetIdentity;
using EPiServer.ServiceLocation;
using EPiServer.Shell.Security;
using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Authentication.Co
using Microsoft.AspNetCore.Authentication.Fa
using Microsoft.AspNetCore.Authentication.Go
using Microsoft.AspNetCore.Authentication.Op
using Microsoft.AspNetCore.Http;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjecti
using Microsoft.IdentityModel.Protocols.Open
using Microsoft.IdentityModel.Tokens;

namespace YourNamespace.Infrastructure.Secur
{
    public static class AuthenticationExtens
    {
        private const string ExternalAppCook

        /// <summary>
        /// Keep Optimizely CMS local Identi
        /// </summary>
        public static IServiceCollection Use
            this IServiceCollection services
            IConfiguration configuration) wh
        {
            services.AddCmsAspNetIdentity<TU
            {
                var conn = configuration.Get
                if (!string.IsNullOrWhiteSpa
                {
                    o.ConnectionStringOption
                    {
                        Name = "EPiServerDB"
                        ConnectionString = c
                    };
                }
            });

            return services;
        }

        /// <summary>
        /// Adds Entra ID (Azure AD) via Ope
        /// Required config keys:
        /// Authentication:AzureClientID, Au
        /// </summary>
        public static IServiceCollection Use
        {
            Microsoft.IdentityModel.Logging.

            var clientId        = configurati
            var clientSecret    = configurati
            var azureAuthority = configurati
            var callbackPath    = configurati

            services.AddAuthentication()
                .AddCookie(ExternalAppCookie
                {
                    // Any external login th
                    options.Events = new Coo
                    {
                        OnSignedIn = async c
                        {
                            if (ctx.Principa
                            {
                                var sync = c
                                await sync.S
                            }
                        }
                    };
                })
                .AddOpenIdConnect("azure", o
                {
                    options.SignInScheme = E
                    options.ResponseType = O
                    options.UsePkce = true;

                    options.ClientId = clien
                    options.ClientSecret = c
                    options.Authority = azur
                    options.CallbackPath = n

                    // Ensure standard profi
                    options.Scope.Clear();
                    options.Scope.Add(OpenId
                    options.Scope.Add(OpenId
                    options.Scope.Add(OpenId

                    options.MapInboundClaims
                    options.GetClaimsFromUse

                    options.ClaimActions.Map
                    options.ClaimActions.Map
                    options.ClaimActions.Map

                    options.TokenValidationP
                    {
                        RoleClaimType = "rol
                        NameClaimType = "pre
                        ValidateIssuer = fal
                    };

                    options.Events = new Ope
                    {
                        OnRedirectToIdentity
                    {
                        if (ctx.Response
                        {
                            ctx.HandleRe
                        }
                        return Task.Comp
                    },
                    OnAuthenticationFail
                    {
                        context.HandleRe
                        context.Response
                        return Task.Comp
                    },
                    OnTokenValidated = c
                    {
                        // Safe redirect
                        var redirect = c
                        if (!string.IsNu
                            Uri.TryCreat
                            uri.IsAbsolu
                        {
                            ctx.Properti
                        }

                        // Tag provider
                        if (ctx.Principa
                        {
                            id.AddClaim(
                        }

                        // Fire-and-forg
                        ServiceLocator.C
                            .GetInstance
                            .Synchronize

                        return Task.Comp
                    }
                };
            });

        return services;
    }

    /// <summary>
    /// Adds Google OAuth. Signs into th
    /// </summary>
    public static IServiceCollection Use
    {
        var clientId     = configuration
        var clientSecret = configuration

        services.AddAuthentication()
            .AddGoogle("google", options
            {
                options.SignInScheme = E
                options.ClientId     = c
                options.ClientSecret = c

                options.ClaimActions.Map
                options.ClaimActions.Map
                options.ClaimActions.Map

                options.Events = new OAu
                {
                    OnCreatingTicket = c
                    {
                        if (ctx.Principa
                            id.AddClaim(
                        return Task.Comp
                    }
                };
            });

        return services;
    }

    /// <summary>
    /// Adds Facebook OAuth. Signs into
    /// Requires Authentication:Facebook
    /// </summary>
    public static IServiceCollection Use
    {
        var appId     = configuration["A
        var appSecret = configuration["A

        services.AddAuthentication()
            .AddFacebook("facebook", opt
            {
                options.SignInScheme = E

                options.AppId     = appI
                options.AppSecret = appS

                options.Scope.Clear();
                options.Scope.Add("email
                options.Scope.Add("publi

                options.Fields.Add("emai
                options.Fields.Add("firs
                options.Fields.Add("last
                options.Fields.Add("name

                options.ClaimActions.Map
                options.ClaimActions.Map
                options.ClaimActions.Map

                options.Events = new OAu
                {
                    OnCreatingTicket = c
                    {
                        if (ctx.Principa
                            id.AddClaim(
                        return Task.Comp
                    }
                };
            });

        return services;
```

```
            }
        }
    }
```

```
// File: Infrastructure/Security/MultiAuthExt
using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Identity;
using Microsoft.Extensions.DependencyInjectio

namespace YourNamespace.Infrastructure.Securi
{
    public static class MultiAuthExtensions
    {
        public static IServiceCollection UseM
        {
            const string AppCookie = "azure-c
            const string IdentityCookie = Ide

            services.AddAuthentication()
                // Authenticate: choose which
                .AddPolicyScheme("smart-auth"
                {
                    options.ForwardDefaultSel
                    {
                        var cookies = ctx.Req
                        if (cookies.ContainsK
                        if (cookies.ContainsK
                        return AppCookie; //
                    };
                })
                // Challenge: route by path
                .AddPolicyScheme("smart-chall
                {
                    options.ForwardDefaultSel
                    {
                        var path = (ctx.Reque

                        if (path.StartsWith("
                        if (path.StartsWith("
                        if (path.StartsWith("
                        if (path.StartsWith("

                        return "azure"; // de
                    };
                });

            // Make our policy schemes the de
            services.PostConfigure<Authentica
            {
                o.DefaultScheme             =
                o.DefaultAuthenticateScheme =
                o.DefaultChallengeScheme    =
                o.DefaultSignInScheme       =
            });

            return services;
        }
    }
}
```

```csharp
// File: Controllers/LoginController.cs
using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Mvc;

namespace YourNamespace.Controllers
{
    [Route("login")]
    public class LoginController : Controller
    {
        [HttpGet("")]
        public IActionResult Microsoft([FromQ
            => Challenge(new AuthenticationPr

        [HttpGet("google")]
        public IActionResult Google([FromQuer
            => Challenge(new AuthenticationPr

        [HttpGet("facebook")]
        public IActionResult Facebook([FromQu
            => Challenge(new AuthenticationPr

        public async Task<IActionResult> Logo
        {
            var authProvider = User.FindFirst

            if (string.IsNullOrEmpty(authProv
            {
                await signInManager.SignOutAs
            }
            else
            {
                await ControllerContext.HttpC
                HttpContext.Response.Cookies.
            }

            return Redirect(urlResolver.GetUr
        }
    }
}
```

In `ConfigureServices`:

```csharp
services.UseOptimizelyCmsIdentity<YourNamespac
services.UseEntraIdForCms(Configuration);
services.UseGoogleForCms(Configuration);
services.UseFacebookForCms(Configuration);
services.UseMultiAuthGateway();
```

In appsettings.json:

```
{
  "ConnectionStrings": {
    "EPiServerDB": "Server=.;Database=YourCms
  },
  "Authentication": {
    "AzureClientID": "YOUR-ENTRA-CLIENT-ID",
    "AzureClientSecret": "YOUR-ENTRA-CLIENT-S
    "AzureAuthority": "https://login.microsof
    "CallbackPath": "/signin-oidc",
    "Google": {
      "ClientId": "YOUR-GOOGLE-CLIENT-ID",
      "ClientSecret": "YOUR-GOOGLE-CLIENT-SEC
    },
    "Facebook": {
      "AppId": "YOUR-FACEBOOK-APP-ID",
      "AppSecret": "YOUR-FACEBOOK-APP-SECRET"
    }
  }
}
```

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