

# Share authentication cookies among ASP.NET apps

09/05/2019 • 5 minutes to read •  +1

## In this article

[Share authentication cookies with ASP.NET Core Identity](#)

[Share authentication cookies without ASP.NET Core Identity](#)

[Share cookies across different base paths](#)

[Share cookies across subdomains](#)

[Encrypt data protection keys at rest](#)

[Share authentication cookies between ASP.NET 4.x and ASP.NET Core apps](#)

[Use a common user database](#)

[Additional resources](#)

By [Rick Anderson](#)

Websites often consist of individual web apps working together. To provide a single sign-on (SSO) experience, web apps within a site must share authentication cookies. To support this scenario, the data protection stack allows sharing Katana cookie authentication and ASP.NET Core cookie authentication tickets.

In the examples that follow:

- The authentication cookie name is set to a common value of `.AspNet.SharedCookie`.
- The `AuthenticationType` is set to `Identity.Application` either explicitly or by default.
- A common app name is used to enable the data protection system to share data protection keys (`SharedCookieApp`).
- `Identity.Application` is used as the authentication scheme. Whatever scheme is used, it must be used consistently *within and across* the shared cookie apps either as the default scheme or by explicitly setting it. The scheme is used when encrypting and decrypting cookies, so a consistent scheme must be used across apps.
- A common [data protection key](#) storage location is used.
  - In ASP.NET Core apps, [PersistKeysToFileSystem](#) is used to set the key storage location.
  - In .NET Framework apps, Cookie Authentication Middleware uses an implementation of [DataProtectionProvider](#). `DataProtectionProvider` provides

data protection services for the encryption and decryption of authentication cookie payload data. The `DataProtectionProvider` instance is isolated from the data protection system used by other parts of the app.

`DataProtectionProvider.Create(System.IO.DirectoryInfo, Action<IDataProtectionBuilder>)` accepts a `DirectoryInfo` to specify the location for data protection key storage.

- `DataProtectionProvider` requires the `Microsoft.AspNetCore.DataProtection.Extensions` NuGet package:
  - In ASP.NET Core 2.x apps, reference the `Microsoft.AspNetCore.App` metapackage.
  - In .NET Framework apps, add a package reference to `Microsoft.AspNetCore.DataProtection.Extensions`.
- `SetApplicationName` sets the common app name.

## Share authentication cookies with ASP.NET Core Identity

When using ASP.NET Core Identity:

- Data protection keys and the app name must be shared among apps. A common key storage location is provided to the `PersistKeysToFileSystem` method in the following examples. Use `SetApplicationName` to configure a common shared app name (`SharedCookieApp` in the following examples). For more information, see [Configure ASP.NET Core Data Protection](#).
- Use the `ConfigureApplicationCookie` extension method to set up the data protection service for cookies.
- The default authentication type is `Identity.Application`.

In `Startup.ConfigureServices`:


C#

 Copy

```
services.AddDataProtection()  
    .PersistKeysToFileSystem( "{PATH TO COMMON KEY RING FOLDER}" )  
    .SetApplicationName( "SharedCookieApp" );  
  
services.ConfigureApplicationCookie(options => {  
    options.Cookie.Name = ".AspNet.SharedCookie";  
});
```

# Share authentication cookies without ASP.NET Core Identity

When using cookies directly without ASP.NET Core Identity, configure data protection and authentication in `Startup.ConfigureServices`. In the following example, the authentication type is set to `Identity.Application`:

C#	 Copy
<pre>services.AddDataProtection()     .PersistKeysToFileSystem( "{PATH TO COMMON KEY RING FOLDER}" )     .SetApplicationName( "SharedCookieApp" );  services.AddAuthentication( "Identity.Application" )     .AddCookie( "Identity.Application", options =&gt;     {         options.Cookie.Name = ".AspNet.SharedCookie";     } );</pre>	

## Share cookies across different base paths

An authentication cookie uses the `HttpRequest.PathBase` as its default `Cookie.Path`. If the app's cookie must be shared across different base paths, `Path` must be overridden:

C#	 Copy
<pre>services.AddDataProtection()     .PersistKeysToFileSystem( "{PATH TO COMMON KEY RING FOLDER}" )     .SetApplicationName( "SharedCookieApp" );  services.ConfigureApplicationCookie(options =&gt; {     options.Cookie.Name = ".AspNet.SharedCookie";     options.Cookie.Path = "/"; } );</pre>	

## Share cookies across subdomains

When hosting apps that share cookies across subdomains, specify a common domain in the `Cookie.Domain` property. To share cookies across apps at `contoso.com`, such as `first_subdomain.contoso.com` and `second_subdomain.contoso.com`, specify the `Cookie.Domain` as `.contoso.com`:

C#	 Copy
<pre>options.Cookie.Domain = ".contoso.com";</pre>	

## Encrypt data protection keys at rest

For production deployments, configure the `DataProtectionProvider` to encrypt keys at rest with DPAPI or an X509Certificate. For more information, see [Key encryption at rest in Windows and Azure using ASP.NET Core](#). In the following example, a certificate thumbprint is provided to [ProtectKeysWithCertificate](#):

C#	 Copy
<pre>services.AddDataProtection()     .ProtectKeysWithCertificate("{CERTIFICATE THUMBPRINT}");</pre>	

## Share authentication cookies between ASP.NET 4.x and ASP.NET Core apps

ASP.NET 4.x apps that use Katana Cookie Authentication Middleware can be configured to generate authentication cookies that are compatible with the ASP.NET Core Cookie Authentication Middleware. This allows upgrading a large site's individual apps in several steps while providing a smooth SSO experience across the site.

When an app uses Katana Cookie Authentication Middleware, it calls `UseCookieAuthentication` in the project's *Startup.Auth.cs* file. ASP.NET 4.x web app projects created with Visual Studio 2013 and later use the Katana Cookie Authentication Middleware by default. Although `UseCookieAuthentication` is obsolete and unsupported for ASP.NET Core apps, calling `UseCookieAuthentication` in an ASP.NET 4.x app that uses Katana Cookie Authentication Middleware is valid.

An ASP.NET 4.x app must target .NET Framework 4.5.1 or later. Otherwise, the necessary NuGet packages fail to install.

To share authentication cookies between an ASP.NET 4.x app and an ASP.NET Core app, configure the ASP.NET Core app as stated in the [Share authentication cookies among ASP.NET Core apps](#) section, then configure the ASP.NET 4.x app as follows.

Confirm that the app's packages are updated to the latest releases. Install the [Microsoft.Owin.Security.Interop](#) package into each ASP.NET 4.x app.


Locate and modify the call to `UseCookieAuthentication`:

- Change the cookie name to match the name used by the ASP.NET Core Cookie Authentication Middleware (`.AspNet.SharedCookie` in the example).
- In the following example, the authentication type is set to `Identity.Application`.
- Provide an instance of a `DataProtectionProvider` initialized to the common data protection key storage location.
- Confirm that the app name is set to the common app name used by all apps that share authentication cookies (`SharedCookieApp` in the example).

If not setting

`http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier` and `http://schemas.microsoft.com/accesscontrolservice/2010/07/claims/identityprovider`, set [UniqueClaimTypeIdentifier](#) to a claim that distinguishes unique users.

*App\_Start/Startup.Auth.cs:*

C#	 Copy
<pre>app.UseCookieAuthentication(new CookieAuthenticationOptions {     AuthenticationType = "Identity.Application",     CookieName = ".AspNet.SharedCookie",     LoginPath = new PathString("/Account/Login"),     Provider = new CookieAuthenticationProvider     {         OnValidateIdentity =             SecurityStampValidator                 .OnValidateIdentity&lt;ApplicationUserManager, ApplicationUser&gt;(                     validateInterval: TimeSpan.FromMinutes(30),                     regenerateIdentity: (manager, user) =&gt;                         user.GenerateUserIdentityAsync(manager))     },     TicketDataFormat = new AspNetTicketDataFormat(         new DataProtectorShim(             DataProtectionProvider.Create("{PATH TO COMMON KEY RING FOLDER}"),             (builder) =&gt; { builder.SetApplicationName("SharedCookieApp"); })         .CreateProtector(             "Microsoft.AspNetCore.Authentication.Cookies." +             "CookieAuthenticationMiddleware",</pre>	

```

        "Identity.Application",
        "v2" )),
    CookieManager = new ChunkingCookieManager()
});

System.Web.Helpers.AntiForgeryConfig.UniqueClaimTypeIdentifier =
    "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name";

```

When generating a user identity, the authentication type (`Identity.Application`) must match the type defined in `AuthenticationType` set with `UseCookieAuthentication` in *App\_Start/Startup.Auth.cs*.

*Models/IdentityModels.cs*:

C#

 Copy

```

public class ApplicationUser : IdentityUser
{
    public async Task<ClaimsIdentity> GenerateUserIdentityAsync(
        UserManager<ApplicationUser> manager)
    {
        // The authenticationType must match the one defined in
        // CookieAuthenticationOptions.AuthenticationType
        var userIdentity =
            await manager.CreateIdentityAsync(this,
                "Identity.Application");

        // Add custom user claims here

        return userIdentity;
    }
}

```

## Use a common user database

When apps use the same Identity schema (same version of Identity), confirm that the Identity system for each app is pointed at the same user database. Otherwise, the identity system produces failures at runtime when it attempts to match the information in the authentication cookie against the information in its database.

When the Identity schema is different among apps, usually because apps are using different Identity versions, sharing a common database based on the latest version of Identity isn't possible without remapping and adding columns in other app's Identity

schemas. It's often more efficient to upgrade the other apps to use the latest Identity version so that a common database can be shared by the apps.

## Additional resources

- [Host ASP.NET Core in a web farm](#)

---

Is this page helpful?

 Yes  No

---