Single sign-on is a session and user authentication service that permits users to use one set of sign-in credentials (for example, name and password) to access multiple websites. The service authenticates the user across a set of websites and eliminates further sign-in prompts when accessing different sites during an existing session.

Single sign-on can be implemented:

- on a single shared domain
- across different domains
- via the Kentico API

Note

When implementing single sign-on for websites spread across multiple Kentico instances, we strongly recommend that all instances target the same version of the Microsoft .NET Framework.

Setting up single sign-on on a shared main domain

This approach allows you to configure single sign-on for multiple sites running on subdomains of a shared main domain (for example *site1.example.com*, *site2.example.com*) on the Internet Information Services (IIS) server. Users authenticated this way have access to both the secured content on the live site and the administration interface of all Kentico instances sharing the main domain. The sites or applications **do not need** to be running on Kentico.

Single sign-on on a shared main domain is supported in the following scenarios:

Forms Authentication

To set up single sign-on with Forms authentication across applications running on a shared main domain and using the standard ASP.NET 2.0 Forms authentication, ensure that:

1. All applications share the user database or at least use the same user names. You may need to integrate the authentication using a custom security handler.
2. The *web.config* file of all applications uses the same authentication cookie name and the path is set to "/":

```
<authentication mode="Forms">
  <forms name=".ASPXFORMSAUTH" path="/" ... />
</authentication>
```

3. The *web.config* file of all applications uses the same machine key.

   - The **machineKey** element is not present in the web.config by default.
   - You can use a PowerShell script to generate the *machineKey* element by following this article from Microsoft. Insert the generated *machineKey* into the *<system.web>* section in the application's **web.config** file:

```
<system.web>
 ...
  <machineKey decryption="..." decryptionKey="..." validation="..."
validationKey="..."  />
 ...
</system.web>
```

4. If your applications run on different sub-domains, such as *www.example.com* and *forums.example.com*, you need to set the **d omain** attribute of the forms-authentication cookie to the main shared domain:

```
<forms name=".ASPXFORMSAUTH" path="/" domain=".mywebsite.com" ... />
```

Windows Authentication

If you are using the Windows AD authentication, the user identity is shared within the Windows domain. No additional configuration is required.

Setting up single sign-on across different domains

Single sign-on across different domains can be enabled only for site switching via the administration interface. This approach requires all sites to be running on a **single** Kentico instance. The sites can still use completely different domains.

To enable single sign-on across different domains:

1. Navigate to **Settings -> Security & Membership**.
2. Select the **Automatically sign-in user when site changes** checkbox under *Administration.*
3. Click **Save**.

No further configuration is necessary. Users can now freely switch sites via the administration interface without the need to re-enter their credentials every time the site changes.

Implementing single sign-on via the Kentico API

You can also implement single sign-on functionality on custom pages using the Kentico API.

The following code example shows how to authenticate a user with a particular username in your code:

```
string userName = "testuser";

// Authenticates the user with the specified user name
CMS.Membership.AuthenticationHelper.AuthenticateUser(userName, true, false);
```

User authentication via the *AuthenticateUser()* method closely mimics standard ASP.NET Forms authentication. The authentication cookie created when a user is authenticated this way will behave according to its settings as described in the Forms authentication s ection above.

The second code example shows how to generate a URL with a user authentication token. The system automatically authenticates users when they access this URL.

```
using CMS.Membership;
using CMS.Helpers;

...

string userName = "testuser";

// Gets the user with the specified user name
UserInfo userInfo = UserInfoProvider.GetUserInfo(userName);

// Gets the authentication URL for a specified user and target URL
string url = AuthenticationHelper.GetUserAuthenticationUrl(userInfo, "/default.aspx");

// Redirects the user to the target URL for authentication
URLHelper.Redirect(url);
```