You can use global events to define custom actions that the system performs after a user tries to sign in to Kentico with multi-factor authentication enabled. See the **SecurityEvents** section of the global event reference to learn more about the available options.

> ⚠ When implementing a custom authentication factor, disable the **Display secret key** option in **Settings -> Security & Membership -> Authentication**.

To set up a custom action that the system performs after a user signs in to Kentico with multi-factor authentication enabled, implement a handler for the **SecurityEvents.MultiFactorAuthenticate.Execute** event. For example, you can implement functionality that sends users an SMS text or email with a passcode.

The following code is an example of a custom module class that sends an email with a valid passcode to users who attempt to sign in with multi-factor authentication enabled.

```
using CMS;
using CMS.DataEngine;
using CMS.Membership;
using CMS.EmailEngine;

// Registers the custom module into the system
[assembly: RegisterModule(typeof(CustomMFAuthenticationModule))]

public class CustomMFAuthenticationModule : Module
{
    // Module class constructor, the system registers the module under the name
"CustomMFAuthentication"
    public CustomMFAuthenticationModule()
        : base("CustomMFAuthentication")
    {
    }

    // Contains initialization code that is executed when the application starts
    protected override void OnInit()
    {
        base.OnInit();

        // Assigns a handler to the SecurityEvents.MultiFactorAuthenticate.Execute
event
        // This event occurs when users try to sign in to Kentico with multi-factor
authentication enabled
        SecurityEvents.MultiFactorAuthenticate.Execute += MFAuthentication_Execute;
    }

    // Handler method that sends the passcode emails
    // You can replace it with your custom code
    private void MFAuthentication_Execute(object sender, AuthenticationEventArgs e)
    {
        // Gets the user's email address
        string userEmail = e.User.Email;

        if (userEmail != null && userEmail != "")
        {
            // Creates the email message
            EmailMessage msg = new EmailMessage();

            msg.From = "system@localhost.local";
            msg.Recipients = userEmail;
            msg.Subject = "Authentication passcode";
            msg.Priority = EmailPriorityEnum.High;
            msg.Body = "<html><body><p>Your authentication passcode: "
                       + e.Passcode + " (valid for 5 minutes)"
                       + "</p></body></html>";

            // Sends out the email message
            EmailSender.SendEmail(msg);
        }
    }
}
```

When the multi-factor authentication event occurs, the system generates a valid passcode for the given user. You can access the passcode in the **Passcode** property of the handler's **AuthenticationEventArgs** parameter, and use any type of API to deliver the information to the authenticating user.