In some scenarios, SSL decryption and encryption may not be performed directly by your application's server. Instead, the decryption and encryption is performed via a reverse proxy, which is equipped with an SSL offload hardware (for example, an SSL accelerator). This means that requests are forwarded to the application internally using the standard HTTP protocol, even when the client accesses the page through HTTPS. If the settings for using SSL are enabled for the website, it may result in a redirection loop.

You can solve this issue by adding custom code to the application's request handlers. It is necessary to appropriately set the **IsSSL** static property of the **CMS.Helpers.RequestContext** class. If set to *true*, the system treats all requests as secured, regardless of their URL format, and redirection to HTTPS page versions is not performed by the application. Of course, it is necessary to correctly identify which requests originally used SSL, for example by checking the request headers.

You also need to set the **SSLUrlPort** property of the **CMS.Helpers.URLHelper** class to 443 (or any other port that you use for HTTPS requests).

Setting the IsSSL and SSLUrlPort properties

1. Open your Kentico project in Visual Studio (using the **WebSite.sln** or **WebApp.sln** file).
2. Create a custom module class.
    - Either add the class into a custom project within the Kentico solution (recommended) or directly into the Kentico web project (into a custom folder under the **CMSApp** project for *web application* installations, into the **App_Code** folder for *web site* installations).

      > For basic execution of initialization code, you only need to register a "code-only" module through the API. You do NOT need to create a new module within the **Modules** application in the Kentico administration interface.

3. Override the module's **OnInit** method and assign a handler to the **RequestEvents.Prepare.Execute** event.

```
using System;
using System.Web;
using System.Collections.Specialized;

using CMS;
using CMS.DataEngine;
using CMS.Base;
using CMS.Helpers;

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

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

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

                // Assigns a handler called before each request is processed
                RequestEvents.Prepare.Execute += HandleSSLRequests;

                // Sets the URL port used for HTTPS requests
                URLHelper.SSLUrlPort = 443;
        }

        // Checks if requests are forwarded as SSL
        private static void HandleSSLRequests(object sender, EventArgs e)
        {
                if ((HttpContext.Current != null) && (HttpContext.Current.Request !
= null))
                {
                        // Loads the request headers as a collection
                        NameValueCollection headers = HttpContext.Current.Request.
Headers;

                        // Gets the value from the X-Forwarded-Ssl header
                        string forwardedSSL = headers.Get("X-Forwarded-Ssl");
                        RequestContext.IsSSL = false;

                        // Checks if the original request used HTTPS
                        if (String.Equals(forwardedSSL, "on", StringComparison.
OrdinalIgnoreCase))
                        {
                                RequestContext.IsSSL = true;
                        }
                }
        }
}
```

The example registers a custom module and overrides its **OnInit()** method (executed automatically when the application starts) to assign a handler that is called before each request is processed. The **X-Forwarded-Ssl** header is used to check if the original request was submitted via HTTPS before the SSL accelerator forwarded it to the application. If this is the case, the **IsSSL** property is set to *true* , so the system processes the request as if it used the HTTPS protocol. Also, the **SSLUrlPort** property is set to 443.

Important

Your proxy device may use a different method to identify requests that were originally secured by SSL. In such cases, you need to write a condition that fits your specific scenario. For example, another typical approach is to check if the value of the **X-Forwarded-Proto** request header is *https*.

You may also include additional custom code to fulfill any other security requirements, such as validation of the proxy's IP address.