You can use the email marketing features of Kentico with websites that are presented by a separate MVC application.

In this scenario, you manage and send newsletters from the administration interface of the Kentico application. All settings related to email marketing that you configure in Kentico apply (both global settings and the configuration of individual newsletters). On the side of the MVC application, you need to handle the actions that visitors perform on the pages of the website – newsletter subscription and unsubscription.

To allow visitors on your MVC website to manage subscriptions for your newsletters:

1. Install the **Kentico.Newsletters** integration package into your MVC project.
2. Initialize the subscription service.
3. Implement the required functionality on your MVC site:
   - Subscription
   - Unsubscription
   - Processing of subscription confirmations (if your newsletters use double opt-in)
4. Enable resource sharing with the Kentico administration application.

Initializing the subscription service

To use the newsletter subscription API, you need to initialize a **NewsletterSubscriptionService** instance and make it available to the related controllers.

We recommend using a dependency injection container to initialize service instances. When configuring the lifetime scope for *NewsletterSubscriptionService*, create a shared instance (singleton) for all requests.

```
using Kentico.Newsletters;
using CMS.SiteProvider;
```

```
var subscriptionService = new NewsletterSubscriptionService();
```

You can now call the methods of the subscription service within your application's code to perform actions related to newsletter subscription and unsubscription.

Setting up subscription

Use the following approach to develop actions that allow visitors to subscribe to newsletters on your website:

Tip: To view the full code of a functional example directly in Visual Studio, download the Kentico MVC solution from GitHub and inspect the **LearningKit** project. You can also run the Learning Kit website after connecting the project to a Kentico database.

1. Create a newsletter for your MVC site in Kentico (or use an existing one). See Creating newsletters for more information.
2. Create a new controller class in your MVC project or edit an existing one.
3. Implement two subscription actions – one basic GET action to display a subscription form and a second POST action to handle the creation of new recipients when the form is submitted.
4. Perform the following steps within the POST action:
   a. Get a *ContactInfo* object representing the new recipient based on the email address posted from the subscription form – call the **ContactProvider.GetContactForSubscribing** method (*CMS.ContactManagement* namespace).
   b. Use the *CMS.Newsletters* API to get a *NewsletterInfo* object representing the newsletter for which you are implementing subscription.
   c. Call the **Subscribe** method of the subscription service with the prepared *ContactInfo* and *NewsletterInfo* objects as parameters.

Notes:

- The **ContactProvider.GetContactForSubscribing** method automatically creates the contact object in Kentico based on the specified email address (if necessary).
- The **Subscribe** method adds the contact as a recipient of the newsletter. Returns a *true* value for new recipients, *false* if a contact with the given email address is already a recipient of the newsletter.

```
using System;
using System.Web.Mvc;

using CMS.Core;
using CMS.ContactManagement;
using CMS.Newsletters;
using CMS.SiteProvider;
using CMS.Helpers;

using Kentico.Newsletters;
```

Subscription actions example

```
        /// <summary>
        /// Basic action that displays the newsletter subscription form.
        /// </summary>
        public ActionResult Subscribe()
        {
            return View();
        }

        /// <summary>
        /// Handles creation of new marketing email recipients when the
subscription form is submitted.
        /// Accepts an email address parameter posted from the subscription
form.
        /// </summary>
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Subscribe(NewsletterSubscribeViewModel model)
        {
            if (!ModelState.IsValid)
            {
                // If the subscription data is not valid, displays the
subscription form with error messages
                return View(model);
            }

            // Gets the default Kentico contact provider
            IContactProvider contactProvider = Service.
Resolve<IContactProvider>();
            // Either gets an existing contact by email or creates a new
contact object with the given email
            ContactInfo contact = contactProvider.GetContactForSubscribing
(model.Email);

            // Gets a newsletter
            // Fill in the code name of your newsletter object in Kentico
```

```
            NewsletterInfo newsletter = NewsletterInfoProvider.
GetNewsletterInfo("SampleNewsletter", SiteContext.CurrentSiteID);

            // Prepares settings that configure the subscription behavior
            var subscriptionSettings = new NewsletterSubscriptionSettings()
            {
                RemoveAlsoUnsubscriptionFromAllNewsletters = true, //
Subscription removes the given email address from the global opt-out list
for all marketing emails (if present)
                SendConfirmationEmail = true, // Allows sending of
confirmation emails for the subscription
                AllowOptIn = true // Allows handling of double opt-in
subscription for newsletters that have it enabled
            };

            // Subscribes the contact with the specified email address to
the given newsletter
            SubscriptionService.Subscribe(contact, newsletter,
subscriptionSettings);

            // Passes information to the view, indicating whether the
newsletter requires double opt-in for subscription
            model.RequireDoubleOptIn = newsletter.NewsletterEnableOptIn;

            // Displays a view to inform the user that the subscription was
successful
            return View("SubscribeSuccess", model);
        }
```

5. We recommend creating a view model for your subscription action *(NewsletterSubscribeViewModel* in the example above). The view model allows you to:
    - Pass parameters from the subscription form (email address and, optionally, the recipient's name).
    - Use data annotations to define validation and formatting rules for the subscription data. See System.ComponentModel.DataAnnotations on MSDN for more information about the available data annotation attributes.
    - Pass information back to the subscription form, for example, a flag indicating that the newsletter requires a double opt-in subscription.

Subscription view model example

```
 »using System.ComponentModel;
using System.ComponentModel.DataAnnotations;

public class NewsletterSubscribeViewModel
{
    [DataType(DataType.EmailAddress)]
    [Required(ErrorMessage = "The Email address cannot be empty.")]
    [DisplayName("Email address")]
    [EmailAddress(ErrorMessage = "Invalid email address.")]
    [MaxLength(254, ErrorMessage = "The Email address cannot be longer than
254 characters.")]
    public string Email
    {
        get;
        set;
    }

    /// <summary>
    /// Indicates whether the newsletter requires double-opt in for
subscription.
    /// Allows the view to display appropriate information to newly
subscribed users.
    /// </summary>
    [Bindable(false)]
    public bool RequireDoubleOptIn
    {
        get;
        set;
    }
}
```

6. Design the user interface required for newsletter subscription on your website:
   - Create a view for the *Subscribe* action and display an appropriate subscription form. We recommend using a strongly typed view based on your subscription view model.
   - Create any other views required by the *Subscribe* action, for example, to inform users about successful subscriptions.

Your website's visitors can now subscribe to newsletters. The system automatically performs all standard Kentico subscription features, such as sending of confirmation emails based on the newsletter's *Subscription confirmation* template.

Continue by setting up:

- Handling of unsubscription requests
- Processing of subscription confirmations (if your newsletters use double opt-in)

Setting up unsubscription

Users can unsubscribe from email campaigns or newsletters by clicking links placed into the content of your emails.

When setting up unsubscription functionality, you first need to ensure that the Kentico application generates correct unsubscription links:

1. Make sure that your site in Kentico is configured to use extensionless URLs by default – set **Settings -> URLs and SEO -> Friendly URLs extension** to an empty value.

2. Add unsubscription links into the content of your marketing emails or email templates (in the **Email marketing** application).

   > Use the *{% EmailFeed.UnsubscribeFromEmailFeedUrl %}* and *{% EmailFeed.UnsubscribeFromAllEmailFeedsUrl %}* macros to generate the URLs of the links.
   >
   > See Preparing email templates to learn more.

3. In the **Email marketing** application, edit your email feeds, open their **Configuration** tab and set the following properties:
   - **Base URL** – must match the *Presentation URL* of your MVC site. For example: *http://www.SiteDomain.com*
   - **Unsubscription page URL** – must match the route of the action that handles unsubscription requests in your MVC application. For example: */NewsletterSubscription/Unsubscribe*

     > To use a shared *Unsubscription page URL* for multiple email feeds, leave the property empty for individual email feeds and set the value in the **Settings -> On-line marketing -> Email marketing -> Unsubscription page URL** setting.

Continue by implementing a controller action within your MVC application for handling of unsubscription requests:

1. Create a new model class in your MVC project. The model will represent the parameters that Kentico generates in newsletter and email campaign unsubscription links.

   Unsubscription model

```
 »using System;
using System.ComponentModel.DataAnnotations;

public class MarketingEmailUnsubscribeModel
{
    /// <summary>
    /// The email address of the recipient who is requesting unsubscription.
    /// </summary>
    [Required]
    public string Email
    {
        get;
        set;
    }

    /// <summary>
    /// The GUID (identifier) of the Kentico email feed related to the
unsubscription request.
    /// </summary>
    [Required]
    public Guid NewsletterGuid
    {
        get;
        set;
    }

    /// <summary>
    /// The GUID (identifier) of the Kentico marketing email related to the
unsubscription request.
    /// </summary>
    [Required]
    public Guid IssueGuid
    {
        get;
        set;
    }

    /// <summary>
    /// Hash for protection against forged unsubscription requests.
    /// </summary>
    [Required]
    public string Hash
    {
        get;
        set;
    }

    /// <summary>
    /// Indicates whether the unsubscription request is for all marketing emails
or only a specific email feed.
    /// </summary>
    public bool UnsubscribeFromAll
    {
        get;
        set;
    }
}
```

2. Add a GET action to your newsletter subscription controller and perform the following steps:
    a. Use the unsubscription model to pass parameters from the unsubscription request.
    b. Verify that the unsubscription hash is valid for the given email address – call the **ValidateEmail** method of the subscription service, with the email address and hash from the unsubscription request as parameters.
    c. To remove subscriptions, call the **Unsubscribe** or **UnsubscribeFromAll** method of the subscription service (with parameters taken from the unsubscription request). Branch the logic based on the *UnsubscribeFromAll* parameter from the unsubscription request.

```
using System.Web.Mvc;

using Kentico.Newsletters;
```

Unsubscription action example

```
        /// <summary>
        /// Handles marketing email unsubscription requests.
        /// </summary>
        public ActionResult Unsubscribe(MarketingEmailUnsubscribeModel model)
        {
            // Verifies that the unsubscription request contains all required
parameters
            if (ModelState.IsValid)
            {
                // Confirms whether the hash in the unsubscription request is
valid for the given email address
                // Provides protection against forged unsubscription requests
                if (SubscriptionService.ValidateEmail(model.Email, model.Hash))
                {
                    // Unsubscribes the specified email address (either from all
marketing emails or one specific email feed)
                    if (model.UnsubscribeFromAll)
                    {
                        SubscriptionService.UnsubscribeFromAll(model.Email, model.
NewsletterGuid, model.IssueGuid);
                    }
                    else
                    {
                        SubscriptionService.Unsubscribe(model.Email, model.
NewsletterGuid, model.IssueGuid);
                    }

                    // Displays a view to inform the user that they were
unsubscribed
                    return View("UnsubscribeSuccess");
                }
            }

            // If the unsubscription was not successful, displays a view to
inform the user
            // Failure can occur if the request does not provide all required
parameters or if the hash is invalid
            return View("UnsubscribeFailure");
        }
```

3. Create views for informing recipients about the result of their unsubscription requests.

When a user clicks on an unsubscription link in a newsletter, the URL leads to the unsubscribe action on your MVC site. The action validates the parameters and either unsubscribes the recipient or displays information if the request is invalid.

The content of the unsubscription page depends on your implementation of the returned views.

Handling confirmations for double opt-in subscription

If your newsletters are configured to use double opt-in, you need to set up additional logic on your MVC site to allow users to confirm their newsletter subscriptions.

When a visitor subscribes to a newsletter with double opt-in enabled, the system automatically adds the recipient with the "Waiting for confirmation" subscription status and the Kentico application sends a confirmation email to the given email address. You need to ensure that the double opt-in email contains a correct confirmation link:

1. Make sure that your site in Kentico is configured to use extensionless URLs by default – set **Settings -> URLs and SEO -> Friendly URLs extension** to an empty value.
2. Add the confirmation link into the content of the **Double opt-in template** assigned to your newsletter (in the **Email marketing** application).

   Use the *{% EmailFeed.SubscriptionConfirmationUrl %}* macro to generate the URL of the subscription confirmation link. See Preparing email templates to learn more.

3. In the **Email marketing** application, edit your newsletters, open their **Configuration** tab and set the following properties:
   - **Base URL** – must match the *Presentation URL* of your MVC site. For example: *http://www.SiteDomain.com*
   - **Approval page URL** – must match the route of the action that handles subscription confirmation requests in your MVC application. For example: */NewsletterSubscription/ConfirmSubscription*

   To use a shared *Approval page URL* for multiple newsletters, leave the property empty for individual newsletters and set the value in the **Settings -> On-line marketing -> Email marketing -> Double opt-in approval page URL** setting.

Continue by implementing a controller action within your MVC application to handle the subscription confirmation requests:

1. Create a new model class in your MVC project. The model will represent the parameters that Kentico generates in subscription confirmation links.

Subscription confirmation view model

```
 »using System.ComponentModel.DataAnnotations;

public class NewsletterConfirmSubscriptionViewModel
{
    /// <summary>
    /// Hash used to identify the subscription and for protection against forged
confirmation requests.
    /// </summary>
    [Required]
    public string SubscriptionHash
    {
        get;
        set;
    }

    /// <summary>
    /// The date and time of the original subscription. Used to detect expired
confirmation requests.
    /// </summary>
    public string DateTime
    {
        get;
        set;
    }
}
```

2. Add a GET action to your newsletter subscription controller and perform the following steps:
    a. Use the subscription confirmation model to pass parameters from the confirmation request.
    b. Parse the date and time parameter from the confirmation request. The value uses a specific date and time format required by the Kentico API (see the code example below for details).
    c. To confirm subscriptions, call the **ConfirmSubscription** method of the subscription service (with parameters taken from the hash and DateTime value in the confirmation request).
    d. Handle the result of the subscription confirmation based on the **ApprovalResult** value returned by the *ConfirmSubscription* method (see the code example below for information about the possible values).

```
using System;
using System.Web.Mvc;

using CMS.Core;
using CMS.ContactManagement;
using CMS.Newsletters;
using CMS.SiteProvider;
using CMS.Helpers;

using Kentico.Newsletters;
```

Subscription confirmation action example

```
        /// <summary>
        /// Handles confirmation requests for newsletter subscriptions (when
using double opt-in).
        /// </summary>
        public ActionResult ConfirmSubscription
```

```
(NewsletterConfirmSubscriptionViewModel model)
        {
            // Verifies that the confirmation request contains the required hash
parameter
            if (!ModelState.IsValid)
            {
                // If the hash is missing, returns a view informing the user that
the subscription confirmation was not successful
                ModelState.AddModelError(String.Empty, "The confirmation link is
invalid.");
                return View(model);
            }

            // Attempts to parse the date and time parameter from the request
query string
            // Uses the date and time formats required by the Kentico API
            DateTime parsedDateTime = DateTimeHelper.ZERO_TIME;
            if (!string.IsNullOrEmpty(model.DateTime) && !DateTimeUrlFormatter.
TryParse(model.DateTime, out parsedDateTime))
            {
                // Returns a view informing the user that the subscription
confirmation was not successful
                ModelState.AddModelError(String.Empty, "The confirmation link is
invalid.");
                return View(model);
            }

            // Attempts to confirm the subscription specified by the request's
parameters
            ApprovalResult result = SubscriptionService.ConfirmSubscription(model.
SubscriptionHash, false, parsedDateTime);

            switch (result)
            {
                // The confirmation was successful or the recipient was already
approved
                // Displays a view informing the user that the subscription is
active
                case ApprovalResult.Success:
                case ApprovalResult.AlreadyApproved:
                    return View(model);

                // The confirmation link has expired
                // Expiration occurs after a certain number of hours from the
time of the original subscription
                // You can set the expiration interval in Kentico (Settings -> Onâ
€‘line marketing -> Email marketing -> Double opt-in interval)
                case ApprovalResult.TimeExceeded:
                    ModelState.AddModelError(String.Empty, "Your confirmation
link has expired. Please subscribe to the newsletter again.");
                    break;

                // The subscription specified in the request's parameters does
not exist
                case ApprovalResult.NotFound:
                    ModelState.AddModelError(String.Empty, "The subscription that
you are attempting to confirm does not exist.");
                    break;

                // The confirmation failed
```

```
                default:
                    ModelState.AddModelError(String.Empty, "The confirmation of
your newsletter subscription did not succeed.");
                    break;
            }

            // If the subscription confirmation was not successful, displays a
view informing the user
            return View(model);
        }
```

3. Create a view for informing users about the result of their subscription confirmation requests.

When a new recipient clicks the link in the confirmation email received after subscribing to a newsletter with double opt-in enabled, the confirmation action on your MVC site attempts to confirm the subscription. If the link is valid (based on the expiration interval configured in the Kentico application in *Settings -> Online marketing -> Email marketing -> Double opt-in interval*), the system updates the recipient to the "Subscribed" status and the newsletter subscription becomes active.

The content of the subscription approval page depends on your implementation of the returned views.

Enabling resource sharing

When running a "content only" site (i.e. a site created using the *MVC Blank Site* template), the **Email marketing** application in the Kentico administration interface automatically checks the accessibility of the URLs specified for each email feed. This includes the following:

- Unsubscription page URL (see Setting up unsubscription)
- Approval page URL (see Handling confirmations for double opt-in subscription)
- Clicked link and Opened email tracking URLs (see Tracking marketing emails on MVC sites)

If any of the given URLs are not accessible, the system displays a warning on the **Emails** tab of the email feed and the **Send** tab of individual emails.

To allow the system to verify the URLs correctly in scenarios where your MVC site and Kentico administration application run on different domains, you need to enable cross-origin resource sharing. Do this by enabling the resource sharing feature for your MVC application:

1. Edit the project's **App_Start\ApplicationConfig.cs** file.
2. Call the **UseResourceSharingWithAdministration()** method of the *ApplicationBuilder* instance in the **RegisterFeatures** method.

```
using Kentico.Web.Mvc;

...

public static void RegisterFeatures(ApplicationBuilder builder)
{
    ...
    // Enables cross-origin resource sharing with the connected Kentico administration
application
    builder.UseResourceSharingWithAdministration();
}
```

When processing requests sent from the domain of the connected Kentico administration application, your MVC application now sends a response with the *Access-Control-Allow-Origin* header, which enables cross origin resource sharing.