

In Kentico, consents are records used to inform website visitors about the means of collecting and manipulating their personal data by the system, site administrators, marketers and anyone else who has access to said data, including third-parties to whom this data is forwarded.

You can use consents to comply with the requirements of the [GDPR](#) and other personal data regulations.

Whenever any personal data of a visitor is obtained, it is necessary to have a consent agreement from the visitor to legally process this data. This includes [tracking of contacts](#) and their [activities](#) on the live site. Every consent agreement of a visitor is directly connected to a corresponding [contact](#). Deleting contacts from the system therefore also causes deletion of the contact's consent agreements.



#### Prerequisite

Since consent agreements are directly connected to contacts, you need to enable [contact tracking](#) in your connected Kentico instance and in the MVC application.

This page describes how to set up consent functionality on MVC sites. To learn how to create consents in the Kentico administration interface or use consents on [portal engine](#) websites, see [Working with consents](#).



**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.

## Setting up tracking consent

When creating a consent for the [tracking of contacts](#) and [activities](#), you need to provide a way for visitors to adjust their allowed [cookie level](#). The system only performs the tracking for visitors whose cookie level is *Visitor* or higher.

To ensure that visitors are not tracked until they give the tracking consent, set the **default cookie level** for the website in your connected Kentico instance:

1. Open the **Settings** application.
2. Navigate to the **System** category in the settings tree.
3. Set the value of the **Default cookie level** setting to *System* or *Essential*.



**Note:** With the **System** cookie level, basic functionality such as authentication may not work for visitors by default (if your authentication cookie is [registered](#) with the *Essential* level). See [Working with users on MVC sites](#) for more information.

4. Click **Save**.

To allow visitors to give and revoke agreements with the tracking consent on your MVC site, you need to develop corresponding components.

## Creating the consent

Start by [defining the tracking consent](#) in the Kentico administration interface (in the **Data protection** application on the **Consents** tab).

## Creating the consent model

We recommend creating a view model class in your MVC project to represent data related to the tracking consent. Include the following properties:

- A string property for storing the consent's short text
- A boolean property to indicate whether an agreement with the consent was given



```
public class TrackingConsentViewModel
{
    public string ShortText { get; set; }

    public bool IsAgreed { get; set; }
}
```

## Creating the controller

Create a new controller class in your MVC project or edit an existing one:

- Add using statements for the following namespace from the Kentico API (and any other namespaces that you need):

```
using CMS.ContactManagement;
using CMS.Core;
using CMS.DataProtection;
using CMS.Helpers;

using Kentico.ContactManagement;
```

- Initialize instances of the following services, which are necessary to manage contacts and tracking consent:
  - **IConsentAgreementService** – to manage consent agreements.
  - **IContactTrackingService** – to obtain the contact linked to the current visitor on the live site.
  - **ICurrentCookieLevelProvider** – to manage the current visitor's cookie level.



We recommend using a [dependency injection container](#) to initialize service instances.

```
private readonly IConsentAgreementService consentAgreementService;
private readonly IContactTrackingService contactTrackingService;
private readonly ICurrentCookieLevelProvider currentCookieLevelProvider;

public TrackingConsentController()
{
    consentAgreementService = Service.Resolve<IConsentAgreementService>();
    contactTrackingService = new ContactTrackingService();
    currentCookieLevelProvider = Service.
Resolve<ICurrentCookieLevelProvider>();
}
```

- Implement a GET action that displays information about the tracking consent:



```
public ActionResult DisplayConsent()
{
    // Gets the related tracking consent
    // Fill in the code name of the appropriate consent object in Kentico
    ConsentInfo consent = ConsentInfoProvider.GetConsentInfo
("SampleTrackingConsent");

    // Gets the current contact
    ContactInfo contact = contactTrackingService.GetCurrentContactAsync
(User.Identity.Name).Result;

    // Sets the default cookie level for contacts who have revoked the
tracking consent
    // Required for scenarios where one contact uses multiple browsers
    if ((contact != null) && !consentAgreementService.IsAgreed(contact,
consent))
    {
        var defaultCookieLevel = currentCookieLevelProvider.
GetDefaultCookieLevel();
        currentCookieLevelProvider.SetCurrentCookieLevel
(defaultCookieLevel);
    }

    var consentModel = new TrackingConsentViewModel
    {
        // Adds the consent's short text to the model
        ShortText = consent.GetConsentText("en-US").ShortText,

        // Checks whether the current contact has given an agreement for
the tracking consent
        IsAgreed = (contact != null) && consentAgreementService.IsAgreed
(contact, consent)
    };

    return View("Consent", consentModel);
}
```

- Implement POST actions for creating and revoking consent agreements:
  - Retrieve the visitor's contact using the **GetCurrentContact** method of the **IContactTrackingService** instance (see [Tracking contacts on MVC sites](#)).
  - Modify the visitor's cookie level according to the performed action – call the **SetCurrentCookieLevel** method of the **ICurrentCookieLevelProvider** instance.
    - Set the *All* level when creating tracking consent agreements.
    - Set the default cookie level from the site's settings when revoking agreements.
  - Create or revoke the consent agreement using the **Agree** or **Revoke** methods of the **IConsentAgreementService** instance.



```
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Accept()
{
    // Gets the related tracking consent
    ConsentInfo consent = ConsentInfoProvider.GetConsentInfo
("SampleTrackingConsent");

    // Sets the visitor's cookie level to 'All' (enables contact tracking)
    currentCookieLevelProvider.SetCurrentCookieLevel(CookieLevel.All);

    // Gets the current contact and creates a consent agreement
    ContactInfo contact = contactTrackingService.GetCurrentContactAsync
(User.Identity.Name).Result;
    consentAgreementService.Agree(contact, consent);

    return RedirectToAction("DisplayConsent");
}
```

```
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Revoke()
{
    // Gets the related tracking consent
    ConsentInfo consent = ConsentInfoProvider.GetConsentInfo
("SampleTrackingConsent");

    // Gets the current contact and revokes the tracking consent agreement
    ContactInfo contact = contactTrackingService.GetCurrentContactAsync
(User.Identity.Name).Result;
    consentAgreementService.Revoke(contact, consent);

    // Sets the visitor's cookie level to the site's default cookie level
    (disables contact tracking)
    int defaultCookieLevel = currentCookieLevelProvider.
GetDefaultCookieLevel();
    currentCookieLevelProvider.SetCurrentCookieLevel(defaultCookieLevel);

    return RedirectToAction("DisplayConsent");
}
```

## Creating the view

To display the tracking consent and inputs for giving or revoking agreements on your website, add an appropriate view to your MVC project. For example, you can add the consent presentation to a partial view that is included in your MVC website's main layout. This ensures that the input to give the tracking consent is displayed on all pages.

Depending on your preferences and legal requirements, you can either hide the view's output for visitors who have given the tracking consent or display content that allows the consent to be revoked.

Visitors on your website now have an option to give agreements with your tracking consent. New visitors are not tracked as contacts until they positively confirm that they agree with the tracking.

## Adding consents to forms

Your MVC site may contain [forms](#) that allow visitors to submit personal data. To collect consent agreements regarding the processing of this data, you need to modify the corresponding form's definition in the Kentico application, as well as related components in the MVC application.

In the Kentico back-end application:

- Create an appropriate [consent](#) in the **Data protection** application.
- Add a consent agreement field to an existing form or create a new form (in the **Forms** application).
- Ensure that the form's fields are [mapped to contact attributes](#).

In the MVC application:

- Modify the appropriate controller to create consent agreements (use an *IFormConsentAgreementService* instance from the *CMS.DataProtection* namespace of the Kentico API).
- Update the class [generated](#) for the form so that it includes the new consent field.
- Extend your form submission view model to store consent data.
- Modify your form's view to include an input option through which visitors can give consent agreements.

The following sections describe how to extend the example from [Working with forms on MVC sites](#). You need to set up a form in your MVC application before you can follow this scenario and add the consent features.

### Modifying the feedback form

Add a field for storing consent agreements to the form:

1. Edit the desired form in the **Forms** application.
2. Switch to the **Fields** tab and create a **New field**.
3. Set the properties of the field:
  - **Field name:** ConsentAgreement
  - **Data type:** *Unique identifier (GUID)*
4. Click **Save**.
5. Select the **Contact mapping** tab and map the form's fields to the appropriate contact attributes (see [Mapping fields to contact attributes](#)).
6. Switch to the **Code** tab.
7. Transfer the updated code to the [generated class](#) representing the form's data items in your MVC application (*FeedbackFormItem* in the example).

### Modifying the form model

Edit the form's view model class and add two properties:

- A string property for storing the consent's short text
- A boolean property to indicate whether an agreement with the consent was given

```
public string ConsentShortText { get; set; }

public bool ConsentIsAgreed { get; set; }
```

### Modifying the controller

To process the consent agreements, you need to modify the controller class that you use to display the form and process the submitted data:

1. Initialize **IFormConsentAgreementService** and **IContactTrackingService** instances, and make them available in the controller's code.



We recommend using a [dependency injection container](#) to initialize service instances.

2. Modify the GET action that displays the form, and set the consent properties of the view model.
3. Modify the POST action that handles the form submission:
  - a. Retrieve the visitor's contact using the **GetCurrentContact** method of the **IContactTrackingService** instance (see [Tracking contacts on MVC sites](#)).
  - b. Evaluate whether consent was given for the submitted form data. If yes, create a new consent agreement using the **Agree** method of the **IFormConsentAgreementService** instance.
  - c. Assign the GUID identifier of the returned consent agreement object into the consent field within the form item's data.

```
»using System.Web.Mvc;

using CMS.Core;
using CMS.DataProtection;
using CMS.OnlineForms.Types;

using Kentico.ContactManagement;

using LearningKit.Models.Form;

namespace LearningKit.Controllers
{
    public class FeedbackFormConsentController : Controller
    {
        private readonly IContactTrackingService contactTrackingService;
        private readonly IFormConsentAgreementService formConsentAgreementService;
        private readonly ConsentInfo consent;

        /// <summary>
        /// Constructor.
        /// You can use a dependency injection container to initialize the services.
        /// </summary>
        public FeedbackFormConsentController()
        {
            contactTrackingService = new ContactTrackingService();
            formConsentAgreementService = Service.
Resolve<IFormConsentAgreementService>();

            // Gets the related consent
            // Fill in the code name of the appropriate consent object in Kentico
            consent = ConsentInfoProvider.GetConsentInfo("SampleFormConsent");
        }

        /// <summary>
        /// Basic action that displays the feedback form.
        /// </summary>
        public ActionResult Fill()
        {
            var model = new FeedbackFormMessageConsentModel
            {
                // Adds the consent text to the form model
                ConsentShortText = consent.GetConsentText("en-US").ShortText,
                ConsentIsAgreed = false
            };
        }
    }
}
```



```
        return View("FormFillConsent", model);
    }

    /// <summary>
    /// Inserts the form data to the connected database when the feedback form is
submitted.
    /// Accepts parameters posted from the feedback form via the
FeedbackFormMessageConsentModel.
    /// </summary>
    [HttpPost]
    [ValidateAntiForgeryToken]
    public ActionResult SendFeedback(FeedbackFormMessageConsentModel model)
    {
        // Validates the received form data based on the view model
        if (!ModelState.IsValid)
        {
            model.ConsentShortText = consent.GetConsentText("en-US").ShortText;

            return View("FormFillConsent", model);
        }

        // Inserts the collected form data to the connected database
        InsertFeedbackFormItem(model);

        return View("FormSendSuccessConsent");
    }

    // Inserts the collected data to the connected database (helper method)
    private void InsertFeedbackFormItem(FeedbackFormMessageConsentModel feedback)
    {
        // Creates a form item containing the collected data
        var item = new FeedbackFormItemConsent
        {
            UserName = feedback.FirstName,
            UserLastName = feedback.LastName,
            UserEmail = feedback.Email,
            UserFeedback = feedback.MessageText,
        };

        // Creates a consent agreement if the consent checkbox was selected
        if (feedback.ConsentIsAgreed)
        {
            // Gets the current contact
            var contact = contactTrackingService.GetCurrentContactAsync(User.
Identity.Name).Result;

            // Creates an agreement for the specified consent and contact
            var agreement = formConsentAgreementService.Agree(contact, consent,
item);

            // Adds the GUID of the agreement into the form's consent field
            item.ConsentAgreement = agreement.ConsentAgreementGuid;
        }

        // Inserts the form data into the database
        item.Insert();
    }
}
```

Visitors on your MVC site now have an option to agree with the consent text when submitting the form. When processing the form's data, the sample controller creates a consent agreement for the given contact and stores the agreement's identifier into the corresponding form record field.

## Adding registration and subscription consents

You can collect consent agreements in any scenarios where visitors submit an input form. For example, when [registering as users](#) on the website or [subscribing to a newsletter](#).

To set up the consent functionality, use a similar approach as described for [form consents](#):

1. Prepare an appropriate [consent](#) in the Kentico administration interface (in the **Data protection** application on the **Consents** tab).
2. Expand the related controllers, models and views in your MVC project:
  - a. Display the consent text in the appropriate form (registration or subscription), along with an input that allows visitors to give their consent.
  - b. Initialize instances of the **IFormConsentAgreementService** and **IContactTrackingService** interfaces, and make them available in the controller's code.
  - c. Get the visitor's contact (see [Tracking contacts on MVC sites](#)) and the appropriate consent object (by calling the **ConsentInfoProvider.GetConsentInfo** method).
  - d. In the post action that handles the submitted data, evaluate whether the consent was given and call the **Agree** method of the **IFormConsentAgreementService** instance to create a consent agreement.



### **IFormConsentAgreementService and IConsentAgreementService**

The *IFormConsentAgreementService.Agree* method automatically handles scenarios where the contact parameter is *null* (for example for visitors who have not given an agreement with the tracking consent). In these cases, the method creates a new contact object with the related data and connects it with the consent agreement.

If you only wish to create the consent in cases where contact tracking is enabled, you can add a null condition for the visitor's contact object and use the *IConsentAgreementService.Agree* method instead.

Visitors can now give their consent with personal data processing when submitting the corresponding forms (e.g. during registration or when subscribing to a newsletter).

## Creating conditions based on consents

If you have any functionality that stores or processes personal data, you can create conditions to ensure that it only runs for contacts who have given consent.

1. Initialize an instance of the **IConsentAgreementService** interface and make it available in your code.
2. Obtain the related contact and consent objects.
3. Evaluate whether the contact has given an agreement with the given consent by calling the **IsAgreed** method of the **IConsentAgreementService** instance.

## Revoking consent agreements

To ensure compliance of your website with the GDPR or other data protection regulations, you need to give visitors the option to revoke their previously given consent agreements. We recommend creating a privacy page, where visitors can view the consents for which they have given agreements and potentially revoke them.



"It shall be as easy to withdraw as to give consent."

(Source: [GDPR Article 7](#), Paragraph 3)



The following sections describe how to create a privacy page on MVC sites. The privacy page lists accepted consents for visitors, and allows them to view the full consent details or revoke their agreements.

- Create a model representing the list of consents for which visitors have given agreements.
- Create a model representing consent details.
- Create a controller that provides actions for displaying and revoking consents.
- Create views for the consent list and details pages.

## Creating the models

- Add a model class that represents the list of consents for which visitors have given agreements.

```
»using System.Collections.Generic;
using System.Linq;

using CMS.DataProtection;


namespace LearningKit.Models.PrivacyPage
{
    public class ConsentListingModel
    {
        public IEnumerable<Consent> Consents { get; set; } = Enumerable.
Empty<Consent>();
    }
}
```

### **Consent and ConsentInfo objects**

The Kentico API uses **ConsentInfo** objects to work with consents in general. However, when you load consents for which a contact has given an agreement, the *ConsentAgreementService.GetAgreedConsents* method returns a collection of **Consent** objects. Instances of the *Consent* class provide the *GetConsentText* method, which automatically retrieves either the current texts of the given consent or the texts of the archived consent version for which the agreement was given.

When displaying accepted consents to visitors, always work with *Consent* objects to ensure that the correct consent text version is used (for scenarios where the consent text was changed after agreements had been given).

For more information, see the *Viewing archived consents* section of the [Working with consents](#) page.

 **Tip:** You can directly use an *IEnumerable<Consent>* collection as a model if you do not need to work with any other data in your controller or views. However, having a dedicated model class allows you to extend the model based on future requirements.

- Add a model class that represents the full details of a consent, including properties for the short text, full text, and any other information about the consent you may need.

```
»namespace LearningKit.Models.PrivacyPage
{
    public class ConsentDetailsModel
    {
        public string ConsentShortText { get; set; }

        public string ConsentFullText { get; set; }

        public string ConsentDisplayName { get; set; }
    }
}
```

## Creating the controller

Create a new controller class in your MVC project or edit an existing one:

- Add using statements for the following namespaces from the Kentico API (and any other namespaces that you need):

```
using CMS.ContactManagement;
using CMS.Core;
using CMS.DataProtection;
using CMS.Helpers;

using Kentico.ContactManagement;
```

- Initialize instances of the **IConsentAgreementService**, **IContactTrackingService** and **ICurrentCookieLevelProvider** interfaces, and make them available in the controller's code:

```
private readonly IContactTrackingService contactTrackingService;
private readonly IConsentAgreementService consentAgreementService;
private readonly ICurrentCookieLevelProvider currentCookieLevelProvider;

public PrivacyPageController()
{
    consentAgreementService = Service.Resolve<IConsentAgreementService>();
    contactTrackingService = new ContactTrackingService();
    currentCookieLevelProvider = Service.
Resolve<ICurrentCookieLevelProvider>();
}
```



We recommend using a [dependency injection container](#) to initialize service instances.

- Implement a GET action which retrieves a list of all consents accepted by the visitor:
  1. Retrieve the visitor's contact using the **GetCurrentContact** method of the **IContactTrackingService** instance (see [Tracking contacts on MVC sites](#)).
  2. Get the consents for which the contact has given an agreement by calling the **GetAgreedConsents** method of the **IConsentAgreementService** instance.
  3. Create an instance of the consent list model class and pass it to the view.

```
/// <summary>
/// Loads and displays consents for which visitors have given agreements.
/// </summary>
public ActionResult Index()
{
    // Gets the current visitor's contact
    ContactInfo currentContact = contactTrackingService.
GetCurrentContactAsync(User.Identity.Name).Result;

    var consentListingModel = new ConsentListingModel();

    // Does not attempt to load consent data if the current contact is
not available
    // This occurs if contact tracking is disabled, or for visitors who
have not given an agreement with the tracking consent
    if (currentContact != null)
    {
        // Gets all consents for which the current contact has given an
agreement
        consentListingModel.Consents = consentAgreementService.
GetAgreedConsents(currentContact);
    }

    return View(consentListingModel);
}
```

- Implement a GET action which retrieves the full details of a specified consent:
  1. Retrieve the current contact using the contact tracking service.
  2. Get the consents for which the contact has given an agreement using the consent agreement service, and select the required consent based on the action's parameter. This approach ensures that the correct consent text version is used in cases where the text was changed after the agreement had been given.
  3. Create an instance of the consent details model class, set its properties, and pass it to the view.

```

    /// <summary>
    /// Display details of the specified consent.
    /// </summary>
    public ActionResult ConsentDetails(int consentId)
    {
        // Gets a list of consents for which the current contact has given an
        agreement
        ContactInfo currentContact = contactTrackingService.
GetCurrentContactAsync(User.Identity.Name).Result;
        IEnumerable<Consent> consents = consentAgreementService.
GetAgreedConsents(currentContact);

        // Gets the consent matching the identifier for which the details
        were requested
        // Using this approach to get objects of the 'Consent' class is
        necessary to ensure that the correct consent text
        // is displayed, either from the current consent text or the archived
        consent version for which the agreement was given
        Consent consent = consents.FirstOrDefault(c => c.Id == consentId);

        // Displays the privacy page (consent list) if the specified consent
        identifier is not valid
        if (consent == null)
        {
            return View("Index");
        }

        // Sets the consent's details into the view model
        var model = new ConsentDetailsModel
        {
            ConsentDisplayName = consent.DisplayName,
            ConsentShortText = consent.GetConsentText("en-US").ShortText,
            ConsentFullText = consent.GetConsentText("en-US").FullText
        };

        return View(model);
    }

```

- Implement a POST action that revokes agreements for a specified consent:
  1. Retrieve the current contact using the contact tracking service and the required consent (by calling the **ConsentInfoProvider.GetConsentInfo** method).
  2. Create a revocation for the consent agreement using the **Revoke** method of the **IConsentAgreementService** instance.



When revoking the **tracking consent**, you also need to lower the visitor's cookie level to ensure that contact tracking is disabled. Call the **SetCurrentCookieLevel** method of the **ICurrentCookieLevelProvider** instance and set the default cookie level from the site's settings.

```
/// <summary>
/// Revokes the current contact's agreement with the specified consent.
/// </summary>
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Revoke(int consentId)
{
    // Gets the related consent object
    ConsentInfo consent = ConsentInfoProvider.GetConsentInfo(consentId);

    // Gets the current visitor's contact
    ContactInfo currentContact = contactTrackingService.
GetCurrentContactAsync(User.Identity.Name).Result;

    // For the tracking consent, lowers the cookie level to the site's
    default in order to disable tracking
    if (consent.ConsentName == "SampleTrackingConsent")
    {
        currentCookieLevelProvider.SetCurrentCookieLevel
(currentCookieLevelProvider.GetDefaultCookieLevel());
    }

    // Revokes the consent agreement
    consentAgreementService.Revoke(currentContact, consent);

    return RedirectToAction("Index");
}
```



#### Clearing personal data for revoked consents

When a visitor revokes a consent agreement, you may also need to delete or anonymize certain types of [personal data](#) stored by the system.

To perform additional actions of this type whenever a consent agreement is revoked, assign a custom handler to the system's **DataProtectionEvents.RevokeConsentAgreement** [global event](#).

See also: [Implementing personal data erasure](#)

## Creating the views

Add appropriate views to your MVC project to display the consent list (privacy page) and the consent details.



Consent text values may contain HTML tags added via the editor in the Kentico administration interface (for formatting or content such as links). To ensure that the text is displayed correctly, disable the HTML encoding for the values using the **Html.Raw** method or the **Html.Kentico().ResolveUrls** extension method (also handles correct resolving of link URLs).

For example:

```
<h3>Short text</h3>
<p>@Html.Raw(Model.ConsentShortText)</p>
<h3>Full text</h3>
<p>@Html.Kentico().ResolveUrls(Model.ConsentFullText)</p>
```

Visitors on your site are now able to view a list of all consents for which they have given an agreement. They can also access the full details of the given consents and revoke their agreements if required.