

To ensure that URLs can be processed correctly, the system checks text values for unsafe characters before they are added to the URL path, and makes any necessary changes. This affects fields such as page aliases, page URL paths, forum names, post subjects etc.

By default, the process includes removal of diacritics and replacement of all forbidden characters according to the settings defined for the given site (as described in [URL format and configuration](#)). Characters with diacritics are replaced by the appropriate base character, for example *ü* is converted to *u*.

If you need to change the default behavior in any way, you can register and implement a handler for one or both of the following system events:

- **OnBeforeGetSafeUrlPart** - occurs whenever a potentially unsafe text value is added to a part of a URL, e.g. when the page alias field is saved (or filled automatically based on the page's name). This event can be used to customize the final format of URL paths. For example, you can convert characters from an international character set to an equivalent in Latin characters (ASCII) or specify a unique replacement string for particular forbidden characters.
- **OnBeforeRemoveDiacritics** - occurs whenever the system removes diacritics from text. This is one of the default steps performed when creating safe URLs, but the event is also triggered during many text parsing operations that are not directly related to URLs (for example when indexing text for the [Smart search](#)).

The events are members of the **URLHelper** and **TextHelper** classes respectively, which can be found under the **CMS.Helpers** namespace.

- The handlers for these events have the source text included as a string parameter passed by reference, so any changes that you make in the code are reflected in the result.
- Both handlers have a *boolean* return value that indicates whether the default functionality should also be performed after the handler is executed. For this reason, it is highly recommended to set the return value to *true* for all but the most extensive customizations.

#### **Warning**

Only return a *false* value if you are sure that your custom handler can take full responsibility for all URL safety or diacritics removal requirements.

Disabling the default system functionality may prevent parts of your website from working correctly, particularly in the case of handlers for the **OnBeforeGetSafeUrlPart** event.

## Example

The following example demonstrates how to define handlers for the forbidden character events:

1. Open your Kentico web project in Visual Studio.
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).
3. Override the module's **OnInit** method and assign handlers to the required events:



```
using System;

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

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

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

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

        // Assigns a handler to the OnBeforeGetSafeUrlPart event
        URLHelper.OnBeforeGetSafeUrlPart += Custom_OnBeforeGetSafeUrlPart;

        // Assigns a handler to the OnBeforeRemoveDiacritics event
        TextHelper.OnBeforeRemoveDiacritics +=
        Custom_OnBeforeRemoveDiacritics;
    }
}
```

4. Add the definition of the **Custom\_OnBeforeGetSafeUrlPart** handler into the module class:

```
static bool Custom_OnBeforeGetSafeUrlPart(ref string url, string siteName,
EventArgs e)
{
    // Replaces all & characters with the word "and"
    url = url.Replace("&", "and");

    // Returns true to indicate that the default URL replacements should also
    be performed - removing forbidden characters and diacritics (both default and
    custom)
    return true;
}
```




This code replaces all ampersand (&) characters with the word *and*. For example, a page named *Products & Services* has its default page alias generated as *Products-and-Services*, which is then used in the page's URL. This example is only meant as a simple demonstration. In realworld scenarios, the handler can be much more complex, for example when mapping the character set of an international language to appropriate ASCII characters or words.

The **siteName** parameter of the handler contains the code name of the site under which the event occurred. This can be useful if you need to access the forbidden character settings of the given site in your custom code.

5. Add the **Custom\_OnBeforeRemoveDiacritics** handler:

```
static bool Custom_OnBeforeRemoveDiacritics(ref string text, EventArgs e)
{
    // Replaces German special characters
    text = text.Replace( "ä", "ae" );
    text = text.Replace( "ö", "oe" );
    text = text.Replace( "ü", "ue" );
    text = text.Replace( "Ä", "Ae" );
    text = text.Replace( "Ö", "Oe" );
    text = text.Replace( "Ü", "Ue" );
    text = text.Replace( "ß", "ss" );

    // Returns true to indicate that the default diacritics removal should
    also be performed
    return true;
}
```

 This ensures that the system uses custom replacements for German special characters rather than simply stripping the diacritics and leaving the base character.

6. **Save** the file (Build your project if it was installed as a web application).

The system applies the changes when creating new URLs and when removing diacritics from text.



**Note:** The customization does not automatically change the aliases and URL paths of existing pages until the current value is changed or saved.