

You can import [products](#) from an external source of data into Kentico using the API. As a source of data, you can use, for example, a CSV (comma-separated values) file.



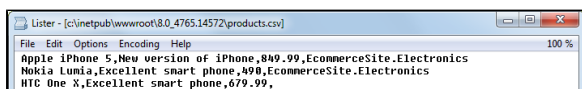
**Note:** Before you start the custom import product import, back up your Kentico database.

## Preparing your product source data

Prepare your external source of data, i.e. a CSV file. Make sure that all product properties required for import into the Kentico database are included.

In this example, the CSV file contains the following records (comma-separated):

- **Field 1** - specifies the name of the product, e.g. *Apple iPhone 5*.
- **Field 2** - specifies the description of the product, e.g. *New version of iPhone*.
- **Field 3** - specifies the price of the product. Use a dot (.) to separate decimal places, e.g. *849.99*.
- **Field 4** - specifies the [department](#) where the product will be placed after import. You need to enter the department code name, e.g. *EcommerceSite.Electronics*. If you leave the record empty or enter an invalid department code name, no department will be assigned to the product after import.



## Creating the product import page

Create the following files in your web project folder.

- [ImportProducts.aspx](#)
- [ImportProducts.aspx.cs](#)
- products.csv

### ImportProducts.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="ImportProducts.aspx.cs"
Inherits="ImportProducts" Theme="Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR
/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
    <title>Import</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Label ID="lblInfo" runat="server" EnableViewState="false" /><br />
            <asp:Label ID="lblError" runat="server" EnableViewState="false" />
        </div>
    </form>
</body>
</html>
```

## ImportProducts.aspx.cs

The following sample code allows you to import products specified in your external source of data to the **Products -> Electronics -> Cellphones** section as **Cellphones**.

Modify the code to suit your needs:

- [Change the import location](#)
- [Change the product type](#)

```
using System;
using CMS.DataEngine;
using CMS.DocumentEngine;
using CMS.Ecommerce;
using CMS.Helpers;
using CMS.IO;
using CMS.Membership;
using CMS.SiteProvider;

public partial class ImportProducts : System.Web.UI.Page
{
    private string userName = "Administrator"; // Page owner
    private TreeProvider tree;
    private string siteName = "";
    private string cultureCode;
    private int siteId;
    private string className = "CMSProduct.CellPhone";

    protected void Page_Load(object sender, EventArgs e)
    {
        // Get site info
        siteName = SiteContext.CurrentSiteName;
        siteId = SiteContext.CurrentSiteID;

        // Get default site culture
        cultureCode = CultureHelper.GetDefaultCultureCode(siteName);

        // Get user info
        UserInfo ui = UserInfoProvider.GetUserInfo(userName);

        if (ui != null)
        {
            tree = new TreeProvider(ui);
        }
        else
        {
            tree = new TreeProvider();
        }

        // Import products
        lblError.Text = Import(Server.MapPath("products.csv"));
    }

    /// <summary>
    /// Imports products from the specified file in *.csv format
    /// </summary>
    /// <param name="filePath"></param>
    /// <returns></returns>
    private string Import(string filePath)
```



```
{
    string error = "";
    FileStream file = null;

    try
    {
        // Specify file with product data
        file = FileStream.New(filePath, FileMode.Open, FileAccess.Read);
    }
    catch (Exception ex)
    {
        error = "Error loading file '" + filePath + "': " + ex.Message;
    }

    if (error == "")
    {
        // Get product parent page
        TreeNode parentPage = GetProductParentPage();

        // Get default product page type
        DataClassInfo productPageType = GetProductPageType();

        if ((parentPage == null) || (productPageType == null))
        {
            error = "Unable to create products, some information missing";
        }
        else
        {
            var reader = StreamReader.New(file);
            int count = 0;

            // Read file
            while (!reader.EndOfStream)
            {
                string line = reader.ReadLine();
                if (line != null && line.Trim() != "")
                {
                    // Get product data
                    string[] productData = line.Trim().Split(',');

                    // Create product (SKU)
                    var sku = new SKUInfo
                    {
                        SKUName = productData[0],
                        SKUDescription = productData[1],
                        SKUPrice = ValidationHelper.GetDecimal(productData[2], 0),
                        SKUEnabled = true,
                        SKUSiteID = siteId,
                        SKUProductType = SKUProductTypeEnum.Product
                    };

                    DepartmentInfo department = DepartmentInfoProvider.
GetDepartmentInfo(productData[3], siteName);
                    if (department != null)
                    {
                        // Assign product to its department
                        sku.SKUDepartmentID = department.DepartmentID;
                    }
                }
            }
        }
    }
}
```



```
        {
            SKUInfoProvider.SetSKUInfo(sku);
        }
        catch
        {
            error += "Unable to create product '" + sku.SKUName + "'.
<br />";
        }

        // If product was created successfully
        if (sku.SKUID > 0)
        {
            // Create product page type and assign SKU to it
            var productDoc = (SKUTreeNode)TreeNode.New(productPageType.
ClassName, tree);

            productDoc.DocumentSKUName = sku.SKUName;
            productDoc.DocumentSKUDescription = sku.SKUDescription;
            productDoc.NodeSKUID = sku.SKUID;
            productDoc.DocumentCulture = cultureCode;
            try
            {
                productDoc.Insert(parentPage, true);
                count++;
            }
            catch
            {
                error += "Unable to create page '" + sku.SKUName + "'.
<br />";

                SKUInfoProvider.DeleteSKUInfo(sku.SKUID);
            }
        }
    }

    // Close file
    if (file != null)
    {
        file.Close();
    }

    // Close reader
    reader.Close();

    // Display the number of created products
    lblInfo.Text = "Number of created products: " + count;
}

return error;
}

/// <summary>
/// Ensures parent page for product pages exists.
/// </summary>
private TreeNode GetProductParentPage()
{
    // Try to get products' parent page
    TreeNode parent = tree.SelectSingleNode(siteName, "/Products/Electronics
/Cellphones", TreeProvider.ALL_CULTURES, true, "cms.menuitem");
```



```
// Parent not found
if (parent == null)
{
    // Get site root
    TreeNode root = tree.SelectSingleNode(siteName, "/", TreeProvider.
ALL_CULTURES, true, "cms.root");
    if (root != null)
    {
        // Create new parent page
        parent = TreeNode.New("cms.menuitem", tree);
        parent.SetValue("MenuItemName", "Custom products");
        parent.DocumentName = "Custom products";
        parent.DocumentCulture = cultureCode;
        parent.Insert(root, true);
    }
}
return parent;
}

/// <summary>
/// Ensures default product page type exists.
/// </summary>
private DataClassInfo GetProductPageType()
{
    DataClassInfo defaultProductType = DataClassInfoProvider.GetDataClassInfo
(className);
    return defaultProductType;
}
}
```

### Changing the import location

If you need to change import location, customize the *GetProductParentPage()* method.

Specifically, you need to modify the value of the *parent* variable:

```
parent = tree.SelectSingleNode(siteName, "/Products/Electronics/Cellphones",
TreeProvider.ALL_CULTURES, true, "cms.menuitem");
```

For example, if you want to import your products directly into the **Products** section, change the path to */Products*.



If you enter an invalid parent node path, the system creates a new page located in the root directory.

The system then adds the imported products under this parent page.

### Changing the product type

If you need to change the product type, customize the value of the *className* variable:

```
string className = "CMSProduct.CellPhone";
```

For example, if you want to import your products as **Tablets**, change the value to *"CMSProduct.Tablet"*.



To be able to perform the import, your website must have appropriate [product types](#) defined.

## Running the product import

Open the *ProductImport.aspx* page in your browser. The system runs the page's code and informs you about the import result.

If you open the **Products** application, you can see the imported products listed under **Electronics -> Cellphones**.

Products

Electronics

Televisions

Cellphones

Apple iPhone 5

HTC One X

Nokia Lumia

Android

BlackBerry

iOS

Symbian

Media Players

Computers

Clothing

Books





















Gifts

Donations

Advanced search

Reset

Search

| Actions  | Product name                    | SKU | Price    |
|--|---------------------------------|-----|----------|
| <input type="checkbox"/>   | Apple iPhone 4S ✓               |     | \$500.00 |
| <input type="checkbox"/>   | Apple iPhone 5 ✓                | 111 | \$869.00 |
| <input type="checkbox"/>   | BlackBerry Storm 9530 ✓         |     | \$159.99 |
| <input type="checkbox"/>   | BlackBerry Torch 9810 Slider ✓  |     | \$599.00 |
| <input type="checkbox"/>   | HTC EVO 3D ✓                    |     | \$349.99 |
| <input type="checkbox"/>   | HTC One X ✓                     |     | \$679.99 |
| <input type="checkbox"/>   | Nokia 701 Dark Steel ✓          |     | \$319.99 |
| <input type="checkbox"/>   | Nokia Lumia ✓                   | 112 | \$539.00 |
| <input type="checkbox"/>   | Nokia X2-01 ✓                   |     | \$99.99  |
| <input type="checkbox"/>   | Samsung Galaxy Nexus GT-i9250 ✓ |     | \$669.99 |