

You can add integration connector classes into your application in two ways:

- [In a separate project \(assembly\)](#)
- [In the App_Code folder](#)

Creating connectors in a separate project

1. Create a **Class Library** project in Visual Studio (for example name it *CustomIntegrationConnector*).
2. Add references to the Kentico libraries:
 - If the connector project is standalone, install the *Kentico.Libraries* NuGet package (see [Using the Kentico API externally](#) for details).
 - If you are adding the connector project to your Kentico solution, reference the DLL files directly:
 - i. Right-click the project and select **Add reference...**
 - ii. Click **Browse...**
 - iii. Add at least the following references from the project's *Lib* directory:
 - CMS.Base
 - CMS.DataEngine
 - CMS.DocumentEngine
 - CMS.Helpers
 - CMS.SiteProvider
 - CMS.Synchronization
 - CMS.SynchronizationEngine
 - CMS.WorkflowEngine
3. Reference the custom class library project from the Kentico application.



Implementing outgoing synchronization for MVC sites

When implementing [outgoing synchronization](#) for [MVC sites](#), you need to reference the custom class library from **both** the MVC and Kentico administration projects. This ensures that the synchronization works for objects created or modified by the MVC live site application (for example users, form data, customers and their orders).

4. Edit and rename the default class in the project and set the class to inherit from **BaseIntegrationConnector**.
5. Override the **Init()** method and set the **ConnectorName** property within this method.
 - The value of the *ConnectorName* property must match the code name of the connector object registered in the administration interface.

```
using CMS.Synchronization;
using CMS.SynchronizationEngine;

public class CMSIntegrationConnector : BaseIntegrationConnector
{
    /// <summary>
    /// Initializes the connector name.
    /// </summary>
    public override void Init()
    {
        // Initializes the connector name (must match the code name of the
        // connector object in the system)
        // GetType().Name uses the name of the class as the
        ConnectorName
        ConnectorName = GetType().Name;
    }
}
```

6. Build the solution.

With the connector class prepared, you now need to:

- Implement [outgoing](#) and/or [incoming](#) synchronization
- [Register the connector in the system](#)

Creating connectors in the App_Code folder



Note: You cannot create your connectors in the *App_Code* folder if you wish to synchronize incoming tasks from external applications logged directly using the API (without a custom communication service). In these scenarios, the external application must be able to reference the connector class, which requires a separate project (and DLL).

1. Open your Kentico web project in Visual Studio (using the **Website.sln** or **WebApp.sln** file).
2. Create a new class in the *App_Code* folder (or *Old_App_Code* on web application installations).
3. Set the class to inherit from **BaseIntegrationConnector**.
4. Override the *Init()* method and set the **ConnectorName** property within this method.
5. Ensure that the system loads the appropriate class when working with the connector using the **RegisterCustomClass** assembly attribute. See [Loading custom classes from App_Code](#) for more information.

```
using CMS;

[assembly: RegisterCustomClass("CMSIntegrationConnector", typeof(
    CMSIntegrationConnector))]
```

6. On web application projects, build the solution.

With the connector class prepared, you now need to:

- Implement [outgoing](#) and/or [incoming](#) synchronization
- Register the connector in the system

You can find an example of a basic connector class on the [Example - Integration connector](#) page.

Registering connectors in the system

Once the connector's class is ready, you need to register the connector as an object in the system:

1. In the Kentico administration interface, open the **Integration bus** application.
2. Select the **Connectors** tab.
3. Click **New connector**.
4. Fill in the **Display name**, **Assembly name** and **Class** and select the **Enabled** check box.

Property	Description
Display name	The name of the connector displayed in the user interface.
Code name	Sets a unique identifier for the connector. Must match the value of the ConnectorName property declared in the connector's class.
Provider class	Specifies the class where the connector class is implemented: <ul style="list-style-type: none"> • Assembly name - the assembly (project) containing the connector class. Select (<i>custom classes</i>) for connectors implemented in the <i>App_Code</i> (or <i>Old_App_Code</i>) folder. • Class - the exact class (including any namespaces) that defines the functionality of the connector. For <i>App_Code</i> classes, the value must match the first parameter of the <i>RegisterCustomClass</i> attribute that loads the class.




Enabled	Indicates if the connector logs and processes integration tasks. Logging and processing of tasks must also be enabled in Settings -> Integration -> Integration bus .
---------	--

5. Click **Save**.



Note: When you add, edit or delete a connector, the system re-initializes all defined connectors.

The system displays a warning icon () next to connectors that are not registered correctly. The most common causes of problems are:

- The value of the **ConnectorName** property in the connector class's **Init** method does not match the **Code name**.
- Incorrect assembly or class name.
- App_Code classes are not loaded correctly. See [Loading custom classes from App_Code](#) for more information.