

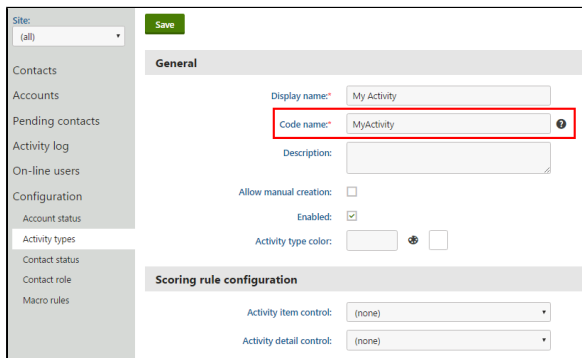
In addition to the [default activity types](#) used by the [contact management](#) feature, you can also create custom activity types:



Note

Only users with the global administrator [privilege level](#) can define custom activity types.

1. Open the **Contact management** application.
2. Switch to **Configuration -> Activity types**.
3. Click **New activity type**.
4. Fill in the display name and note the code name of your new activity type.



The screenshot shows the 'Activity types' configuration page in Kentico. On the left is a sidebar with navigation links: Site, Contacts, Accounts, Pending contacts, Activity log, On-line users, Configuration, Account status, Activity types (selected), Contact status, Contact role, and Macro rules. The main area has a 'Save' button at the top left. Below it is the 'General' section with fields for 'Display name*' (My Activity), 'Code name*' (MyActivity, highlighted with a red box), and 'Description'. There are also checkboxes for 'Allow manual creation' and 'Enabled', and a color picker for 'Activity type color'. Below the 'General' section is the 'Scoring rule configuration' section with dropdowns for 'Activity item control' and 'Activity detail control', both set to '(none)'.

After you add a custom activity type, you can use the Kentico API to log the activity for contacts. You need to:

1. Open your Kentico project in Visual Studio.
2. Create a class representing your activity type. The class must inherit from **CMS.Activities.CustomActivityInitializerBase**.



Either add the class into a custom project within the Kentico solution (recommended) or directly into the Kentico web project (anywhere in the **CMSApp** project for *web application* installations, into the **App_Code** folder for *web site* installations).



Custom activity class example

```
using CMS.Activities;

/// <summary>
/// Custom activity inherits from the CustomActivityInitializerBase class.
/// </summary>
public class MyActivityInitializer : CustomActivityInitializerBase
{
    private readonly string mActivityValue;

    /// <summary>
    /// Default constructor that takes activity data as parameters.
    /// </summary>
    public MyActivityInitializer(string activityValue)
    {
        mActivityValue = activityValue;
    }

    /// <summary>
    /// Initializes an activity with required data.
    /// </summary>
    public override void Initialize(IActivityInfo activity)
    {
        activity.ActivityTitle = "My custom activity title";
        activity.ActivityValue = mActivityValue;
    }

    /// <summary>
    /// The code name of the corresponding activity type in Kentico.
    /// </summary>
    public override string ActivityType
    {
        get
        {
            return "MyActivity";
        }
    }
}
```

3. Call the **Log** method of an instance of the **IActivityLogService** interface within the code where you want the activity to be logged.



Logging the activity

```
using CMS.Core;
using CMS.Activities;
using CMS.Helpers;

...

// Gets an instance of the activity logging service
var service = Service.Resolve<IActivityLogService>();

// Prepares an initializer for logging the activity
var activityInitializer = new MyActivityInitializer("value");

// Logs the activity
service.Log(activityInitializer, CMSHttpContext.Current.Request);
```



Tip: If you need to log your activity in code where the HTTP request context is not available (for example in certain types of custom [scheduled tasks](#)), call the **LogWithoutModifiersAndFilters(IActivityInitializer activityInitializer)** method instead of *Log*.

The system now automatically logs the custom activity when your custom code is triggered.