By handling events, you can customize how the system logs records into its Event log.

The system raises the following types of related events:

- **EventLogEvents.LogEvent.Before** - allows you to customize the data logged for events or cancel logging for certain types of events.
- **EventLogEvents.LogEvent.After** - allows you to perform custom actions after events are successfully logged, for example duplicate certain types of events into a custom log.

Note: Performing demanding operations during event logging can have a negative impact on the performance of the website (particularly on sites under heavy load where a large number of events is logged).

Example

The following example demonstrates how to disable logging for events of a specific type and modify the *Description* text for certain events.

1. Open your Kentico project in Visual Studio (using the **WebSite.sln** or **WebApp.sln** file).
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 a handler method to the **EventLogEvents.LogEvent.Before** event.
4. Perform the required actions within the handler method.
    - Access the data of the logged event through the **Event** property of the handler's **LogEventArgs** parameter.
    - To cancel the logging of an event, call the **Cancel** method of the handler's **LogEventArgs** parameter.

```
using CMS;

using CMS.DataEngine;
using CMS.EventLog;
using CMS.Base;

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

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

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

                // Assigns a handler to the LogEvent.Before event
                EventLogEvents.LogEvent.Before += LogEvent_Before;
        }

        private void LogEvent_Before(object sender, LogEventArgs e)
        {
                // Gets an object representing the event that is being logged
                EventLogInfo eventLogRecord = e.Event;

                // Cancels logging for events with the "CREATEOBJ" or "UPDATEOBJ"
event codes.
                // Disables event log records notifying about the creation or update
of objects in the system,
                // but still allows events related to object deletion.
                string eventCode = eventLogRecord.EventCode;
                if (eventCode.EqualsCSafe("CREATEOBJ") || eventCode.EqualsCSafe
("UPDATEOBJ"))
                {
                        e.Cancel();
                }

                // Adds a custom message to the Description field for events of the
Information type
                if (eventLogRecord.EventType.EqualsCSafe("I"))
                {
                        eventLogRecord.EventDescription += " Custom message";
                }
        }
}
```