You can use event handlers to automatically synchronize content staging and integration bus tasks. Use the API to synchronize the required tasks during the following events:

- **StagingEvents.LogTask.After**
- **IntegrationEvents.LogInternalTask.After**
- **IntegrationEvents.LogExternalTask.After**

Tip: Access the data of the synchronization task through the **Task** property of the event handler's **StagingLogTaskEventArgs** or **IntegrationTaskEventArgs** parameter.

Example

The following example demonstrates how to automatically synchronize staging tasks and outgoing integration tasks:

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 handlers to the **StagingEvents.LogTask.After** and **IntegrationEvents.LogInternalTask.After** events.

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

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

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

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

                // Assigns a handler to the StagingEvents.LogTask.After event
                // This event occurs after the system creates content staging
synchronization tasks (separately for each task)
                StagingEvents.LogTask.After += LogStagingTask_After;

                // Assigns a handler to the IntegrationEvents.LogInternalTask.
After event
                // This event occurs after the system creates outgoing
integration tasks (separately for each task)
                IntegrationEvents.LogInternalTask.After +=
LogIntegrationTask_After;
        }
}
```

4. Define the handler methods (inside the custom module class):

```
// Automatically synchronizes staging tasks
private void LogStagingTask_After(object sender, StagingTaskEventArgs e)
{
        if (e.Task != null)
        {
                // Gets the identifiers of the staging servers for which the task
was created
                var taskServerIds = SynchronizationInfoProvider.
GetSynchronizations()

.Column("SynchronizationServerID")

.WhereEquals("SynchronizationTaskID", e.Task.TaskID);

                // Gets the task's staging servers based on the retrieved
identifiers
                var targetServers = ServerInfoProvider.GetServers().WhereIn
("ServerID", taskServerIds);

                // Processes the task for all relevant servers
                foreach (ServerInfo server in targetServers)
                {
                        // Synchronizes the processed staging task to the target
server
                        new StagingTaskRunner(server.ServerID).RunSynchronization
(e.Task.TaskID);
                }
        }
}

// Automatically synchronizes outgoing integration tasks
private void LogIntegrationTask_After(object sender, IntegrationTaskEventArgs e)
{
        // Gets the info object for an integration connector
        IntegrationConnectorInfo connectorInfo = IntegrationConnectorInfoProvider.
GetIntegrationConnectorInfo("MyConnector");

        if ((connectorInfo != null) && (e.Task != null))
        {
                // Gets an instance of the integration connector class
                BaseIntegrationConnector connector = IntegrationHelper.
GetConnector(connectorInfo.ConnectorName) as BaseIntegrationConnector;

                // Synchronizes the processed integration task
                connector.ProcessInternalTask(e.Task);
        }
}
```

5. Save the class.

The handlers ensure that the system immediately synchronizes:

- All staging tasks to all relevant target servers
- All outgoing integration tasks using a specified connector