OBEC 1.0
OBEC Installation Guide

A developer's guide to installing OBEC

Table of Contents

Introduction

This guide is for developers who are looking for information about how to install the OBEC framework and OBEC Starter Kit and how to create product repositories.

This guide also contains useful information for Sitecore developers who implement webshops and for external commerce system developers who want to use OBEC for integration.

This guide describes how to:

- Install the OBEC framework
- Integrate OBEC with an external commerce system and configure the OBEC connector for this system
- Install the OBEC Developer Starter Kit
- Configure a webshop

Installing the OBEC Framework

You must install OBEC as a package in Sitecore to use it for webshop development or integration with external commerce systems.

System Requirements

The following are the requirements to install the OBEC framework:

- .NET Framework 4.5
- Microsoft SQL Server 2008 R2
- Sitecore CMS 7.0 rev. 130810 or later with DMS. This CMS version is also known as CMS 7.0 Update-1.There are major changes in Update-1, and this is why we do not support the initial 7.0 release anymore.

Note
You must run the application pool in **integrated mode**. For more information, see CMS 7.0 Installation Guide.

Installing OBEC

To Install the Sitecore OBEC package:

1. Use the Sitecore installation wizard to install the Sitecore OBEC package.
2. Select **Republish** to publish everything.
3. If your external commerce system uses the product synchronization feature, you must create a product repository. To create a product repository, see the section *Creating a Product Repository*.

Installed DMS Related Items

When you install the OBEC package, the following DMS items are automatically installed:

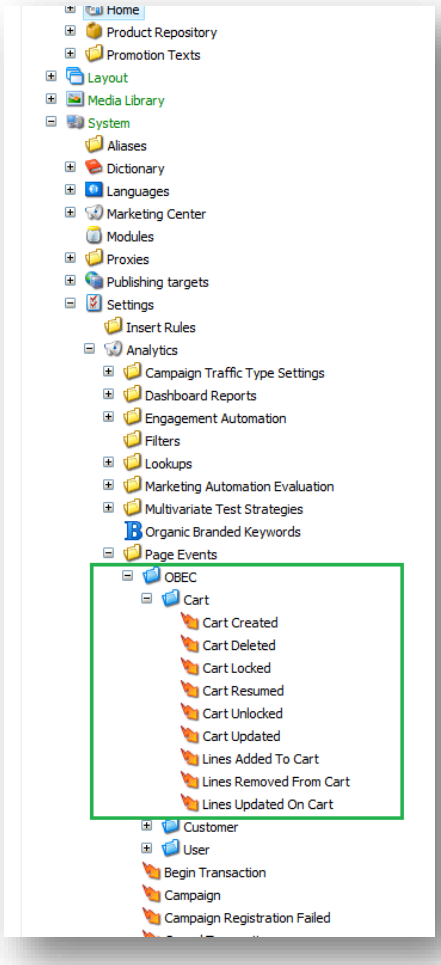
- Page events
- Conditional rendering rule conditions
- Engagement automation plan
- Engagement automation conditions and actions
- Templates and branch templates

Page Events

The OBEC package automatically installs, and deploys the page events with the path `/sitecore/system/Settings/Analytics/Page Events/OBEC`.

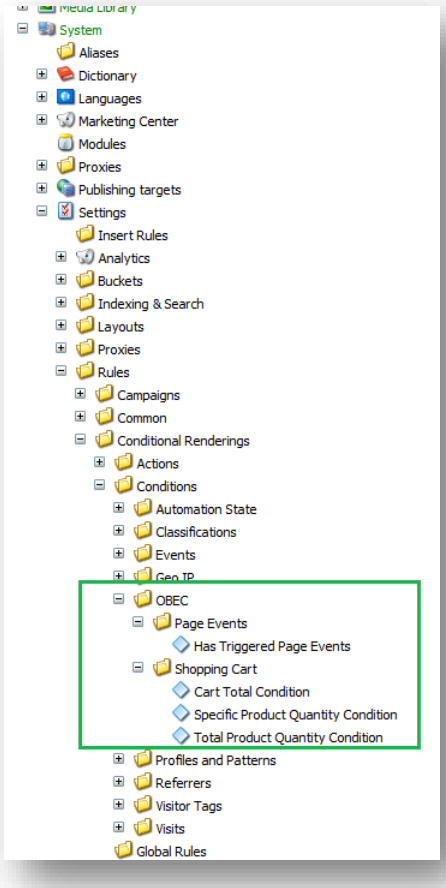
The deployment of the page events transfers the events to the Analytics database.

The following image shows the OBEC events that are installed:



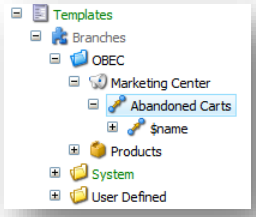
Rendering Rule Conditions

The following image shows the installed OBEC rule conditions in the Content Tree:



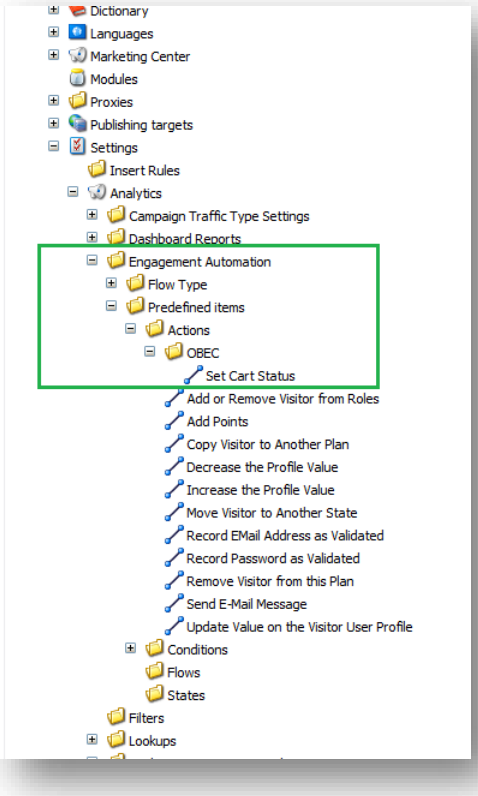
Engagement Automation

The following image shows the installed OBEC engagement automation plans in the Content Tree:

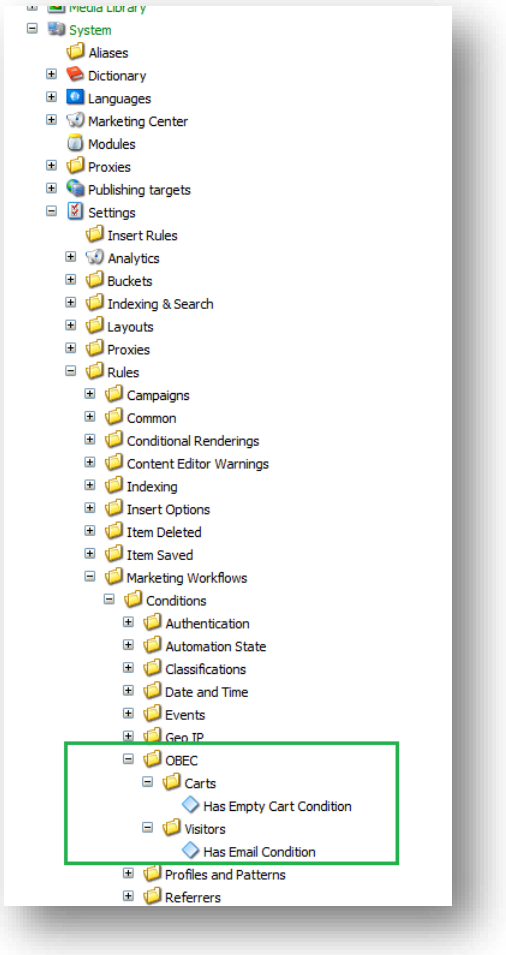


Engagement Automation Conditions and Actions

The following image shows the installed OBEC engagement automation actions in the Content Tree:

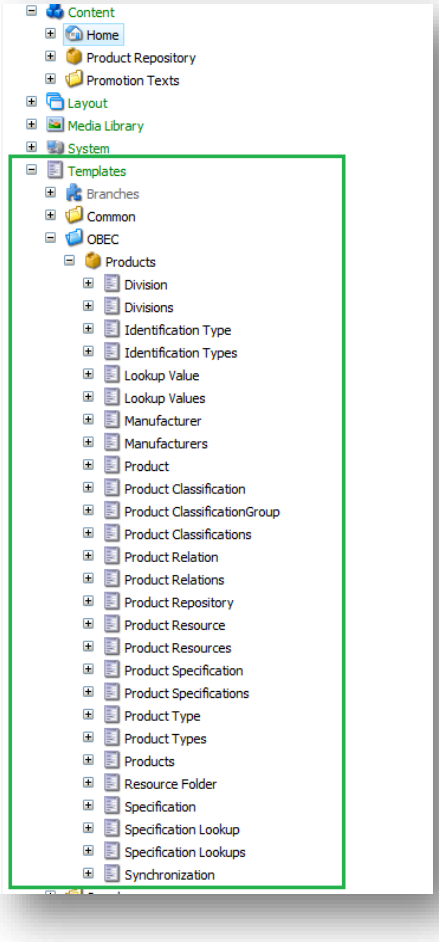


The following image shows the installed OBEC engagement automation conditions in the Content Tree:



Templates

The following image shows the installed OBEC templates in the Content Tree:



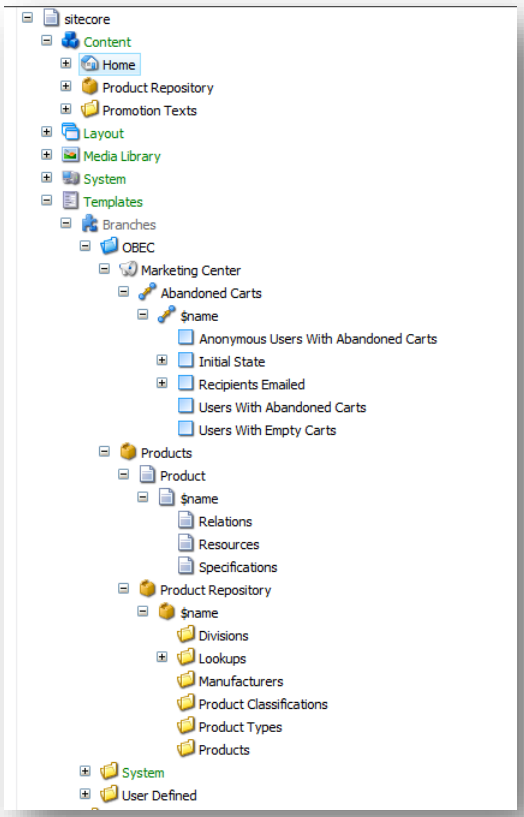
Branch Templates

The following templates are installed as branch templates.

- Abandoned Carts EA plan
- Product
- Product Repository

For more information about branch templates, see the Siteore *Data Definition API Cookbook*.

The following image shows the installed OBEC branch templates in the Content Tree:



Integrating an External Commerce System

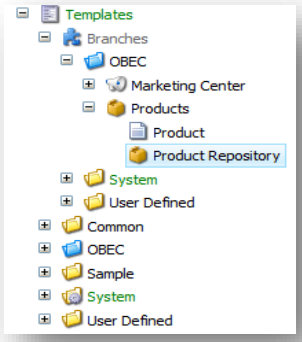
Some external commerce systems use product synchronization, and some do not.

If you want to integrate Sitecore with an external commerce system that uses product synchronization, you must create a product repository. If the external commerce system does not use the product synchronization approach, skip this step.

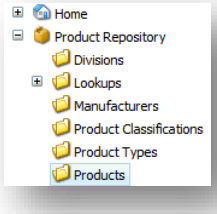
Creating a Product Repository

After you install the OBEC framework package, you can use the following steps to create a product repository:

1. Open the Content Editor and create a product repository item that is based on the `Templates/Branches/OBEC/Products/Product Repository` branch.



The following image shows the created product repository:



Note
You should create only one shared repository across all shops to avoid redundancy.

2. Copy the item path or ID of the new item and paste it in the `Sitecore.OBEC.Products.Config` file, in the value of the `<paths productrespository>` attribute.
3. You can use this attribute to use a custom product repository name or ID or to change the location of the product repository in Sitecore content.

```
<paths productRepository="/sitecore/content/Product Repository">
```

If you change the location of the product repository, you must also change it in the following two files (specified in bold in the following samples):

- In the `Sitecore.Obec.Products.Lucene.Index.Master.config` file:

```
<index id="obec_products_master_index"
type="Sitecore.ContentSearch.LuceneProvider.LuceneIndex,
Sitecore.ContentSearch.LuceneProvider">
    <param desc="name">$(id)</param>
    <param desc="folder">$(id)</param>
    <!-- This initializes index property store. The ID has to be set to the index
    id -->
    <param desc="propertyStore" ref="contentSearch/databasePropertyStore"
      param1="$(id)" />
    <strategies hint="list:AddStrategy">
    <!-- NOTE: order of these is controls the execution order -->
    <strategy ref="contentSearch/indexUpdateStrategies/manual" />
    </strategies>
    <commitPolicyExecutor type="Sitecore.ContentSearch.CommitPolicyExecutor,
      Sitecore.ContentSearch">
    <policies hint="list:AddCommitPolicy">
        <policy type="Sitecore.ContentSearch.TimeIntervalCommitPolicy,
          Sitecore.ContentSearch" />
    </policies>
    </commitPolicyExecutor>
    <locations hint="list:AddCrawler">
        <crawler type="Sitecore.ContentSearch.SitecoreItemCrawler,
          Sitecore.ContentSearch">
            <Database>master</Database>
            <Root>/sitecore/content/Product Repository/Products</Root>
        </crawler>
    </locations>
</index>
```

- The `Sitecore.Obec.Products.Lucene.Index.Web.config` file:

```
<index id="obec_products_web_index"
type="Sitecore.ContentSearch.LuceneProvider.LuceneIndex,
Sitecore.ContentSearch.LuceneProvider">
    <param desc="name">$(id)</param>
    <param desc="folder">$(id)</param>
    <!-- This initializes index property store. The ID has to be set to the index
    id -->
    <param desc="propertyStore" ref="contentSearch/databasePropertyStore"
    param1="$(id)" />
    <strategies hint="list:AddStrategy">
    <!-- NOTE: order of these is controls the execution order -->
    <strategy ref="contentSearch/indexUpdateStrategies/onPublishEndAsync" />
    </strategies>
    <commitPolicyExecutor type="Sitecore.ContentSearch.CommitPolicyExecutor,
      Sitecore.ContentSearch">
    <policies hint="list:AddCommitPolicy">
        <policy type="Sitecore.ContentSearch.TimeIntervalCommitPolicy,
          Sitecore.ContentSearch" />
    </policies>
    </commitPolicyExecutor>
    <locations hint="list:AddCrawler">
        <crawler type="Sitecore.ContentSearch.SitecoreItemCrawler,
          Sitecore.ContentSearch">
```

```
                <Database>web</Database>
                <Root>/sitecore/content/Product Repository/Products</Root>
            </crawler>
        </locations>
    </index>
```

Note
You should store product repository under `/sitecore/content`, so that it does not get mistaken for the main website that typically goes under `/sitecore/content/Home`.

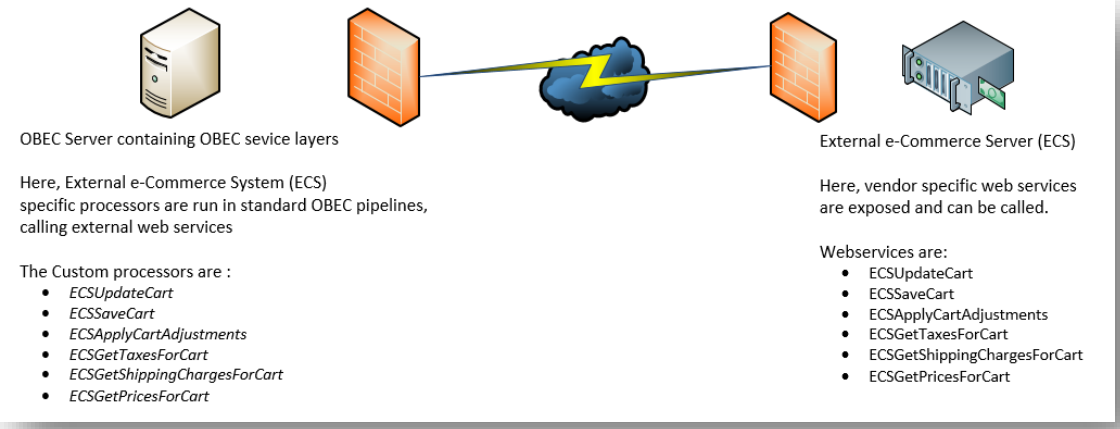
4. In the Publish group, choose **Publish Site**, and then choose **Republish** to publish everything.
5. Rebuild all indexes.

OBEC Connector for NopCommerce

NopCommerce is a reference implementation for a sample external commerce system used to illustrate how to use an integration connector.

The nopCommerce integration consists of two parts:

- A Sitecore package that contains an assembly with processors for the OBEC pipelines and a separate configuration file. The configuration file references the processors and patches the default configuration that inserts the processors in the right places. The configuration also contains a definition of WCF client bindings to web services that are exposed on the server instance that runs NOP.
- nopCommerce plugins that each expose a web service in the service layer. There are currently three plugins for Shopping Cart, Pricing, and Product Synchronization. It is the responsibility of the plugin webservices to manage the interaction with the nopCommerce API.

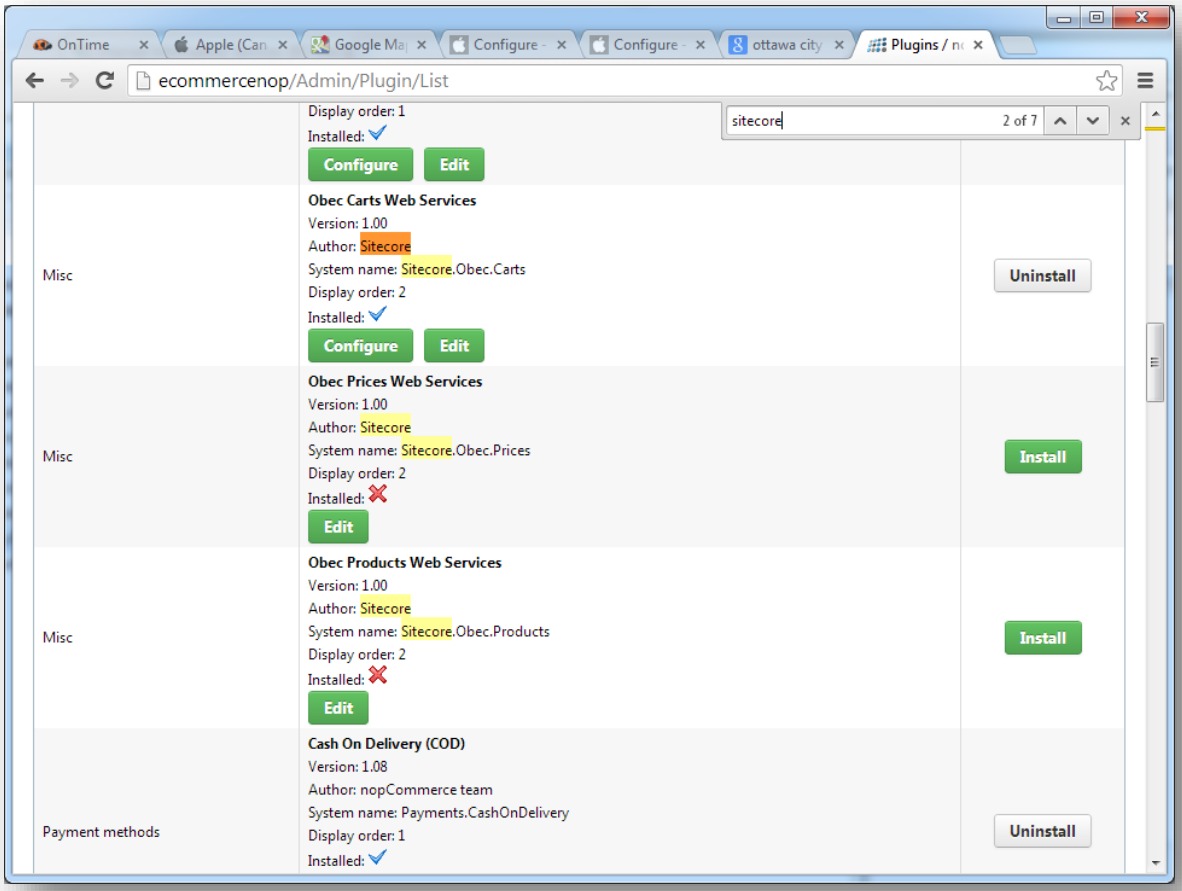


Installing NopCommerce

1. Download and install nopCommerce 3.1 from http://www.nopcommerce.com/, and choose to include sample data to populate Sitecore with.
2. Unzip the following solutions into the `NopCommerceInstallation\Website\Plugins` folder:
   - `Nop.Plugin.Sitecore.Obec.Carts`

- `Nop.Plugin.Sitecore.Obec.Products`
- `Nop.Plugin.Sitecore.Obec.Prices`

3. Use Visual Studio to build the solutions that you extracted in the previous `Plugins` folder.
4. Login to the Administration part of NopCommerce and navigate to the `Configuration\Plugins` section.
5. Install the plugins for the Obec Carts Web Services, Obec Prices Web Services, and Obec Products Web Services:



Installing the NopCommerce Connector

To install the Sitecore OBEC NopCommerce Connector:

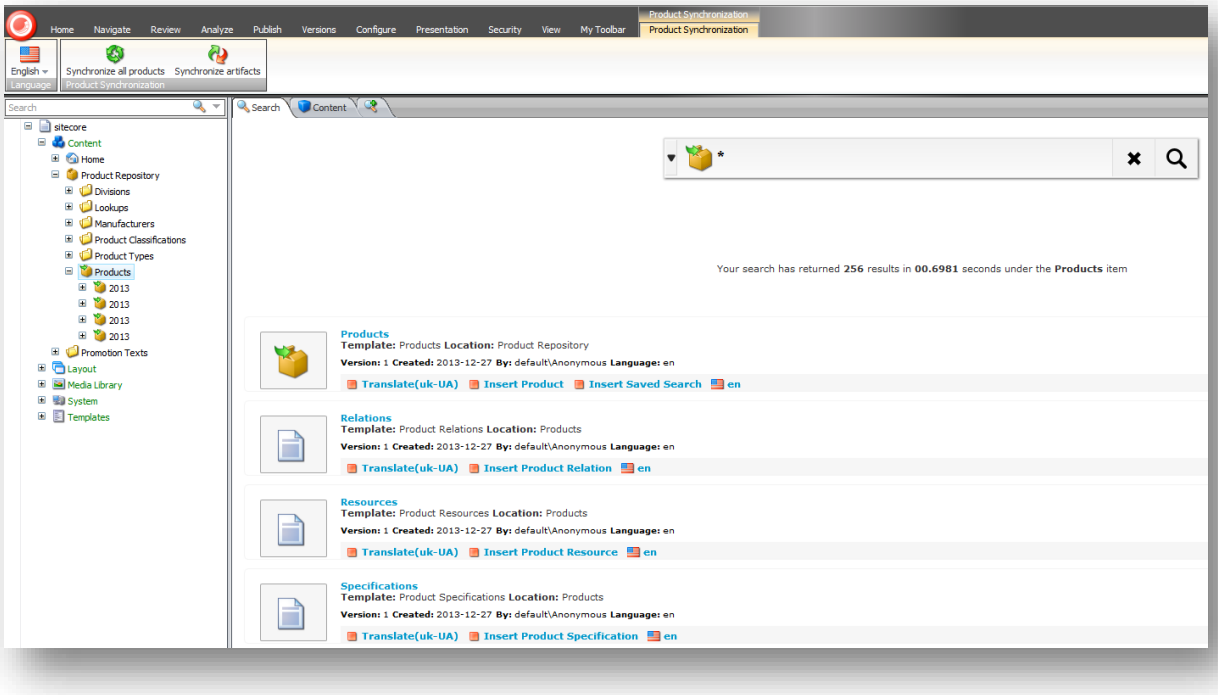
1. Use the Sitecore installation wizard to install Sitecore Obec Connectors NopCommerce 1.0.
2. Enable the `/App_Config/Include/Connectors/Sitecore.Obec.Connectors.NopCommerce.config.disabled` configuration file by removing the last extension `.disabled`.

   In the `/App_Config/Sitecore.Obec.Connectors.NopCommerce.config` file, in the `configuration/system.ServiceModel/client` section, for every Windows

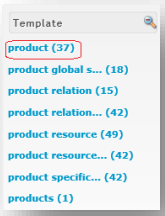
Communication Foundation (WCF) endpoint, in the `address` attribute, change the host name `ecommercenop` to the address of your particular commerce system.

3. On the nopCommerce side for each installed OBEC plugin, update the WCF client endpoints. Navigate to the `website\Plugins` folder, and open the following configuration files:
   - `Nop.Plugin.Sitecore.Obec.Carts.dll.config`
   - `Nop.Plugin.Sitecore.Obec.Products.dll.config`
   - `Nop.Plugin.Sitecore.Obec.Prices.dll.config`

   In each of these files, in the `applicationSettings` node, there is a setting with a value node that contains the URI to the WCF service on the OBEC side. Change the host name `ecommerceobec` to the host name of your webshop solution or your Starter Kit installation. For more information about the Starter kit, see the following section.
4. To synchronize the products from nopCommerce into your OBEC installation, launch Sitecore, open the Content Editor, navigate to the **Product Repository** item, and in the **Product Synchronization** group, choose **Synchronize all products.**
5. To verify that the product repository is now populated, navigate to the `Product Repository\Products` item, and on the **Search** tab, use the asterisk `*` to return all the products:



On the right, select the template facet to return the template products only:



The OBEC Developer Starter Kit

The OBEC Developer Starter Kit is a reference implementation that contains a sample webshop solution and shows how to use the OBEC API. You can easily modify this solution.

For information about the starter kit and its components, see the Architecture Overview document.

In addition to the content that the starter kit adds to Sitecore, the OBEC Developer Starter Kit also contains a Visual Studio project that has a compilable source code and shows how you can use the OBEC API. Before you can run the installation, you need to build it.

The OBEC Developer Starter Kit is only intended as a learning tool to illustrate how to use OBEC API – it is not intended for solutions going to production.

The following section describes the system requirements, installation steps, and the technologies used in the starter kit.

System Requirements

To work with the starter kit, you must have following installed:

- .NET Framework 4.5
- Microsoft SQL Server 2008 R2
- Visual Studio 2012 or later
- ASP.NET MVC 4
- Sitecore CMS 7.0 rev. 130810 or later. This is also called CMS 7.0 Update 1.
- Sitecore OBEC 1.0

Installing the OBEC Developer Starter Kit

The OBEC Developer Starter Kit works with the OBEC Product Data Model. It is assumed that products are stored in the associated product repository item structure. For more information about the product item structure, see the section *Creating a Product Repository*.

To install the OBEC Developer Starter Kit package:

1. In the `~/App_Config/Include/` folder, enable the Sitecore MVC functionality by:
   - Renaming `Sitecore.Mvc.config.disabled` to `Sitecore.Mvc.config`
   - Renaming `Sitecore.MvcAnalytics.config.disabled` to `Sitecore.MvcAnalytics.config`
2. Use the Sitecore installation wizard to install the OBEC Starter Kit package.

3. In the `Web.config` file , ensure that ASP.NET MVC 4 is configured in the `<system.web>` section:

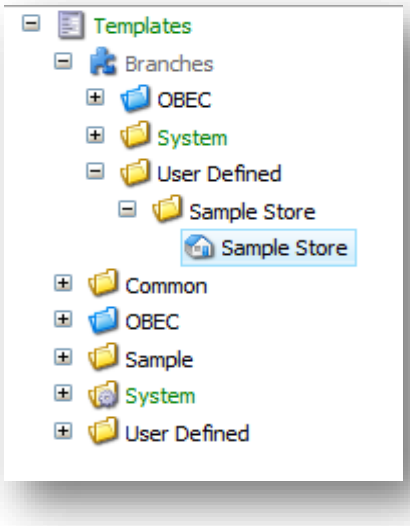
```
<compilation defaultLanguage="c#" debug="false" targetFramework="4.5">
  <assemblies>
    ...
    <add assembly="System.Web.Helpers, Version=2.0.0.0,
    Culture=neutral,PublicKeyToken=31BF3856AD364E35" />
    <add assembly="System.Web.Mvc, Version=4.0.0.0,
    Culture=neutral,PublicKeyToken=31BF3856AD364E35" />
    <add assembly="System.Web.WebPages, Version=2.0.0.0,
    Culture=neutral,PublicKeyToken=31BF3856AD364E35" />
  </assemblies>
</compilation>
```

4. In the `Web.config` file, add the following nodes to the `configuration/runtime` section:

```
<assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
  <dependentAssembly>
    <assemblyIdentity name="System.Web.Mvc" publicKeyToken="31bf3856ad364e35" />
    <bindingRedirect oldVersion="0.0.0.0-4.0.0.0" newVersion="4.0.0.0" />
  </dependentAssembly>
      <dependentAssembly>
    <assemblyIdentity name="Castle.Core" publicKeyToken="407dd0808d44fbdc"
    culture="neutral" />
    <bindingRedirect oldVersion="0.0.0.0-3.2.0.0" newVersion="3.2.0.0" />
  </dependentAssembly>
  <dependentAssembly>
    <assemblyIdentity name="Microsoft.Data.OData"
    publicKeyToken="31bf3856ad364e35" culture="neutral" />
    <bindingRedirect oldVersion="0.0.0.0-5.5.0.0" newVersion="5.5.0.0" />
  </dependentAssembly>
  <dependentAssembly>
    <assemblyIdentity name="Microsoft.Data.Edm" publicKeyToken="31bf3856ad364e35"
    culture="neutral" />
    <bindingRedirect oldVersion="0.0.0.0-5.5.0.0" newVersion="5.5.0.0" />
  </dependentAssembly>
  <dependentAssembly>
    <assemblyIdentity name="System.Spatial" publicKeyToken="31bf3856ad364e35"
    culture="neutral" />
    <bindingRedirect oldVersion="0.0.0.0-5.5.0.0" newVersion="5.5.0.0" />
  </dependentAssembly>
  <dependentAssembly>
    <assemblyIdentity name="WebGrease" publicKeyToken="31bf3856ad364e35"
    culture="neutral" />
    <bindingRedirect oldVersion="0.0.0.0-1.3.0.0" newVersion="1.3.0.0" />
  </dependentAssembly>
</assemblyBinding>
```

5. Open the Content Editor and rename the *Home* item in `sitecore/content/`.

6. In `sitecore/content/`, create a new website *Home* item based on the `/Branches/User Defined/Sample Store/Sample Store` branch.



Now, the `~/App_Config/Include/Obec/Sitecore.Obec.StarterKit.config` file includes the site definition:

```
<site name="StarterKit" patch:before="site[@name='website']"
            virtualFolder="/"
            physicalFolder="/"
            rootPath="/sitecore/content"
            startItem="/home"
            database="web"
            domain="extranet"
            allowDebug="true"
            cacheHtml="true"
            htmlCacheSize="10MB"
            enablePreview="true"
            enableWebEdit="true"
            enableDebugger="true"
            disableClientData="false" />
```

7. Delete the original Home item.
8. Navigate to `/Marketing Center/Engagement Plans/OBEC` and create a Starter Kit Abandoned Carts item based on the `/Branches/OBEC/Marketing Center/Abandoned Carts` branch.

Note
The default implementation of the Cart service layer uses an engagement automation (EA) plan and stores the cart data along with the EA state information. For more information, see the *Configuring a Webshop* section.

In OBEC, the default implementation searches for an EA Plan that has a name that consists of the name of the site and `Abandoned Carts` (see the previous configuration snippet). If you use a different name for the plan, you need to update the configuration or the implementation.

The configuration approach is explained in the *Configuring a Webshop* section.

In the implementation, there is an interface called `IEaPlanProvider` with a default implementation called `EaPlanProvider`. The `EaPlanProvider` class returns the EA plan based on the naming scheme described in it.

In the `/App_Config/Include/Sitecore.Obec.Config` file, you can use the following setting to replace the default implementation:

```
<eaPlanProvider type="Sitecore.Obec.Multishop.EaPlanProvider, Sitecore.Obec"
 singleInstance="true" />
```

9. Deploy the engagement plan and publish the content.
10. In Visual Studio, build and run `Website\Sitecore.Obec.StarterKit.sln`.
11. If you have already installed the connector for nopCommerce, you can synchronize the product data, publish the content, and then rebuild all indexes.

Technologies Used in the Starter Kit

The OBEC Starter Kit uses the following technologies:

- Bootstrap: a free collection of tools for creating websites and web applications. It contains HTML and CSS-based design templates for typography, forms, buttons, navigation, and other interface components. This is in addition to optional JavaScript extensions.

  In the starter kit, Bootstrap HTML and CSS design templates are the output of the Sitecore renderings. For more information, see the *getBootstrap* website.

- Castle Windsor: the mature Inversion of Control container available for .NET

  For more information, see the *Castle website*.

- Glass-mapper: an object mapping framework that maps Sitecore items directly into an object model. In the starter kit, Glass-mapper is used to map the product data into the models for the MVC-based UI components.

- KnockoutJS: a JavaScript library that helps you to create rich, responsive display and editor user interfaces with a clean underlying data model. For more information, see the *knockoutjs* website.

Configuring a Webshop

The default configuration of OBEC uses the Engagement Automation state for the following purposes:

- To store a copy of the shopping cart. For this purpose an EA plan needs to be created and configured for the shop.

  If you do not want to save the cart to an EA state, you can disable the cart by removing the following processors from the pipelines found in the `/App_Config/Include/Sitecore.Obec.Carts.config` file:

  - The `SaveToEAState` processor in the `SaveCart` pipeline
  - The `FindCartInEaState` processor in the `CreateOrResumeCart` pipeline
  - The `LoadCartFromEaState` processor in the `LoadCart` pipeline
  - The `DeleteCartFromEaState` processor in the `DeleteCart` pipeline

- To track visitors and follow up when they leave the shop prematurely and abandon their shopping cart.

  In the `/App_Config/Include/Sitecore.Obec.Carts.config` file, if you do not want to use the Abandoned Cart plan, you can disable it by removing the following processors:

  - The `AddVisitorToEaPlan` processor in the `SaveCart` pipeline
  - The `MoveVisitorToInitialState` processor in the `ResumeCart` pipeline

To make OBEC work with carts, you need to configure an engagement automation plan for each shop:

1. Create the engagement plan based on the preconfigured Abandoned Carts branch under the path: `sitecore/system/Marketing Center/Engagement Plans/OBEC`.
2. Deploy the plan and publish it.
3. Bind the plan with the OBEC framework
   - The default implementation uses the following naming convention: `{Site name} Abandoned Carts`. For the Starter Kit, it becomes `StarterKit Abandoned Carts`.
   - If you want to use a custom EA Plan name, add the attribute `engagementPlanName` under the site definition `engagementPlanName ="My store"`. When you do this the website is connected to the `sitecore/system/Marketing Center/Engagement Plans/OBEC/My store` plan.

```
<site name="StarterKit" patch:before="site[@name='website']"
    engagementPlanName ="My store"
… />
```