ACTIVE COMMERCE

TAKE BACK YOUR CUSTOMER.

INSTALLATION / QUICK START

SITECORE E-COMMERCE DONE RIGHT. ➡ ACTIVECOMMERCE.COM

Active Commerce Installation

And Developer Quick Start

Prerequisites

- Sitecore CMS 6.5 and DMS 2.0.1 (rev. 120706)
- .NET Framework 4.0
- ASP.NET MVC 2
- SQL Server 2008 or higher
- Team Development for Sitecore (TDS) (if using Visual Studio solution template)

Installation

1. If creating a new site:
    a. Use the Sitecore 6.5 installer as normal to install new instance of Sitecore.
    b. Switch to .NET Framework v4.0 when defining IIS web site (under Advanced).
    c. Install DMS 2.0.1
        i. Follow Sitecore instructions to attach DMS databases, install Analytics configuration files, and configure analytics connection string.
2. If adding on to an existing site:
    a. Ensure IIS app pool is set to .NET Framework v4.0 and all prerequisites are met.
3. Place the supplied Active Commerce license file, “license_activecommerce.xml”, in the Data folder of your site.
4. In Sitecore, use the Installation Wizard to install the following modules, in this order:

   a. Sitecore E-Commerce Services (version 1.2)
      i. Download from the Sitecore Developer Network
      ii. *NOTE: Do not install SES Examples package.*
   b. Active Commerce
      i. Sitecore install package can be obtained from your Active Commerce representative.
      ii. Select "Merge - Merge" for any SES conflicts.
5. Modify Web.config
   a. You can skip this step if following the *Visual Studio Solution Setup*, which includes an adjusted Web.config as part of the template.
   b. Make necessary web.config changes for integrating ASP.NET MVC 2 + .NET Framework 4.0
      i. Use the "web.config.Net4.MVC" file provided as part of the SItecore install (found in your web root).
      ii. Alternatively, follow instructions provided by Sitecore, found here.
      iii. *NOTE: Only perform the web.config changes (section 3.1.1)*
   c. Add to <system.webServer><modules> section (after last node):
```
<add name="CassetteHttpModule"
type="Cassette.Aspnet.CassetteHttpModule, Cassette.Aspnet" />
```
   d. Add to <system.webServer><handlers> section:
   (after "WebDAVRoot64" node)
```
<add verb="*" path="skinned_resource.ashx"
type="ActiveCommerce.Skinning.SkinningHandler,
ActiveCommerce.Kernel, Version=1.0.0.0, Culture=neutral"
name="SkinnedResourceHandler" resourceType="Unspecified" />
```
   (after last node)
```
<add name="CassetteHttpHandler" verb="*" path="cassette.axd"
preCondition="integratedMode" allowPathInfo="true"
type="Cassette.Aspnet.CassetteHttpHandler, Cassette.Aspnet" />
```
   e. Add to <system.web><httpModules> section (after last node):
```
<add name="CassetteHttpModule"
type="Cassette.Aspnet.CassetteHttpModule, Cassette.Aspnet" />
```
   f. Add to <system.web><httpHandlers> section:
   (before first node)
```
<add verb="*" path="skinned_resource.ashx"
type="ActiveCommerce.Skinning.SkinningHandler,
ActiveCommerce.Kernel, Version=1.0.0.0, Culture=neutral" />
```
   (after last node)
```
<add verb="*" path="cassette.axd"
type="Cassette.Aspnet.CassetteHttpHandler, Cassette.Aspnet" />
```
6. Run your site, log in to Sitecore, and make sure you don't receive any errors.

See Active Commerce Configuration Guide - Scaling Active Commerce for information about setting up Active Commerce in a distributed environment.

Troubleshooting

- **Problem:** I'm getting a "Could not load file or assembly 'xxx' or one of its dependencies. The located assembly's manifest definition does not match the assembly reference." error.
- **Solution:** These can usually be resolved by adding a <runtime><assemblyBinding> redirection. This is added to the root <configuration> node. Example <runtime> section:

```
<runtime>
  <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
      <assemblyIdentity name="System.Web.Mvc"
publicKeyToken="31bf3856ad364e35" culture="neutral" />
      <bindingRedirect oldVersion="0.0.0.0-2.0.0.0"
newVersion="2.0.0.0" />
      </dependentAssembly>
      <dependentAssembly>
      <assemblyIdentity name="AjaxMin"
publicKeyToken="21ef50ce11b5d80f" culture="neutral" />
      <bindingRedirect oldVersion="0.0.0.0-4.67.4639.17277"
newVersion="4.67.4639.17277" />
      </dependentAssembly>
  </assemblyBinding>
</runtime>
```

Creating a New Site

1. In the Sitecore Content Editor, select the sitecore\Content item.
2. Click "Active Commerce Website," which is now an insert option.
3. Enter the name of your site and click OK.
   NOTE: The name entered here will be used throughout as a default name for various auto-generated items.
4. Once complete, you should be able to preview the site (Home item).
5. Modify the site definition.
   a. You can skip this step if following the *Visual Studio Solution Setup*, which includes an adjusted SiteDefinition.config as part of the template.
   b. Add the following attribute:
      EcommerceSiteSettings="/Site Settings"
6. Note that workflow is enabled on much of the default Active Commerce content tree, and it is necessary to complete workflow on these items in order to publish them.

Visual Studio Solution Setup

The Visual Studio solution template is intended to speed up the startup process for new Active Commerce projects. The solution not only includes references and configuration specific to Active

Commerce, but it also includes Sitecore best practices and extensions we've found helpful along the way. Even if you don't use the template directly, i.e., you're integrating into an existing site/ solution, it can be a handy reference.

1. Download the Visual Studio solution template: https://github.com/ActiveCommerce/solution-template
2. Before opening, do a search and replace of file contents, replacing *MyActiveCommerce* with the name of your project.
    a. This should replace namespaces, names and paths of projects in .csproj files, etc.
    b. Won't always be this simple, but it gets you most of the way there.
3. Don't open it yet. Rename the solution file to your project name, and replace *MyActiveCommerce* in folder and project file names with the same name as you used above.
4. Confirm src\deploy.targets contains the appropriate path name for your Sitecore install.
    a. This is used to resolve project assembly references.
5. Open the Visual Studio solution.
    a. Confirm project names, class namespaces, etc. are as you need them for your project.
    b. Confirm Sitecore DLL references aren't broken.
6. Configure the TDS projects (Sitecore.Core and Sitecore.Master) for your project's host name and Sitecore path.
    a. See TDS Getting Started Guide for details.
    b. See recommendations below. Remove these projects if not using TDS.
7. Edit the App_Config\Include\SiteDefinition.config
    a. Change the site name.
    b. Change the site hostName.
    c. Check the site rootPath.
    d. Fix the site name within the publish:end handler.
    e. Fix the site name within the publish:end:remote handler.
8. Confirm App_Config\Include\DataFolder.config contains the appropriate path name for your Sitecore install.
9. Edit App_Config\Include\MailServer.config
    a. Change MailServer, Port, etc appropriately for your SMTP configuration.
10. Pull the site's ConnectionStrings.config into the project. Ensure the *analytics* connection string is there (should have been done as part of the DMS install).
11. Build the solution.
    a. The build should be successful (address any errors).
    b. If using TDS, files are automatically copied out to your site web root.
        i. If files were not copied, check the Sitecore.Master project properties, and ensure Build > Sitecore Deploy Folder is set correctly.
    c. If not using TDS, publish the web project and copy files to to your site web root.
12. Run your site and make sure all is good.

Recommended Sitecore Configuration, Enhancements, and Tools

Configuration

(Included in the solution template)

- Web.config
    - Active Commerce modifications as noted in installation.
    - Changes to make URLs more SEO-friendly:
        - Replace spaces with underscores ("_").
        - Get rid of the .aspx extension for content and the .ashx extension for media.
    - compilation section pulled out to App_Config\Compilation.config
        - Allows setting debug="false" in a Release build using TDS file replacement.
    - customErrors section pulled out to App_Config\CustomErrors.config
        - Allows disabling in Debug and enabling in QA and Release using TDS file replacement.
        - QA and Release implementations are configured for use of the Sitecore Error Manager module (see *Sitecore Modules* below).
- dataFolder configured in App_Config\Include\DataFolder.config
    - Allows configuring per environment using TDS file replacement.
- Mail server configured in App_Config\Include\MailServer.config
    - Allows configuring per environment using TDS file replacement.
- Sitecore Error Manager module (see *Sitecore Modules* below) configured in Unic.Modules.ErrorManager.x*Env*.configs
    - Use common "Error" and "404" pages. These pages correspond to those automatically added when inserting a new "Active Commerce Website" (see *Creating a New Site* above).

Enhancements

(Included in the solution template)

- EnsureDataSourceIsGUID
    - Ensures that data sources are saved as GUIDs. This (or some form thereof) is a must for any site that heavily utilizes the Page Editor, as Active Commerce does for the home page, promo banners, and "generic content" functionality.
    - Files involved:
        - EnsureDataSourceIsGUID.config
        - EnsureDataSourceIsGUID.cs
- AddParagraphTagIfMissing
    - Many locations where Active Commerce outputs rich text assume that the RTE is wrapping content in a <p> or other block tag. Sometimes it fails to do so.
    - Files involved:
        - AddParagraphTagIfMissing.config

        - AddParagraphTagIfMissing.cs
- LayoutField
    - Ensure our now-GUID-based data sources are resolved in the Links Database.
    - Files involved:
        - FieldTypes.config
        - LayoutField.cs
- Indexing.UpdateInterval
    - Forces Sitecore to check for items to re-index more frequently. This setting really only applies to staged environments, but is a real necessity there to ensure that product data is updated quickly.
    - File involved:
        - Indexing.UpdateInterval.config
- Scheduled Incremental Publishing in combination with Workflow:
    - Active Commerce includes basic workflow out of the box. Only modifications to items marked "Ready to Publish" will be pushed live on site publish. This means we can take advantage of automatic, scheduled publishing without worrying about inadvertent changes going live.
    - This is done via configuration.
    - An implementation has been included to be used on the live site (src\Config\Release\App_Config\Include\Agents.config), but this can easily be added for other environments.

Sitecore Modules

- Sitecore Error Manager - This module fixes (among other things) the incorrect use of a 302 redirect when issuing a 404 or 500.

Visual Studio Plugins

- Team Development for Sitecore (TDS) - Highly recommended. Sitecore items in source control, deployment management, and more!
- Sitecore Rocks — "Makes Sitecore developers happy." What more do you need to say?