Sitecore E-Commerce Services 2.2 for CMS 7.0 or Later

E-Commerce Installation Guide

A developer's guide to install Sitecore E-Commerce Services

Table of Contents

Chapter 1 Introduction 4
Chapter 2 Preparation 5
2.1 Packages Overview 6
2.2 Requirements 7
2.2.1 Operating System Requirements 7
2.2.2 Hardware Requirements 7
2.2.3 Database Requirements 7
2.2.4 .NET Framework Requirements 7
2.2.5 ADO.NET Entity Framework 7
2.2.6 Package Requirements 8
Sitecore E-Commerce Services 2.2 8
Sitecore E-Commerce Order Manager 2.2 8
Sitecore E-Commerce Examples 2.2 8
Chapter 3 Installing Sitecore E-Commerce Services 10
3.1 Installing the Core Package 11
3.2 Installation Troubleshooting 14
3.2.1 CMS Content Editor Error 14
Chapter 4 Installing Order Manager 15
4.1 Order Manager Architecture Options 16
4.1.1 Simple Architecture 16
4.1.2 Third Party Integration 17
4.1.3 Distributed Architecture 18
4.2 Installing Order Manager 19
4.3 Order Manager Post Installation Steps 21
4.3.1 Setting the Web Site Definition 21
4.3.2 Setting the Order Manager Website Definition 23
4.3.3 Setting the Order Manager Website Context Name 23
Chapter 5 Installing Sitecore E-Commerce Example Pages 25
5.1 Installing the Example Pages Package 26
Chapter 6 Configuring E-Commerce Search 30
6.1 E-Commerce Search Overview 31
6.2 Configuring E-Commerce Search 32
6.2.1 Publish the Entire Website 32
6.2.2 Configure Search in the E-Commerce Product Repository 32
Editing the Root Node to Access the Product Repository 32
6.2.3 Configure Search in the E-Commerce Webshop 34
Editing the Root Node to Access the Site Root 34
6.2.4 Rebuild the Search Indexes 35
6.2.5 Rebuild the Link Database 36
Chapter 7 Setting up a Distributed Environment 38
7.1 Installing SES and OM in a Distributed Environment 39
7.1.1 Installing a CM Server Instance and a CD Server Instance 40
7.1.2 Installing an Order Manager Server Instance 41
7.1.3 Installing Multiple CD Instances 42
Chapter 8 Scaling E-Commerce Services 43
8.1 Installing Multiple Instances 44
8.1.1 Scaling Sitecore E-Commerce Services 44
8.1.2 Scaling SES with Sitecore CMS 7.0 or Later 44
8.1.3 Scaling SES with the Service Model 44
The Separation of Content Management and Content Delivery Servers 45

Orders 46
Product Price Information 46
8.1.4 The Service Model 47
8.2 Configuring Multiple Instances 49
8.2.1 Content Management Server Configuration 49
8.2.2 Content Delivery Server Configuration 49
Configuring the Lucene Search Index 52
SES Services Configuration Requirements 53
8.2.3 WCF Configuration Notes 53
Chapter 9 Deploying SES on Azure 54
9.1 Requirements 55
9.2 Deploying the Content Editing Environment 57
9.2.1 First deployment 57
9.2.2 Deploying the Order and Logging databases from SES to Azure 58
9.2.3 Second Deployment 59
9.3 Deploying the Content Delivery Environment 60
9.3.1 First Deployment 60

Chapter 1 Introduction

Sitecore E-Commerce Services is a framework for implementing e-commerce solutions on the Sitecore platform.

This document explains how you prepare for installation and the steps you need to take to install Sitecore E-Commerce Services and other Sitecore E-Commerce Services applications, such as Sitecore E-Commerce Services Example Pages and Order Manager. It also includes more advanced topics such as how to improve scalability by setting up Sitecore E-Commerce Services in a distributed environment.

This document contains the following chapters:

- **Chapter 1 — Introduction**
  This chapter gives a summary of the content contained in each chapter.
- **Chapter 2 — Preparation**
  This chapter outlines the requirements and prerequisites you need before you can install Sitecore E-Commerce Services.
- **Chapter 3 — Installing Sitecore E-Commerce Services**
  This chapter explains how to install Sitecore E-Commerce Services and its prerequisite components.
- **Chapter 4 — Installing Order Manager**
  This chapter explains how to install Order Manager and its prerequisites.
- **Chapter 5 — Installing Sitecore E-Commerce Example Pages**
  This chapter explains how to install the Sitecore E-Commerce example webshop pages.
- **Chapter 6 — Configuring E-Commerce Search**
  This chapter explains how you configure search for the Sitecore E-Commerce Services product repository and the webshop.
- **Chapter 7 — Setting up a Distributed Environment**
  This chapter explains how to plan and set up a distributed environment that includes Sitecore E-Commerce Services and Order Manager.
- **Chapter 8 — Scaling E-Commerce Services**
  This chapter explains how you can improve performance, scalability and security in your E-Commerce solution.
- **Chapter 9 – Deploying SES on Azure**
  This chapter explains how you can deploy SES on Azure.

Chapter 2

Preparation

Use the information in this section to prepare yourself for the installation of Sitecore E-Commerce Services. This chapter outlines the Sitecore E-Commerce Solution packages you can install and specifies the hardware and software requirements you need before proceeding any further with the installation process.

This chapter contains the following sections:

- Packages Overview
- Requirements

2.1 Packages Overview

Before installing the Sitecore E-Commerce Services or Order Manager, familiarize yourself with the contents of the Sitecore E-Commerce packages and the installation prerequisites.

Note
For more information about installation prerequisites for the Sitecore E-Commerce Services, see the Sitecore Developers Network (SDN).

There are three different Sitecore E-Commerce packages that you can install:

- Sitecore E-Commerce Services 2.2 rev. 140219 (core package).
- Sitecore E-Commerce Order Manager 2.2 rev. 140219.
- Sitecore E-Commerce Examples 2.2 rev. 140219.

Important
When you install these Sitecore E-Commerce packages, Sitecore DMS is an optional component and not a requirement.

2.2 Requirements

Sitecore E-Commerce Services and Order Manager run on Sitecore CMS and DMS, and requires the same hardware and software.

2.2.1 Operating System Requirements

You can host Sitecore CMS 7.0 or later on the operating systems that support .NET Framework 4.5, such as:

- Windows Server 2008 (32/64-bit) SP2+
- Windows Server 2008 R2 (32/64-bit) SP1+
- Windows Server 2012 (32/64-bit)
- Windows Vista (32/64-bit) SP2+
- Windows 7 (32/64-bit, Home Premium and higher) SP1+
- Windows 8 (32/64-bit, Professional and higher)

Important

Visit Windows Update website http://windowsupdate.microsoft.com and install all the appropriate service packs and security updates on all of your Sitecore CMS host and client computers.

For more information see the *Sitecore CMS 7.0 (or higher) Installation Guide* on SDN.

2.2.2 Hardware Requirements

See hardware requirements for Sitecore CMS 7. For more information, see the *Sitecore CMS 7.0 Installation Guide* on SDN.

2.2.3 Database Requirements

- MS SQL Server 2008 R2 SP1.
- MS SQL Server 2012.

For more information on database requirements, see the *Sitecore CMS 7.0 Installation Guide* on SDN.

2.2.4 .NET Framework Requirements

Sitecore CMS 7 requires .NET Framework 4.5. You should apply the available updates to the .NET Framework to every Sitecore host.

For information about the database requirements, see the *Sitecore CMS 7.0 Installation Guide* on SDN.

2.2.5 ADO.NET Entity Framework

ADO.NET Entity Framework (EF) is an object-relational mapping (ORM) framework for the .NET Framework. It allows developers to write .NET object oriented code and map it to relational databases such as SQL Server.

Depending on the database that you want to use, install the Microsoft Entity Framework 5.0 or later for SQL Server. The assembly of ADO. NET is already in the SES package.

2.2.6 Package Requirements

This section lists the requirements that you need for each of the following E-Commerce products:

Sitecore E-Commerce Services 2.2

This package contains the core functionality of Sitecore E-Commerce Services. It does not contain any example pages.

Before installing the core package, install the following components in this order:

- Sitecore CMS 7.0 or later.
- Sitecore DMS 7.0 or later (optional).
- Microsoft ADO.NET Entity Framework 5.0 or later for SQL Server is also dependency. The assembly of ADO. NET is already in the SES package.

Note
If you want to run the `OrderCreated` pipeline, you must also configure the mail server.

Sitecore E-Commerce Order Manager 2.2

This package contains the Order Manager application and Sitecore Process Enablement and Accelerator Kit (SPEAK). You should have the following requirements before you install the Order Manager package:

- Sitecore CMS 7.0 or later.
- Sitecore DMS 7.0 or later (optional).
- Sitecore E-Commerce Services 2.2 rev. 140219 (core package).
- Microsoft MVC 3 or later.
- Microsoft ADO.NET Entity Framework 5.0 or higher for SQL Server.

Note
SPEAK is a rapid development kit that consists of several Sitecore controls and the main components of the Order Manager application. The SPEAK package runs automatically when you install Order Manager. This is a previous version of the SPEAK framework which is different from the one that is in Sitecore CMS 7.1, but the two versions can work in the same environment without interrupting each other.

Important

Before installing Sitecore E-Commerce Order Manager package, you must open the **`web.config`** file and increase the shutdown timeout value in the **`<system.web />`** section:

```
<system.web>
<hostingEnvironment shutdownTimeout="300" idleTimeout="100" />
</system.web>
```

After the installation, you can remove the **`<hostingEnvironment/>`** section.

Sitecore E-Commerce Examples 2.2

This package only contains the example pages that illustrate some of the functionality of Sitecore E-Commerce Services. However, it does not contain the core functionality or Sitecore CMS.

Install the following components in this order before you install the examples package:

- Sitecore CMS 7.0 or later.
- Sitecore DMS 7.0 or later (optional).Sitecore E-Commerce Services 2.2 rev. 140219 (core package).
- Web Forms for Marketers 2.3.0 rev. 130118 or later.

Note
For smooth performance of the Sitecore E-Commerce Services, you must also configure the mail server (SMTP).

Important
When you install the Sitecore E-Commerce Services core package or Examples, you must follow the correct installation sequence otherwise the installation will fail.

Important
You must have a Sitecore license that includes *Sitecore.Ecommerce*. If you are unsure whether you have the right license, search you license file for the words '*Sitecore.Ecommerce'*.

Chapter 3

Installing Sitecore E-Commerce Services

This chapter outlines the steps you need to take to install the Sitecore E-Commerce Services core package. You must install the core package before you can install other Sitecore E-Commerce applications such as Order Manager or Example Pages.

This chapter includes the following sections:

- Installing the Core Package
- Installation Troubleshooting

3.1 Installing the Core Package

Before you install Sitecore E-Commerce Services read Chapter 2, Preparation and then follow the steps in this section.

Note
It is important that you perform all these steps in the correct order.

Install Sitecore CMS

To Install Sitecore CMS:

1. Download Sitecore CMS from SDN.
2. Install the Sitecore CMS using either the .exe file or the zip archive. Ensure that you have a valid license for *Sitecore.Ecommerce*. Your license file should include the term *Sitecore.Ecommerce*.

For more information on which version of Sitecore CMS or DMS to use, see the ***Error! Reference source ot found.*** section.

Install Sitecore DMS (optional)

To Install Sitecore DMS:

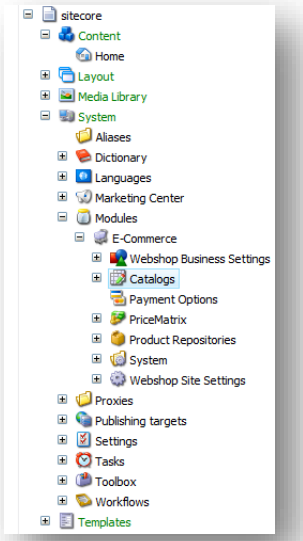
1. Download Sitecore DMS from SDN.
2. Unzip the *Analytics* database to the *Databases* folder in your site root: `<Website>\Databases`
3. In the `<Website>\App_Config\Include` folder, unzip these two configuration files:
   - `Sitecore.Analytics.config`
   - `Sitecore.Analytics.ExcludeRobots.config`
4. Edit the `ConnectionStrings.config` file to point to the *Analytics* database.
5. In SQL Server, attach the *Analytics* database.
6. In Sitecore Desktop, verify that the DMS is running.
7. Clear the browser cache and run IIS reset.

Install Sitecore E-Commerce Services Core Package

To Install Sitecore E-Commerce Services Core Package:

1. Download Sitecore E-Commerce Services 2.2 and save the zip file to the packages folder in your website:
   `<Website>\Data\packages`
2. In the Sitecore CMS Desktop, click **Sitecore**, **Development Tools, Installation Wizard** to install the Sitecore E-Commerce package.
3. In the **Installation Wizard**, click **Next** and click **Browse** to locate the SES package.
4. Select the SES package you want to install and click **Open**.
5. Read and accept the terms of the license agreement.
6. Study the *ReadMe* file. If necessary make a copy of the text and click **Next**.
7. Click **Install**.

After you have installed the core package, an E-Commerce folder with sub items appears in the content tree with the path: `/sitecore/system/Modules/Ecommerce`:



You can see that the `Sitecore/Content/Home` folder is empty. To build your own website, you need to copy the `Sitecore/System/Modules/Ecommerce` subfolders to the `Sitecore/content/Home` folder and build your own web pages.

Attach Order Manager Databases

After installing *Sitecore E-Commerce Services*, you must attach two databases that are required for *Order Manager*:

- *SitecoreEcommerce_Orders.mdf*
- *SitecoreEcommerce_ActionLog.mdf*

To attach the Order Manager databases:

1. Move the *Orders* and *Logging* databases from the `App_Data` folder under the website root to the same folder as your CMS databases (`<Website>/Databases`).
2. Open SQL Server and attach the *Orders* and *Action Log* databases.
3. Use the following path to navigate to the connection strings and open the `ConnectionStrings.config` file:
   `<Website>/App_Config`
4. To point to the databases, add the following two entries to the `ConnectionStrings.config` file:

```
<add name="orders" connectionString="user id=sa;password=*****;Data
Source=.\SQLEXPRESS;Database=SitecoreEcommerce_Orders;MultipleActiveResultSets=true"
providerName ="System.Data.SqlClient"/>

<add name="logging" connectionString="user id=sa;password=*****;Data
Source=.\SQLEXPRESS;Database=SitecoreEcommerce_ActionLog;MultipleActiveResultSets=true"
providerName ="System.Data.SqlClient"/>
```

5. Save your changes and close the `ConnectionStrings.config` file.

Important
Before you begin to use Sitecore E-Commerce Services you must first configure search in the Sitecore Desktop. To find out how to configure search, see Chapter 6, Configuring E-Commerce Search.

3.2 Installation Troubleshooting

Use this section if you encounter problems during the installation of Sitecore E-Commerce Services.

3.2.1 CMS Content Editor Error

In some cases, particularly during high server load, the CMS Content Editor may display the following error message:

Multiple controls with the same ID were found. FindControl requires that controls have unique IDs.

Workaround

Comment out the following processor in the `web.config` file:

```
<processor type="Sitecore.Shell.Applications.ContentEditor.Pipelines.RenderContentEditor.
       RenderSkinedContentEditor, Sitecore.Client"/>
```

This instructs the Content Editor to ignore Skin settings for the current item. These settings are taken from the value of the **__Skin** field or `ContentEditor.DefaultSkin` setting. All items will now be rendered with the default skin and you no longer see the error message.

Chapter 4

Installing Order Manager

This chapter describes the steps you need to take to install Sitecore E-Commerce Services Order Manager. It also outlines three different approaches you can take to planning the architecture that you need for Sitecore E-Commerce Services and Order Manager.

This chapter contains the following sections:

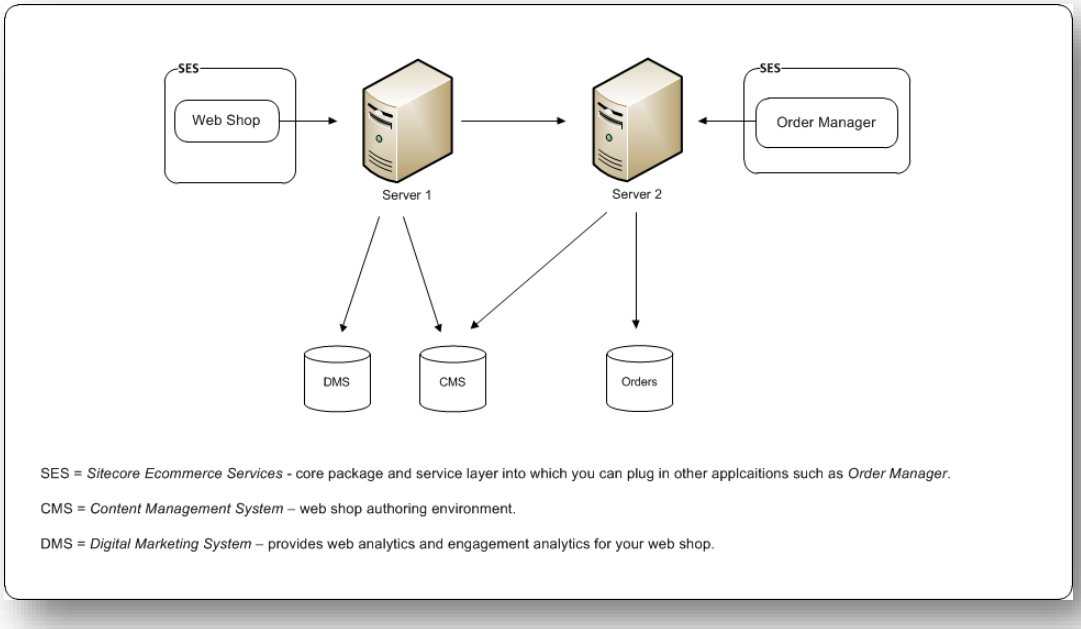
- Order Manager Architecture Options
- Installing Order Manager
- Order Manager Post Installation Steps

4.1 Order Manager Architecture Options

Before installing Order Manager you need to decide which type of architecture is most suitable for your needs. This section outlines three different approaches you can take to setting up Sitecore E-Commerce Services and Order Manager.

4.1.1 Simple Architecture

Simple Order Manager architecture without third party integration:

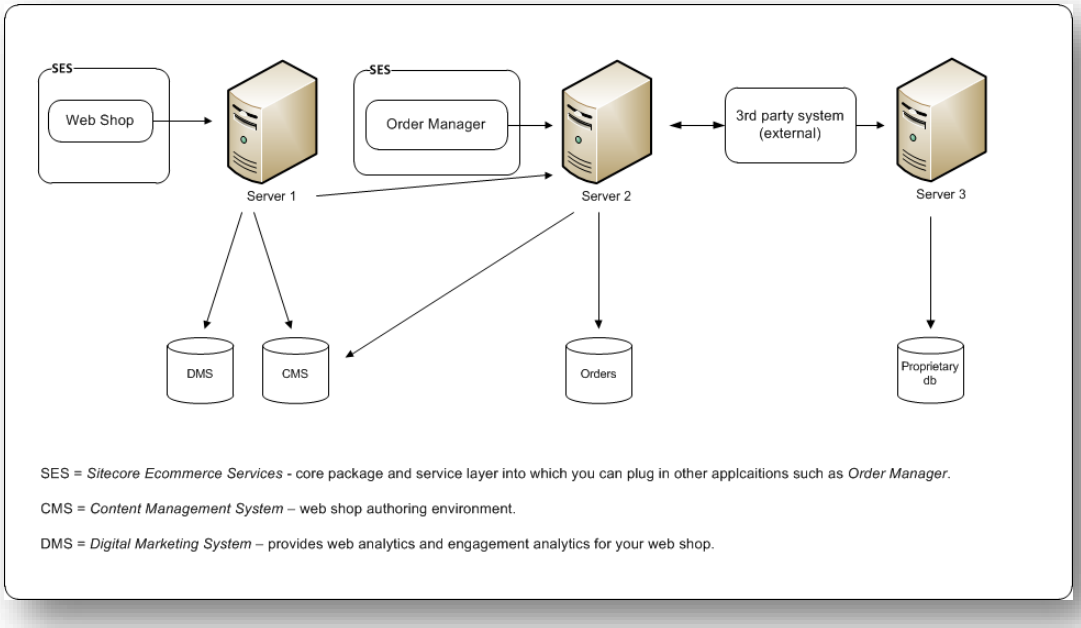


Benefits of Order Manager on its own:

- Simplicity.
- Order Manager handles all customer orders.
- No need for a third party external system to manage orders (basic order handling).
- Stores all orders in its own dedicated SQL Server or Oracle database.

4.1.2 Third Party Integration

Order Manager and integration with a third party ERP system:

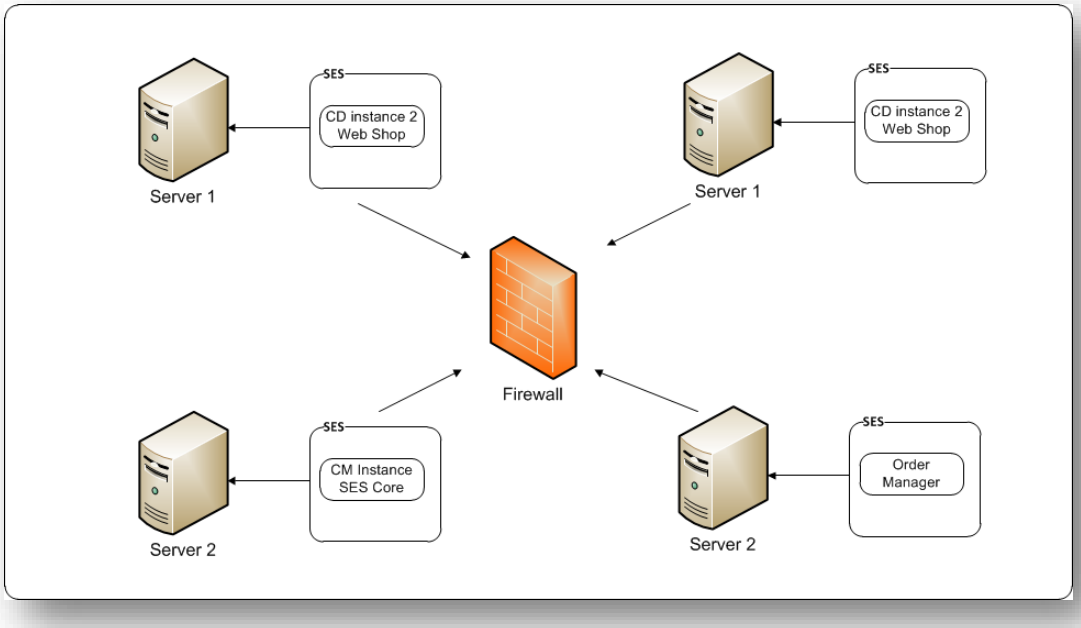


Benefits of integrating Order Manager with a third party ERP system:

- Ease of integration - you can continue to use the existing 3rd party external system to generate invoices, credit notes, package slips, stock maintenance, purchase orders (procurement) and returns. Basically, all the things that are not covered by Order Manager.
- Orders are stored in two places.
  - Order Manager - where you can view orders on the web. For example, when a customer needs to view their order history and status.
  - 3rd party system - where orders can undergo further processing.
- Business processes - You can either implement business processes in Order Manager or in the 3rd party system.

4.1.3 Distributed Architecture

Order Manager in a simple distributed architecture:



Benefits of Order Manager in a distributed architecture:

- Improved scalability.
- Better performance and load balancing.
- Stronger security.

4.2 Installing Order Manager

Follow the steps in this section to install Sitecore E-Commerce Order Manager and all its prerequisite components. Before following these steps ensure that you are familiar with the contents of Chapter 2, Preparation.

Note
It is important that you perform these steps in the correct order.

Install Sitecore CMS

To install the CMS:

1. Download Sitecore CMS from SDN.
2. Install the Sitecore CMS using either the .exe file or the zip archive. Ensure that you have a valid license for Sitecore E-Commerce Services. Your license file should include the term *Sitecore.Ecommerce*.

   For more information on which version of Sitecore CMS or DMS to use, see the **Error! eference source not found.** section.

Install Sitecore DMS (optional)

To install the DMS:

1. Download the Sitecore DMS from SDN.
2. Unzip and attach the *Analytics* database file in SQL Server. Add the include files and verify that the DMS is running.

   For more detailed steps, see Installing the Core Package.

Install SES Core Package

To install the core package:

1. Download the Sitecore E-Commerce Services 2.2 core package from SDN.
2. Install using the Sitecore Installation Wizard.

   For more detailed steps, see Installing the Core Package.

Install SES Order Manager

To install the Order Manager:

1. In the `web.config` file, in the `<system.web>` section, increase the shutdown timeout value by adding the following line:

```
<system.web>
…
<hostingEnvironment shutdownTimeout="300" idleTimeout="100" />
…
</system.web>
```

2. Download Sitecore E-Commerce Services Order Manager from SDN and save the zip file to the packages folder in your website:
   `<Website>\Data\packages`
3. In the Sitecore CMS Desktop, click **Sitecore**, **Development Tools, Installation Wizard** to install the Order Manager package.

4. In the **Installation Wizard**, click **Next** and click **Browse** to locate the Order Manager package.
5. Select the Order Manager package and click **Open**.
6. Read and accept the terms of the license agreement.
7. Study the *ReadMe* file. If necessary make a copy of the text and click **Next**.
8. Click **Install**.
9. After the installation is finished, in the `web.config` file, remove the `<hostingEnvironment />` section.

4.3 Order Manager Post Installation Steps

To complete the installation process after you have installed Order Manager, you must:

1. Set the web site definition.
2. Set the Order Manager website definition.
3. Set the Order Manager website context name.

4.3.1 Setting the Web Site Definition

This section provides some general information and outlines the steps you need to follow to change the web site definition name.

Web Site Definition

Each Sitecore web site has a site name that is defined in the `web.config` file. Sitecore CMS uses the site name "`website`" by default. If you want to specify a different name or configure a different site for your webshop, you need to enter this information in the correct section of the `web config` file. To do this you could change the default name or add an additional line with your site name to the web.config file.

Default web site definition in the `web.config`:

```
        <site name="website"…
```

Note
The default website name definition is "website". Please note that it is not the best practice to change the default website name. It is because this name is used in a few places in the web.config file, such as the Preview.DefaultSite setting, the setting for clearing the cache in the <event name="publish:end"> node, default cache sizes for the website in the <cacheSizes> node and others. You will need to change the name of the website in all these settings, too.

Order Manager Site Context Name

When you install Order Manager, you need to make a reference to the web site definition in the `Sitecore.Ecommerce.Apps.OrderManagement.config`. This setting is the site context name.

Default site context name in the `Sitecore.Ecommerce.Apps.OrderManagement.config`:

```
        <ShopContextName></ShopContextName>
```

When you install Sitecore CMS, SES and Order Manager there are two possible architecture options:

Architecture	Description	Setting
Single Server	CMS Core and Order Manager installed on the same server.	Both instances share the same web.config and site definition settings. See Setting the Web Site Definition.

Architecture	Description	Setting
Distributed Environment	CMS Core and Order Manager installed on separate servers and in different locations	Both instances share the same web.config and site definition settings. See Setting the Order Manager Website Definition You also need to set the site context name for the Order Manager instance. See Setting the Order Manager Website Context Name.

If you need to change the web site definition name, follow the steps in this section.

Important
The default website definition is *website*. In most cases you do not need to change the default name.

To change the default web site definition on the CM Server:

1. In the site root folder, navigate to the Website folder and open the `web.config` file.
2. In the `web.config` file, navigate to the node `<site name="website"`.
3. Change the default name to the name of your web site.
4. Save your changes.

Note
If you have also installed the Sitecore E-Commerce Services Example pages, the `Examples.config` file also contains a site definition that refers to the example pages. The default name is "`example`". You do not need to change this setting.

4.3.2 Setting the Order Manager Website Definition

Order Manager requires a site context to read the Webshop Business settings. This is where, among other things, order states are configured, access is given to products for editing order lines and where product information is used.

If you are using Order Manager in a distributed environment, you must ensure that a web site definition is available in the Order Manager instance that points to the same site as the CM instance. One way to do this is to copy and paste the site definition directly from the CM instance to the OM instance. After copying the site definition, the unnecessary attributes can be removed. The only attributes necessary on the OM instance are as follows:"

```
<site name="example" virtualFolder="/" physicalFolder="/" content="master"
rootPath="/sitecore/content/E-Commerce Examples" startItem="/home"
EcommerceSiteSettings="/Site Settings" patch:before="site[@name='website']"/>
```

4.3.3 Setting the Order Manager Website Context Name

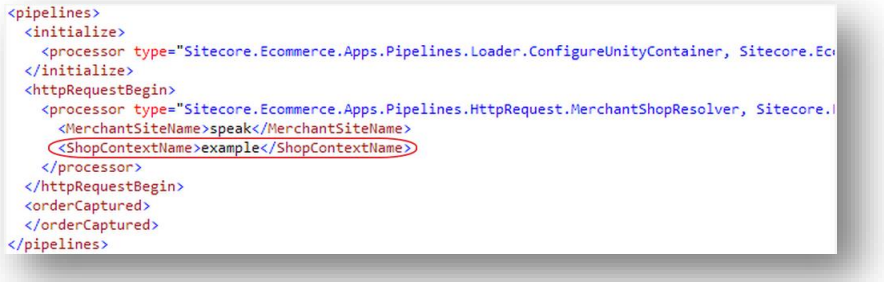
Note
You only need to read this section if you do not configure the context switcher.

When editing order lines in Order Manager, the *Webshop Business Settings (Business Catalog)* must be accessible so that Order Manager can retrieve product and price data from the webshop. Since the *Webshop Business Settings* is indirectly linked to the website, you must set the Order Manager instance context name to match the website definition used in the CM and OM instances.

To set the context name of your Order Manager web site:

1. In the Order Manager site root folder, navigate to the `Sitecore.Ecommerce.Apps.OrderManagement.config` file using the following path: `<Website>/App_Config/Include`
2. Open the `Sitecore.Ecommerce.Apps.OrderManagement.config` file.
3. In the `Sitecore.Ecommerce.Apps.OrderManagement.config` file, edit the `<ShopContextName>` element name to match the name of your webshop site definition name.

By default this name is *website*. Therefore in most cases change the name to *website*, so it matches the site definition used in the CM instance and the OM instance.



4. Save your changes.

Note
The `<MerchantSiteName>speak</MerchantSiteName>` element is set automatically by the system. Do not change this setting.

Chapter 5

Installing Sitecore E-Commerce Example Pages

Sitecore E-Commerce Example Pages enable you to create an example webshop that sells cameras and photography equipment. Installing the example pages allows you to test the functionality in Sitecore E-Commerce Services without actually running a live production web site.

This chapter includes the following sections:

- Installing the Example Pages Package

5.1 Installing the Example Pages Package

Follow these steps to install the Sitecore E-Commerce Services Example Pages package.

Note
It is important that you perform these steps in the correct order.

Install Sitecore CMS

To install Sitecore CMS:

1. Download Sitecore CMS from SDN.
2. Install the Sitecore CMS using either the .exe file or the zip archive. Ensure that you have a valid license for *Sitecore.Ecommerce*. Your license file should include the term *Sitecore.Ecommerce*.

   For more information on which version of Sitecore CMS or DMS to use, see the **Error! eference source not found.** section.

Install Sitecore DMS (optional)

To install Sitecore DMS:

1. Download the Sitecore DMS from SDN.
2. Unzip and attach the *Analytics* database file in SQL Server. Add the include files and verify that the DMS is running.

   For more detailed steps see Installing the Core Package.

Install SES Core Package

To install SES Core Package:

1. Download the Sitecore E-Commerce Services 2.2 core package from SDN.
2. Install using the Sitecore Installation Wizard.

   For more detailed steps, see Installing the Core Package.

Install Web Forms for Marketers

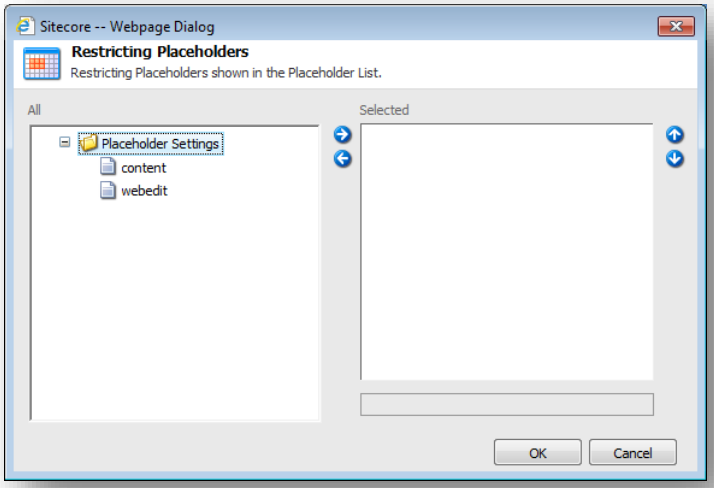
To install Web Forms for Marketers:

1. Download the Sitecore Web Forms for Marketers module from SDN and install it.
2. In the Sitecore CMS Desktop, click **Sitecore**, **Development Tools**, **Installation Wizard** to install *Web Forms for Marketers*.
3. In the Installation Wizard, click **Next** and click **Browse** to locate the **Web Forms for Marketers** package.
4. Select the **Web Forms for Marketers** package and click **Open**.
5. Read and accept the terms of the license agreement.
6. Study the *ReadMe* file. If necessary make a copy of the text.

   The *ReadMe* file suggests that you choose a placeholder during installation and that you republish content to the web database.

   Click **Next**.
7. Click **Install**.

8. During installation you can choose a placeholder.



   This is an optional step. To skip this step click **Cancel** and continue with the installation.

9. When the installation is complete, click Finish to close the Installation Wizard.

   If you are running Web Forms for Marketers 2.3.0 on Sitecore CMS 7.0, after opening a form on the frontend or in the Form Designer, you might encounter the following error message:

   "Could not load file or assembly 'HtmlAgilityPack, Version=1.4.0.0, Culture=neutral, PublicKeyToken=bd319b19eaf3b43a' or one of its dependencies. The located assembly's manifest definition does not match the assembly reference. (Exception from HRESULT: 0x80131040)"

   Workaround:
   Add the following "runtime" section to the web.config file after the </configSections> tag:

```
        <runtime>
                        <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
                                        <dependentAssembly>
                                                        <assemblyIdentity
name="HtmlAgilityPack" publicKeyToken="bd319b19eaf3b43a" culture="neutral" />
                                                        <bindingRedirect
oldVersion="1.4.0.0" newVersion="1.4.6.0" />
                                        </dependentAssembly>
                                        <dependentAssembly>
                                                        <assemblyIdentity
name="Newtonsoft.Json" publicKeyToken="30ad4fe6b2a6aeed" culture="neutral" />
                                                        <bindingRedirect
oldVersion="3.5.0.0" newVersion="4.5.0.0" />
                                        </dependentAssembly>
                                </assemblyBinding>
         </runtime>
```

10. Save the changes and restart the website.

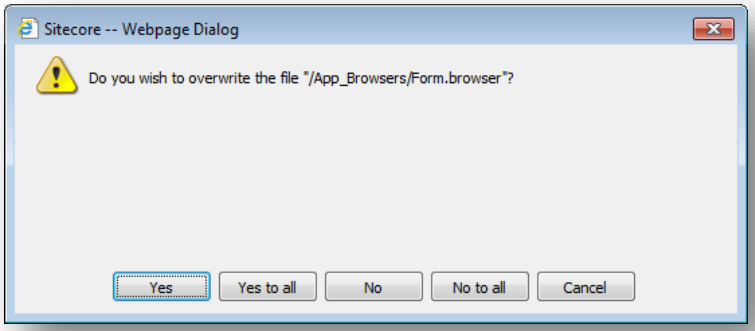
11. If you get the MVC 2.0 error, add the following section to the `web.config` file into `<assemblyBinding>` section:

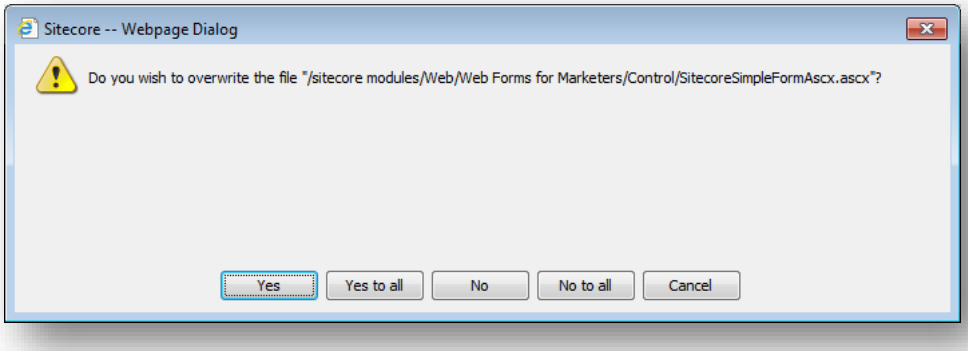
```
<dependentAssembly>
<assemblyIdentity name="System.Web.Mvc" publicKeyToken="31BF3856AD364E35" />
<bindingRedirect oldVersion="2.0.0.0-3.0.0.0" newVersion="4.0.0.0" />
</dependentAssembly>
```

Install the Sitecore E-Commerce Services Example Pages

To install the Sitecore E-Commerce Services Example Pages:

1. Download the Sitecore E-Commerce Example Pages package from SDN and install it.
2. In the Sitecore Desktop, open the Installation Wizard.
3. In the Installation Wizard, click **Next** and click **Upload** to locate the **Sitecore E-Commerce Example Pages** package.
4. Accept the terms and conditions.
5. Study the *ReadMe* file. Check that you have installed all the pre-requisites correctly.
6. Click **Next,** and then click **Install**.
7. During installation, you are asked to overwrite the following files.





   Click **Yes** or **Yes to all**.
8. In the dialog boxes **Item being installed already exists in database**, choose overwrite and click apply.
9. When the installation is complete, click **Finish** to close the Installation Wizard.

Important Note
Before you begin to use Sitecore E-Commerce Services Example Pages you must first configure search in the Sitecore Desktop and the E-Commerce webshop. The next chapter, Configuring E-Commerce Search explains how to perform these post installation steps.

Configure the SMPT Mail Server

In the `OrderCreated` pipeline, which sends out an email when executed, the default configuration of Sitecore E-Commerce Services 2.2 contains the following processor:

```
<processor type="Sitecore.Ecommerce.Visitor.Pipelines.OrderCreated.NotifyCustomer,
Sitecore.Ecommerce.Visitor"/>
```

The pipeline is provided as part of the Sitecore E-Commerce Services 2.2 package, but is not executed in the package. The pipeline is executed in the Sitecore E-Commerce Examples 2.2 package when an order is created, as the last step in the checkout process.

If an exception is thrown, you need to configure the mail server.

Chapter 6

Configuring E-Commerce Search

After you have installed the Sitecore E-Commerce Services module and Example Pages, you must configure E-Commerce search to support different methods of searching for products.

This chapter contains the following sections:

- E-Commerce Search Overview
- Configuring E-Commerce Search

6.1 E-Commerce Search Overview

After you have installed the E-Commerce module you must configure E-Commerce search.

Sitecore E-Commerce Services supports three different search methods:

- Sitecore Query
- Fast Query
- Lucene Search.

SES uses Lucene search indexes to search the Sitecore Desktop and the webshop.

You must ensure that the Lucene search engine can create the appropriate indexes to search for products in the product repository in Sitecore and ensure that visitors can search for products in the webshop.

To configure the search indexes for your E-Commerce installation:

Important
You must perform the steps in this order.

Summary of steps:

1. Publish the entire website.
2. Make changes to the appropriate SES config file:

   SES Core Package

   - Edit the `Sitecore.Ecommerce.config` file to point to the Ecommerce product repository.

   SES Core Package and Example Pages

   - Edit the `Sitecore.Ecommerce.config` file to point to the Ecommerce product repository.
   - Edit the `Sitecore.Ecommerce.Examples.config` file to point to the E-Commerce site root.
3. Rebuild the search indexes.
4. Rebuild the link database.

Note
If you do not install the example pages, the `Sitecore.Ecommerce.Examples.config` file is not added to your installation. To ensure that the web index for your webshop is generated, you must manually add the web index described in this section to the `Sitecore.Ecommerce.config` file or to the `web.config` file.

See the following section for a more detailed description of SES search configuration steps.

6.2 Configuring E-Commerce Search

Perform the following steps to configure Sitecore E-Commerce search.

6.2.1 Publish the Entire Website

To publish the entire website:

1. In the **Content Editor**, on the **Publish** tab, in the **Publish** group, click **Publish** and then click **Publish Site**.
2. In the **Publish** wizard, **Settings** page, select **Republish** *(Publish Everything)*.
3. Select all languages (*English* and *Danish*).
4. Click **Publish**.

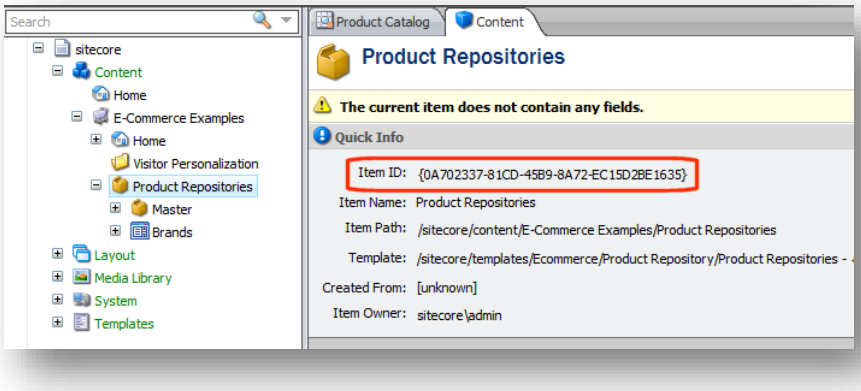
6.2.2 Configure Search in the E-Commerce Product Repository

You must follow the steps in this section if you have installed either the SES core package or the SES Example Pages package.

Editing the Root Node to Access the Product Repository

Before you can use the product search forms to search for products within Sitecore, you must ensure that the `Sitecore.Ecommerce.config` file points to the product repository in Sitecore where you store your product items.

1. In the **Sitecore Desktop**, open the **Content Editor**.
2. In the content tree, navigate to the **Product Repositories** node.



3. Open the `Sitecore.Ecommerce.config` file and navigate to the search section.
   Path to the `Sitecore.Ecommerce.config` file:
   `<Website>\App_Config\Include`

4. Remove the comments tags in the code to make the *locations* section active:

```
    <search>
      <configuration>
        <indexes>
          <index id="products" type="Sitecore.Search.Index, Sitecore.Kernel">
            <param desc="name">$(id)</param>
            <param desc="folder">__products</param>
            <Analyzer type="Sitecore.Ecommerce.Search.LuceneAnalyzer, Sitecore.Ecommerce.Kernel" />
            <!--<locations hint="list:AddCrawler">
              <master type="Sitecore.Ecommerce.Search.DatabaseCrawler, Sitecore.Ecommerce.Kernel">
                <Database>master</Database>
                <Root>{D62FF240-A6E3-4FE3-AF8C-2C6DECB40D03}</Root>
                <Tags>master products</Tags>
              </master>
              <web type="Sitecore.Ecommerce.Search.DatabaseCrawler, Sitecore.Ecommerce.Kernel">
                <Database>web</Database>
                <Root>{D62FF240-A6E3-4FE3-AF8C-2C6DECB40D03}</Root>
                <Tags>web products</Tags>
              </web>
            </locations>-->
          </index>
        </indexes>
      </configuration>
    </search>
  </sitecore>
</configuration>
```

5. Replace both instances of the `<Root>{0A702337-81CD-45B9-8A72-EC15D2BE1635}</Root>` GUID with the Item ID or GUID of the product repository where you store your products.

```
    <search>
      <configuration>
        <indexes>
          <index id="products" type="Sitecore.Search.Index, Sitecore.Kernel">
            <param desc="name">$(id)</param>
            <param desc="folder">__products</param>
            <Analyzer type="Sitecore.Ecommerce.Search.LuceneAnalyzer, Sitecore.Ecommerce.Kernel" />
            <locations hint="list:AddCrawler">
              <master type="Sitecore.Ecommerce.Search.DatabaseCrawler, Sitecore.Ecommerce.Kernel">
                <Database>master</Database>
                <Root>{0A702337-81CD-45B9-8A72-EC15D2BE1635}</Root>
                <Tags>master products</Tags>
              </master>
              <web type="Sitecore.Ecommerce.Search.DatabaseCrawler, Sitecore.Ecommerce.Kernel">
                <Database>web</Database>
                <Root>{0A702337-81CD-45B9-8A72-EC15D2BE1635}</Root>
                <Tags>web products</Tags>
              </web>
            </locations>
          </index>
        </indexes>
      </configuration>
    </search>
  </sitecore>
</configuration>
```

This ensures that the Lucene search engine can index all the items that make up the products that are selected on the various search forms.

6.2.3 Configure Search in the E-Commerce Webshop

You only need to follow the steps in this section if you have installed the SES Example Pages package.

If you have installed the SES Example Pages package there are two things you need to do:

1. Ensure that you can search for products in your product repository.

   See the previous section: Editing the Root Node to Access the Product Repository.

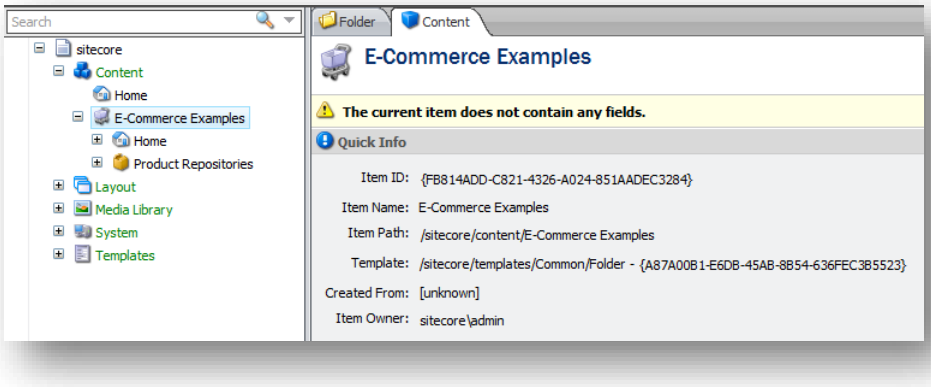
2. Ensure that visitors to your website can search for products in the webshop.

   See this section: Editing the Root Node to Access the Site Root.

Editing the Root Node to Access the Site Root

You must ensure that the `Sitecore.Ecommerce.Examples.config` file points to the root node of your website.

1. In the **Sitecore Desktop**, open the **Content Editor**.
2. In the content tree, navigate to the site root or **Home** node.



3. Open the `Sitecore.Ecommerce.Examples.config` file and navigate to the search section. Path to the `Sitecore.Ecommerce.Examples.config` file: `<Website>\App_Config\Include`
4. Remove the comments tags in the code to make the *locations* section active.

5. Replace the `<Root>{FB814ADD-C821-4326-A024-851AADEC3284}</Root>` GUID with the Item ID or GUID of the root or *Home* node of your E-Commerce website.

```
    <search>
      <configuration>
        <indexes>
          <index id="web" type="Sitecore.Search.Index, Sitecore.Kernel">
            <param desc="name">$(id)</param>
            <param desc="folder">__web</param>
            <Analyzer type="Sitecore.Ecommerce.Search.LuceneAnalyzer, Sitecore.Ecommerce.Kernel" />
            <locations hint="list:AddCrawler">
              <web type="Sitecore.Ecommerce.Search.VirtualProductsCrawler, Sitecore.Ecommerce.Kernel">
                <Database>web</Database>
                <Root>{FB814ADD-C821-4326-A024-851AADEC3284}</Root>
                <Tags>web content</Tags>
              </web>
            </locations>
          </index>
        </indexes>
      </configuration>
    </search>
  </sitecore>
</configuration>
```

This ensures that the Lucene search engine can index all the items and products in your webshop.

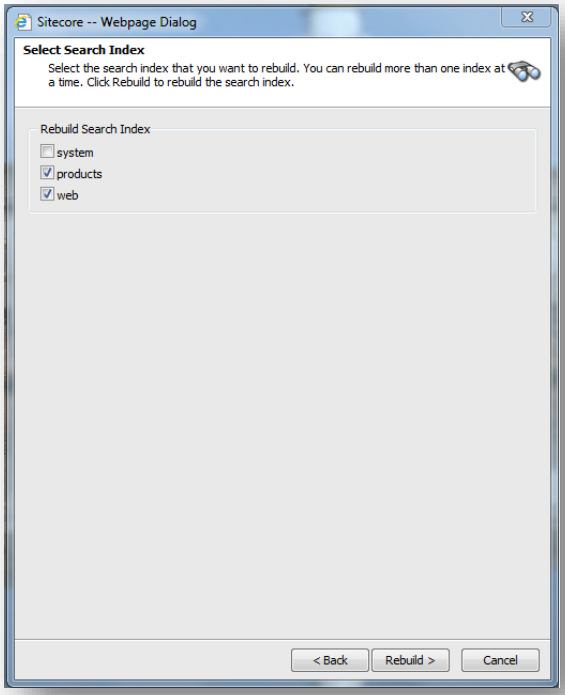
6.2.4 Rebuild the Search Indexes

After you have implemented changes to the appropriate configuration files, you must rebuild the search indexes.

To rebuild the search index:

1. Open the **Sitecore Desktop**.
2. Click **Sitecore**, **Control Panel**, **Database**, and then **Rebuild the Search Index**.

3. In the **Select Search Index** wizard, select *products* and *web* and then click **Rebuild**.



4. Click **Finish** when the wizard has completed.

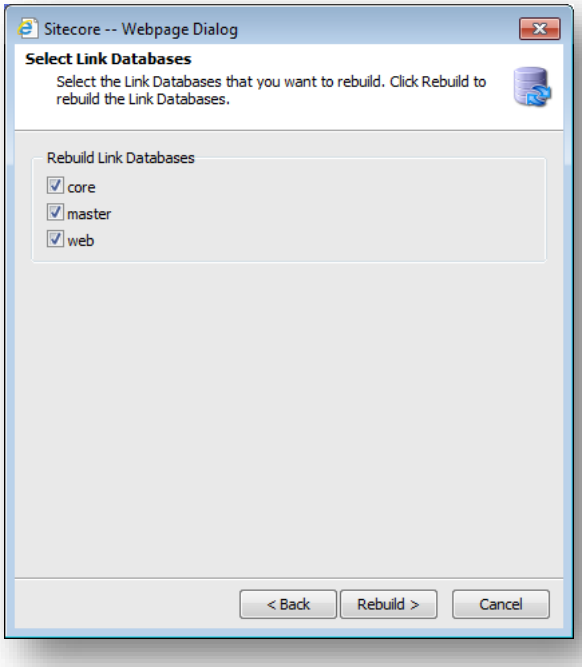
Note
The procedures described in this section only apply if you are using Lucene search.

6.2.5 Rebuild the Link Database

To rebuild the link database:

1. Open the **Sitecore Desktop**.
2. Click **Sitecore**, **Control Panel**, **Database**, and then click **Rebuild the Link Database**.

3. In the wizard, select all of the available link databases and click **Rebuild**.



Click **Finish** when the wizard has completed.

Chapter 7

Setting up a Distributed Environment

This chapter explains how to install Sitecore E-Commerce Services and Order Manager in a distributed environment. Set up a distributed environment to improve the scalability, security and performance of your Sitecore E-Commerce Services solution.

This chapter includes the following sections:

- Installing SES and OM in a Distributed Environment

7.1 Installing SES and OM in a Distributed Environment

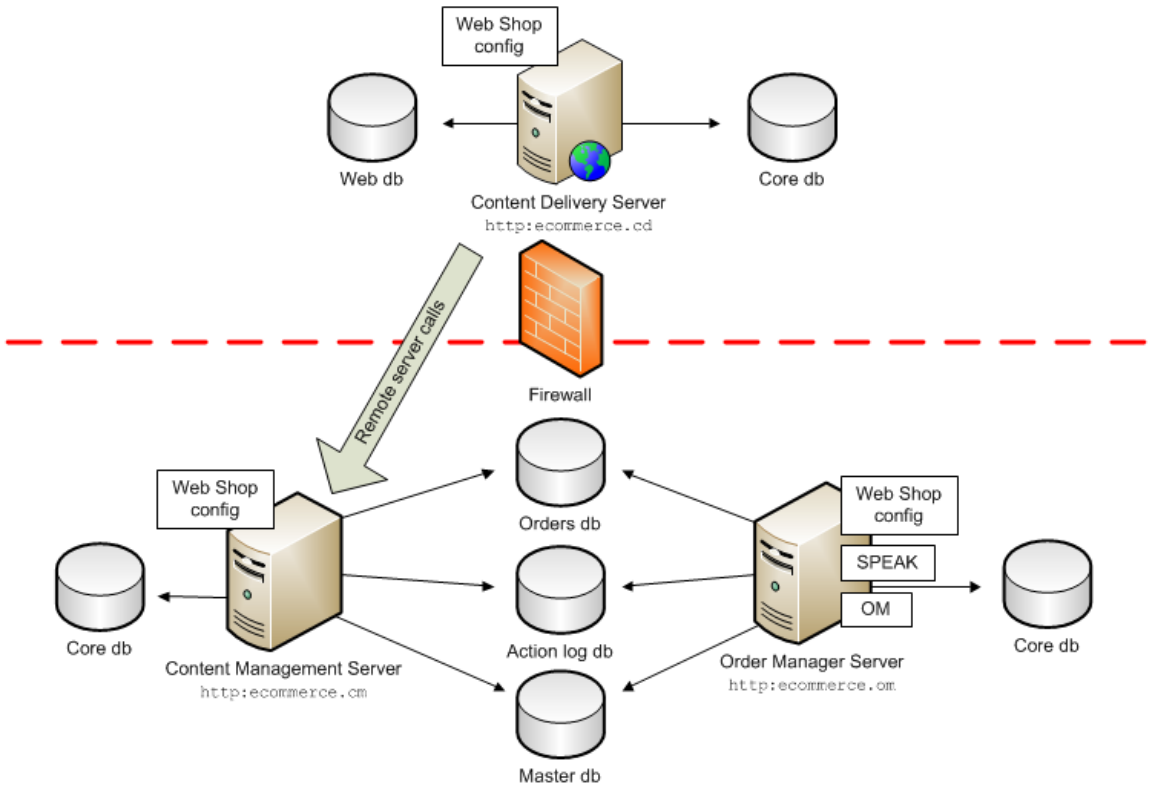
Install Sitecore E-Commerce Services and Order Manager in a distributed environment to improve scalability, security and performance. In a distributed environment the Content Management (CM) and Content Delivery (CD) servers are separated by a firewall.

This chapter provides an overview of the architecture and describes the steps you need to follow to install and configure Sitecore E-Commerce Services and Order Manager in a distributed environment.

For more detailed information on how to set up a distributed environment for Sitecore CMS (excluding Order Manager) see the *Sitecore CMS see Sitecore CMS 7.0 Scaling Guide*.

For information on how to set up a distributed environment for Sitecore CMS and E-Commerce Services see Chapter 8, Scaling E-Commerce Services.

Sitecore E-Commerce Services and Order Manager in a simple distributed environment:



Summary of components:

- Content Delivery Server (CD) – CMS, DMS, SES Core and SES Example pages.
- Content Management Server (CM) – CMS, DMS, SES Core and SES Example pages.
- Order Manager Server (OM) – CMS, DMS, SES Core, SES Example pages and Order Manager (OM).

Note
In some advanced scenarios, for example a distributed architecture in a cloud installation, the CM and OM server instances could also sit outside the firewall.

The following sections explain how you install Sitecore E-Commerce Services and Order Manager in a simple distributed environment.

7.1.1 Installing a CM Server Instance and a CD Server Instance

To install and configure content management and content delivery servers:

1. Install two identical instances of Sitecore CMS with the DMS, WFFM, SES Core and SES Example pages packages.

   For more information on how to do this, follow the instructions in the *Sitecore CMS 7.0 Scaling Guide* on the Sitecore Developer Network (SDN).

   You should then have the following two server instances installed:

   - Content Delivery Server (outside the firewall)
   - Content Management Server (inside the firewall)

2. Configure remote server calls between these two instances so that they can communicate with each other across the firewall. You enable remote server calls using the Windows Communication Foundation (WCF). To do this, use the SES service model approach outlined in Chapter 8 - Scaling E-Commerce Services. Follow the steps described in section 8.2 - Configuring Multiple Instances.

3. In the Content Delivery (CD) instance, navigate to the `unity.config` file. Use the following path: `<Website>/App_Config`

   According to the instructions in section *The Service Model,* make the changes to the `unity.config` file.

4. Also in the CD instance, rename the `SwitchMasterToWeb.config.example` file to `SwitchMasterToWeb.config`. This activates the configuration file and ensures that the content delivery server (CD instance) cannot access the Master database.

   Use the following path to locate the file:
   `<Website>/App_Config/include/SwitchMasterToWeb.config.example`

5. In the Content Management (CM) instance (after installing the SES core) attach the following SQL Server databases:

   - *SitecoreEcommerce_Orders.mdf* – stores all customer orders.
   - *SitecoreEcommerce_ActionLog.mdf* – stores a log of all actions related to the processing of customer orders.

   After you have installed the SES core, the orders and logging databases can be found in the `<SiteRootFolder>/Databases` folder.

6. Add connection strings for the orders and logging databases to the `ConnectionStrings.config` file. Use the following path to locate the `ConnectionStrings.config` file file:
   `<Website>/App_Config`

7. In the `Sitecore.Ecommece.config` file, configure the `orderCreated` pipeline.

   - For the CM server:

```
<orderCreated>
<processor type="Sitecore.Ecommerce.Merchant.Pipelines.OrderCreated.CheckProductQuantity,
Sitecore.Ecommerce.Merchant">
<MaximumQuantity>100</MaximumQuantity>
```

```
</processor>
<processor type="Sitecore.Ecommerce.Merchant.Pipelines.OrderCreated.TryOpenOrder,
Sitecore.Ecommerce.Merchant" />
</orderCreated>
```

- For the CD server:

```
<orderCreated>
<processor type="Sitecore.Ecommerce.Visitor.Pipelines.OrderCreated.NotifyCustomer,
Sitecore.Ecommerce.Visitor" />
</orderCreated>
```

7.1.2 Installing an Order Manager Server Instance

To install and configure an Order Manager (OM) Server:

1. Install a third Sitecore CMS instance for Order Manager. Ensure that this server also has the SES Core packages installed.
2. Install the SES Order Manager package. For instructions on how to install Order Manager, see the section Installing Order Manager.
3. In the Order Manager server (OM) instance, navigate to the `ConnectionStrings.config` file: `<Website>/App_Config`
4. Open the `ConnectionStrings.config` file and make the following changes:
   1) Remove or comment out the line that connects the *Web* database. This is necessary to prevent the Order Manager instance from connecting to the web database.
   2) Edit the `Master` connection string to point to the *Master* database of the CM instance. This ensures that Order Manager uses the Master database.
   3) Add the `Orders` connection string. Edit the “Orders” connection string to point to the *Orders* database of the CM instance. This ensures that Order Manager stores customer orders in the Orders database.

      ```
      <add name="orders" connectionString="user
      id=user_name;password=user_password;Data
      Source=<server name>;Database=db name;MultipleActiveResultSets=true"
      providerName ="System.Data.SqlClient"/>
      ```

   4) Add the `Logging` connection string. Edit the “Logging” connection string to point to the *Logging* database of the CM instance. This ensures that Order Manager uses the Logging database to store all order processing actions.

      ```
      <add name="logging" connectionString="user
      id=user_name;password=user_password;Data
      Source=<server_name>;Database=db_name;MultipleActiveResultSets=true"
      providerName ="System.Data.SqlClient"/>
      ```

   Note
   You can re-use the CM instance connection strings for *b*, *c* and *d*.

5. Edit the `Sitecore.Ecommerce.Apps.OrderManagement.config` file in case you need to set the default context name of your Order Manager web site.

   Important
   For more information on how to set the context name see, Order Manager Post Installation Steps, Setting the Order Manager Website Context Name.

6. Rename the `/App_Config/include/SwitchWebToMaster.config.example` file to `SwitchMasterToWeb.config`. Enable this config file to prevent the Order Manager instance from accessing the *Web* database.

   Note
   In step 3, you edited the Order Manager connection string to point to the CM instance Master database. If you wish, you can remove and delete the *Master* and *Web* databases from SQL Server. You can also remove the logging database as this is now also obsolete. However, check that your CM instance has the correct database configuration before making these changes.

7.1.3 Installing Multiple CD Instances

You may want to deploy several CD instances.

To install several CD instances, perform the following steps:

1. Extend the `/App_Config/ConnectionStrings.config` file of the CM instance with an additional `web2` database:

```
<add name="web" connectionString="… " />
<add name="web2" connectionString="…" />
```

2. Extend the `/web.config` file of the CM instance with the additional `web2` database in the `//configuration/databases/database` section that is fully copied from the `<database name="web">` configuration.
3. Register an additional `web2` publishing target in the CM instance.
4. Point all CD instances to the same `core` database of the first CD instance.
   For the second CD instance the configuration should look like:

```
<add name="analytics" connectionString="cd.2__Analytics" />
 <add name="core" connectionString="cd.1 __Core" />
 <add name="web" connectionString="cd.2 __Web" />
```

   If you added the third CD instance, its configuration should look like:

```
<add name="analytics" connectionString="cd.3__Analytics" />
 <add name="core" connectionString="cd.1 __Core" />
 <add name="web" connectionString="cd.3 __Web" />
```

Chapter 8

Scaling E-Commerce Services

You can set up multiple instances of Sitecore E-Commerce Services and Order Manager to improve the scalability, performance and security of your solution.

The information contained in this chapter was previously contained in the SES Scaling Guide.

This chapter contains the following sections:

- Installing Multiple Instances
- Configuring Multiple Instances

8.1 Installing Multiple Instances

You may need to run SES on multiple Sitecore instances for one or more of the following reasons:

- Scalability — you might need to have multiple content delivery (CD) servers for handling web-requests and multiple content management (CM) servers for managing the content as well as the products and orders.
- Separation — you might need to use a firewall to separate the CD servers from the CM servers. You can place the content management servers behind the firewall where they won't steal resources from the content delivery servers that are managing the web-requests.
- Security — managing orders, products, and prices on CM servers that are behind the firewall makes this information inaccessible and protects your data.

8.1.1 Scaling Sitecore E-Commerce Services

You can scale SES in two different ways:

- You can use the native Sitecore CMS 6.6 scalability features that allow you to run multiple Sitecore instances. This approach will primarily aim at running multiple CM instances behind a firewall and will allow you to run multiple CD servers on the other side of the firewall.
- You can use SES's service model to add multiple CD instances that manage all the web-requests as well as communicating with the backend CM servers that store the orders placed by customers and handle all the requests for price and stock information.

You can use either of these approaches on their own or combine them to facilitate more advanced setups.

8.1.2 Scaling SES with Sitecore CMS 7.0 or Later

Sitecore CMS 6.6 provides native support for configuring multiple related instances in order to achieve better load balancing on the CM servers. By using CMS 7.0 or later to scale SES, you can increase performance by having the content distributed across several CM servers.

For more information about using CMS 6.6 to configure SEFE, see the *Sitecore CMS 7.0 Scaling Guide*.

http://sdn.sitecore.net/Reference/Sitecore%207/Scaling%20Guide.aspx

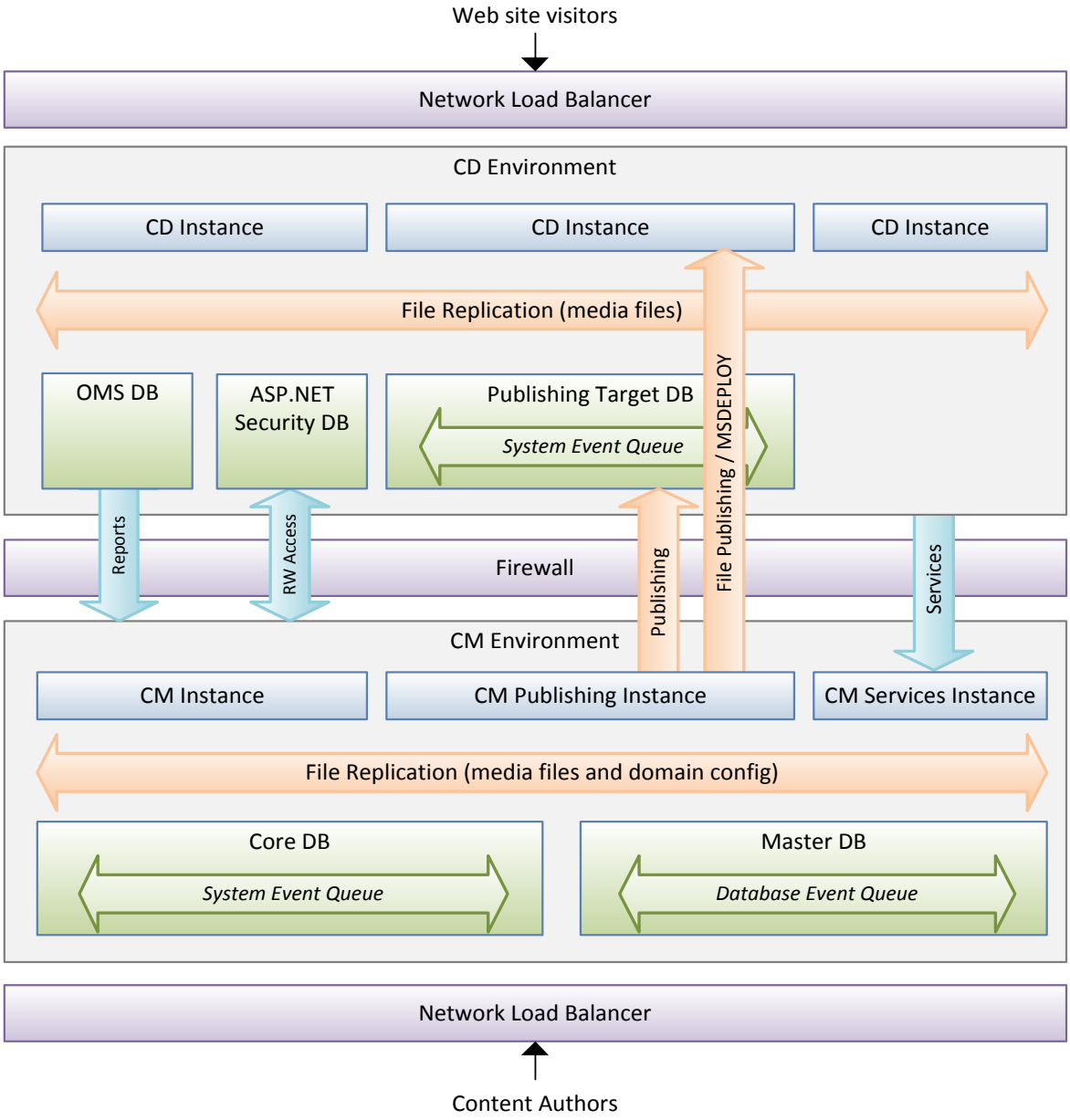
8.1.3 Scaling SES with the Service Model

SES contains an additional mechanism that you can use to scale your solution — the Service Model.

The Service Model allows you to avoid storing sensitive information such as orders, prices, and stocks on the CD servers. You can store this information on the CM servers that are protected by a firewall.

The Service Model consists of a number of service interfaces that are designed to link separate CM and CD servers. A service layer has been implemented using WCF technology that provides powerful configuration options in terms of communication protocols and security.

The following diagram shows the CD and CM instances and how the SES Service Model extends the CMS architecture.



The lower section illustrates the CM environment. The rightmost CM server is configured to host the services that the CD servers use to communicate across the firewall. The upper section illustrates the CD environment. The CD instances use the service layer to access the CM environment and manage orders, product stock, and price information.

The Separation of Content Management and Content Delivery Servers

The CD servers use the service layer to access the CM servers that contain the:

- Orders.

- Product price information.
- Stock information.

The service layer is divided into three separate services that corresponding with the previous list. You can use one or all of the services in this split CD and CM environment.

If you are running on a single server environment or CMS 7.0 scaled environment, none of the services are used. For more information, see also *Configuring Multiple Instances* section.

Orders

By default all customer orders are stored in Order Manager. It is easy to integrate Order Manager with a third party external system.

However, if you use a 3rd party system, you must extend the part of the domain model that handles the orders.

Note
It is still possible to store customer orders in the Sitecore E-Commerce Services content tree but this functionality will be deprecated in future versions of Sitecore E-Commerce Services.

Important
The CM servers should manage integration with the external systems. The CD servers should never do this. There is a single point of integration.

You must configure the *Order* service if you want to run separate CD and CM environments.

Product Price Information

The type of webshop you are running often determines where you should store the product price data.

B2C Shop

A business-to-consumer shop typically has fairly static prices. In which case, it might make sense to store the prices on the CD servers as well as on the CM servers. You do this by storing the product price data in the *Master* database and publishing the price in the same way as you publish ordinary CMS data. The *Example Pages* package demonstrates how you can store price data in the product template and publish it from there to the *Web* database.

B2B Shop

A business-to-business webshop typically has some more advanced price structures that can vary depending on a number of rules and the customer relationships that you support. The prices can be more variable in nature. It therefore makes sense not to store price data on the CD servers, but to request the price from the CM server through the service layer. In this case, the product price data should not be stored along with the product information as is done in the *Example Pages* package.

If the solution is integrated with a 3rd party system, the prices are typically stored there and the `ProductPriceManager` should be replaced with a customized version that integrates with the external system.

Important
The CM servers should manage integration with the external systems. The CD servers should never do this. There is a single point of integration.

Stock Information

Stock information fluctuates a lot and therefore it should never be stored on the CD servers. Stock information needs to be synchronized and the correct way to do that is on the CM server through the Service Layer.

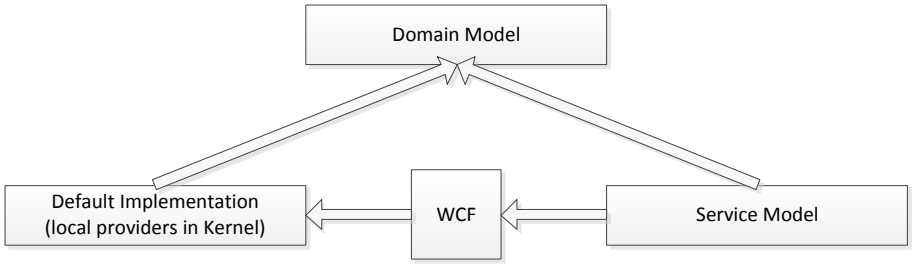
If the solution is integrated with a 3rd party system, stock information is typically stored there and the `ProductStockManager` should be replaced with a customized version that integrates with the external system.

Important
The CM servers should manage integration with the external systems. The CD servers should never do this. There is a single point of integration.

8.1.4 The Service Model

The SES domain model represents the E-Commerce abstraction layer and describes the API contracts.

You use Windows Communication Foundation (WCF) to implement the service model that is represented by a set of manager classes which implement main Domain Model contracts.



The default implementation of the domain model is stored in the `Sitecore.Ecommerce.Kernel` and the `Sitecore.Ecommerce.Visitor` assemblies and works with the local CMS instance and accesses the items that are local for that instance:

- OrderManager
- ProductPriceManager
- ProductStockManager
- VisitorOrderManager
- VisitorOrderProcessor
- ItemBasedOrderIDGenerator

To set up a separate CD environment and use the service model, a set of additional remote managers are added. These remote managers are stored in the `Sitecore.Ecommerce.ServiceModel` assembly:

- RemoteOrderManager
- RemoteProductPriceManager
- RemoteProductStockManager
- RemoteVisitorOrderManager
- RemoteOrderProcessor

- RemoteOrderIDGenerator

Both the *local* and *remote* managers are set up in the `Unity.Config` file. By default, the *local* managers are configured, which means that they work with items from the local CMS instance.

To configure the remote managers, you must map a manager registration to the appropriate alias.

In the `Unity.Config` file, locate the `<register>` elements section and change the `mapTo` attribute to the appropriate *remote* alias:

```
<register type="OrderIDGenerator" mapTo="RemoteOrderIDGenerator">
  <lifetime type="hierarchical" />
</register>
<register type="IProductStockManager" mapTo="RemoteProductStockManager">
  <lifetime type="hierarchical" />
</register>
<register type="IProductPriceManager" mapTo="RemoteProductPriceManager">
  <lifetime type="hierarchical" />
</register>
<register type="VisitorOrderProcessorBase" mapTo="RemoteOrderProcessor">
  <lifetime type="hierarchical" />
  <interceptor type="VirtualMethodInterceptor" />
  <policyInjection />
</register>
<register type="VisitorOrderManager" mapTo="RemoteVisitorOrderManager">
  <lifetime type="hierarchical" />
  <interceptor type="VirtualMethodInterceptor" />
  <policyInjection />
</register>
```

You must not change the registration of the `IOrderManager`, because the default `TransientOrderManager` is designed for backwards compatibility with the previous item-based approach of storing orders as items in the Sitecore Content Editor.

8.2 Configuring Multiple Instances

This section explains how to use the Service Model approach to configure multiple instances of Sitecore E-Commerce Services.

8.2.1 Content Management Server Configuration

You must configure a content management (CM) server instance to host the SES Services. The CM server instance can share the *Master* and *Core* databases with other CM server instances in a CM environment.

You should configure the publishing targets of the CM server to point to the content delivery (CD) instances. For more information about configuring publishing targets, see the *Sitecore 7.0 Scaling Guide*.

Also in the `web.config` file of the CM instance, you must add the following two sections.

Add to the `<configuration>` section:

```
<system.runtime.serialization>
    <dataContractSerializer >
      <declaredTypes>
        <add type="Sitecore.Ecommerce.DomainModel.Products.ProductStock,
          Sitecore.Ecommerce.DomainModel">
          <knownType type="Sitecore.Ecommerce.Products.ProductStock,
            Sitecore.Ecommerce.Kernel"/>
        </add>
      </declaredTypes>
    </dataContractSerializer>
</system.runtime.serialization>
```

Also in `<configuration>` add this to the `<runtime>` section:

```
        <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
          <dependentAssembly>
            <codeBase version="4.5.0.0" href="sitecore
             modules/Shell/Ecommerce/Services/servicebin/Newtonsoft.Json.dll" />
            <assemblyIdentity name="Newtonsoft.Json" publicKeyToken="30ad4fe6b2a6aeed"
             version="4.5.0.0" culture="neutral" />
            <bindingRedirect oldVersion="3.5.0.0" newVersion="4.5.0.0" />
          </dependentAssembly>
        </assemblyBinding>
```

8.2.2 Content Delivery Server Configuration

You must configure the Windows Communication Foundation (WCF) Service Model for every CD instance. You must configure each CD instance in the `web.config` file.

The CD configuration consists of a set of bindings, one for each service and a corresponding set of endpoints that map the bindings to a server instance.

Add bindings to the `<system.serviceModel>` section of the `web.config` file. This is the default child section for all ASP.NET applications. Also there must be registrations of known types that were shown in the previous section. In the following example, there are three bindings: `OrderService`, `ProductPriceService` and `ProductStockService`:

```
        <system.serviceModel>
          <bindings>
            <basicHttpBinding>
              <binding name="BasicHttpBinding_IOrderService" closeTimeout="00:01:00"
                  openTimeout="00:01:00" receiveTimeout="00:10:00" sendTimeout="00:01:00"
                  allowCookies="false" bypassProxyOnLocal="false"
```

```
            hostNameComparisonMode="StrongWildcard"
            maxBufferSize="65536" maxBufferPoolSize="524288"
            maxReceivedMessageSize="65536"
            messageEncoding="Text" textEncoding="utf-8" transferMode="Buffered"
            useDefaultWebProxy="true">
          <readerQuotas maxDepth="32" maxStringContentLength="8192"
            maxArrayLength="16384"
              maxBytesPerRead="4096" maxNameTableCharCount="16384" />
          <security mode="None">
            <transport clientCredentialType="None" proxyCredentialType="None"
                realm="" />
            <message clientCredentialType="UserName" algorithmSuite="Default" />
          </security>
        </binding>
        <binding name="BasicHttpBinding IOrderIDGeneratorService"
            closeTimeout="00:01:00"
            openTimeout="00:01:00" receiveTimeout="00:10:00" sendTimeout="00:01:00"
            allowCookies="false" bypassProxyOnLocal="false"
            hostNameComparisonMode="StrongWildcard"
            maxBufferSize="65536" maxBufferPoolSize="524288"
            maxReceivedMessageSize="65536"
            messageEncoding="Text" textEncoding="utf-8" transferMode="Buffered"
            useDefaultWebProxy="true">
          <readerQuotas maxDepth="32" maxStringContentLength="8192"
            maxArrayLength="16384"
              maxBytesPerRead="4096" maxNameTableCharCount="16384" />
          <security mode="None">
            <transport clientCredentialType="None" proxyCredentialType="None"
                realm="" />
            <message clientCredentialType="UserName" algorithmSuite="Default" />
          </security>
        </binding>
        <binding name="BasicHttpBinding_IProductPriceService" closeTimeout="00:01:00"
            openTimeout="00:01:00" receiveTimeout="00:10:00" sendTimeout="00:01:00"
            allowCookies="false" bypassProxyOnLocal="false"
            hostNameComparisonMode="StrongWildcard"
            maxBufferSize="65536" maxBufferPoolSize="524288"
            maxReceivedMessageSize="65536"
            messageEncoding="Text" textEncoding="utf-8" transferMode="Buffered"
            useDefaultWebProxy="true">
          <readerQuotas maxDepth="32" maxStringContentLength="8192"
            maxArrayLength="16384"
              maxBytesPerRead="4096" maxNameTableCharCount="16384" />
          <security mode="None">
            <transport clientCredentialType="None" proxyCredentialType="None"
                realm="" />
            <message clientCredentialType="UserName" algorithmSuite="Default" />
          </security>
        </binding>
        <binding name="BasicHttpBinding_IProductStockService" closeTimeout="00:01:00"
            openTimeout="00:01:00" receiveTimeout="00:10:00" sendTimeout="00:01:00"
            allowCookies="false" bypassProxyOnLocal="false"
            hostNameComparisonMode="StrongWildcard"
            maxBufferSize="65536" maxBufferPoolSize="524288"
            maxReceivedMessageSize="65536"
            messageEncoding="Text" textEncoding="utf-8" transferMode="Buffered"
            useDefaultWebProxy="true">
          <readerQuotas maxDepth="32" maxStringContentLength="8192"
           maxArrayLength="16384"
              maxBytesPerRead="4096" maxNameTableCharCount="16384" />
          <security mode="None">
            <transport clientCredentialType="None" proxyCredentialType="None"
                realm="" />
            <message clientCredentialType="UserName" algorithmSuite="Default" />
          </security>
        </binding>
        <binding name="BasicHttpBinding_IOrderProcessorService" closeTimeout="00:01:00"
            openTimeout="00:01:00" receiveTimeout="00:10:00" sendTimeout="00:01:00"
            allowCookies="false" bypassProxyOnLocal="false"
```

```
            hostNameComparisonMode="StrongWildcard"
            maxBufferSize="65536" maxBufferPoolSize="524288"
            maxReceivedMessageSize="65536"
            messageEncoding="Text" textEncoding="utf-8" transferMode="Buffered"
            useDefaultWebProxy="true">
          <readerQuotas maxDepth="32" maxStringContentLength="8192"
            maxArrayLength="16384"
              maxBytesPerRead="4096" maxNameTableCharCount="16384" />
          <security mode="None">
            <transport clientCredentialType="None" proxyCredentialType="None"
                realm="" />
            <message clientCredentialType="UserName" algorithmSuite="Default" />
          </security>
        </binding>
        <binding name="BasicHttpBinding IOrderRepositoryService" closeTimeout="00:10:00"
            openTimeout="00:10:00" receiveTimeout="01:40:00" sendTimeout="00:10:00"
            allowCookies="false" bypassProxyOnLocal="false"
            hostNameComparisonMode="StrongWildcard"
            maxBufferSize="524288" maxBufferPoolSize="524288"
            maxReceivedMessageSize="524288"
            messageEncoding="Text" textEncoding="utf-8" transferMode="Buffered"
            useDefaultWebProxy="true">
          <readerQuotas maxDepth="32" maxStringContentLength="524288"
            maxArrayLength="524288"
              maxBytesPerRead="4096" maxNameTableCharCount="16384" />
          <security mode="None">
            <transport clientCredentialType="None" proxyCredentialType="None"
                realm="" />
            <message clientCredentialType="UserName" algorithmSuite="Default" />
          </security>
        </binding>
      </basicHttpBinding>
    </bindings>
    <client>
      <endpoint address="http://localhost/sitecore
        modules/shell/ecommerce/services/OrderService.svc"
          binding="basicHttpBinding"
          bindingConfiguration="BasicHttpBinding_IOrderService"
          contract="OrderService.IOrderService"
          name="BasicHttpBinding IOrderService" />
      <endpoint address="http://localhost/sitecore
        modules/shell/ecommerce/services/OrderIDGeneratorService.svc"
          binding="basicHttpBinding"
          bindingConfiguration="BasicHttpBinding IOrderIDGeneratorService"
          contract="OrderIDGeneratorService.IOrderIDGeneratorService"
          name="BasicHttpBinding_IOrderIDGeneratorService" />
      <endpoint address="http://localhost/sitecore
        modules/shell/ecommerce/services/ProductPriceService.svc"
          binding="basicHttpBinding"
          bindingConfiguration="BasicHttpBinding_IProductPriceService"
          contract="ProductPriceService.IProductPriceService"
          name="BasicHttpBinding_IProductPriceService" />
      <endpoint address="http://localhost/sitecore
        modules/shell/ecommerce/services/ProductStockService.svc"
          binding="basicHttpBinding"
          bindingConfiguration="BasicHttpBinding_IProductStockService"
          contract="ProductStockService.IProductStockService" `
          name="BasicHttpBinding_IProductStockService" />
      <endpoint address="http://localhost/sitecore
        modules/shell/ecommerce/services/OrderProcessorService.svc"
          binding="basicHttpBinding"
          bindingConfiguration="BasicHttpBinding_IOrderProcessorService"
          contract="OrderProcessorService.IOrderProcessorService"
          name="BasicHttpBinding_IOrderProcessorService" />
      <endpoint address="http://localhost/sitecore
        modules/shell/ecommerce/services/OrderRepositoryService.svc"
          binding="basicHttpBinding"
          bindingConfiguration="BasicHttpBinding_IOrderRepositoryService"
          contract="OrderRepositoryService.IOrderRepositoryService"
```

```
            name="BasicHttpBinding_IOrderRepositoryService" />
      </client>
    </system.serviceModel>
    <system.runtime.serialization>
      <dataContractSerializer >
        <declaredTypes>
          <add type="Sitecore.Ecommerce.DomainModel.Products.ProductStock,
    Sitecore.Ecommerce.DomainModel">
            <knownType type="Sitecore.Ecommerce.Products.ProductStock,
    Sitecore.Ecommerce.Kernel"/>
          </add>
        </declaredTypes>
      </dataContractSerializer>
    </system.runtime.serialization>
```

You can use the bindings as shown in this example but you must configure the endpoints to use the right URI (Uniform Resource Identifier). Endpoints give the clients access to the functionality of a WCF service that is hosted on a server instance. Each endpoint has an address attribute that is based on the URI protocol which has the following format:

`<scheme://hostname[:port]/path>`

Important
The http scheme is configured and each endpoint is given the `localhost` hostname by default. You must replace this with the address of the CM server that hosts the service model.

If the network configuration allows, you can change protocols used for binding the services to improve performance and security. You must configure this on both the CD and CM server instances. This is a feature of the WCF framework.

For more information about configuring the Service Model, see http://msdn.microsoft.com/en-us/library/ms731734.aspx.

For more information about configuring endpoints, see http://msdn.microsoft.com/en-us/library/ms733107.aspx.

Configuring the Lucene Search Index

SES uses Lucene for product indexing and searching by default. The search indexes configured in the `App_Config/Include/Sitecore.Ecommerce.config` file contain the configuration for both the *Master* and *Web* databases by default. In the CD environment, the servers are typically not associated with the *Master* database. Therefore, you should not configure the product index for the *Master* database on the CD server instances.

You must remove the highlighted section from the `Sitecore.Ecommerce.config` file:

`search/configuration/indexes/index[id="products"]/locations/master`:

```
<search>
  <configuration>
    <indexes>
      <index id="products" type="Sitecore.Search.Index, Sitecore.Kernel">
        <param desc="name">$(id)</param>
        <param desc="folder">__products</param>
        <Analyzer type="Sitecore.Ecommerce.Search.LuceneAnalyzer, Sitecore.Ecommerce.Kernel" />
        <locations hint="list:AddCrawler">
          <master type="Sitecore.Ecommerce.Search.DatabaseCrawler, Sitecore.Ecommerce.Kernel">
            <Database>master</Database>
            <Root>{0A702337-81CD-45B9-8A72-EC15D2BE1635}</Root>
            <Tags>master products</Tags>
          </master>
          <web type="Sitecore.Ecommerce.Search.DatabaseCrawler, Sitecore.Ecommerce.Kernel">
            <Database>web</Database>
            <Root>{0A702337-81CD-45B9-8A72-EC15D2BE1635}</Root>
            <Tags>web products</Tags>
          </web>
        </locations>
      </index>
    </indexes>
  </configuration>
</search>
```

SES Services Configuration Requirements

The service model does not require a specific CMS version. You can therefore use any CMS version that SES supports.

8.2.3 WCF Configuration Notes

For more information about configuring WCF see:

http://msdn.microsoft.com/en-us/library/ms735093.aspx

http://msdn.microsoft.com/en-us/library/ms731884.aspx

http://msdn.microsoft.com/en-us/library/ms731316.aspx

http://www.devx.com/codemag/Article/33342/1763/page/1

Chapter 9

Deploying SES on Azure

This chapter describes how to deploy SES 2.2 with CMS 7.0 on Azure. The deployment consists of Content Editing (CE) environment and the Content Delivery environment. This chapter also describes how to deploy a Content Editing environment from a previously deployed one.

It contains the following sections:

- Requirements
- Deploying the Content Editing Environment
- Deploying the Content Delivery Environment

9.1 Requirements

You should install the following on the machine that deploys the Sitecore installation to the cloud:

1. Install Sitecore CMS 7.0 rev.140120 or later and DMS 7.0 rev.140120 or later. You should configure Sitecore as a single server and the configuration changes when deploying to Azure. You should also configure all the installations of Sitecore modules according to their installation instructions.
2. Install the Web Forms for Marketers module v2.3.0 rev.131126 or later for Sitecore CMS 6.5, 6.6 and 7.0 with the following SQL server configuration:
   - Add the connection string to the `connectionstring.config` file and name it WFM.
   - Remove the settings in the `forms.config` file. It must so it looks like the following:

```
<formsDataProvider type="Sitecore.Forms.Data.DataProviders.WFMDataProvider,
Sitecore.Forms.Core">
</formsDataProvider>
```

3. Install Azure SDK 2.0 and configure this to be used in Visual Studio 2012-2013.
4. Install Sitecore Azure 3.1 rev.130731.
5. Install Sitecore E-Commerce Services 2.2
6. Install Sitecore E-Commerce Order Manager 2.2
7. Install Sitecore E-Commerce Services 2.2 Example Pages package
8. Configure Sitecore to use MVC.
9. In Visual Studio 2012 or 2013, create a web project for the installation. For information on how to do it, see the installation guide for CMS 7.0.
10. Add MVC 4 to the website. For information on how to do it, see http://www.nuget.org/packages/Microsoft.AspNet.Mvc/4.0.30506
11. Configure MVC to redirect the old assemblies to the installed version of MVC. For information on how to do it, see http://www.asp.net/whitepapers/mvc4-release-notes#_Toc303253806
12. Configure Sitecore mail settings to use a mail server that is also available in Azure. For development purpose, mailgun can be used to send 10K free mails per month www.mailgun.org
13. In the `web.config` file of the deployment machines, add the following

```
<system.runtime.serialization>
    <dataContractSerializer >
      <declaredTypes>
        <add type="Sitecore.Ecommerce.DomainModel.Products.ProductStock,
         Sitecore.Ecommerce.DomainModel">
          <knownType type="Sitecore.Ecommerce.Products.ProductStock,
           Sitecore.Ecommerce.Kernel"/>
        </add>
      </declaredTypes>
    </dataContractSerializer>
</system.runtime.serialization>
```

14. If Sitecore is not already configured to use NewtonSoft Jason versions 4.5, install it and configure it.
15. Check that Sitecore runs from the deployment machine and SES is working in all areas as expected.

16. To deploy your solution on Azure, you should request the deployment file as described in the Azure module documentation.
17. In the `App_Config/Include/Sitecore.Ecommerce.config` file, set the value of the `Catalog.OpenInNewWindow` setting to `false`. This opens the items that represent the entries of the product and order catalogs in a new Content Editor tab in the same window. The default value is `true` to make catalogs use the Field Editor in a new window. This functionality is not supported in the Azure environment.

9.2 Deploying the Content Editing Environment

This section describes how to deploy the following on Azure:

- The CE environment for the first time
- The `Order` and `Logging` databases for the first time
- The `Order` and `Logging` database connections from the first deployment

9.2.1 First deployment

To deploy the CE for the first time:

1. Create a folder in the website root `App_Data` and name it `CEAzureOverrideFiles`.
2. Copy the content of the `App_Data/AzureOverrideFiles` folder to the `App_Data/CEAzureOverrideFiles` folder.
3. In the `\App_Data\CEAzureOverrideFiles\App_Config\Include` folder, create an empty file and call it `SwitchToCE.config`.
4. Add the following to the new empty file:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration xmlns:patch="http://www.sitecore.net/xmlconfig/"
xmlns:set="http://www.sitecore.net/xmlconfig/set/">
  <sitecore>
    <pipelines>
      <orderCreated>
        <processor
          type="Sitecore.Ecommerce.Visitor.Pipelines.OrderCreated.NotifyCustomer,
          Sitecore.Ecommerce.Visitor">
          <patch:delete />
        </processor>
      </orderCreated>
    </pipelines>
  </sitecore>
</configuration>
```

5. Open the Sitecore desktop.
6. Open the Sitecore Azure application. If Local Emulator is the only environment type that you have, press the Add icon and follow the instruction to add an environment file. By following the instructions, you configure the environment type that is named Development [Staging] .This name can vary according to the environment file that you have requested from Sitecore.
7. In the dropdown menu, choose the environment that is named Development [Staging].
8. To select the location to which you want to deploy the instance, click one of the grey nodes. In this example the chosen grey node is *North Europe*.
9. In the pop-out menu, click *Add Editing Farm*.
10. When the dialog box appears, add the publishing settings file, and then follow the instructions. For more information, see the Azure documentation.
11. When the New Deployment dialog box appears, do not click **Start deployment**. Instead, in the left lower corner, click **More Options**.

12. Sitecore content editor must appear and the content tree is expanded to the new deployment, as follows:

    `/sitecore/system/Modules/Azure/[YOUR DEPLOYMENT NAME]/North Europe/Editing01/Role01/Staging`

13. On the staging item, change the field Override Sources Folder from `App_Data/AzureOverrideFiles` to `App_Data/CEAzureOverrideFiles`
14. Expand the Staging node.
15. Expand the Database references node.
16. Right click one of the database reference nodes, choose Duplicate, and then name the new duplicated node *wfm*.
17. If there is no analytics node, right click one of the database reference nodes and select Duplicate – call the new duplicated node *analytics*.
18. Switch to the Sitecore Azure dialog and click **Start deployment**.
19. Wait until deployment has completed and North Europe is now green on the world map, and then choose North Europe again
20. In the menu, choose **Editing01**, click **Browse** and verify that the instance is running.

9.2.2 Deploying the Order and Logging databases from SES to Azure

Before deploying `Order` and `logging` databases to the cloud it is needed to get the server name, login and password for the SQL Azure server that you created in the First Deployment section.

To get this server information:

1. Open Sitecore Content Editor and expand the content tree to the following node: `/sitecore/system/Modules/Azure/[YOUR DEPLOYMENT NAME]/North Europe/Editing01/Sql01/Set01/core`
2. In the field **Target database**, copy the connection string, for example, to a Notepad file. The fields that you need are Data source, User ID and Password.

To deploy the `Order` and `Logging` databases to the cloud:

3. Open Visual Studio 2012 or 2013.
4. Create an SQL Server Database project.
5. Rename the default project to `Orders` and choose yes to rename target to `Orders` as well.
6. Right click the `Orders` project in the **Solution Explorer** and click **Properties**.
7. On the **Project Settings** tab, click Windows Azure SQL Database as Target platform and save the project.
8. In the **Solution Explorer**, right click the `Orders` Project and from the menu, choose Import and click **Database**.
9. Follow the wizard instructions and import the local `Orders` database that is installed with SES 2.2. Leave other properties as default unless you need other things.
10. After importing the database, right click the `Orders` Project in the **Solution Explorer** and click **Publish**.
11. Follow the instructions in the dialog to set the target database to the database in Azure. You must use the Data Source, User ID and password from previous steps.

12. Follow the same steps to deploy the `Logging` database.

9.2.3 Second Deployment

This section describes how to deploy the `Order` and `Logging` database connections from the first deployment.

1. Open Sitecore Content Editor and expand the content tree to the following node: `/sitecore/system/Modules/Azure/[YOUR DEPLOYMENT NAME]/North Europe/Editing01/Role01/Staging`
2. In the Connection Strings Patch, copy the existing data, for example, to a Notepad file.
3. Add the following two connection strings and change the fields in the square brackets to match the information from the previous step.

```
<add name="orders" connectionString="Data Source=tcp:[YOUR AZURE DATABASE SERVER];
Initial Catalog=[YOUR AZURE ORDERS DATABASE];
Integrated Security=False;
User ID=[YOUR AZURE DATABASE USER ID];
Password=[YOUR AZURE DATABASE USER PASSWORD];
Encrypt=True;
MultipleActiveResultSets=true" providerName="System.Data.SqlClient" />
<add name="logging" connectionString="Data Source=tcp:[YOUR AZURE DATABASE SERVER];
Initial Catalog=[YOUR AZURE ACTIONLOG DATABASE];
Integrated Security=False;
User ID=[YOUR AZURE DATABASE USER ID];
Password=[YOUR AZURE DATABASE USER PASSWORD];
Encrypt=True" providerName="System.Data.SqlClient" />
```

4. Open the Sitecore Azure dialog box.
5. Click the green North Europe node.
6. Choose Editing01 and click **Upgrade Files**.
7. Wait until deployment is completed.
8. Click the green North Europe node.
9. Choose Editing01 and click **browse**.
10. Test the application:
    - Frontend should accept orders.
    - Speak OM UI should be working.
    - Sitecore desktop should be working.

Now, you have your CE instance containing Sitecore SES 2.2 and OM.

9.3 Deploying the Content Delivery Environment

This section describes the how to deploy the CD environment on Azure.

9.3.1 First Deployment

To deploy the CD for the first time:

1. Create a folder in the website root and name it `App_Data/CDAzureOverrideFiles`.
2. Copy content from `App_Data/AzureOverrideFiles` to `App_Data/CDAzureOverrideFiles`
3. Copy the `Unity.config` file form the `App_Config` folder to the `App_Data/CDAzureOverrideFiles/App_Config` folder.
4. Copy `App_Config/Include/ SwitchMasterToWeb.config.example` to the `App_Data/CDAzureOverrideFiles/App_Config/Include/` folder.
5. In the `App_Data/CDAzureOverrideFiles/App_Config/Include/ SwitchMasterToWeb.config.example` file, add the following to the `configuration/sitecore` element:

```
<pipelines>
    <orderCreated>
        <processor
        type="Sitecore.Ecommerce.Merchant.Pipelines.OrderCreated.CheckProductQuantity,
         Sitecore.Ecommerce.Merchant">
          <patch:delete />
        </processor>
        <processor
        type="Sitecore.Ecommerce.Merchant.Pipelines.OrderCreated.TryOpenOrder,
         Sitecore.Ecommerce.Merchant">
          <patch:delete />
        </processor>
    </orderCreated>
</pipelines>
```

6. In the the `App_Data/CDAzureOverrideFiles/App_Config/Include/` folder, rename the `SwitchMasterToWeb.config.example` file to `SwitchMasterToWeb.config`.
7. Open the `App_Data/CDAzureOverrideFiles/App_Config/Unity.config` file and change the registrations of the following types to use Remote as described in *The Service Model* section:
   - `OrderIDGenerator`
   - `IProductStockManager`
   - `IProductPriceManager`
   - `VisitorOrderManager`
   - `VisitorOrderProcessorBase`
8. Open the Sitecore desktop.
9. Open the Sitecore Azure dialog box.
10. In the dropdown menu, click the environment that is named Development [Staging].

11. To select the location to which you want to deploy the instance, click one of the grey nodes. In this example the chosen grey node is *West Europe*.
12. In the pop-out menu, click *Add Delivery Farm*.
13. When the New Deployment dialog box appears, do not click **Start deployment**. Instead, in the left lower corner, click **More Options**.
14. Sitecore content editor must appear and the content tree is expanded to the new deployment, as follows:

    `/sitecore/system/Modules/Azure/[YOUR DEPLOYMENT NAME]/West Europe/Delivery01/Role01/Staging`

15. On the staging item:
    - Change the field Override Sources Folder from `App_Data/AzureOverrideFiles` to `App_Data/CDAzureOverrideFiles`.
    - In the Custom Web Config Patch field, add the following custom web config patch and change the URL fields to points to the CE that you already installed.

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:msxsl="urn:schemas-microsoft-com:xslt" exclude-result-prefixes="msxsl">
  <xsl:output method="xml" indent="yes" />
  <xsl:template match="@* | node()">
    <xsl:copy>
      <xsl:apply-templates select="@* | node()" />
    </xsl:copy>
  </xsl:template>
  <xsl:template match="configuration/system.serviceModel/bindings/basicHttpBinding">
    <xsl:copy>
      <xsl:apply-templates select="node()|@*"/>
      <binding name="BasicHttpBinding_IOrderService" closeTimeout="00:01:00"
        openTimeout="00:01:00" receiveTimeout="00:10:00" sendTimeout="00:01:00"
        allowCookies="false" bypassProxyOnLocal="false"
        hostNameComparisonMode="StrongWildcard"
        maxBufferSize="65536" maxBufferPoolSize="524288"
        maxReceivedMessageSize="65536"
        messageEncoding="Text" textEncoding="utf-8" transferMode="Buffered"
        useDefaultWebProxy="true">
        <readerQuotas maxDepth="32" maxStringContentLength="8192"
          maxArrayLength="16384"
            maxBytesPerRead="4096" maxNameTableCharCount="16384" />
        <security mode="None">
          <transport clientCredentialType="None" proxyCredentialType="None"
              realm="" />
          <message clientCredentialType="UserName" algorithmSuite="Default" />
        </security>
      </binding>
      <binding name="BasicHttpBinding_IOrderIDGeneratorService"
        closeTimeout="00:01:00"
        openTimeout="00:01:00" receiveTimeout="00:10:00" sendTimeout="00:01:00"
        allowCookies="false" bypassProxyOnLocal="false"
        hostNameComparisonMode="StrongWildcard"
        maxBufferSize="65536" maxBufferPoolSize="524288"
        maxReceivedMessageSize="65536"
        messageEncoding="Text" textEncoding="utf-8" transferMode="Buffered"
        useDefaultWebProxy="true">
        <readerQuotas maxDepth="32" maxStringContentLength="8192"
          maxArrayLength="16384"
          maxBytesPerRead="4096" maxNameTableCharCount="16384" />
        <security mode="None">
          <transport clientCredentialType="None" proxyCredentialType="None"
            realm="" />
          <message clientCredentialType="UserName" algorithmSuite="Default" />
        </security>
      </binding>
```

```
            <binding name="BasicHttpBinding_IProductPriceService" closeTimeout="00:01:00"
              openTimeout="00:01:00" receiveTimeout="00:10:00" sendTimeout="00:01:00"
              allowCookies="false" bypassProxyOnLocal="false"
              hostNameComparisonMode="StrongWildcard"
              maxBufferSize="65536" maxBufferPoolSize="524288"
              maxReceivedMessageSize="65536"
              messageEncoding="Text" textEncoding="utf-8" transferMode="Buffered"
              useDefaultWebProxy="true">
              <readerQuotas maxDepth="32" maxStringContentLength="8192"
               maxArrayLength="16384"
               maxBytesPerRead="4096" maxNameTableCharCount="16384" />
              <security mode="None">
                <transport clientCredentialType="None" proxyCredentialType="None"
                    realm="" />
                <message clientCredentialType="UserName" algorithmSuite="Default" />
              </security>
            </binding>
            <binding name="BasicHttpBinding_IProductStockService" closeTimeout="00:01:00"
                openTimeout="00:01:00" receiveTimeout="00:10:00" sendTimeout="00:01:00"
                allowCookies="false" bypassProxyOnLocal="false" hostNameComparisonMode="Stro
                ngWildcard"
                maxBufferSize="65536" maxBufferPoolSize="524288" maxReceivedMessageSize="655
                36"
                messageEncoding="Text" textEncoding="utf-8" transferMode="Buffered"
                useDefaultWebProxy="true">
              <readerQuotas maxDepth="32" maxStringContentLength="8192"
                maxArrayLength="16384"
                maxBytesPerRead="4096" maxNameTableCharCount="16384" />
              <security mode="None">
                <transport clientCredentialType="None" proxyCredentialType="None"
                  realm="" />
                <message clientCredentialType="UserName" algorithmSuite="Default" />
              </security>
            </binding>
            <binding name="BasicHttpBinding_IOrderProcessorService" closeTimeout="00:01:00"
                openTimeout="00:01:00" receiveTimeout="00:10:00" sendTimeout="00:01:00"
                allowCookies="false" bypassProxyOnLocal="false"
                hostNameComparisonMode="StrongWildcard"
                maxBufferSize="65536" maxBufferPoolSize="524288"
                maxReceivedMessageSize="65536"
                messageEncoding="Text" textEncoding="utf-8" transferMode="Buffered"
                useDefaultWebProxy="true">
              <readerQuotas maxDepth="32" maxStringContentLength="8192"
                maxArrayLength="16384"
                maxBytesPerRead="4096" maxNameTableCharCount="16384" />
              <security mode="None">
                <transport clientCredentialType="None" proxyCredentialType="None"
                  realm="" />
                <message clientCredentialType="UserName" algorithmSuite="Default" />
              </security>
            </binding>
            <binding name="BasicHttpBinding_IOrderRepositoryService" closeTimeout="00:10:00"
              openTimeout="00:10:00" receiveTimeout="01:40:00" sendTimeout="00:10:00"
              allowCookies="false" bypassProxyOnLocal="false"
              hostNameComparisonMode="StrongWildcard"
              maxBufferSize="524288" maxBufferPoolSize="524288"
              maxReceivedMessageSize="524288"
              messageEncoding="Text" textEncoding="utf-8" transferMode="Buffered"
              useDefaultWebProxy="true">
              <readerQuotas maxDepth="32" maxStringContentLength="524288"
                maxArrayLength="524288"
                maxBytesPerRead="4096" maxNameTableCharCount="16384" />
              <security mode="None">
                <transport clientCredentialType="None" proxyCredentialType="None"
                  realm="" />
                <message clientCredentialType="UserName" algorithmSuite="Default" />
              </security>
            </binding>
          </xsl:copy>
```

```
    </xsl:template>

    <xsl:template match="configuration/system.serviceModel">
      <xsl:copy>
        <xsl:apply-templates select="node()|@*"/>
            <client>
              <endpoint
                address="http://[YOUR AZURE CE INSTANCE URL]/sitecore modules/shell
                /ecommerce/services/OrderService.svc" binding="basicHttpBinding"
                  bindingConfiguration="BasicHttpBinding_IOrderService"
                  contract="OrderService.IOrderService"
                  name="BasicHttpBinding_IOrderService" />
              <endpoint
                address="http://[YOUR AZURE CE INSTANCE URL]/sitecore modules/shell
                /ecommerce/services/OrderIDGeneratorService.svc"
                binding="basicHttpBinding"
                bindingConfiguration="BasicHttpBinding_IOrderIDGeneratorService"

                contract="OrderIDGeneratorService.IOrderIDGeneratorService"
                name="BasicHttpBinding_IOrderIDGeneratorService" />
              <endpoint
                address="http://[YOUR AZURE CE INSTANCE URL]/sitecore modules/shell
                /ecommerce/services/ProductPriceService.svc"
                 binding="basicHttpBinding"
                 bindingConfiguration="BasicHttpBinding_IProductPriceService"
                 contract="ProductPriceService.IProductPriceService"
                 name="BasicHttpBinding_IProductPriceService" />
              <endpoint
                address="http://[YOUR AZURE CE INSTANCE URL]/sitecore modules/shell
                /ecommerce/services/ProductStockService.svc"
                 binding="basicHttpBinding"
                 bindingConfiguration="BasicHttpBinding IProductStockService"
                 contract="ProductStockService.IProductStockService"
                 name="BasicHttpBinding_IProductStockService" />
              <endpoint
                address="http://[YOUR AZURE CE INSTANCE URL]/sitecore modules/shell
                /ecommerce/services/OrderProcessorService.svc"
                binding="basicHttpBinding"
                bindingConfiguration="BasicHttpBinding_IOrderProcessorService"
                contract="OrderProcessorService.IOrderProcessorService"
                name="BasicHttpBinding_IOrderProcessorService" />
              <endpoint
                address="http://[YOUR AZURE CE INSTANCE URL]/sitecore modules/shell
                /ecommerce/services/OrderRepositoryService.svc"
                binding="basicHttpBinding"
                bindingConfiguration="BasicHttpBinding_IOrderRepositoryService"
                contract= "OrderRepositoryService.IOrderRepositoryService"
                name="BasicHttpBinding_IOrderRepositoryService" />
            </client>
      </xsl:copy>
    </xsl:template>
  </xsl:stylesheet>
```

16. Expand the Staging node.
17. Expand the Database references node.
18. Duplicate any of the nodes and name the new node *wfm*.
19. If the node *analytics* does not exist, duplicate any node and name it *analytics*.
20. For the `core`, `analytic` and `wfm` databases nodes, change the **Database Id** field to point to the following database:

    `/sitecore/system/Modules/Azure/[YOUR DEPLOYMENT NAME]//North Europe/Editing01/Sql01/Set01`:
21. Switch to the **Azure** dialog box that is already open and click **Start deployment**.

22. Select OK to use the existing databases so they are not recreated in the dialogs that will appear.
23. Wait for the deployment to complete.
24. Test your CD instance.

Now, you have SES installation that is deployed on Azure.