



Enterprise Content Management apps in SharePoint 2013 and SharePoint Online solution pack (Module 3 of 8)

Microsoft Corporation

November 2014

**Applies to:** SharePoint 2013 and SharePoint Online

**Summary:** This solution pack includes code and documents that demonstrate and describe techniques that use enterprise content management features in SharePoint 2013 and SharePoint Online that can be delivered using apps.

©2014 Microsoft Corporation. All rights reserved.

This document is provided "as-is." Information and views expressed in this document, including URL and other Internet website references, may change without notice. You bear the risk of using it.

Some examples are for illustration only and are fictitious. No real association is intended or inferred.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes. You may modify this document for your internal, reference purposes.

# Contents

The Enterprise Content Management apps in SharePoint 2013 and SharePoint Online solution pack contains eight modules, which are listed in Table 1.

**Table 1. Enterprise Content Management apps in SharePoint 2013 and SharePoint Online solution pack modules**

|  |  |  |
| --- | --- | --- |
| **Module** | **Name** | **Describes how to…** |
| 1 | Document library templates | Implement a custom document library template when creating a document library. This sample describes how to use site columns, site content types, taxonomy fields, and version settings, and how to remove the default document content type from a document library. |
| 2 | Document auto tagging | Automatically tag documents with metadata when documents are created or uploaded to SharePoint. This sample describes creation of taxonomy fields and content types, creation of document libraries with content types, registration of the ItemAdding and ItemAdded Remote Event Receiver, removal of Remote Event Receivers, retrieval of User Profile properties, and setting of taxonomy fields. |
| **3** | **Information Management** | **Get or set site policies to manage the site lifecycle (closure and deletion of sites after a period of time).** |
| 4 | Records management extensions | Enable and change in-place records management settings on your sites and lists. |
| 5 | Taxonomy operations | Create and read taxonomy data. |
| 6 | Bulk uploading documents | Bulk upload documents to document libraries (including OneDrive for Business). |
| 7 | Upload large files | Use different methods to upload large files to a document library. |
| 8 | Synchronize term groups | Synchronize term groups across multiple term stores. |

# [Core.InformationManagement app](https://github.com/OfficeDev/PnP/tree/dev/Scenarios/Core.InformationManagement)

|  |  |  |
| --- | --- | --- |
| **What this demonstrates** | **Why you would want to use this sample** | **How this app works** |
| This code sample uses a provider-hosted app for SharePoint to show how to get and set a site policy on a site. | Consider using this sample when you want to:   * Apply policy settings during your custom site provisioning process.   **Important:** Currently, you cannot do any of the following items programmatically:   * Create a site policy (however you can modify an existing site policy). * Create a custom expiration formula. | This app implements an ASP.Net web application as a provider-hosted app that:   * Gets the site policy applied to a site, and the details of the site policy (the closure and expiration dates of the site based on the applied site policy). * Allows you to select and apply a site policy to a site. |

**Related samples**:

[OfficeDevPnP.Core](https://github.com/OfficeDev/PnP/tree/dev/OfficeDevPnP.Core)

[Core.SiteClassification](https://github.com/OfficeDev/PnP/tree/dev/Scenarios/Core.SiteClassification)

[ECM.AutoTagging](https://github.com/OfficeDev/PnP/tree/dev/Scenarios/ECM.AutoTagging)

[ECM.DocumentLibraries](https://github.com/OfficeDev/PnP/tree/dev/Scenarios/ECM.DocumentLibraries)

[ECM.RecordsManagement](https://github.com/OfficeDev/PnP/tree/dev/Scenarios/ECM.RecordsManagement)

**Before you run this app:**

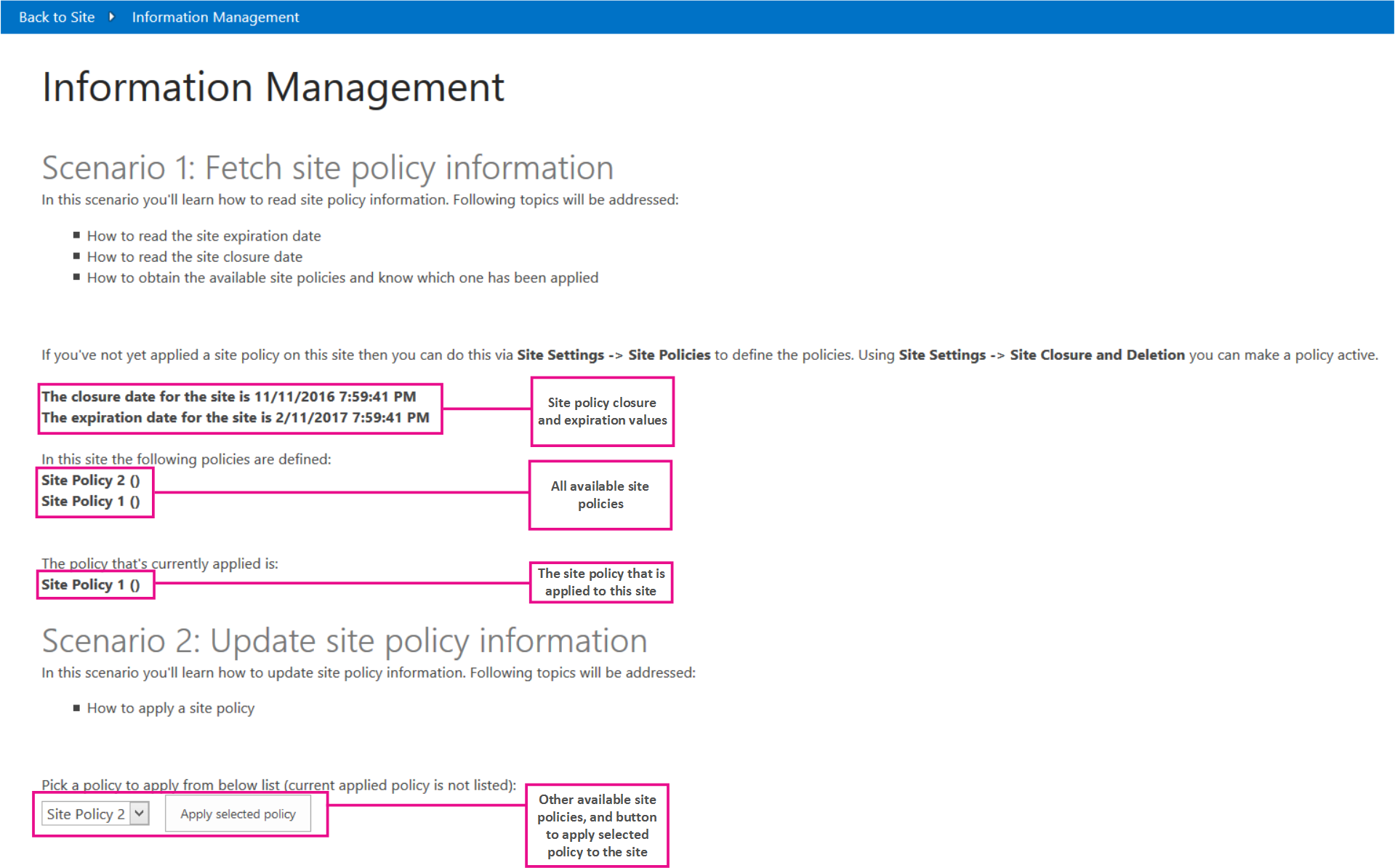
* We recommend that you create at least one site policy, and assign it to your site before running this app. If you don’t do this step, the app will start without displaying sample data (Figure 1 shows the app displaying sample data). For more information on site policies, see [Overview of site policies in SharePoint 2013](http://technet.microsoft.com/en-US/library/jj219569(v=office.15).aspx).

When you start the app, the launch page displays the following information (as shown in Figure 1):

* The site’s closure and expiration dates. These dates are specific to a site and depend on the applied site policy’s configuration settings.
* All site policies that can be applied to this site.
* The currently applied site policy.
* The ability to select and apply a new site policy to the site.

From your SharePoint site, you can navigate to the app, which runs on the remote host, by choosing **Recent > Core.InformationManagement**. To return to your SharePoint site, choose **Back to Site.**

**Figure 1. Information Management launch page**



In the **Core.InformationManagementWeb** project, the **Pages\Default.aspx.cs** filecontains the code for the page displayed in Figure 1.

The following code in the **Page\_Load** method of the **Default.aspx.cs** page fetches and displays the closure and expiration dates of the site, based on the applied site policy. The following code calls the **GetSiteExpirationDate** and **GetSiteCloseDate** extension methods of the OfficeDevPnP.Core project.

// Get site expiration and closure dates.

if (cc.Web.HasSitePolicyApplied())

{

lblSiteExpiration.Text = String.Format("The expiration date for the site is {0}", cc.Web.GetSiteExpirationDate());

lblSiteClosure.Text = String.Format("The closure date for the site is {0}", cc.Web.GetSiteCloseDate());

}

The following code in the **Page\_Load** method of the **Default.aspx.cs** page displays the names of all site policies that can be applied to the site (including the currently applied site policy). The following code calls the **GetSitePolicies** extension method of the OfficeDevPnP.Core project.

// List the defined policies.

List<SitePolicyEntity> policies = cc.Web.GetSitePolicies();

string policiesString = "";

foreach (var policy in policies)

{

policiesString += String.Format("{0} ({1}) <BR />", policy.Name, policy.Description);

}

lblSitePolicies.Text = policiesString;

};

The following code in the **Page\_Load** method of the **Default.aspx.cs** page displays the name of the site policy currently applied to the site. The following code calls the **GetAppliedSitePolicy** extension method of the OfficeDevPnP.Core project.

// Show the assigned policy.

SitePolicyEntity appliedPolicy = cc.Web.GetAppliedSitePolicy();

if (appliedPolicy != null)

{

lblAppliedPolicy.Text = String.Format("{0} ({1})", appliedPolicy.Name, appliedPolicy.Description);

}

else

{

lblAppliedPolicy.Text = "No policy has been applied";

}

The following code in the **Page\_Load** method of the **Default.aspx.cs** page fills the drop-down list with all the site policies available, except the site policy that is currently assigned to the site.

// Fill the policies combo.

foreach (var policy in policies)

{

if (appliedPolicy == null || !policy.Name.Equals(appliedPolicy.Name, StringComparison.InvariantCultureIgnoreCase))

{

drlPolicies.Items.Add(policy.Name);

}

}

btnApplyPolicy.Enabled = drlPolicies.Items.Count > 0;

The following code in the **Default.aspx.cs** page applies the selected site policy to the site. The original site policy no longer applies and is replaced by the new site policy.

protected void btnApplyPolicy\_Click(object sender, EventArgs e)

{

if (drlPolicies.SelectedItem != null)

{

cc.Web.ApplySitePolicy(drlPolicies.SelectedItem.Text);

Page.Response.Redirect(Page.Request.Url.ToString(), true);

}

}