# Office AMS: Site Classification

|  |  |
| --- | --- |
| Summary: | Applies to: |
| This sample demonstrates how to implement a site classification solution using various samples found in AMS as well as leverage Site Policies | * Office 365 Multi-Tenant (MT) * Office 365 Dedicated (D) * SharePoint 2013 on-premises |
| Solution: | Core.SiteClassification, version 1.0 |
| Author: | Brian Michely, Vesa Juvonen, Bert Jansen, Frank Marasco (Microsoft) |
| //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  // THIS CODE IS PROVIDED \*AS IS\* WITHOUT WARRANTY OF  // ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING ANY  // IMPLIED WARRANTIES OF FITNESS FOR A PARTICULAR  // PURPOSE, MERCHANTABILITY, OR NON-INFRINGEMENT.  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* | |

## Version log

Change log between different releases.

|  |  |
| --- | --- |
| Version | Notes |
| 1.0 | Initial |

# General comments

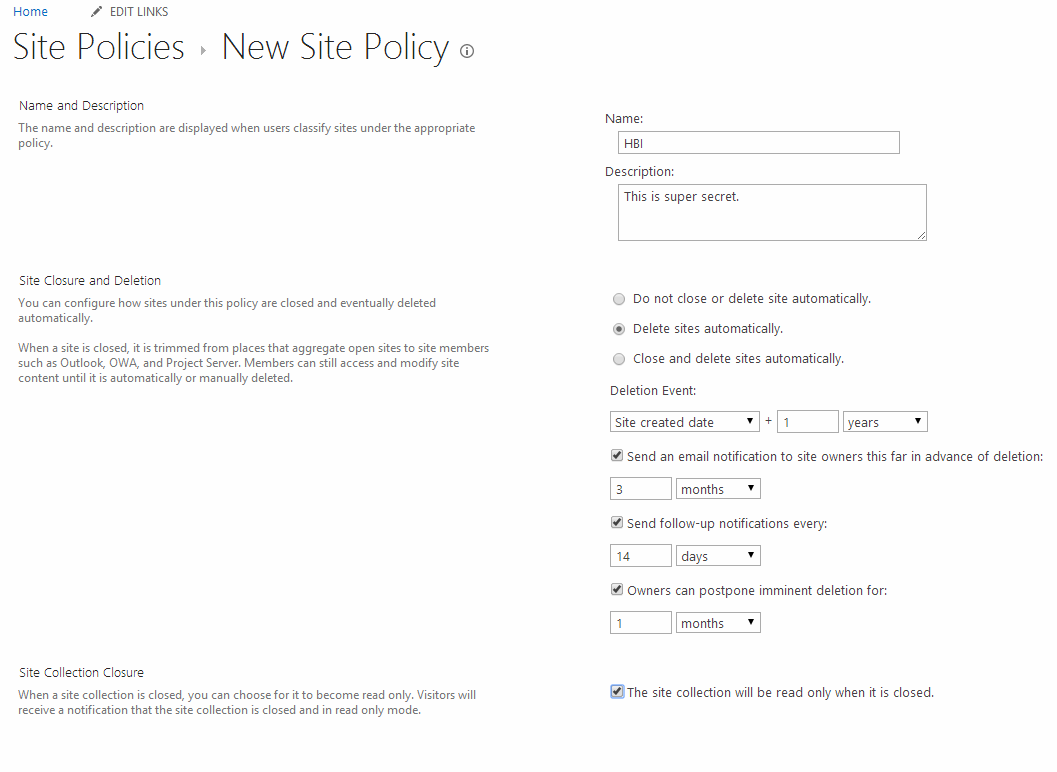
Even with good governance, SharePoint sites can proliferate and grow out of control. Sites are created as they are needed, but sites are rarely deleted. Many organization have search crawl burdened by unused site collections, difficulty with outdated and irrelevant results. With site classification it allows sensitive data in the environment to be identified. In this scenario, will demonstrate on how to implement a site classification solution that leverages many of the existing AMS samples. This solution will also leverage SharePoint Site Policies to enforce the deletion. This solution can also be integrated into your existing Site Provisioning solution to form a unified solution for your governance needs.

# SETUP

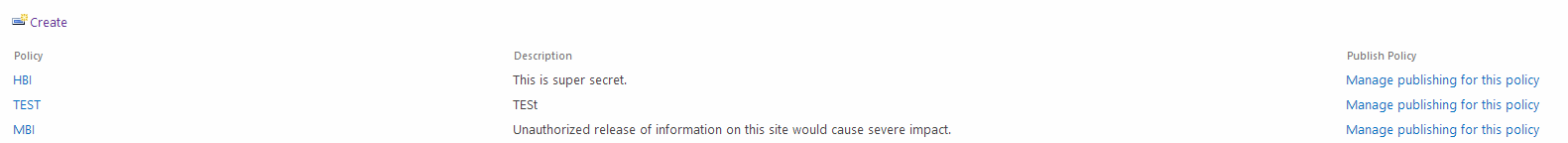
First of we need to define the site policies that will be available in all your sites collections. We are going to define the Site Policies in the content type hub and publish. In this example we are using SharePoint Online MT, but this same approach is available in SharePoint Online Dedicated as well as SharePoint on-premises. If your environment is hosted in SharePoint Online MT, your content type hub would be located at the following URL. <https://[tenanatname]/sites/contentTypeHub>. Navigate to Settings, then Site Policies under Site Collection Administration, and then finally create.

**Note:** See Overview of site policies in SharePoint 2013 at <http://technet.microsoft.com/en-US/library/jj219569(v=office.15).aspx> for an overview of Site Policies

We are going to create three site policies, HBI, MBI and then LBI. Create an HBI Policy that mimics the below screen.



Repeat the above setup 2 more times for MBI and LBI. You should end up with the below



Once we have the policies we are going to publish the Site Policies.

# Scenario 1: Insert a custom action

Here we are going to add the custom action to the Settings page and the SharePoint gear. That is only available to users with ManageWeb Permission.

|  |  |
| --- | --- |
|  |  |

/// <summary>

/// Adds a custom Action to a Site Collection

/// </summary>

/// <param name="ctx">The Authenticaed client context.</param>

/// <param name="hostUrl">The Provider hosted URL for the Application</param>

static void AddCustomAction(ClientContext ctx, string hostUrl)

{

var \_web = ctx.Web;

ctx.Load(\_web);

ctx.ExecuteQuery();

//we only want the action to show up if you have manage web permissions

BasePermissions \_manageWebPermission = new BasePermissions();

\_manageWebPermission.Set(PermissionKind.ManageWeb);

CustomActionEntity \_entity = new CustomActionEntity()

{

Group = "SiteTasks",

Location = "Microsoft.SharePoint.SiteSettings",

Title = "Site Classification",

Sequence = 1000,

Url = string.Format(hostUrl, ctx.Url),

Rights = \_manageWebPermission,

};

CustomActionEntity \_siteActionSC = new CustomActionEntity()

{

Group = "SiteActions",

Location = "Microsoft.SharePoint.StandardMenu",

Title = "Site Classification",

Sequence = 1000,

Url = string.Format(hostUrl, ctx.Url),

Rights = \_manageWebPermission

};

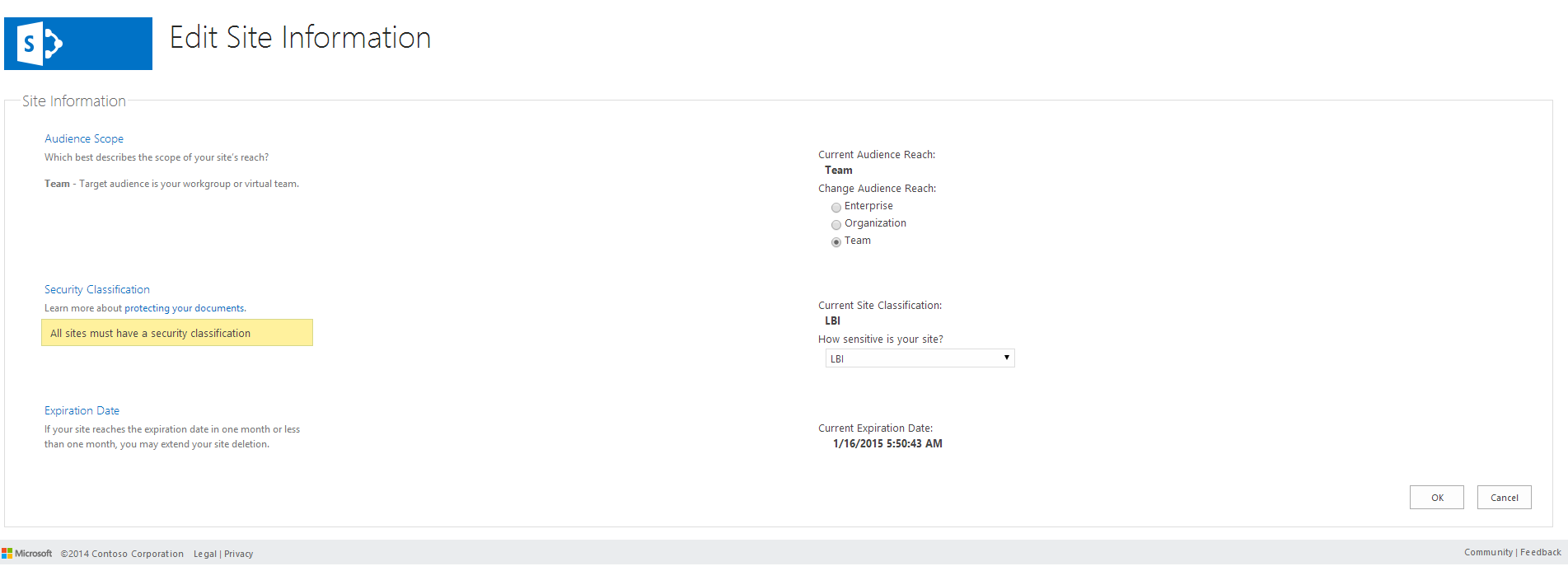
\_web.AddCustomAction(\_entity);

\_web.AddCustomAction(\_siteActionSC);

}

See [here](http://msdn.microsoft.com/en-us/library/office/bb802730(v=office.15).aspx) for more information on the custom action settings.

# Scenario 2: CUSTOM SITE CLASSIFCATION



This pages defines the necessary options that are available. Here we define the intended Audience reach of the Site as well as the defined Site Policy. We also show the Expiration date of the site, which is based on the site policy that you created earlier. Both the audience reach and Site classification are searchable and will have managed properties associated after a crawl has taken place. You can then use these properties to search for specific types of sites from.



These are searchable via a custom hidden list that is implemented in the Site Collection. This implement in the Core.SiteClassification.Common project in the SiteManagerImpl class.

private void CreateSiteClassificationList(ClientContext ctx)

{

var \_newList = new ListCreationInformation()

{

Title = SiteClassificationList.SiteClassificationListTitle,

Description = SiteClassificationList.SiteClassificationDesc,

TemplateType = (int)ListTemplateType.GenericList,

Url = SiteClassificationList.SiteClassificationUrl,

QuickLaunchOption = QuickLaunchOptions.Off

};

if(!ctx.Web.ContentTypeExistsById(SiteClassificationContentType.SITEINFORMATION\_CT\_ID))

{

//ct

ContentType \_contentType = ctx.Web.CreateContentType(SiteClassificationContentType.SITEINFORMATION\_CT\_NAME,

SiteClassificationContentType.SITEINFORMATION\_CT\_DESC,

SiteClassificationContentType.SITEINFORMATION\_CT\_ID,

SiteClassificationContentType.SITEINFORMATION\_CT\_GROUP);

FieldLink \_titleFieldLink = \_contentType.FieldLinks.GetById(new Guid("fa564e0f-0c70-4ab9-b863-0177e6ddd247"));

\_titleFieldLink.Required = false;

\_contentType.Update(false);

//Key Field

ctx.Web.CreateField(SiteClassificationFields.FLD\_KEY\_ID,

SiteClassificationFields.FLD\_KEY\_INTERNAL\_NAME,

FieldType.Text,

SiteClassificationFields.FLD\_KEY\_DISPLAY\_NAME,

SiteClassificationFields.FIELDS\_GROUPNAME);

//value field

ctx.Web.CreateField(SiteClassificationFields.FLD\_VALUE\_ID,

SiteClassificationFields.FLD\_VALUE\_INTERNAL\_NAME,

FieldType.Text,

SiteClassificationFields.FLD\_VALUE\_DISPLAY\_NAME,

SiteClassificationFields.FIELDS\_GROUPNAME);

//Add Key Field to content type

ctx.Web.AddFieldToContentTypeById(SiteClassificationContentType.SITEINFORMATION\_CT\_ID,

SiteClassificationFields.FLD\_KEY\_ID.ToString(),

true);

//Add Value Field to content type

ctx.Web.AddFieldToContentTypeById(SiteClassificationContentType.SITEINFORMATION\_CT\_ID,

SiteClassificationFields.FLD\_VALUE\_ID.ToString(),

true);

}

var \_list = ctx.Web.Lists.Add(\_newList);

\_list.Hidden = true;

\_list.ContentTypesEnabled = true;

\_list.Update();

ctx.Web.AddContentTypeToListById(SiteClassificationList.SiteClassificationListTitle, SiteClassificationContentType.SITEINFORMATION\_CT\_ID, true);

this.CreateCustomPropertiesInList(\_list);

ctx.ExecuteQuery();

this.RemoveFromQuickLaunch(ctx, SiteClassificationList.SiteClassificationListTitle);

}

By default, when you create list either using out-of-box or if you are using CSOM, the list will be available in the Recent menu. Now, we don’t want that right? It’s supposed to be hidden. We need some simple code to remove the item from the recent menu.

See for more information: http://blogs.technet.com/b/speschka/archive/2014/05/07/create-a-list-in-the-host-web-when-your-sharepoint-app-is-installed-and-remove-it-from-the-recent-stuff-list.aspx

private void RemoveFromQuickLaunch(ClientContext ctx, string listName)

{

Site \_site = ctx.Site;

Web \_web = \_site.RootWeb;

ctx.Load(\_web, x => x.Navigation, x => x.Navigation.QuickLaunch);

ctx.ExecuteQuery();

var \_vNode = from NavigationNode \_navNode in \_web.Navigation.QuickLaunch

where \_navNode.Title == "Recent"

select \_navNode;

NavigationNode \_nNode = \_vNode.First<NavigationNode>();

ctx.Load(\_nNode.Children);

ctx.ExecuteQuery();

var vcNode = from NavigationNode cn in \_nNode.Children

where cn.Title == listName

select cn;

NavigationNode \_cNode = vcNode.First<NavigationNode>();

\_cNode.DeleteObject();

ctx.ExecuteQuery();

}

So you’re probably thinking that site admin or someone with permission can remove that list. Well we thought about that too. When this page is accessed we will create the recreate the list, however in the sample we don’t set the properties back. If the values in the list are not present, we know that someone deleted the list and you can leverage the AMS Sample Core.SiteEnumeration to do checks on the list and send nasty emails to your site administrators. You may also extend this sample and modify the permissions on the list so that only site collections administrators have access.

The list verification check is also implemented in the SiteManagerImpl in the Initialize member.

internal void Initialize(ClientContext ctx)

{

try {

var \_web = ctx.Web;

var lists = \_web.Lists;

ctx.Load(\_web);

ctx.Load(lists, lc => lc.Where(l => l.Title == SiteClassificationList.SiteClassificationListTitle));

ctx.ExecuteQuery();

if (lists.Count == 0) {

this.CreateSiteClassificationList(ctx);

}

}

catch(Exception \_ex)

{

}

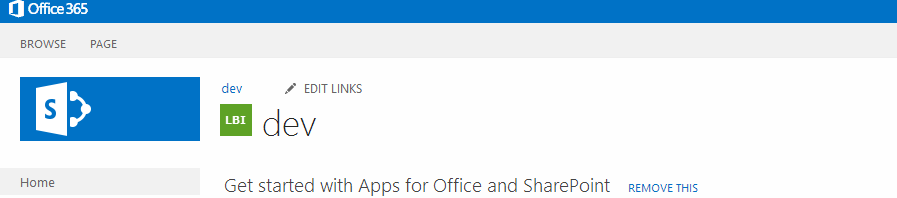
}

}

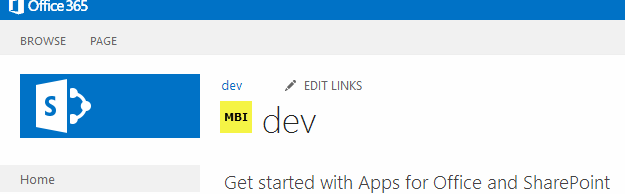
# Scenario 3: SITE CLASSIFICATION DISPLAY

In the final scenario, we are going to add an indicator. I chose to inject an image next to the Site Title. In the old days we would use a Delegate control or a custom master page and add some JavaScript. Delegate controls are Server Side implemented we don’t want to use that. Modifying the master page would work, but I chose our friendly JavaScript injection pattern. When you change the Site Policy in the Edit Site Information page, this will change the site indicator like below.

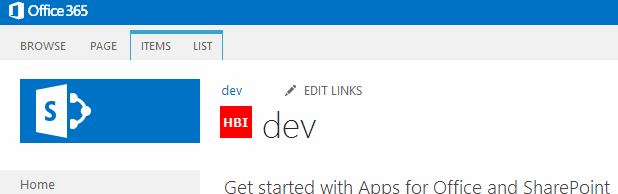
### LBI



### MBI



### HBI



The below method is defined in the Core.SiteClassificationWeb project, scripts and classifier.js. I chose to store the images in an Azure Web Site, so you will have to change the URL’s to match your environment. Maybe in the next release I can remove the hard-coded urls.

function setClassifier() {

if (!classified)

{

var clientContext = SP.ClientContext.get\_current();

var query = "<View><Query><Where><Eq><FieldRef Name='SC\_METADATA\_KEY'/><Value Type='Text'>sc\_BusinessImpact</Value></Eq></Where></Query><ViewFields><FieldRef Name='ID'/><FieldRef Name='SC\_METADATA\_KEY'/><FieldRef Name='SC\_METADATA\_VALUE'/></ViewFields></View>";

var list = clientContext.get\_web().get\_lists().getByTitle("Site Information");

clientContext.load(list);

var camlQuery = new SP.CamlQuery();

camlQuery.set\_viewXml(query);

var listItems = list.getItems(camlQuery);

clientContext.load(listItems);

clientContext.executeQueryAsync(Function.createDelegate(this, function (sender, args) {

var listItemInfo;

var listItemEnumerator = listItems.getEnumerator();

while (listItemEnumerator.moveNext()) {

listItemInfo = listItemEnumerator.get\_current().get\_item('SC\_METADATA\_VALUE');

var pageTitle = $('#pageTitle')[0].innerHTML;

if (pageTitle.indexOf("img") > -1) {

classified = true;

}

else {

var siteClassification = listItemInfo;

if (siteClassification == "HBI") {

var img = $("<a href='http://insertyourpolicy' target=\_blank><img id=classifer name=classifer src='https://spmanaged.azurewebsites.net/content/img/hbi.png' title='Site contains personally identifiable information (PII), or unauthorized release of information on this site would cause severe or catastrophic loss to Contoso.' alt='Site contains personally identifiable information (PII), or unauthorized release of information on this site would cause severe or catastrophic loss to Contoso.'></a>");

$('#pageTitle').prepend(img);

classified = true;

}

else if (siteClassification == "MBI") {

var img = $("<a href='http://insertyourpolicy' target=\_blank><img id=classifer name=classifer src='https://spmanaged.azurewebsites.net/content/img/mbi.png' title='Unauthorized release of information on this site would cause severe impact to Contoso.' alt='Unauthorized release of information on this site would cause severe impact to Contoso.'></a>");

$('#pageTitle').prepend(img);

classified = true;

}

else if (siteClassification == "LBI") {

var img = $("<a href='http://insertyourpolicy' target=\_blank><img id=classifer name=classifer src='https://spmanaged.azurewebsites.net/content/img/lbi.png' title='Limited or no impact to Contoso if publically released.' alt='Limited or no impact to Contoso if publically released.'></a>");

$('#pageTitle').prepend(img);

classified = true;

}

}

}

}));

}

# Dependencies

* Microsoft.SharePoint.Client.dll
* Microsoft.SharePoint.Client.Runtime.dll
* Microsoft.Office.Client.Policy
* OfficeAMS.Core