



## **Customizing the Ribbon in Windows® SharePoint® Services "14"**

This document supports a preliminary release of a software product that may be changed substantially prior to final commercial release. This document is provided for informational purposes only and Microsoft makes no warranties, either express or implied, in this document. Information in this document, including URL and other Internet Web site references, is subject to change without notice. The entire risk of the use or the results from the use of this document remains with the user. Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in examples herein are fictitious. No association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

© 2009 Microsoft Corporation. All rights reserved.

All other trademarks are property of their respective owners.

# Customizing the Ribbon in Windows<sup>®</sup> SharePoint<sup>®</sup> Services "14"

Dallas Tester  
Microsoft Corporation  
July 2009

**Applies to:** Windows<sup>®</sup> SharePoint<sup>®</sup> Services "14"  
Microsoft<sup>®</sup> SharePoint<sup>®</sup> Server 2010

**Summary:** The Ribbon is now included in the user interface for Windows<sup>®</sup> SharePoint<sup>®</sup> Services "14" and Microsoft<sup>®</sup> SharePoint<sup>®</sup> Server 2010. It can be extended by using a combination of XML and ECMAScript (JavaScript, JScript). Customizations to the Ribbon are created by using the Feature infrastructure and can be deployed by using the Windows PowerShell™ command-line interface, a solution package, or a user custom action.

## Contents

Contents .....	ii
Basic Objects in the Ribbon .....	ii
Deploying a Ribbon Customization.....	ii
Add a Button to the Ribbon .....	iii
Remove a Button on the Ribbon.....	iv
Replace a Button on the Ribbon .....	v
Using ECMAScript on the Ribbon .....	vi
Conclusion.....	viii
Additional Resources .....	viii

## Basic Objects in the Ribbon

The top-level elements in the Ribbon architecture are tabs. Tabs appear across the top of the page in a SharePoint site. Each tab organizes a set of groups. These groups contain a set of controls. The controls inside the groups include buttons, drop-down menus, check boxes, combo boxes, split buttons, and galleries. Each of these controls is tied to a unique command.

The Ribbon is defined by using XML in a Feature manifest or a user custom action. The XML used for the Ribbon defines each tab, group, and control. The Tab element contains one Groups element. Each Groups element has multiple Group elements. Inside the Group element is a single Controls element that contains multiple different types of controls.

## Deploying a Ribbon Customization

Each of the following customizations in this article shows you the Manifest.xml file. A Feature.xml file defines the scope of and information about the Feature. A sample Feature.xml could resemble the following.

```
<?xml version="1.0" encoding="utf-8" ?>
<Feature Id="FeatureId"
  Title="Name of the Feature"
  Description="Description of the Feature."
```

```

Version="1.0.0.0"
Scope="Web"
xmlns="http://schemas.microsoft.com/sharepoint/">
<ElementManifests>
  <ElementManifest Location="manifest.xml" />
</ElementManifests>
</Feature>

```

To deploy a Feature by using Windows PowerShell, you install it and then enable it by using the following commands.

```

Install-SPFeature FeatureId
Enable-SPFeature FeatureId -Url http://server/site/subsite

```

**Note** After installing a customization by using Windows PowerShell, you may need to reset Internet Information Services (IIS). To do this, enter **iisreset** at a command prompt.

## Add a Button to the Ribbon

To add a button, you define the location on the Ribbon where you want the button to appear. The following procedure adds a new button to the **Library** tab in the **Actions** group for a document library.

### To add a button to the Ribbon

1. Create a folder named **AddARibbonButton** in %ProgramFiles%\Common Files\Microsoft Shared\web server extensions\14\TEMPLATE\FEATURES.
2. Create a feature.xml file as discussed in the Deploying a Ribbon Customization section.
 

**Note** You must replace the *Id*, *Title*, and *Description* attributes in the Feature.xml file. You use the *Id* attribute when deploying the Feature.
3. Copy and paste the following XML into a new text file. This XML will add a new button on the **Library** tab in the **Actions** group for a document library. The images used for *Image16by16* and *Image32by32* are included with Windows SharePoint Services "14".

```

<?xml version="1.0" encoding="utf-8"?>
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">
  <CustomAction Id="Ribbon.Library.Actions.AddAButton"
    Location="ViewToolbar"
    RegistrationId="101"
    RegistrationType="List"
    Title="Add a Ribbon Button">
    <CommandUIExtension>
      <CommandUIDefinitions>
        <CommandUIDefinition
          Location="Ribbon.Library.Actions.Controls._children">
          <Button Id="Ribbon.Library.Actions.NewRibbonButton"

```

```

        Command="NewRibbonButtonCommand"
        Image16by16="/_layouts/images/FILMSTRP.GIF"
        Image32by32="/_layouts/images/PPEOPLE.GIF"
        LabelText="New Button"
        TemplateAlias="o2" />
    </CommandUIDefinition>
</CommandUIDefinitions>
<CommandUIHandlers>
    <CommandUIHandler
        Command="NewRibbonButtonCommand"
        CommandScript="javascript:alert('This is a new button!');" />
</CommandUIHandlers>
</CommandUIExtension>
</CustomAction>
</Elements>

```

4. Save the file as Manifest.xml.
5. Deploy the customization by using Windows PowerShell as shown in the Deploying a Ribbon Customization section.
6. To test the new button, navigate to a document library in your site or subsite. Click the **Library** tab, look in the **Actions** group, and click the **New Button** button.

## Remove a Button on the Ribbon

To remove a button from the Ribbon, you define the location of the button you want to remove. You use a **HideCustomAction** element to remove a button from the Ribbon. The following procedure removes the **Connect to Outlook** button from the **Library** tab in the **Actions** group for a document library.

### To remove a button on the Ribbon

1. Create a folder named **RemoveARibbonButton** in %ProgramFiles%\Common Files\Microsoft Shared\web server extensions\14\TEMPLATE\FEATURES.
2. Create a Feature.xml file as discussed in the Deploying a Ribbon Customization section.
 

**Note** You must replace the *Id*, *Title*, and *Description* attributes in the Feature.xml file. You use the *Id* attribute when deploying the Feature.
3. Copy and paste the following XML into a new text file. This XML removes the **Connect to Outlook** button from the **Library** tab in the **Actions** group for a document library.

```

<?xml version="1.0" encoding="utf-8"?>
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">
    <HideCustomAction Id="RemoveRibbonButton"
        Location="CommandUI.Ribbon.Library.Actions.ConnectToClient">
    </HideCustomAction>

```

```
</Elements>
```

4. Save the file as Manifest.xml.
5. Deploy the customization by using Windows PowerShell as shown in the Deploying a Ribbon Customization section.
6. To see the results, navigate to a document library in your site or subsite. Click the **Library** tab, look in the **Actions** group, and observe the absence of the **Connect to Outlook** button.

## Replace a Button on the Ribbon

To replace a button on the Ribbon, you define the location of the button you want to replace. The following procedure replaces the **Connect to Outlook** button on the **Library** tab in the **Actions** group for a document library.

### To replace a button on the Ribbon

1. Create a folder named **ReplaceARibbonButton** in %ProgramFiles%\Common Files\Microsoft Shared\web server extensions\14\TEMPLATE\FEATURES.
2. Create a Feature.xml file as discussed in the Deploying a Ribbon Customization section. **Note** You must replace the *Id*, *Title*, and *Description* attributes in the Feature.xml file. You use the *Id* attribute when deploying the Feature.
3. Copy and paste the following XML into a new text file. This XML replaces the **Connect to Outlook** button on the **Library** tab in the **Actions** group for a document library. The images used for *Image16by16* and *Image32by32* are included with Windows SharePoint Services "14".

```
<?xml version="1.0" encoding="utf-8"?>
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">
  <CustomAction Id="Ribbon.Library.Actions.ReplacementButton"
    Location="ViewToolbar"
    RegistrationId="101"
    RegistrationType="List"
    Title="Replace a Ribbon Button">
    <CommandUIExtension>
      <CommandUIDefinitions>
        <CommandUIDefinition
          Location="Ribbon.Library.Actions.ConnectToClient">
          <Button Id="Ribbon.Library.Actions.ConnectToClient.
ReplacementButton"
            Command="ReplacementButtonCommand"
            Image16by16="/_layouts/images/FILMSTRP.GIF"
            Image32by32="/_layouts/images/PPEOPLE.GIF"
            LabelText="Replaced Button"
```

```

        TemplateAlias="o1" />
    </CommandUIDefinition>
</CommandUIDefinitions>
<CommandUIHandlers>
    <CommandUIHandler
        Command="ReplacementButtonCommand"
        CommandScript="javascript:alert('This button has been
replaced.');" />
    </CommandUIHandler>
</CommandUIHandlers>
</CommandUIExtension>
</CustomAction>
</Elements>

```

4. Save the file as Manifest.xml.
5. Deploy the customization by using Windows PowerShell as shown in the Deploying a Ribbon Customization section.
6. To test the new button, navigate to a document library in your site or subsite. Click the **Library** tab, look in the **Actions** group, and click on **Replaced Button** button.

## Using ECMAScript on the Ribbon

Putting custom ECMAScript (JavaScript, JScript) on the page can be accomplished in three ways.

- Override a delegate control on the page.
- Put a Web Part on the page.
- Use ScriptLink for the Location attribute on a custom action.

Replacing a delegate should be used when working with a document library or list. The AdditionalPageHead delegate can be used to insert ECMAScript in the page. This type of customization can be scoped to a single Web site, Web application, or the entire farm.

Web Parts can be placed on the page and register script by using the **RegisterClientScriptBlock** method of the **System.Web.UI.Page** object. This method of adding ECMAScript to the page should be used when the Ribbon button is related to the Web Part that emits the ECMAScript.

The ScriptLink location on a custom action places script on the page at the **SPSite** level. All pages in the site will have the script on the page. This type of customization should be used when creating a customization that affects most pages within a site.

## To replace a button on the Ribbon for a Site Collection

1. Create a folder named **RibbonButtonInScriptLink** in %ProgramFiles%\Common Files\Microsoft Shared\web server extensions\14\TEMPLATE\FEATURES.
2. Create a Feature.xml file as discussed in the Deploying a Ribbon Customization section.

**Note** You must replace the *Id*, *Title*, and *Description* attributes in the Feature.xml file. You use the *Id* attribute when deploying the Feature. The *Scope* attribute in the Feature.xml file must be set to **Site** to enable the button in a site collection.

- Copy and paste the following XML into a new text file. The first custom action replaces the **Connect to Outlook** button on the **Library** tab in the **Actions** group for a document library. The second custom action defines the **HelloWorld** ECMAScript function that is called by the **CommandUIHandler** element. The images used for *Image16by16* and *Image32by32* are included with Windows SharePoint Services "14".

```
<?xml version="1.0" encoding="utf-8"?>
<Elements xmlns="http://schemas.microsoft.com/sharepoint/">
  <CustomAction Id="Ribbon.Library.Actions.ConnectToClient"
    Location="ViewToolbar"
    RegistrationId="101"
    RegistrationType="List"
    Title="Custom ECMAScript Button">
    <CommandUIExtension>
      <CommandUIDefinitions>
        <CommandUIDefinition
          Location="Ribbon.Library.Actions.ConnectToClient">
          <Button Id="Ribbon.Library.Actions.ConnectToClient"
            Command="CustomECMAScriptCommand"
            Image16by16="/_layouts/images/FILMSTRP.GIF"
            Image32by32="/_layouts/images/PPEOPLE.GIF"
            LabelText="Hello World"
            TemplateAlias="o1" />
          </CommandUIDefinition>
        </CommandUIDefinitions>
        <CommandUIHandlers>
          <CommandUIHandler
            Command="CustomECMAScriptCommand"
            CommandScript="javascript:HelloWorld();" />
          </CommandUIHandlers>
        </CommandUIExtension>
      </CustomAction>
      <CustomAction Id="Ribbon.Library.Actions.NewButton.Script"
        Location="ScriptLink"
        ScriptBlock="
          function HelloWorld()
```

```
{
    alert('Hello, world!');
}" />
</Elements>
```

4. Save the file as Manifest.xml.
5. Deploy the customization by using Windows PowerShell as shown in the Deploying a Ribbon Customization section.
6. To test the new button, navigate to a document library in your site or subsite. Click the **Library** tab, look in the **Actions** group, and click the **Hello World** button.

## Conclusion

The Ribbon provides a cohesive user interface for Windows SharePoint Services "14" and Microsoft SharePoint Server 2010. The Ribbon can be customized by using XML and ECMAScript (JavaScript, JScript). These customizations can be scoped and use default ECMAScript, the Windows SharePoint Services ECMAScript object model, or custom ECMAScript.

## Additional Resources

[How To: Create a Simple Feature  
Feature.xml Files](#)