Guide for IT Pros for Office Web Apps (Installed on SharePoint 2010 Products)

Microsoft Corporation

Published: February 2011

Author: Microsoft Office System and Servers Team (itspdocs@microsoft.com)

Abstract

This book includes information to help IT professionals use Microsoft Office Web Apps on Microsoft SharePoint 2010 Products in an organization.

The content in this book is a copy of the selected content in the Office Web Apps technical library (*http://go.microsoft.com/fwlink/?LinkId=195436*). For the most current content, see the technical library on the Web.

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

Some examples depicted herein are provided for illustration only and are fictitious. No real association or connection is intended or should be 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.

© 2011 Microsoft Corporation. All rights reserved.

Microsoft, Access, Active Directory, Backstage, Excel, Groove, Hotmail, InfoPath, Internet Explorer, Outlook, PerformancePoint, PowerPoint, SharePoint, Silverlight, Windows, Windows Live, Windows Mobile, Windows PowerShell, Windows Server, and Windows Vista are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

Contents

Getting help xxi

Understanding Office Web Apps (Installed on SharePoint 2010 Products) 1
Office Web Apps versions 1
Integration with SharePoint 2010 Products 2
Understanding the Office Web Apps user experience 2
Viewing and editing Office documents 2
Improving the user experience with Silverlight 3
Configuring the default open behavior for browser enabled documents 3
Understanding Office Web Apps architecture 3
Understanding how Office Web Apps work 5
Viewing and editing Word documents using the Word Web App 5
Viewing and editing PowerPoint presentations using the PowerPoint Web App 5
PowerPoint Broadcast Slide Show presentations using the PowerPoint Web App 6
Viewing and editing Excel workbooks using the Excel Web App 7
Viewing and editing OneNote notebooks using the OneNote Web App 7
Planning Office Web Apps in your organization 7
Deploying Office Web Apps in your organization 8
Summary 8

Planning Office Web Apps (Installed on SharePoint 2010 Products) 9
Software prerequisites 9
Browser support 10
Mobile device support 11
Plan server topology 11
Plan default open behavior for documents 13
Plan Office Web Apps Feature activation 13
Optimizing performance 14
Scenarios that require the Office client applications 15
Creating documents from inside a SharePoint site 17
Summary 17

Deploy Office Web Apps (Installed on SharePoint 2010 Products) 18
Understanding Office Web Apps deployment 19
Install and configure Office Web Apps on an existing stand-alone SharePoint server 20
Run Office Web Apps setup 20
Run PSConfig to register the services 20
Start the service instances 21
Create the service applications and the service application proxies 22
Activate the Office Web Apps Feature 23

Install and configure Office Web Apps on a new stand-alone SharePoint server 25
Run Office Web Apps setup 25
Run PSConfig to register the services, start the service instances, create the service applications and proxies, and activate the Office Web Apps Feature 25
Install and configure Office Web Apps on an existing SharePoint server farm 26
Run Office Web Apps setup 26
Run PSConfig to register services 26
Start the service instances 27
Create the service applications and the service application proxies 28
Activate the Office Web Apps Feature 29
Install and configure Office Web Apps on a new SharePoint server farm 31
Run Office Web Apps setup 31
Run PSConfig to register services 31
Run the SharePoint Farm Configuration Wizard to start the service instances, create the service applications and proxies, and activate the Office Web Apps Feature 32
Additional configuration (optional) 33
Configure the SharePoint default open behavior for browser-enabled documents 33
Troubleshooting 35

PowerPoint Broadcast Slide Show 36

Understanding Broadcast Slide Show (Office Web Apps) 37
Broadcast Slide Show Overview 37
Understanding the presenter experience 38
Understanding the attendee experience 38
Broadcast Slide Show features 38
Managing Broadcast Slide Show 39
Broadcast Slide Show and Microsoft Live Meeting 39
Summary 40

Planning Broadcast Slide Show (Office Web Apps) 41
Plan service types 41
Plan broadcast site location 42
Plan permissions 42
Use Group Policy to manage Broadcast Slide Show 43
Summary 43

Configure a Broadcast site (Office Web Apps) 44
Configure a PowerPoint Broadcast site 44

Configure Broadcast Slide Show performance (Office Web Apps) 48
Deploy a dedicated SharePoint server farm 48
Configure broadcast capacity 49

Configure the polling interval 50
Modify throttling to limit Web front-end resource usage 51
Change the maximum duration of a broadcast 52
Set a time limit for idle broadcasts 52

Configure Group Policy settings (Broadcast Slide Show) 54
Manage Broadcast Slide Show by using Group Policy 54
Disable Broadcast Slide Show by using Group Policy 55

Manage Office Web Apps (Installed on SharePoint 2010 Products) 56

Activate the Office Web Apps Feature on site collections 57
Activate the Office Web Apps Feature for a site collection 57

Manage the Office Web Apps cache 60
Manage the Office Web Apps cache 60

Configure the default open behavior for browser-enabled documents (Office Web Apps) 64
Configure the default open behavior for documents 64

Configure PowerPoint service application settings 68
Configure the PowerPoint service application settings 68
Set maximum worker processes by using Windows PowerShell 69

Configure Word Viewing service settings 71
Configure the Word Viewing service application settings 71
Set maximum worker processes by using Windows PowerShell 73
Set maximum rendering time by using Windows Powershell 74

Configure Excel Services Application settings (Office Web Apps) 76

Technical reference (Office Web Apps) 77

Windows PowerShell for Office Web Apps 78

SharePoint 2010 Products administration by using Windows PowerShell 79
Overview 79
Accessing Windows PowerShell for SharePoint 2010 Products 80
Permissions 80
Learning Windows PowerShell 81

Windows PowerShell for Office Web Apps reference 83

Excel Web App cmdlets (Office Web Apps) 84

Get-SPExcelBlockedFileType 87
Syntax 87

Detailed Description 87
Parameters 87
AutoGenParams 90
Input Types 90
Return Types 90
Example 90
Description 90
Code 90
Comments 91

Get-SPExcelDataConnectionLibrary 92
Syntax 92
Detailed Description 92
Parameters 92
AutoGenParams 95
Input Types 95
Return Types 95
Example 95
Description 95
Code 95
Comments 95
Example 95
Description 95
Code 96
Comments 96

Get-SPExcelDataProvider 97
Syntax 97
Detailed Description 97
Parameters 97
AutoGenParams 99
Input Types 100
Return Types 100
Example 100
Description 100
Code 100
Comments 100
Example 100
Description 100
Code 100
Comments 100

Get-SPExcelFileLocation 101

Syntax 101
Detailed Description 101
Parameters 101
AutoGenParams 104
Input Types 105
Return Types 105
Example 105
Description 105
Code 105
Comments 105
Example 105
Description 105
Code 105
Comments 105
Example 105
Description 105
Code 106
Comments 106

Get-SPExcelServiceApplication 107
Syntax 107
Detailed Description 107
Parameters 107
AutoGenParams 109
Input Types 110
Return Types 110
Example 110
Description 110
Code 110
Comments 110
Example 110
Description 110
Code 110
Comments 110
Example 110
Description 110
Code 110
Comments 111

Get-SPExcelUserDefinedFunction 112
Syntax 112
Detailed Description 112

Parameters 112
AutoGenParams 114
Input Types 114
Return Types 114
Example 114
Description 114
Code 115
Comments 115
Example 115
Description 115
Code 115
Comments 115

New-SPExcelBlockedFileType 116
Syntax 116
Detailed Description 116
Parameters 116
AutoGenParams 119
Input Types 120
Return Types 120
Example 120
Description 120
Code 120
Comments 120

New-SPExcelDataConnectionLibrary 121
Syntax 121
Detailed Description 121
Parameters 121
AutoGenParams 124
Input Types 125
Return Types 125
Example 125
Description 125
Code 125
Comments 125

New-SPExcelDataProvider 126
Syntax 126
Detailed Description 126
Parameters 126
AutoGenParams 130
Input Types 130

Return Types 130
Example 130
Description 130
Code 130
Comments 130
Example 131
Description 131
Code 131
Comments 131

New-SPExcelFileLocation 132
Syntax 132
Detailed Description 132
Parameters 132
AutoGenParams 146
Input Types 148
Return Types 148
Example 148
Description 148
Code 148
Comments 148
Example 148
Description 148
Code 148
Comments 149

New-SPExcelServiceApplication 150
Syntax 150
Detailed Description 150
Parameters 150
AutoGenParams 159
Input Types 160
Return Types 160
Example 160
Description 160
Code 161
Comments 161
Example 161
Description 161
Code 161
Comments 161

New-SPExcelUserDefinedFunction 162

Syntax 162
Detailed Description 162
Parameters 162
AutoGenParams 165
Input Types 166
Return Types 166
Example 166
Description 166
Code 166
Comments 166
Example 166
Description 166
Code 166
Comments 166

Remove-SPExcelBlockedFileType 167
Syntax 167
Detailed Description 167
Parameters 167
AutoGenParams 170
Input Types 171
Return Types 171
Example 171
Description 171
Code 171
Comments 171

Remove-SPExcelDataConnectionLibrary 172
Syntax 172
Detailed Description 172
Parameters 172
AutoGenParams 175
Input Types 175
Return Types 175
Example 175
Description 175
Code 176
Comments 176
Example 176
Description 176
Code 176
Comments 176

Remove-SPExcelDataProvider 177
Syntax 177
Detailed Description 177
Parameters 177
AutoGenParams 179
Input Types 180
Return Types 180
Example 180
Description 180
Code 180
Comments 180
Example 180
Description 180
Code 181
Code 181
Comments 181

Remove-SPExcelFileLocation 182
Syntax 182
Detailed Description 182
Parameters 182
AutoGenParams 185
Input Types 186
Return Types 186
Example 186
Description 186
Code 186
Comments 186
Example 186
Description 186
Code 186
Comments 187

Remove-SPExcelUserDefinedFunction 188
Syntax 188
Detailed Description 188
Parameters 188
AutoGenParams 190
Input Types 191
Return Types 191
Example 191
Description 191

Code 191
Comments 191
Example 191
Description 191
Code 191
Comments 191

Set-SPExcelDataConnectionLibrary 192
Syntax 192
Detailed Description 192
Parameters 192
AutoGenParams 196
Input Types 196
Return Types 196
Example 196
Description 196
Code 196
Comments 197
Example 197
Description 197
Code 197
Comments 197

Set-SPExcelDataProvider 198
Syntax 198
Detailed Description 198
Parameters 198
AutoGenParams 201
Input Types 202
Return Types 202
Example 202
Description 202
Code 202
Comments 202
Example 202
Description 202
Code 202
Code 202
Code 203
Comments 203

Set-SPExcelFileLocation 204
Syntax 204

Detailed Description 204
Parameters 204
AutoGenParams 219
Input Types 220
Return Types 220
Example 220
Description 220
Code 220
Comments 220
Example 221
Description 221
Code 221
Comments 221

Set-SPExcelServiceApplication 222
Syntax 222
Detailed Description 222
Parameters 222
AutoGenParams 231
Input Types 232
Return Types 232
Example 232
Description 232
Code 232
Comments 232
Example 232
Description 232
Code 232
Comments 233

Set-SPExcelUserDefinedFunction 234
Syntax 234
Detailed Description 234
Parameters 234
AutoGenParams 237
Input Types 238
Return Types 238
Example 238
Description 238
Code 238
Comments 238
Example 238

Description 238
Code 238
Comments 238

Office Web Apps cmdlets 240

Get-SPOfficeWebAppsCache 242
Syntax 242
Detailed Description 242
Parameters 242
AutoGenParams 244
Input Types 245
Return Types 245
Example 245
Description 245
Code 245
Code 245
Code 245
Code 245
Comments 245
Example 245
Description 245
Code 245
Code 246
Code 246
Comments 246
Example 246
Description 246
Code 246
Code 246
Code 246
Comments 246

Get-SPOneNoteWebAppConfig 247
Syntax 247
Detailed Description 247
Parameters 247
AutoGenParams 249
Input Types 249
Return Types 249
Example 249
Description 249
Code 249

Code 249
Code 250
Comments 250

Get-SPPowerPointServiceApplication 251
Syntax 251
Detailed Description 251
Parameters 251
AutoGenParams 254
Input Types 254
Return Types 254
Example 254
Description 254
Code 254
Comments 255
Example 255
Description 255
Code 255
Comments 255

Get-SPPowerPointServiceApplicationProxy 256
Syntax 256
Detailed Description 256
Parameters 256
AutoGenParams 259
Input Types 259
Return Types 259
Example 259
Description 259
Code 259
Comments 259
Example 260
Description 260
Code 260
Comments 260

New-SPOfficeWebAppsCache 261
Syntax 261
Detailed Description 261
Parameters 261
AutoGenParams 264
Input Types 265
Return Types 265

Example 265
Description 265
Code 265
Code 265
Comments 265

New-SPPowerPointServiceApplication 266
Syntax 266
Detailed Description 266
Parameters 266
AutoGenParams 269
Input Types 270
Return Types 270
Example 270
Description 270
Code 270
Comments 270

New-SPPowerPointServiceApplicationProxy 271
Syntax 271
Detailed Description 271
Parameters 271
AutoGenParams 275
Input Types 275
Return Types 275
Example 275
Description 275
Code 275
Comments 275

New-SPWordViewingServiceApplication 277
Syntax 277
Detailed Description 277
Parameters 277
AutoGenParams 280
Input Types 280
Return Types 280
Example 280
Description 280
Code 280
Comments 280

New-SPWordViewingServiceApplicationProxy 282

Syntax 282
Detailed Description 282
Parameters 282
AutoGenParams 284
Input Types 285
Return Types 285
Example 285
Description 285
Code 285
Comments 285

Remove-SPOfficeWebAppsCache 286
Syntax 286
Detailed Description 286
Parameters 286
AutoGenParams 288
Input Types 289
Return Types 289
Example 289
Description 289
Code 289
Comments 289

Remove-SPPowerPointServiceApplication 290
Syntax 290
Detailed Description 290
Parameters 290
AutoGenParams 292
Input Types 293
Return Types 293
Example 293
Description 293
Code 293
Comments 293
Example 293
Description 293
Code 293
Comments 293

Remove-SPPowerPointServiceApplicationProxy 294
Syntax 294
Detailed Description 294
Parameters 294

AutoGenParams 297
Input Types 297
Return Types 297
Example 297
Description 297
Code 297
Comments 297
Example 297
Description 297
Code 298
Comments 298

Set-SPOfficeWebAppsCache 299
Syntax 299
Detailed Description 299
Parameters 299
AutoGenParams 303
Input Types 304
Return Types 304
Example 304
Description 304
Code 304
Comments 304

Set-SPOneNoteWebAppConfig 305
Syntax 305
Detailed Description 305
Parameters 305
AutoGenParams 309
Input Types 310
Return Types 310
Example 310
Description 310
Code 310
Comments 310
Example 310
Description 310
Code 310
Comments 311

Set-SPPowerPointServiceApplication 312
Syntax 312
Detailed Description 312

Parameters 312
AutoGenParams 318
Input Types 319
Return Types 319
Example 319
Description 319
Code 319
Comments 320
Example 320
Description 320
Code 320
Comments 320

Set-SPPowerPointServiceApplicationProxy 321
Syntax 321
Detailed Description 321
Parameters 321
AutoGenParams 326
Input Types 327
Return Types 327
Example 327
Description 327
Code 327
Comments 327
Example 327
Description 327
Code 328
Code 328
Comments 328

Set-SPWordViewingServiceApplication 329
Syntax 329
Detailed Description 329
Parameters 329
AutoGenParams 334
Input Types 335
Return Types 335
Example 335
Description 335
Code 336
Comments 336

Set-SPWordViewingServiceApplicationProxy 337

Syntax 337
Detailed Description 337
Parameters 337
AutoGenParams 339
Input Types 340
Return Types 340
Example 340
Description 340
Code 340
Comments 340

System Center Operations Manager knowledge articles (Office Web Apps) 341

Merge failures high (Microsoft OneNote Web App) 342

Request processing slow (Microsoft OneNote Web App) 343

PowerPoint document high download time (Microsoft PowerPoint Web App) 344

PowerPoint high number of queued conversion requests (Microsoft PowerPoint Web App) 345

PowerPoint high rate of "No Available Server" errors (Microsoft PowerPoint Web App) 347

PowerPoint high rate of cache write operations (Microsoft PowerPoint Web App) 349

PowerPoint rate of BroadcastGetData requests too high (Microsoft PowerPoint Web App) 351

Conformance statement A-level (Office Web Apps) 353
Conformance statement for configuration and administration experience at A-level 353

Conformance statement AA-level (Office Web Apps) 355
Conformance statement for core reader, author, and contributor experience at AA-level 355

Getting help

Every effort has been made to ensure the accuracy of this book. This content is also available online in the Office System TechNet Library, so if you run into problems you can check for updates at:

http://technet.microsoft.com/office

If you do not find your answer in our online content, you can send an e-mail message to the Microsoft Office System and Servers content team at:

itspdocs@microsoft.com

If your question is about Microsoft Office products, and not about the content of this book, please search the Microsoft Help and Support Center or the Microsoft Knowledge Base at:

http://support.microsoft.com

Understanding Office Web Apps (Installed on SharePoint 2010 Products)

Microsoft Office Web Apps is the online companion to Office Word, Excel, PowerPoint and OneNote applications that enables users regardless of their location to access documents and edit documents. Users can view, share, and work on documents with others online across personal computers, mobile phones, and the Web. Office Web Apps is available to users through Windows Live and to business customers with Microsoft Office 2010 volume licensing and document management solutions based on Microsoft SharePoint 2010 Products.

Note:
An appropriate device, Internet connection, and supported Internet Explorer, Firefox, or Safari browser are required. Some mobile functionality requires Office Mobile 2010, which is not included in Office 2010 applications, suites, or Web Apps. There are some differences between the features of Office Web Apps, Office Mobile 2010 and the Office 2010 applications.

Information provided in this article is intended for IT Pros planning to use Office Web Apps on SharePoint 2010 Products on-premise in their organizations. SharePoint 2010 Products in this article refers to Microsoft SharePoint Server 2010 and Microsoft SharePoint Foundation 2010 unless otherwise noted. New and updated content will be published on a regular basis.

In this article:

- Office Web Apps versions
- Integration with SharePoint 2010 Products
- Understanding the Office Web Apps user experience
- Understanding Office Web Apps architecture
- Understanding how Office Web Apps work
- Planning Office Web Apps in your organization
- Deploying Office Web Apps in your organization

Office Web Apps versions

Office Web Apps is available to consumers and businesses through:

Windows Live For consumers and small-business users, Office Web Apps is available on Windows Live as a free service. For more information, see Using Office Web Apps in Windows Live (*http://go.microsoft.com/fwlink/?LinkId=180798&clcid=0x409*).

On-premises Business customers licensed for Microsoft Office 2010 through a Volume Licensing program can run Office Web Apps on a server that runs Microsoft SharePoint Foundation 2010 or Microsoft SharePoint Server 2010. Information in this article relates to an on-premises Office Web Apps solution with SharePoint 2010 Products. Office Web Apps can be downloaded from the Microsoft Volume Licensing Service Center (*http://go.microsoft.com/fwlink/?LinkID=191841&clcid=0x409*).

Integration with SharePoint 2010 Products

Office Web Apps is tightly integrated with SharePoint 2010 Products. When you install Office Web Apps, the Office Web Apps Services are added to the list of SharePoint Services and the Office Web Apps Feature is added to the available SharePoint Features.

Office Web Apps services include the Word Viewing Service, PowerPoint Service, and Excel Calculation Services that are created and run within the context of SharePoint Services.

The Office Web Apps Feature and services integrate with SharePoint's robust enterprise content management capabilities to provide users the ability to access and work on your organization's documents from anywhere using a Web browser.

Understanding the Office Web Apps user experience

Viewing and editing Office documents

Office Web Apps gives users a browser-based viewing and editing experience by providing a representation of an Office document in the browser. When a user clicks on a document stored in a SharePoint document library, the document opens directly in the browser. The document appears in the browser similar to how it appears in the Office client application. The Web app also provides many of the same editing features as an Office client application.

Office Web Apps provides this representation of an Office Word document, PowerPoint presentation, Excel workbook, or OneNote notebook using native browser objects such as HTML, JavaScript, and images. Each document type is handled differently depending on the Office Web Apps services started and whether the Office Web Apps Feature is activated.

A document in the Word Web App, PowerPoint Web App, or Excel Web App can be edited in the browser or can be opened for editing in the associated Office client application. If while viewing or working in a Web app a user clicks the **Edit in Browser** button on the Home tab of the toolbar, the user can perform light editing tasks in the browser. A notebook in the OneNote Web App can be edited in the browser natively without having to click the **Edit in Browser** button or it can be opened for editing in the OneNote client application by clicking **Open in OneNote**.

If while in a Web app a user clicks the **Open in Word**, **Open in PowerPoint**, **Open in Excel**, or **Open in OneNote** button on the toolbar, the document will open in the associated Office client application if it is installed on the client computer.

Improving the user experience with Silverlight

Silverlight is a free plugin that can provide richer Web experiences for many browsers. The Silverlight plugin is not required to be installed on the client browser to use Office Web Apps. However, having the Silverlight plugin installed on the browser can provide the following benefits:

- When using the Word Web App on browsers with the Silverlight plugin installed, users can experience faster page loading, improved text fidelity at full zoom, ClearType tuner settings support, and improved accuracy in location of search string instances when using the find on this page feature.
- When using the PowerPoint Web App on browsers with the Silverlight plugin installed, users can experience faster page loading, animations will appear smoother than without, and presentation slides will scale with the browser window size.

Having Silverlight installed on the client browser does not provide any additional benefits in Excel Web App and OneNote Web App.

For more information about Silverlight, see http://www.microsoft.com/silverlight/. (*http://go.microsoft.com/fwlink/?LinkId=188454&clcid=0x409*)

Configuring the default open behavior for browser enabled documents

SharePoint 2010 Products uses the **Default open behavior for browser enabled documents** setting to determine how a document is opened when a user clicks on a document in SharePoint. By default, this option is set to open documents in their associated client application. When Office Web Apps is installed, setup will change this setting to open documents in the browser. However, if the Office Web Apps Services and Feature have not been activated, when a user clicks on a document, the user may receive an error message that indicates that the service cannot be found. The error message includes a button that the user clicks to open the document in the associated Office client application.

For information about how to configure the default open behavior setting in SharePoint, see Configure the default open behavior for browser-enabled documents (Office Web Apps).

Understanding Office Web Apps architecture

Office Web Apps is made up of multiple server components that create and provide renditions of Word documents, PowerPoint presentations, Excel workbooks, and OneNote notebooks stored in SharePoint for viewing and editing in a browser. Each Web app uses some or all these server components depending on content type, user changes, and feature activation. Office Web Apps server components include the following:

Services When you install Office Web Apps on a server, the Word Viewing Service, PowerPoint Service, and Excel Calculation Services are created in SharePoint Services. Each of these services acts as a *service instance* for each Office Web App on a stand-alone server or a service instance for each Office Web App on each server in a server farm.

Only one service instance for each Office Web App can exist on a single server, but multiple service instances can exist for each Office Web App in a server farm. Each service instance provides a location where a *service application* is run.

The Office Web Apps service applications are middle-tier components that manage requests from the Web front-end components, cache renditions, store, calculate, and render documents for consumption by the Web front-end. Not all of the Office Web Apps have a service application. The OneNote Web App for example enables the user to edit .one documents using only the OneNote.aspx without needing a service application. Each service application operates within a service instance property.

Administrators can load-balance services in a server farm by choosing to run service instances on particular servers and not on others. Load balancing can be especially useful in environments where one kind of Web app might be used more than others. Administrators create and start service instances when they deploy Office Web Apps. The service applications will run in their associated service instances automatically. After Office Web Apps is installed on at least one server and the service applications are running in a service instance, administrators can configure service application settings by using SharePoint Central Administration and by using Windows PowerShell.

Each service uses a service application *proxy* in the SharePoint service application proxy group. The proxies provide the location of a service instance within the farm and manage sessions between the Web front-end components and the service applications. The Word Viewing service application, PowerPoint service application, and Excel Calculation Services each has service application proxies in the SharePoint service application proxy group.

Worker processes The Word Viewing service application and the PowerPoint service application uses worker processes to convert documents and presentations into a series of PNG images or into XAML (if Silverlight is installed), and temporarily stores output locally on-disk. Administrators can configure worker process settings to optimize performance by using SharePoint Central Administration and by using Windows PowerShell.

Web front-end components Office Web Apps includes a series of .ASPX, .ASHX, and JavaScript files that run on the front-end SharePoint servers. These are ASP.NET Web applications that render markup for the browser and respond to and manage client events (state) that run on the server. Office Web Apps also uses a series of handlers that run as part of the Web front-end to support rendering HTML and servicing requests made in JavaScript.

Office Web Apps Cache Word Web App and the PowerPoint Web App store document renditions in a cache. Renditions in the cache are then used for future requests of a view of the same document. Using the cache can reduce use of too many system resources to create a rendition. When you deploy Office Web Apps, by default the cache is created as a site collection in the SPContentDatabase. Like other site collections, the cache can be moved to another database depending on performance or backup needs. Cache size and expiration period can also be configured depending on usage and performance requirements. Office Web Apps also includes two timer jobs associated with the cache. Managing the cache can be done by using SharePoint Central Administration and by using Windows PowerShell. For more information about how to manage the cache, see Manage the Office Web Apps cache.

Understanding how Office Web Apps work

Each Web app uses some or all of the components described in the previous section to provide a viewable Office document in the browser and an editable document in either the browser or the client application.

Viewing and editing Word documents using the Word Web App

For viewing Word document types, a combination of images, HTML, and Java Script are used to render a document viewable by the user. If Silverlight is installed on the client browser, some images will be displayed using XAML.

When a user clicks on a Word document in SharePoint, and the **Default open behavior for browser enabled documents** setting in SharePoint is set to open in the browser, WordViewer.aspx will check the Office Web Apps cache to determine whether a rendition of the document is already available.

If a rendition of the document is available in the cache, Word Web App front-end components will display the document in *view* mode in the browser. If a rendition of the document is not available in the cache, the front-end component will call the Word Viewing service application to render the document. The Word Viewing service application then initializes and manages a worker process to render the document by converting it into a series of PNG images or XAML (if Silverlight is installed). The Word Viewing service application then returns the rendered document to WordViewer.aspx and to the cache. WordViewer.aspx then displays the document in the browser in view mode. The Word Viewing service application can initialize and manage multiple worker processes, each process dedicated to a single session.

If while in view mode, the user clicks the **Edit in Browser** button on the Microsoft Word Web App toolbar, WordViewer.aspx will navigate to the Word Editor (edit mode). The Word Editor is a Web front-end component that creates the browser based editing surface that enables the user to work on the document without loss of fidelity.

If while in view mode or edit mode the user clicks the **Open in Word** button on the Microsoft Word Web App toolbar, the document will open in the Word client application, if it is installed on the client computer.

Viewing and editing PowerPoint presentations using the PowerPoint Web App

For viewing PowerPoint presentation types, a combination of images, HTML, and JavaScript is used to render a presentation viewable by the user. If Silverlight is installed on the client browser, some images will be displayed using XAML.

When a user clicks on a PowerPoint presentation in SharePoint, and the **Default open behavior for browser enabled documents** setting in SharePoint is set to open in the browser, the front-end components will check the Office Web Apps cache to determine whether a rendition of the presentation is already available.

If a rendition of the presentation is available in the cache, PowerPoint Web App front-end components will display the presentation in view mode in the browser. If a rendition of the presentation is not available in the cache, PowerPointFrame.aspx will call the PowerPoint service application to render the presentation. The PowerPoint service application then initializes and manages a worker process to render the document by converting it into a series of PNG images or XAML (if Silverlight is installed). The PowerPoint service application then returns the rendered document to PowerPointFrame.aspx and to the cache. PowerPointFrame.aspx then displays the document in the browser in view mode. The PowerPoint service application can initialize and manage multiple worker processes, each process dedicated to a single editing session.

If while in view mode, the user clicks the **Edit in Browser** button on the Microsoft PowerPoint Web App toolbar, PowerPointFrame.aspx will move to PowerPoint Editor (edit mode). PowerPoint Editor is a Web front-end component that creates the browser based editing surface that enables the user to work on the document without loss of fidelity. PowerPointFrame.aspx will send updates to the PowerPoint service application which services editing requests, manages a single worker process for each editing session, manages re-rendering, auto saves, and returns the updated rendition to PowerPointFrame.aspx. PowerPointFrame.aspx then displays the updated presentation rendition in the browser.

If while in view mode or edit mode, the user clicks the **Open in PowerPoint** button on the Microsoft PowerPoint Web App toolbar, the presentation will open in the PowerPoint client application, if it is installed on the client computer.

PowerPoint Broadcast Slide Show presentations using the PowerPoint Web App

When Office Web Apps is installed and the PowerPoint service application is created, the PowerPoint Broadcast Service and a default broadcast site are also created. When a user (presenter) opens a presentation in SharePoint using PowerPoint 2010, the user can then click **Broadcast Slide Show** on the Slide Show tab to start a Broadcast Slide Show presentation. When the user clicks **Broadcast Slide Show**, the presentation will automatically be uploaded to the Broadcast site, and the Broadcast Service starts a broadcast session.

The PowerPoint service application will request the presentation from the Broadcast site. The PowerPoint service application then instructs worker processes to convert the presentation into a series of images or XAML (if Silverlight is installed), and temporarily store output locally on disk. The PowerPoint service application then creates an attendee URL and returns the URL and broadcast session information to PowerPointFrame.aspx. PowerPointFrame.aspx then returns the attendee URL to the presenter’s browser; the presenter can then send the URL in e-mail or post the link for attendees.

During the presentation session, PowerPointFrame.aspx determines the presenter's current location in the presentation, retrieves the current slide from the cache, and renders the updated slide in the browser. Attendee browsers poll PowerPoint.aspx once per second to check for updates in the slide show presentation.

Viewing and editing Excel workbooks using the Excel Web App

The Excel Web App uses DHTML and JavaScript to render and enable editing an Excel workbook without the need for downloading ActiveX controls to the browser running on the client computer. The Excel Web App uses Excel Calculation Services to load the workbook, calculate in full fidelity, refresh external data if it is needed, and maintain the session.

When a user clicks on an Excel workbook in SharePoint, and the **Default open behavior for browser enabled documents** setting in SharePoint is set to open in the browser, the Excel Web App will use the Excel Services Application Web Service Application Proxy to direct calls to and from Excel Calculation Services.

If while in view mode, the user clicks the **Edit in Browser** button on the Microsoft Excel Web App toolbar, the user can then edit the workbook in the browser. The browser will send updates to the Excel Calculation Services which will service those editing requests and return updated results to the browser.

If while in view mode or edit mode, the user clicks the **Open in Excel** button on the Microsoft Excel Web App toolbar, the workbook will open in the Excel client application, if it is installed on the client computer.

Viewing and editing OneNote notebooks using the OneNote Web App

The OneNote Web App uses a combination of HTML and JavaScript to create a rendition of a notebook that can be viewed and edited by the user.

When a user clicks on a OneNote notebook in SharePoint, and the **Default open behavior for browser enabled documents** setting in SharePoint is set to open in the browser, OneNote.aspx will display the notebook in the browser. The user can edit the notebook in the browser.

If the user clicks the **Open in OneNote** button on the Microsoft OneNote Web App toolbar, the notebook will open in the OneNote client application, if it is installed on the client computer.

Planning Office Web Apps in your organization

How you deploy Office Web Apps will depend on how users in your organization will use the Web apps to view and edit Office documents in your SharePoint environment. Your Office Web Apps deployment will also depend on your SharePoint server configuration and hardware resources.

In smaller organizations, all of the Office Web Apps components can be deployed on a single SharePoint 2010 Products server. Although this kind of solution does have limitations, as your needs change, you can later expand your Office Web Apps and SharePoint solution to a multiple server farm.

For medium and large-scale organizations, you can install Office Web Apps on multiple servers in a SharePoint 2010 Products server farm. You can optimize performance not only for users viewing and editing Office documents using Office Web Apps, but also for other SharePoint services that are running on the same server farm.

For more information about how to plan an Office Web Apps solution for your organization, see Planning Office Web Apps (Installed on SharePoint 2010 Products).

Deploying Office Web Apps in your organization

Deploying Office Web Apps involves three primary phases: Installing Office Web Apps by running setup, activating the services, and then activating the Office Web Apps Feature.

Office Web Apps can be installed on a SharePoint 2010 Products stand-alone server or server farm. When deploying on a server farm, Office Web Apps must be installed on every server in the farm, however, the services do not have to be activated on every server in the farm. You can choose to balance load by activating the Office Web Apps services on particular servers in the farm. You can also optimize performance by configuring the Office Web Apps cache and worker processes to your particular requirements.

For more information about how to deploy Office Web Apps on-premise in your organization, see Deploy Office Web Apps (Installed on SharePoint 2010 Products).

Summary

Information in this article provides an understanding of how Office Web Apps installed on SharePoint 2010 Products work. This information can help you when planning Office Web Apps in your organization. The next step in your on-premises Office Web Apps solution is to plan your SharePoint and Office Web Apps server configuration. For more information, see Planning Office Web Apps (Installed on SharePoint 2010 Products).

Planning Office Web Apps (Installed on SharePoint 2010 Products)

This article discusses planning considerations for Microsoft Office Web Apps. Office Web Apps is the online companion to Word, Excel, PowerPoint and OneNote applications that enables users regardless of their location to access and edit documents. Users can view, share, and work on documents with others online across personal computers, mobile telephones, and the Web. Office Web Apps is available to users through Windows Live and to business customers with Microsoft Office 2010 volume licensing and document management solutions that are based on Microsoft SharePoint 2010 Products.

Information provided in this article is intended for IT Pros planning to use Office Web Apps on SharePoint 2010 Products on-premises in their organizations. Before planning your Office Web Apps solution, it is recommended you first read Understanding Office Web Apps (Installed on SharePoint 2010 Products).

SharePoint 2010 Products in this article refers to Microsoft SharePoint Server 2010 and Microsoft SharePoint Foundation 2010 unless otherwise noted. New and updated content will be published on a regular basis.

In this article:

- Software prerequisites
- Browser support
- Mobile device support
- Plan server topology
- Plan default open behavior for documents
- Plan Office Web Apps Feature activation
- Optimizing performance
- Scenarios that require the Office client applications
- Creating documents from inside a SharePoint site
- Summary

Software prerequisites

To deploy Office Web Apps in your organization requires one of the following SharePoint versions:

- SharePoint Server 2010 Standard edition
- SharePoint Server 2010 Enterprise edition
- SharePoint Foundation 2010

Office Web Apps cannot be installed on SharePoint trial editions or SharePoint Server 2010 for Internet Sites.

Important:
Do not install Office Web Apps on a server that is configured as a domain controller. Instead, run OfficeWebAccess-2nd_CurrentVer and the domain controller on separate servers. If you only have one computer, consider running Office Web Apps and the domain controller configuration as separate virtual server instances on a single machine, using Hyper-V.

For information about the different versions of SharePoint and how to implement SharePoint in your organization, see Microsoft SharePoint 2010 Products online at http://go.microsoft.com/fwlink/?LinkId=189311&clcid=0x409 (*http://go.microsoft.com/fwlink/?LinkId=189311&clcid=0x409*).

Browser support

Supported browsers

When planning your Office Web Apps solution, for users who are working either via intranet or extranet, all client computer Web browsers must meet minimum version requirements. Office Web Apps is supported in the following browsers:

- Internet Explorer 7.0
- Internet Explorer 8.0
- Firefox 3. *x*
- Safari 4. *x* for the Macintosh
- Google Chrome

Known issues

If while using an Office Web App, a user clicks the **Open in Word**, **Open in PowerPoint**, **Open in Excel**, or **Open in OneNote** button on the toolbar, a message may be displayed stating the file cannot be opened in the client application. This can be caused by one or more of the following conditions:

- The user is running a 64-bit version of Internet Explorer.
- The ActiveX control necessary for opening the file in the Office client application not being installed or not running in the browser.
- Microsoft Office is not installed on the client computer.
- The version of Office that is installed is earlier than Office 2003
- Security settings in Internet Explorer prevent ActiveX controls from running.
- The browser does not support ActiveX.
- The client browser is Firefox, but the plugin that enables Firefox to open Office documents is not installed.

Mobile device support

Users in your organization can use browser-enabled cell phones and mobile devices to read Excel, PowerPoint, and Word documents stored on a SharePoint server running Office Web Apps, if views and content are enabled for mobile access published externally outside a firewall.

The following devices provide mobile support for Office Web Apps in SharePoint:

- Windows Mobile
- BlackBerry
- iPhone, iPod Touch
- Nokia S60
- Japan feature telephones, including NTT DOCOMO, SoftBank and au by KDDI

To access pages using mobile devices, the URL is the identical to that used by browsers run on client computers. However, this can vary depending on the configuration and presence of Web proxies. Users can click E-mail a Link on the Share & Track tab of the SharePoint ribbon to receive the address in e-mail on an SMTP service enabled SharePoint server.

Plan server topology

How you deploy Office Web Apps in your organization will depend largely on how your SharePoint installation is configured. If you will be installing Office Web Apps on an existing SharePoint server farm, it is important to plan for, test, and monitor how Office Web Apps will affect other aspects of your SharePoint server performance. When installing Office Web Apps in a server farm, it must be installed on every server in that farm. However, the Office Web Apps services do not have to be activated on every server in the farm.

Some of the information provided in this section was derived from the Estimate performance and capacity requirements for Office Web Apps (*http://go.microsoft.com/fwlink/?LinkID=191156&clcid=0x409*) (OfficeWebAppsCapacityPlanningDoc.docx) whitepaper available at Microsoft Download Center. In addition to providing performance and capacity planning information, this whitepaper provides detailed testing information performed by Microsoft, as well as additional recommendations and troubleshooting information.

For the purposes of planning an Office Web Apps server configuration, components can be divided among the following tiers:

- **Web front-end** Office Web Apps front-end components run within the SharePoint Web front-end. Office Web Apps front-end components include a series of ASP.NET Web applications that render markup for the browser, and respond to and manage client events (state) that run on the server.
- **Middle-tier** Office Web Apps middle-tier components include the Web apps services. These services run within the same context as other SharePoint services.

- **Back-end** Office Web Apps will create and use a cache to render documents viewable by the user. The cache exists as a site collection in the SPContentDatabase as part of the SharePoint back-end. Cache performance efficiency will have a direct effect on rendered page load times.

To determine what an optimal deployment should look like depends heavily on expected user count, how heavy the usage is expected to be, and the type of usage. As a starting point for your own deployments, consider the following guidelines:

Application servers and front-end Web servers hardware

Processor(s)	2 quad core @2.33 GHz
RAM	16 GB
Operating System	Windows Server 2008, 64 bit
Size of the SharePoint drive	3x146GB 15K SAS (3 RAID 1 Disks) Disk 1: operating system Disk 2: Swap and BLOB Cache Disk 3: Logs and Temp directory
Number of NICs	2
NIC Speed	1 GBt
Authentication	NTLM
Load balancer type	Hardware load balancing

For specific information about minimum and recommended system requirements for both front-end Web server and application servers, see Hardware and software requirements (SharePoint Server 2010) (*http://technet.microsoft.com/library/4d88c402-24f2-449b-86a6-6e7afcfec0cd(Office.14).aspx*).

Office Web Apps server topology

Average number of daily unique visitors	Average concurrent users	Recommended topology
100	10	1 front-end Web server, 1 application server
1000	30	2 front-end Web servers, 2 application servers
10000	300	4 front-end Web servers, 3 application servers

For heavy usage of the PowerPoint Broadcast feature, a separate server farm is recommended. For more information, see Configure Broadcast Slide Show performance (Office Web Apps).

Worksheet action
The Office Web Apps worksheet (Office Web Apps Worksheet) (*http://go.microsoft.com/fwlink/?LinkId=180808&clcid=0x409*) is provided to record your Office Web Apps planning decisions. Use this worksheet to record the number of Web front-end and middle-tier servers that are required.

Plan default open behavior for documents

SharePoint 2010 Products use a default open behavior setting to determine how a document is opened when a user clicks on a document in SharePoint. By default, when Office Web Apps is installed, setup will configure SharePoint registration so that Office documents are opened in the browser with the associated Web app. You can override the default open behavior at the site collection and document library level before or after Office Web Apps is installed by using Central Administration or by using Windows PowerShell.

For information about how to configure the default open behavior setting in SharePoint, see Configure the default open behavior for browser-enabled documents (Office Web Apps).

Worksheet action
The Office Web Apps worksheet (Office Web Apps Worksheet) (*http://go.microsoft.com/fwlink/?LinkId=180808&clcid=0x409*) is provided to record your Office Web Apps planning decisions. Use this worksheet to record the default open behavior for site collections and libraries.

Plan Office Web Apps Feature activation

The Office Web Apps Feature can be activated at the SharePoint site collection level. The feature should be activated on every site collection for which any of the Web apps will be available. Site collections created after Office Web Apps is installed will have the feature activated automatically. Typically, you activate the feature on existing site collections during the deployment phase after you run setup and activate the services; however, in some cases, you can decide to later activate the feature only on particular site collections and deactivate on others. To provide a consistent user experience, we strongly recommend the feature be activated on all site collections.

If the feature is activated for a site collection, and the Default open behavior for browser-enabled documents setting in SharePoint is set to open in browser (SharePoint OpenInClient feature is disabled for the site collection), when a user clicks on a document in a document library, the document will open in view mode in the browser. The user can edit the document in the browser by clicking the **Edit in Browser** button on the toolbar. If the user clicks on the Edit Control Block (ECB) for a document, the **View in Browser** and **Edit in Browser** menu items will appear together with **Edit in Microsoft Word**, **Edit in Microsoft PowerPoint**, **Edit in Microsoft Excel**, or **Edit in Microsoft OneNote** menu items.

If the feature is not activated for a site collection, and the Default open behavior for browser-enabled documents setting in SharePoint is set to open in browser, when a user clicks on a document in a document library in that site collection, the document will open in view mode in the browser. The user can edit the document in the browser by clicking the **Edit in Browser** button on the toolbar. However, if the user clicks on the ECB for a Word, PowerPoint, or OneNote document, the **View in Browser** and **Edit in Browser** menu items will not appear.

For more information about how to activate the Office Web Apps Feature for a site collection, see Deploy Office Web Apps (Installed on SharePoint 2010 Products) and Activate the Office Web Apps Feature on site collections.

Worksheet action
The Office Web Apps worksheet (Office Web Apps Worksheet) (*http://go.microsoft.com/fwlink/?LinkId=180808&clcid=0x409*) is provided to record your Office Web Apps planning decisions. Use this worksheet to record which site collections will have the Office Web Apps Feature activated.

Optimizing performance

The most significant effect on performance is determined by server topology. Provided usage requirements are within server topology guidelines, there are additional settings that may affect performance you can configure to suit your unique requirements:

- You can set the maximum number of worker processes that are permitted to service requests for the Word Viewing service application and the PowerPoint service application. Configuring this setting can allow for a maximum number of service requests without diminishing worker processes from other applications. For more information, see Configure Word Viewing service settings, and Configure PowerPoint service application settings.
- By default, the maximum upload size for a single document in SharePoint is 50 mb. Administrators can increase the Maximum Upload Size setting for the Web application (SharePoint Web application General Settings) to allow for extremely large documents. To support rendering extremely large Word documents by the Word Web App, you can increase the maximum file rendering time (MaxRenderingLifetimeInSeconds) setting for the Word Viewing service application by using Windows PowerShell. For more information, see Configure Word Viewing service settings.
- You can reduce the resource demands on databases by the Word Viewing service application and the PowerPoint service application by configuring the cache size, expiration period, and cache location. For more information, see Manage the Office Web Apps cache.

The Estimate performance and capacity requirements for Office Web Apps (*http://go.microsoft.com/fwlink/?LinkID=191156&clcid=0x409*) (OfficeWebAppsCapacityPlanningDoc.docx) whitepaper available at Microsoft Download Center provides additional information on how to monitor performance.

For more information about performance and capacity planning for SharePoint 2010 Products, see Plan for server farms and environments (SharePoint Server 2010) (*http://technet.microsoft.com/library/fd081152-5fdd-4e47-9afb-c887fbf1ce22(Office.14).aspx*) and Server farm and environment planning (SharePoint Foundation 2010) (*http://technet.microsoft.com/library/a8e97903-c472-4c13-a1e1-2c075b2f8585(Office.14).aspx*).

Scenarios that require the Office client applications

Office Web Apps are online companions to Word, Excel, PowerPoint and OneNote. Following is a list of scenarios that cannot be performed in Office Web Apps alone. When you want to use features described in these scenarios, you can use the Office Web Apps command for opening the document in their Office client application (**Open in Word**, **Open in Excel**, **Open in PowerPoint**, or **Open in OneNote**). When you save the document, it is updated in the SharePoint library where it was opened.

Note that most items that cannot be created in Office Web Apps can be displayed in Office Web Apps. For example, comments and tracked changes are visible in Word Web App but they must be created or edited in the Word client application.

- **Editing documents**
  - Protecting documents by using Information Rights Management (IRM).

    IRM is not available in Office Web Apps. If you must protect confidential or sensitive information, use the full Office client applications.
  - Copying formatting inside a document.

    The format painter is not available in Office Web Apps. Reapply formatting manually, or use the Office client applications.
  - Inserting complex shapes or symbols in documents (textboxes and shapes, equations, dates and times).

    Complex shapes and symbols are not available in the editing interfaces in Office Web Apps. If you have to insert these shapes or symbols, use the Office client applications. Note that you can view these shapes and symbols in Office Web Apps.
  - Looking up synonyms using a thesaurus or translating words.

    Thesaurus and translation features are not available in Office Web Apps. If you want to use these features, use the Office client applications.
  - Drawing images by using Ink in a OneNote notebook.

    Inking is not available in OneNote Web App. Use the Office client version of OneNote if you want to use Ink. Note that you can still use OneNote Web App to edit other types of content within the notebook.
  - Embedding media or other files into a OneNote notebook.

    If you are using OneNote notebooks as repositories for media or other embedded files, you must use the OneNote client to access the embedded files. Note that you can still use OneNote Web App to edit other types of content within the notebook.

- Editing documents and using track changes to mark revisions.

  Tracking changes in a document is not available in Word Web App. Use the Word client application if you want to track specific revisions to a document. Note that you can view tracked changes in Word Web App.
- Real time co-authoring scenarios are different for Office Web Apps:
  - For Word and PowerPoint documents, use the Office 2010 client applications for co-authoring, and be sure that no one is editing the document at the same time in Word Web App or PowerPoint Web App.
  - For Excel workbooks, use the Excel Web App for co-authoring and be sure that no one is editing the document at the same time in the Excel client.
  - For OneNote notebooks, you can use either OneNote 2010 or OneNote Web App for co-authoring.
- Editing Word objects, such as SmartArt or document headers.

  Some objects, such as SmartArt and document headers and footers, are not available for editing in Office Web Apps. Use the Office client applications to edit these objects. Note that you can view these objects in Office Web Apps.
- Using macros in Word, Excel and PowerPoint documents.

  Macros are not enabled in Office Web Apps. Use the Office client applications if you use macros in Word, Excel and PowerPoint documents.
- Updating external tables or query tables in Excel.

  You cannot refresh or update data in external tables and queries from Office Web Apps. Use the Office client applications or Excel Services in SharePoint to update the data.
- Checking the spelling in Excel and PowerPoint documents.

  Spell check is available only for Word documents or OneNote notebooks in Office Web Apps. Use the Office client applications to check the spelling in other document types.

- **Mobile devices**
  - Editing documents on a mobile device.

    From most smartphone browsers, you can view Word, Excel, and PowerPoint documents in Office Web Apps. However, you cannot edit documents. For more information about supported browsers, see the Browser support section.
  - Viewing OneNote documents on a device that is not a Windows Phone 7 device or an iPhone.

    With OneNote Mobile for Windows Phone 7 and for iPhones, you can view and edit OneNote notebooks. OneNote Mobile is not available on other devices at this time.

Creating documents from inside a SharePoint site

When you create files from inside a SharePoint site by using the browser, Microsoft SharePoint Server uses the Office client applications if they are installed. For more information, see the MSDN blog post: Using Office Web Apps to Create New Files (*http://go.microsoft.com/fwlink/?LinkId=207426&clcid=0x409*).

Summary

Information provided in this article can help you when planning an Office Web Apps solution for your organization. The next step in your on-premises Office Web Apps solution is to deploy Office Web Apps on a stand-alone SharePoint server or a SharePoint server farm. For more information, see Deploy Office Web Apps (Installed on SharePoint 2010 Products).

Deploy Office Web Apps (Installed on SharePoint 2010 Products)

This article discusses how to deploy Microsoft Office Web Apps. Office Web Apps is the online companion to Office Word, Excel, PowerPoint and OneNote applications that enables users to access documents from anywhere. Users can view, share, and work on documents with other users online across personal computers, mobile telephones, and the Web. Office Web Apps is available to users through Windows Live and to business customers with Microsoft Office 2010 volume licensing and document management solutions that are based on Microsoft SharePoint 2010 Products.

Note:
An appropriate device, Internet connection, and supported Internet Explorer, Firefox, or Safari browser are required. Some mobile functionality requires Office Mobile 2010, which is not included in Office 2010 applications, suites, or Office Web Apps. There are some differences between the features of Office Web Apps, Office Mobile 2010 and the Office 2010 applications.

Information provided in this article is intended for IT Pros planning to deploy Office Web Apps on SharePoint 2010 Products on-premises in their organizations. SharePoint 2010 Products in this article refers to Microsoft SharePoint Server 2010 and Microsoft SharePoint Foundation 2010 unless otherwise noted. This article does not provide guidance or instructions about how to deploy SharePoint 2010 Products. New and updated content will be published on a regular basis.

It is recommended you read Understanding Office Web Apps (Installed on SharePoint 2010 Products) and Planning Office Web Apps (Installed on SharePoint 2010 Products) available on Microsoft TechNet (*http://go.microsoft.com/fwlink/?LinkID=78003&clcid=0x409*) before deploying Office Web Apps in your organization. Office Web Apps can be downloaded from the Microsoft Volume Licensing Service Center (*http://go.microsoft.com/fwlink/?LinkID=191841&clcid=0x409*) on the Office Professional Plus 2010 32Bit or Office Standard 2010 32Bit site.

In this article:

- Understanding Office Web Apps deployment
- Additional configuration (optional)
- Troubleshooting

In addition to the previous sections, one of the following sections will apply to your specific Office Web Apps deployment:

- Install and configure Office Web Apps on an existing stand-alone SharePoint server
- Install and configure Office Web Apps on a new stand-alone SharePoint server
- Install and configure Office Web Apps on an existing SharePoint server farm
- Install and configure Office Web Apps on a new SharePoint server farm

Understanding Office Web Apps deployment

For both stand-alone SharePoint servers and SharePoint server farms, deploying Office Web Apps involves three primary phases:

Running setup and PSConfig Tasks include running Setup.exe and SharePoint Products and Technologies Post Setup and Configuration Wizard (PSConfig) on a stand-alone SharePoint server or each server in a SharePoint server farm. Running Setup.exe installs Office Web Apps files and components on a server. Running PSConfig is required as part of Office Web Apps setup in order to register the Office Web Apps services and, depending on the SharePoint installation type, start the service instances, create the service applications and service application proxies, and activate the Office Web Apps Feature.

Activating the Office Web Apps services Includes starting the service instances, and creating the service applications and service application proxies. Whether you must activate the services will depend on the state of SharePoint and whether PSconfig and the SharePoint Farm Configuration Wizard have previously been run.

Activating the Office Web Apps Feature Includes activating the Office Web Apps Feature on all existing SharePoint site collections where the Office Web Apps should be available. If PSconfig or the SharePoint Farm Configuration Wizard have been run before installing Office Web Apps, at least one site collection will exist. The feature will be activated automatically for new site collections created after Office Web Apps is installed.

Note:

Some procedures in this article require using Windows PowerShell or SharePoint 2010 Management Shell to run cmdlets. To run SharePoint 2010 cmdlets in Windows PowerShell, you must add the Microsoft.SharePoint.PowerShell snap-in by using the **Add-PSSnapin** cmdlet, or you can run the SharePoint 2010 cmdlets by using SharePoint 2010 Management Shell. By default, the Microsoft.SharePoint.PowerShell snap-in is already registered, and the snap-in is added to the SharePoint 2010 Management Shell. To run SharePoint 2010 cmdlets, you must be a member of the SharePoint_Shell_Access role on the configuration database and a member of the WSS_ADMIN_WPG local group on the computer where SharePoint 2010 Products is installed. To run scripts in Windows PowerShell or SharePoint 2010 Management Shell, you must set the execution policy by using the **set-executionpolicy** cmdlet, using the **unrestricted** parameter. For more information about the **Add-PSSnapin** cmdlet, see Add-PSSnapin (*http://go.microsoft.com/fwlink/?LinkId=188450&clcid=0x409*). For more information about how to use Windows PowerShell with SharePoint 2010 Products, see SharePoint 2010 Products administration by using Windows PowerShell.

Install and configure Office Web Apps on an existing stand-alone SharePoint server

This section applies only if you are installing Office Web Apps on an existing SharePoint server and PSConfig was previously run as part of SharePoint setup.

Caution:
When you run Setup.exe, Office Web Apps setup configures the default open behavior for browser-enabled documents in SharePoint to open documents in the browser. If Office Web Apps setup was run, but the Office Web Apps Services and Feature has not yet been activated, a user may get a broken link when opening a document in the browser. When deploying Office Web Apps on a live production server farm, to prevent broken links to documents while completing additional deployment tasks after running setup, we recommend you enable the OpenInClient feature on existing site collections before running setup. For more information, see Additional configuration (optional).

Run Office Web Apps setup

Complete this task to install Office Web Apps components and files on a server.

To run Office Web Apps setup

1. From the root folder, run **Setup.exe**.
2. On the **Enter your Product Key** page, enter your product key, and then click **Continue**.
3. On the **Choose a file location** page, click **Install Now** to install to the default location. To install to a different location, specify the location that you want to install to and then click **Install Now**.
4. When Setup finishes, a dialog box prompts you to complete the configuration of your server. Be sure that the **Run the SharePoint Products and Technologies Configuration Wizard now** check box is selected and then click **Close** to start PSConfig.

Run PSConfig to register the services

Complete this task to register the Office Web Apps services on the SharePoint server.

To run PSConfig to register the services

1. If you left the **Run the SharePoint Products and Technologies Configuration Wizard now** check box selected in the previous step, on the PSconfig **Welcome to SharePoint Products** page, click **Next**.
2. In the dialog box that notifies you that some services might have to be restarted or reset during configuration, click **Yes**.
3. On the **Configuration Successful** page, click **Finish**. Your new SharePoint site opens.

Start the service instances

A service instance provides the physical location for a service application. You must start the service instances before you create the service applications and the service application proxies. You can start the service instances by using SharePoint Central Administration or by using Windows PowerShell.

Procedures in this task start the service instances on the server specified.

To start the service instances by using Central Administration

1. Click **Start**, point to **All Programs**, **Microsoft SharePoint 2010 Products**, and then **SharePoint 2010 Central Administration**.
2. On the SharePoint Central Administration home page, in **System Settings**, click **Manage services on this server**.
3. On the Services on server:<servername>page, start **Excel Calculation Services**, **Word Viewing Service**, and **PowerPoint Service**. The OneNote Web App does not use a SharePoint service.

To start the service instances by using Windows PowerShell

1. Using Notepad, open a new text file, and then copy and paste the following script into the file.

```
$machinesToActivate = @("<servername>")

$serviceInstanceNames = @("Word Viewing Service", "PowerPoint Service", "Excel
Calculation Services")

foreach ($machine in $machinesToActivate)

{

  foreach ($serviceInstance in $serviceInstanceNames)

  {

     $serviceID = $(Get-SPServiceInstance | where {$_.TypeName -match
$serviceInstance} | where {$_.Server -match "SPServer Name="+$machine}).ID

     Start-SPServiceInstance -Identity $serviceID

  }

}
```

2. Specify the following parameters:

Parameter	Value
$machinesToActivate	Server name

3. Save the file with a **.ps1** file name extension to a folder where you run scripts (typically C:\scripts).
4. From the Windows PowerShell command prompt (that is, PS C:\>), type the following command and press ENTER:

   C:\<path>\<filename>.ps1

Create the service applications and the service application proxies

After the service instances have been started, the service applications and the service application proxies which connect the SharePoint Web front-ends to the service applications must be created. You can create the service applications and the service application proxies for the Word, PowerPoint, and Excel Web apps by using Central Administration or by using Windows PowerShell. The OneNote Web App does not require a service application. After created, the service applications will run on started service instances automatically. Procedures in this task will create the Office Web Apps service applications and service application proxies for the Word, PowerPoint, and Excel Web apps. This task can only be completed after you have started the service instances.

To create the service applications and the service application proxies by using Central Administration

1. Click **Start**, point to **All Programs**, **Microsoft SharePoint 2010 Products**, and then **SharePoint 2010 Central Administration**.
2. On the SharePoint Central Administration home page, in **Application Management**, click **Manage service applications**.
3. On the Service Applications page, click **New**, and then click **Word Viewing Service**.
4. In the Word Viewing Service Application dialog box, in **Name**, type **Word Viewing Service Application**. In **Application Pool**, select **Use existing application pool**, and then in the listbox, select **SharePoint Web Services Default**. In **Add to default proxy list**, verify **Add this service application's proxy to the farm's default proxy list** is selected (default), and then click **OK**.
5. On the Service Applications page, click **New**, and then click **PowerPoint Service Application**.
6. In the PowerPoint Service Application dialog box, in **Name**, type **PowerPoint Service Application**. In **Application Pool**, select **Use existing application pool**, and then in the listbox, select **SharePoint Web Services Default**. In **Add to default proxy list**, verify **Add this service application's proxy to the farm's default proxy list** is selected (default), and then click **OK**.
7. On the Service Applications page, click **New**, and then click **Excel Services Application**.
8. In the Excel Services Application dialog box, in **Name**, type **Excel Services Application**. In **Application Pool**, select **Use existing application pool**, and then in the listbox, select **SharePoint Web Services Default**. In **Add to default proxy list**, verify **Add this service application's proxy to the farm's default proxy list** is selected (default), and then click **OK**.

Note:

You can choose to create a new application pool to be used with a service application. When creating a new application pool, you can specify the security account used by the application pool to be a predefined Network Service account, or you can specify a managed account. The account must have read/write and execute permissions for the SPContent database and SPConfig database. For more information about services account permissions in SharePoint, see Account permissions and security settings (SharePoint Server 2010) (*http://technet.microsoft.com/library/55b99d80-3fa7-49f0-bdf4-adb5aa959019(Office.14).aspx*).

To create the service applications and the service application proxies by using Windows PowerShell

1. Using Notepad, open a new text file, and then copy and paste the following script into the file.

```
$appPool = Get-SPServiceApplicationPool -Name "SharePoint Web Services Default"

New-SPWordViewingServiceApplication -Name "WdView" -ApplicationPool $appPool |
New-SPWordViewingServiceApplicationProxy -Name "WdProxy"

New-SPPowerPointServiceApplication -Name "PPT" -ApplicationPool $appPool | New-
SPPowerPointServiceApplicationProxy -Name "PPTProxy"

New-SPExcelServiceApplication -Name "Excel" -ApplicationPool $appPool
```

2. Save the file with a **.ps1** file name extension to a folder where you run scripts (typically C:\scripts).
3. From the Windows PowerShell command prompt (that is, PS C:\>), type the following command and press ENTER:

   C:\<path>\<filename>.ps1

Activate the Office Web Apps Feature

After the service instances have been started, and the service applications and service application proxies have been created, to use Office Web Apps require the Office Web Apps Feature be activated on existing site collections. You can activate the feature on a single site collection in a browser on the Site collection features page or by using Windows PowerShell. If you have a large number of site collections, you can activate the feature on all site collections at the same time by using Windows PowerShell.

Procedures in this task will activate the Office Web Apps Feature on one or more existing site collections. This task must be completed only once after the service applications and the service application proxies have been created. If you have more than one existing site collection, and you are activating the feature on one site collection at a time, you may have to perform the procedure for each site collection.

Note:
This task does not need to be completed for new site collections created after Office Web Apps has been installed.

To activate the Office Web Apps Feature on a site collection on the Site collection features page

1. In a browser, in the SharePoint site, click **Site Actions**, and then click **Site Settings**.
2. On the Site Settings page, in **Site Collection Administration**, click **Site Collection Features**.
3. On the Features page, for **Office Web Apps**, click **Activate**.

To activate the Office Web Apps Feature on a site collection by using Windows PowerShell

1. Using Notepad, open a new text file, and then copy and paste the following script into the file.

```
$webAppsFeatureId = $(Get-SPFeature -limit all | where {$_.displayname -eq
"OfficeWebApps"}).Id
$singleSiteCollection = Get-SPSite -Identity http://<site_name> Enable-SPFeature
$webAppsFeatureId -Url $singleSiteCollection.URL
```

2. Specify the following parameters:

Parameter	Value
–Identity	URL

3. Save the file with a **.ps1** file name extension to a folder where you run scripts (typically C:\scripts).
4. In the Windows PowerShell console, at the command prompt (that is, PS C:\>), type the following command and press ENTER:

   C:\<path>\<filename>.ps1

To activate the Office Web Apps Feature on all site collections by using Windows PowerShell

1. Using Notepad, open a new text file, and then copy and paste the following script into the file.

```
$webAppsFeatureId = $(Get-SPFeature -limit all | where {$_.displayname -eq
"OfficeWebApps"}).Id
Get-SPSite -limit ALL |foreach{Enable-SPFeature $webAppsFeatureId -url $_.URL }
```

2. Save the file with a **.ps1** file name extension to a folder where you run scripts (typically C:\scripts).

3. From the Windows PowerShell command prompt (that is, PS C:\>), type the following command and press ENTER:

   C:\<path>\<filename>.ps1

Install and configure Office Web Apps on a new stand-alone SharePoint server

This section applies only if you are installing Office Web Apps on a new SharePoint installation where PSConfig has not previously been run as part of SharePoint setup.

Run Office Web Apps setup

Complete this task to install Office Web Apps components and files on a server.

To run Office Web Apps setup

1. From the root folder, run **Setup.exe**.
2. On the **Enter your Product Key** page, enter your product key, and then click **Continue**.
3. On the **Choose a file location** page, click **Install Now** to install to the default location. To install to a different location, specify the location that you want to install to, and then click **Install Now**.
4. When Setup finishes, a dialog box prompts you to complete the configuration of your server. Be sure that the **Run the SharePoint Products and Technologies Configuration Wizard now** check box is selected, and then click **Close** to start PSConfig.

Run PSConfig to register the services, start the service instances, create the service applications and proxies, and activate the Office Web Apps Feature

Complete this task to register the services, start the service instances, create the service applications and service application proxies, and activate the Office Web Apps Feature.

To run PSConfig to register the services, start the service instances, create the service applications and proxies, and activate the Office Web Apps Feature

1. If you left the **Run the SharePoint Products and Technologies Configuration Wizard now** check box selected in the previous task, on the PSconfig **Welcome to SharePoint Products** page, click **Next**.
2. In the dialog box that notifies you that some services might need to be restarted or reset during configuration, click **Yes**.
3. On the **Configuration Successful** page, click **Finish**. Your new SharePoint site opens.

Install and configure Office Web Apps on an existing SharePoint server farm

Peform the tasks in this section only if you are installing Office Web Apps on an existing SharePoint server farm where the Farm Configuration Wizard has previously been run.

Caution:
When you run Setup.exe, Office Web Apps setup configures the default open behavior for browser-enabled documents in SharePoint to open documents in the browser. If Office Web Apps setup has been run, but the Office Web Apps Services and Feature has not yet been activated, a user may get a broken link when opening a document in the browser. When deploying Office Web Apps on a live production server farm, to prevent broken links to documents while completing additional deployment tasks after running setup, it is recommended you enable the OpenInClient feature on existing site collections prior to running setup. For more information, see Additional configuration (optional).

Run Office Web Apps setup

Complete this task to install Office Web Apps on a single SharePoint server. This task must be performed on each server in the server farm.

To run Office Web Apps Setup

1. From the root folder, run **Setup.exe**.
2. On the **Enter your Product Key** page, enter your product key, and then click **Continue**.
3. On the **Choose a file location** page, click **Install Now** to install to the default location. To install to a different location, specify the location that you want to install to, and then click **Install Now**.
4. When Setup finishes, a dialog box prompts you to complete the configuration of your server. Be sure that the **Run the SharePoint Products and Technologies Configuration Wizard now** check box is selected.
5. Click **Close** to start the configuration wizard.

Run PSConfig to register services

Complete this task to register the Office Web Apps services on a single SharePoint server. This task must be performed on each server in the server farm.

To run PSConfig to register the services

1. On the **Welcome to SharePoint Products** page, click **Next**.
2. In the dialog box that notifies you that some services might need to be restarted or reset during configuration, click **Yes**.

3. On the **Modify server farm settings** page, select **Do not disconnect from this server farm**, and then click **Next**.
4. On the **Configuration Successful** page, click **Finish**. Your new SharePoint site opens.

Start the service instances

A service instance provides the physical location for a service application. For each server that you want to run the Office Web Apps service applications; you must start the service instances. You can start the service instances by using SharePoint Central Administration or by using Windows PowerShell.

Procedures in this task will start the service instances on those servers specified. This task must be completed after you have run WCSetup and PSConfig on each server in the farm.

To start the service instances by using Central Administration

1. Click **Start**, point to **All Programs**, **Microsoft SharePoint 2010 Products**, and then **SharePoint 2010 Central Administration**.
2. On the SharePoint Central Administration home page, in **System Settings**, click **Manage services on this server**.
3. On the Services on server:<servername>page, in **Server**, select a server, and then start **Excel Calculation Services**, **Word Viewing Service**, and **PowerPoint Service**. Repeat this step for each server in the farm you want to run Office Web Apps services. The OneNote Web App does not use a SharePoint service.

To start the service instances by using Windows PowerShell

1. Using Notepad, open a new text file, and then copy and paste the following script into the file.

```
$machinesToActivate = @("<servername1>", "<servername2>")

$serviceInstanceNames = @("Word Viewing Service","PowerPoint Service","Excel
Calculation Services")

foreach ($machine in $machinesToActivate)

{

  foreach ($serviceInstance in $serviceInstanceNames)

  {

     $serviceID = $(Get-SPServiceInstance | where {$_.TypeName -match
$serviceInstance} | where {$_.Server -match "SPServer Name="+$machine}).ID

     Start-SPServiceInstance -Identity $serviceID

  }

}
```

2. Specify the following parameters:

Parameter	Value
$machinesToActivate	Server name

3. Save the file with a **.ps1** file name extension to a folder where you run scripts (typically C:\scripts).
4. From the Windows PowerShell command prompt (that is, PS C:\>), type the following command and press ENTER:

   C:\<path>\<filename>.ps1

Create the service applications and the service application proxies

After the service instances have been started, the service applications and the service application proxies that connect the SharePoint Web front-ends to the service applications must be created. You can create the service applications and the service application proxies for the Word, PowerPoint, and Excel Web apps by using Central Administration or by using Windows PowerShell. The OneNote Web app does not require a service application. After created, the service applications will run on started service instances automatically.

In this task you will create the Office Web Apps service applications and service application proxies for the Word, PowerPoint, and Excel Web apps. This task can be completed only after you have started the service instances.

To create the service applications and the service application proxies by using Central Administration

1. Click **Start**, point to **All Programs**, **Microsoft SharePoint 2010 Products**, and then **SharePoint 2010 Central Administration**.
2. On the SharePoint Central Administration home page, in **Application Management**, click **Manage service applications**.
3. On the Service Applications page, click **New**, and then click **Word Viewing Service**.
4. In the Word Viewing Service Application dialog box, in **Name**, type **Word Viewing Service Application**. In **Application Pool**, select **Use existing application pool**, and then in the listbox, select **SharePoint Web Services Default**. In **Add to default proxy list**, verify **Add this service application's proxy to the farm's default proxy list** is selected (default), and then click **OK**.
5. On the Service Applications page, click **New**, and then click **PowerPoint Service Application**.
6. In the PowerPoint Service Application dialog box, in **Name**, type **PowerPoint Service Application**. In **Application Pool**, select **Use existing application pool**, and then in the

listbox, select **SharePoint Web Services Default**. In **Add to default proxy list**, verify **Add this service application's proxy to the farm's default proxy list** is selected (default) and then click **OK**.

7. On the Service Applications page, click **New** and then click **Excel Services Application**.
8. In the Excel Services Application dialog box, in **Name**, type **Excel Services Application**. In **Application Pool**, select **Use existing application pool**, and then in the listbox, select **SharePoint Web Services Default**. In **Add to default proxy list**, verify **Add this service application's proxy to the farm's default proxy list** is selected (default) and then click **OK**.

   Note:
   You can choose to create a new application pool to be used with a service application. When creating a new application pool, you can specify the security account used by the application pool to be a predefined Network Service account, or you can specify a managed account. The account must have read/write and execute permissions for the SPContent database and SPConfig database. For more information about services account permissions in SharePoint, see Account permissions and security settings (SharePoint Server 2010) (*http://technet.microsoft.com/library/55b99d80-3fa7-49f0-bdf4-adb5aa959019(Office.14).aspx*).

To create the service applications and the service application proxies by using Windows PowerShell

1. Using Notepad, open a new text file, and then copy and paste the following script into the file.

```
$appPool = Get-SPServiceApplicationPool -Name "SharePoint Web Services Default"

New-SPWordViewingServiceApplication -Name "WdView" -ApplicationPool $appPool |
New-SPWordViewingServiceApplicationProxy -Name "WdProxy"

New-SPPowerPointServiceApplication -Name "PPT" -ApplicationPool $appPool | New-
SPPowerPointServiceApplicationProxy -Name "PPTProxy"

New-SPExcelServiceApplication -Name "Excel" -ApplicationPool $appPool
```

2. Save the file with a **.ps1** file name extension to a folder where you run scripts (typically C:\scripts).
3. From the Windows PowerShell command prompt (that is, PS C:\>), type the following command and press ENTER:

   C:\<path>\<filename>.ps1

Activate the Office Web Apps Feature

After the service instances have been started, and the service applications and service application proxies have been created, to use Office Web Apps require the Office Web Apps Feature be activated on existing site collections. You can activate the feature on a single site collection in a browser on the Site collection features page or by using Windows PowerShell.

If you have a large number of site collections, you can activate the feature on all site collections at the same time by using Windows PowerShell.

Procedures in this task will activate the Office Web Apps Feature on one or more existing site collections. This task must be completed only once after the service applications and the service application proxies have been created. If you have more than one existing site collection, and you are activating the feature on one site collection at a time, you may have to perform the procedure for each site collection.

Note:
This task does not need to be completed for new site collections created after Office Web Apps has been installed.

To activate the Office Web Apps Feature on a site collection on the Site collection features page

1. In a browser, in the SharePoint site, click **Site Actions**, and then click **Site Settings**.
2. On the Site Settings page, in **Site Collection Administration**, click **Site Collection Features**.
3. On the Features page, for **Office Web Apps**, click **Activate**.

To activate Office Web Apps Feature on a site collection by using Windows PowerShell

1. Using Notepad, open a new text file, and then copy and paste the following script into the file.

```
$webAppsFeatureId = $(Get-SPFeature -limit all | where {$_.displayname -eq
"OfficeWebApps"}).Id
$singleSiteCollection = Get-SPSite -Identity http://<site_name> Enable-SPFeature
$webAppsFeatureId -Url $singleSiteCollection.URL
```

2. Specify the following parameters:

Parameter	Value
–Identity	URL

3. Save the file with a **.ps1** file name extension to a folder where you run scripts (typically C:\scripts).
4. In the Windows PowerShell console, at the command prompt (that is, PS C:\>), type the following command and press ENTER:

   C:\<path>\<filename>.ps1

To activate the Office Web Apps Feature on all site collections by using Windows PowerShell

1. Using Notepad, open a new text file, and then copy and paste the following script into the file.

```
$webAppsFeatureId = $(Get-SPFeature -limit all | where {$_.displayname -eq
"OfficeWebApps"}).Id
Get-SPSite -limit ALL |foreach{Enable-SPFeature $webAppsFeatureId -url $_.URL }
```

2. Save the file with a **.ps1** file name extension to a folder where you run scripts (typically C:\scripts).
3. From the Windows PowerShell command prompt (that is, PS C:\>), type the following command and press ENTER:

   C:\<path>\<filename>.ps1

Install and configure Office Web Apps on a new SharePoint server farm

Perform the tasks in this section only if you are installing Office Web Apps on a new SharePoint server farm where the Farm Configuration Wizard has not previously been run.

Run Office Web Apps setup

In this task you will install Office Web Apps files and components on a single SharePoint server in a new server farm where the Farm Configuration Wizard has not previously been run. This task must be completed on each server in the server farm.

To run Office Web Apps Setup

1. From the root folder, run **Setup.exe**.
2. On the **Enter your Product Key** page, enter your product key, and then click **Continue**.
3. On the **Choose a file location** page, click **Install Now** to install to the default location. To install to a different location, specify the location that you want to install to, and then click **Install Now**.
4. When Setup finishes, a dialog box prompts you to complete the configuration of your server. Be sure that the **Run the SharePoint Products and Technologies Configuration Wizard now** check box is selected.
5. Click **Close** to start the Farm Configuration Wizard.

Run PSConfig to register services

In this task you will register the Office Web Apps services on a single SharePoint server. This task must be completed on each server in the server farm.

To run PSConfig to register the services

1. On the **Welcome to SharePoint Products** page, click **Next**.
2. In the dialog box that notifies you that some services might need to be restarted or reset during configuration, click **Yes**.
3. On the **Modify server farm settings** page, select **Do not disconnect from this server farm**, and then click **Next**.
4. On the **Configuration Successful** page, click **Finish**. Your new SharePoint site opens.

Run the SharePoint Farm Configuration Wizard to start the service instances, create the service applications and proxies, and activate the Office Web Apps Feature

In this task you will start the service instances on all servers in the farm, create the service applications and service application proxies, and activate the Office Web Apps Feature on all existing site collections. This task must be completed after Setup.exe and PSConfig has been run on each server in the server farm.

To run the SharePoint Farm Configuration Wizard to start the service instances, create the service applications and proxies, and activate the Office Web Apps Feature

1. Click **Start**, point to **All Programs**, **Microsoft SharePoint 2010 Products**, and then **SharePoint 2010 Central Administration**.
2. On the **SharePoint Central Administration** home page, click **Configuration Wizards**.
3. On the **Configuration Wizards** page, click **Launch the Farm Configuration Wizard**.
4. In the **Farm Configuration Wizard** welcome page, choose **Walk me through the settings using this wizard**, and then click **Next**.
5. On the **Configure your SharePoint Farm** page, in **Service Account**, type a name for the Farm admin account.
6. In **Services**, select the Office Web Apps services that you want to activate, and then click **Next**.
7. Create an optional new top-level site. On the **Create Site Collection** page, follow the wizard steps to create a new top-level site.
8. On the **Configure your SharePoint Farm** page, click **Finish**.

Additional configuration (optional)

This section discusses additional configurations that are optional.

Configure the SharePoint default open behavior for browser-enabled documents

In SharePoint, you can configure whether browser-enabled documents are opened in a client application or in the browser. By default, when Office Web Apps is installed, Office documents will then open in the browser. You can override this setting using the SharePoint OpenInClient feature. The OpenInClient feature can be configured in Central Administration or by using the SPFeature cmdlet in Windows PowerShell.

How documents open in SharePoint varies depending on whether or not the OpenInClient feature is present, and either enabled or disabled:

- If the OpenInClient feature is not present and Office Web Apps is not installed, documents will open in the client application (SharePoint default).
- If the OpenInClient feature is not present, Office Web Apps is installed and Office Web Apps service applications are activated, documents will open in the browser (Office Web Apps default).
- If the OpenInClient feature is present and enabled, and Office Web Apps service applications are activated, documents will open in the client application.
- If the OpenInClient feature is present and disabled, and Office Web Apps service applications are activated, documents in will open in the browser.

Caution:
When you run Setup.exe to install Office Web Apps, setup will take control of the default open behavior in SharePoint to register Word, PowerPoint, Excel, and OneNote documents to be opened in their associated Web app. If a user clicks on a document in SharePoint after Setup.exe has been run, but before the Office Web Apps Services and Feature have been activated, the user can get a broken link in the browser. When installing Office Web Apps in a live production environment, it is strongly recommended that you enable the OpenInClient Feature prior to running Office Web Apps setup.

To set the default open behavior for site collections by using Central Administration

1. In SharePoint Central Administration, click **Site Actions**, and then click **Site Settings**.
2. On the Site Settings page, under **Site Collection Administration**, click **Site Collection Features**.
3. On the Features page, for the **Open Documents in Client Applications by Default** feature, click **Activate** (OpenInClient Feature is enabled) to open documents in the client application. Click **Deactivate** (OpenInClient Feature is disabled) to open documents in the browser.

To set the SharePoint Default open behavior for browser-enabled documents to open in the browser by using Windows PowerShell

1. Using Notepad, open a new text file, and then copy and paste the following script into the file.

   This example disables the default open behavior in SharePoint.

   ```
   $defaultOpenBehaviorFeatureId = $(Get-SPFeature -limit all | where {$_.displayname -eq "OpenInClient"}).Id
   Get-SPSite -limit ALL |foreach{ Disable-SPFeature $defaultOpenBehaviorFeatureId -url $_.URL }
   ```

2. Save the file with a **.ps1** file name extension to a folder where you run scripts (typically C:\scripts).
3. In the Windows PowerShell console, at the command prompt (that is, PS C:\>), type the following command and press ENTER:

   C:\<path>\<filename>.ps1

To set the SharePoint Default open behavior for browser-enabled documents to open in the client application by using Windows PowerShell

1. Using Notepad, open a new text file, and then copy and paste the following script into the file.

   This example sets the default open behavior for all documents in all sites to open in the client application (if available).

   ```
   $defaultOpenBehaviorFeatureId = $(Get-SPFeature -limit all | where {$_.displayname -eq "OpenInClient"}).Id
   Get-SPSite -limit ALL |foreach{ Enable-SPFeature $defaultOpenBehaviorFeatureId -url $_.URL }
   ```

2. Save the file with a **.ps1** file name extension to a folder where you run scripts (typically C:\scripts).
3. In the Windows PowerShell console, at the command prompt (that is, PS C:\>), type the following command and press ENTER:

   C:\<path>\<filename>.ps1

Troubleshooting

Problem Office Web Apps is installed, but documents do not open in their associated Web app in the browser.

Solution Verify the Office Web Apps Feature has been activated for the site collection in which the document resides. For more information, see Activate the Office Web Apps Feature.

Solution Verify the service instances have been started. For more information, see Start the service instances.

Solution Verify the service applications and proxies have been created. In SharePoint Central Administration, in **Application Management**, click **Manage service applications**. Verify the Word Viewing service application, PowerPoint service application, and Excel Services Application are started. If they are not started, verify the service instances have been started.

Solution Verify the SharePoint OpenInClient Feature is not enabled. For more information, see Additional configuration (optional).

Problem The Office Web Apps opens fine in view mode, but when a user clicks the Edit in Word, Edit in PowerPoint, or Edit in Excel button on the toolbar, an error is displayed.

Solution Verify that the Office Web Apps Feature is activated and the Word Viewing Service, PowerPoint Service, and Excel Calculation Services are started.

Problem When running setup, the product key will not validate.

Solution Verify you are installing an Office Web Apps version compatible with your version of SharePoint 2010 Products. Office Web Apps Trial Edition cannot be installed on a server with licensed SharePoint 2010 products.

Solution Verify you have a valid Microsoft Office 2010 volume license.

PowerPoint Broadcast Slide Show

Articles in this section provide IT Pros information about understanding, planning, and configuring Microsoft PowerPoint 2010 Broadcast Slide Show as part of Microsoft Office Web Apps installed on Microsoft SharePoint 2010 Products on-premises in their organization.

In this section:

Article	Description
Understanding Broadcast Slide Show (Office Web Apps)	This article contains valuable information to help IT Pros understand Broadcast Slide Show and how it can benefit users in their organizations.
Planning Broadcast Slide Show (Office Web Apps)	This article contains information to help IT Pros plan how to use Broadcast Slide Show in their organizations.
Configure a Broadcast site (Office Web Apps)	This article contains IT Pros information about how to configure a Broadcast site.
Configure Broadcast Slide Show performance (Office Web Apps)	This article contains information for IT Pros about how to configure a Broadcast Slide Show for maximum performance.
Configure Group Policy settings (Broadcast Slide Show)	This article contains information about how to use Group Policy to manage certain features in Broadcast Slide Show.

Understanding Broadcast Slide Show (Office Web Apps)

Broadcast Slide Show is a new capability in Microsoft Office 2010 that enables presenters to broadcast a slide show from Microsoft PowerPoint 2010 to remote viewers who watch in a Web browser. Broadcast Slide Show provides organizations with a low-infrastructure presentation broadcast capability that works through the Web.

The proliferation of high-bandwidth networks and growing ubiquity of wireless connectivity have brought significant changes in the way people communicate and collaborate. After its initial release, PowerPoint was used primarily as a tool for making presentations to an in-person, on-site audience. Over time, however, it has become more and more common for users to present a slide show over a Web or network connection to remote attendees or to mix remote and on-site attendees for the same presentation. Broadcast Slide Show now offers this capability to users through the public PowerPoint Broadcast Service (*http://go.microsoft.com/fwlink/?LinkId=182216&clcid=0x409*), and to business customers with Microsoft Office Web Apps installed on Microsoft SharePoint 2010 Products on-premises in their organizations.

Information provided in this article is intended for IT Pros planning to use Broadcast Slide Show in their organizations. New and updated content will be published on a regular basis. For more information about how to plan effectively for Broadcast Slide Show in your organization, see Planning Broadcast Slide Show (Office Web Apps).

SharePoint 2010 Products in this article refers to Microsoft SharePoint Server 2010 and Microsoft SharePoint Foundation 2010 unless otherwise noted.

In this article:

- Broadcast Slide Show Overview
- Understanding the presenter experience
- Understanding the attendee experience
- Broadcast Slide Show features
- Managing Broadcast Slide Show
- Broadcast Slide Show and Microsoft Live Meeting
- Summary

Broadcast Slide Show Overview

Broadcast Slide Show provides a low-infrastructure remote slide show capability through PowerPoint 2010.

Common broadcast scenarios include:

- Provide an impromptu, one-on-one broadcast

- Invite multiple viewers at remote locations to view a presentation at any time
- Present a slide show to both on-site and remote attendees at the same time in the context of a training session, meeting, or conference call

Understanding the presenter experience

Broadcast Slide Show gives presenters a streamlined user experience that works through PowerPoint 2010 without any additional client software.

Within PowerPoint 2010, the presenter clicks the **Slide Show** tab, and selects **Broadcast Slide Show**. The presenter is then provided a list of broadcast services to choose from. This list may include internal and external providers, and can be centrally controlled by administrators. The presenter then clicks **Start Broadcast** and is provided with a URL that attendees can use to connect to the slide show through the Web.

The presenter can then send this URL to attendees in a number of ways, including an automatically generated e-mail message, or through instant messaging (IM). During the broadcast, the presenter can pause the slide show at any time, resend the Broadcast URL to any attendees, or switch to another application without interrupting the broadcast or showing their computer's desktop to attendees.

Understanding the attendee experience

Broadcast Slide Show enables attendees to quickly connect to a slide show broadcast through the Web without requiring any additional client software installation.

Supported browsers include:

- Microsoft Internet Explorer 7 and 8
- Firefox 3.5
- Safari 4 on the Macintosh

An attendee receives the slide show URL or link from the presenter through e-mail, IM, or other channel. The attendee then clicks the link and connects to the slide show URL in a Web browser. Until the presenter begins, attendees will see a message telling them that the slide show has not yet started.

When the presenter begins, attendees will then see the slide show presentation in their local browser in real-time, as it is presented. When the presenter ends the broadcast, all attendees will see a message telling them that the broadcast is over.

Broadcast Slide Show features

Attendees can see a near full-fidelity slide show synchronized with the presenter, including animations that play at the same time.

While a Broadcast Slide Show can provide a near full-fidelity experience, the following limitations do apply:

- **Audio** Slide show audio will not be heard by attendees through PowerPoint. For remote presentations, users may elect to supplement slide show broadcasts with audio through a personal phone call or conference call.
- **Video** Attendees will only see the static poster frame of any video clips.
- **Transitions** Attendees will only see a fade-in transition between slides.
- **Ink drawings and markup** Presenters will not be able to add ink annotations or markup during the broadcast.
- **ActiveX controls** Attendees will see only a static image of the control and will not be able to interact with it.
- **Hyperlinks to other presentations** If the presenter clicks a hyperlink to another presentation, attendees will continue to see the most recent slide the presenter showed in the original presentation.

Managing Broadcast Slide Show

Broadcast Slide Show is designed to provide administrators a simple management experience without requiring extensive additional infrastructure. Broadcast Slide Show provides the following:

- **Available Services** Administrators can choose between a locally-hosted broadcast service, the public PowerPoint Broadcast Service, or both. Up to 10 broadcast services can be used.
- **Service List** A presenter who initiates a broadcast is given a list of broadcast services to choose from. This can be managed through Group Policy. For example, administrators can decide whether to include the public PowerPoint Broadcast Service in this list, a local server, or both.
- **Permissions** Administrators can determine which users or groups have permission to broadcast slide shows or attend slide show broadcasts.
- **Group Policy** Broadcast Slide Show can be managed using Group Policy and includes the ability to enable or disable the service, manage the service list, and customize the user experience.

For more information about managing Broadcast Slide Show using Group Policy, see Configure Group Policy settings (Broadcast Slide Show).

Broadcast Slide Show and Microsoft Live Meeting

Broadcast Slide Show and Microsoft Live Meeting are complementary technologies, each providing a different level of functionality and end-user experience. Microsoft Live Meeting provides a hosted solution for broad communication and collaboration, including sharing applications, interactive collaboration, chat, and other collaboration features.

Broadcast Slide Show and Microsoft Live Meeting

	Broadcast Slide Show	Microsoft Live Meeting
Client Software	Included in Office PowerPoint 2010	LiveMeeting client
Hosting	Available with or without internally hosted server	Internally hosted or through an externally hosted service
Broadcast	PowerPoint slide show only	Any desktop application
Purpose	Just-in-time shared slideshows	Broad communication and collaboration capabilities
Use when…	You want a low-infrastructure solution for impromptu slideshow broadcasts	You want a complete Web conferencing solution with collaboration tools

Summary

Broadcast Slide Show gives organizations options to provide broadcast slideshow services to users either as an internally-hosted service or as a service accessed over the Internet. This flexibility helps organizations to provide a low-infrastructure slide show option that fits their business scenarios and requirements.

The next step in implementing Broadcast Slide Show in your organization is to plan how both presenters and attendees will use Broadcast Slide Show and how administrators can configure Broadcast Slide Show. For more information, see Planning Broadcast Slide Show (Office Web Apps).

Planning Broadcast Slide Show (Office Web Apps)

Broadcast Slide Show is a new capability in Microsoft Office 2010 that enables presenters to broadcast a slide show from Microsoft PowerPoint 2010 to remote viewers who watch in a Web browser. Broadcast Slide Show provides companies with a low-infrastructure presentation broadcast capability that works through the Web.

Information provided in this article is intended for IT Pros planning to use Broadcast Slide Show as part of Microsoft Office Web Apps installed on Microsoft SharePoint 2010 Products on-premises in their organizations. SharePoint 2010 Products in this article refers to Microsoft SharePoint Server 2010 and Microsoft SharePoint Foundation 2010 unless otherwise noted. New and updated content will be published on a regular basis.

Before planning your Broadcast Slide Show solution, we recommend you first read Understanding Broadcast Slide Show (Office Web Apps).

In this article:

- Plan service types
- Plan broadcast site location
- Plan permissions
- Use Group Policy to manage Broadcast Slide Show
- Summary

Plan service types

Two kinds of broadcast services are available:

- **PowerPoint Broadcast Service** By default, PowerPoint 2010 provides all presenters with a link to the public PowerPoint Broadcast Service hosted by Microsoft. This service requires presenters to sign in with a Windows Live ID. Presenters who use this service receive a public Internet link that they can share with anyone on the Internet they invite.
- **Internal Services** You can host your own broadcast service with Office Web Apps installed on SharePoint 2010 Products. You create one or more broadcast services by creating sites collections that use the PowerPoint Broadcast site template and set permissions for who can use the service through group membership on the site. Up to ten services can be specified.

Service types

Service type	User type	Connection type	Intended for
PowerPoint Broadcast Service	Internal & External	Internet Connection	Smaller businesses with limited IT resources Organizations wanting a low-impact solution for internal and external presentations Organizations with fewer content security or compliance requirements
Internal Service	Users with permissions to SharePoint broadcast site	Office Web Apps installed on SharePoint 2010 Products	Larger businesses with dedicated IT resources Organization whose communication and collaboration needs are primarily internal Organizations with strict content security or compliance requirements

Plan broadcast site location

If you perform a stand-alone installation of Office Web Apps, or run the Farm Configuration Wizard with the PowerPoint service application service selected, the PowerPoint service application is automatically created. A default broadcast site collection is also created under the default Web application at "http://<*default web application*>/sites/broadcast". In most cases, this URL is "http://<*server name*>/sites/broadcast", where <*server name*> is the name of the server.

NT Authority/Authenticated Users is automatically added to the Broadcast Presenters and Broadcast Attendees groups of this default site, granting all users access to create and view broadcasts.

You can also create one or more broadcast sites manually by creating a new site collection in SharePoint using the PowerPoint Broadcast Site template.

For more information about how to create a Broadcast Slide Show site, see Configure a Broadcast site (Office Web Apps).

Plan permissions

An internal deployment of Broadcast Slide Show enables broadcasting using the functionality that is provided by PowerPoint Web App and a SharePoint site. Administrators can control which users can

present and attend by giving users the appropriate permissions to this site. Three levels of permission are available:

- **Broadcast Presenter** Gives users permission to broadcast.
- **Broadcast Attendees** Gives users permission to attend broadcasts.
- **Broadcast Administrators** Gives users permission to change membership on the site.

Most organizations will be able to provide broadcast services to all users through a single default installation of Broadcast Slide Show. For additional flexibility, an organization may decide to configure multiple broadcast sites with different permissions. For example, a company wants to provide broadcast services to their product marketing department to share presentations internally. They create a new broadcast site and grant permissions only to members of the marketing department.

Use Group Policy to manage Broadcast Slide Show

Organizations can use Group Policy to configure and manage Broadcast Slide Show. Broadcast Slide Show supports the following Group Policy controls:

- List of Broadcast Services
  - Up to 10 services can be specified
  - Ability to specify name, URL, and description for each
- Disable access to the public PowerPoint Broadcast Service
- Prevent users from adding to the list of broadcast services
- Turn off access to Broadcast Slide Show through the PowerPoint desktop application
- Turn off programmatic access to the Broadcast Slide Show object model

For more information about how to use Group Policy to manage Broadcast Slide Show, see Configure Group Policy settings (Broadcast Slide Show).

Summary

Information provided in this article can help you when planning a Broadcast Slide Show solution for your organization. The next step in your Broadcast Slide Show solution is to configure broadcast sites. For more information, see Configure a Broadcast site (Office Web Apps).

Configure a Broadcast site (Office Web Apps)

The information that is provided in this article about how to configure a Broadcast Slide Show site is intended for IT pros who maintain Microsoft Office Web Apps installed on Microsoft SharePoint 2010 Products on a stand-alone server or on a server farm. SharePoint 2010 Products in this article refers to Microsoft SharePoint Server 2010 and Microsoft SharePoint Foundation 2010 unless otherwise noted.

Configure a PowerPoint Broadcast site

Broadcast Slide Show requires that Office Web Apps be installed on SharePoint 2010 Products on a stand-alone server or server farm, the PowerPoint service application be created, and a broadcast site collection be created. If the PowerPoint service application is created during post-setup configuration of a stand-alone installation or by running the Farm Configuration Wizard, a default broadcast site is also created. In most cases, the default site's URL is "http://<server name>/sites/broadcast". Setup also installs a broadcast site template. Multiple site collections that use the broadcast site template can be created.

If the PowerPoint service application has not been created during post-setup configuration of a stand-alone installation or by running the Farm Configuration Wizard, you can create the PowerPoint service application manually. To learn about how to create Office Web Apps service applications manually, see Activate the Office Web Apps services and feature by using Windows PowerShell in Deploy Office Web Apps (Installed on SharePoint 2010 Products). If the PowerPoint service application is created manually, to use PowerPoint Broadcast Slide Show, a PowerPoint broadcast site must also be created manually.

For Broadcast Slide Show sites that were created manually, administrators must give users appropriate permissions through SharePoint 2010 Products to broadcast a slide show and to attend broadcast slide shows. For Broadcast Slide Show sites that were created by default, authenticated users are added to both the Broadcast Presenters and Broadcast Attendees groups; however, we recommend that administrators verify that appropriate permissions are given for their organization.

Important:
Users should not be granted Full Control or Full Read policy permissions to a Web Application that contains a broadcast site collection. This circumvents the permissions on the broadcast site and might allow users to access broadcast documents.

Procedures in this task:

- To create a PowerPoint broadcast site by using Central Administration
- To create a PowerPoint broadcast site by using Windows PowerShell
- To configure permissions on the broadcast site by using a Web browser
- To configure permissions on the broadcast site by using Windows PowerShell

Some procedures in this task require using Windows PowerShell or SharePoint 2010 Management Shell to run cmdlets. To run SharePoint 2010 cmdlets in Windows PowerShell, you must add the Microsoft.SharePoint.PowerShell snapin by using the **Add-PSSnapin** cmdlet, or you can run the SharePoint 2010 cmdlets by using SharePoint 2010 Management Shell. By default, the Microsoft.SharePoint.PowerShell snapin is already registered, and the snapin is added to the SharePoint 2010 Management Shell. To run SharePoint 2010 cmdlets, you must be a member of the SharePoint_Shell_Access role on the configuration database and a member of the WSS_ADMIN_WPG local group on the computer where SharePoint 2010 Products is installed. To run scripts in Windows PowerShell or SharePoint 2010 Management Shell, you must set the execution policy by using the **set-executionpolicy** cmdlet, using the **unrestricted** parameter. For more information about the **Add-PSSnapin** cmdlet, see Add-PSSnapin (*http://go.microsoft.com/fwlink/?LinkId=188450&clcid=0x409*). For more information about how to use Windows PowerShell with SharePoint 2010 Products, see SharePoint 2010 Products administration by using Windows PowerShell.

To create a PowerPoint broadcast site by using Central Administration

1. Verify that you have the following administrative credentials:
   - To create a PowerPoint broadcast site, you must be a member of the Farm Administrators SharePoint group on the computer that is running the SharePoint Central Administration Web site.
2. On the Central Administration Web site, in **Application Management**, click **Create site collections**.
3. On the Create Site Collection page, specify the following settings.

   Web Application Specify the Web application that will contain the site or use the default selection (recommended).

   Title Enter the user-friendly title of the site.

   Description Enter an optional description of the broadcast service

   URL This is the URL where users will access this broadcast service.

   Template Selection Click **Enterprise** and then select **PowerPoint Broadcast Site**.

   Primary Site Collection Administrator Enter or browse for a user to be the primary site collection administrator.

   Secondary Site Collection Administrator Enter or browse for a user to be the secondary site collection administrator (optional).

   Quota Template Add a quota template to this site (optional).
4. Verify settings and then click **OK**.

To create a PowerPoint broadcast site by using Windows PowerShell

1. Using Notepad, create a text file and then copy and paste the following script into the file.

```
$rootSites = Get-SPWebApplication | Get-SPSite -Limit ALL | where {
```

```
$_.ServerRelativeUrl -eq "/" }
$bcastTmpl = Get-SPWebTemplate | where { $_.Name -eq "PowerPointBroadcast#0"
foreach ($rootSite in $rootSites) { $url = $rootSite.Url + "/sites/broadcast" New-
SPSite -Url $url -OwnerAlias "contoso\admin" -Template $bcastTmpl}
```

2. Specify the following parameters:

Parameter	Value
URL	Broadcast Site URL
–OwnerAlias	Domain\alias

3. Save the file with a **.ps1** file name extension to a folder where you run scripts (typically C:\scripts).
4. In the Windows PowerShell console, at the command prompt (that is, PS C:\>), type the following command and press ENTER:

   C:\<path>\<filename>.ps1

To configure permissions on the broadcast site by using a Web browser

1. In a Web browser, browse to the broadcast site.
2. In the broadcast site, click **Site Actions**, and then click **Site Permissions**.
3. On the **Permissions** menu bar, click **Grant Permissions**, and then select one of the following.

   To allow users or groups to broadcast, add them to the **Broadcast Presenters** group.

   To allow users or groups to attend a slide show presented through Broadcast Slide Show, add them to the **Broadcast Attendees** group.

   To allow users or groups to administer the site and change group membership, add them to **the Broadcast Administrators** group.

To configure permissions on the broadcast site by using Windows PowerShell

1. Using Notepad, create a text file and then copy and paste the following script into the file.

```
New-SPUser -web $url -UserAlias "NT AUTHORITY\Authenticated Users" -Group
"Broadcast Attendees"
New-SPUser -web $url -UserAlias "NT AUTHORITY\Authenticated Users" -Group
"Broadcast Presenters"
```

2. Specify the following parameters:

Parameter	Value
-UserAlias	Domain\Name Domain\Group
-Group	Broadcast Attendees Broadcast Presenters

3. Save the file with a **.ps1** file name extension to a folder where you run scripts (typically C:\scripts).
4. In the Windows PowerShell console, at the command prompt (that is, PS C:\>), type the following command and press ENTER:

   C:\<path>\<filename>.ps1

See Also

Get-SPWebApplication (*http://technet.microsoft.com/library/11d6521f-f99c-433e-9ab5-7cf9e953457a(Office.14).aspx*)

Get-SPSite (*http://technet.microsoft.com/library/f3422bf4-0f9b-4f22-94c8-2a0606a31b16(Office.14).aspx*)

Get-SPWebTemplate (*http://technet.microsoft.com/library/dfd10bac-c304-4f3f-bea9-eb0af5f96df5(Office.14).aspx*)

New-SPUser (*http://technet.microsoft.com/library/b8d7f8df-d5df-4497-a55b-dbe56b1c6fbb(Office.14).aspx*)

Configure Broadcast Slide Show performance (Office Web Apps)

The information that is provided in this article about how to configure Microsoft PowerPoint 2010 Broadcast Slide Show performance is intended for IT pros who maintain Microsoft Office Web Apps installed on Microsoft SharePoint 2010 Products on a stand-alone server or on a server farm on-premises in their organization. SharePoint 2010 Products in this article refers to Microsoft SharePoint Server 2010 and Microsoft SharePoint Foundation 2010 unless otherwise noted.

Important:
Performance and capacity guidance provided in this article is preliminary and subject to change. Your results may vary.

When using Microsoft PowerPoint Broadcast Slide Show, the Broadcast Slide Show attendee's browser will poll the Broadcast Slide Show server once per second to check for updates in the slide show presentation. If there are many attendees, this can cause an excessive load on the Web front-end servers, and in extreme cases can disrupt other services on the server farm. By default, Broadcast Slide Show is optimized to support broadcasts of fewer than 50 attendees; however, additional attendees are not prevented from joining the presentation. In cases in which a Broadcast Slide Show presentation may have more attendees, administrators can optimize Broadcast Slide Show performance by using one or more of the methods described in this article.

In this article:

- Deploy a dedicated SharePoint server farm
- Configure broadcast capacity
- Configure the polling interval
- Modify throttling to limit Web front-end resource usage
- Change the maximum duration of a broadcast
- Set a time limit for idle broadcasts

Deploy a dedicated SharePoint server farm

In organizations that will use PowerPoint Broadcast Slide Show for many attendees on a regular basis, administrators can deploy a dedicated SharePoint Server farm for PowerPoint Broadcast Slide Show. For more information about how to plan and deploy a dedicated server farm, see Planning Office Web Apps (Installed on SharePoint 2010 Products) and Deploy Office Web Apps (Installed on SharePoint 2010 Products).

Configure broadcast capacity

To increase the rate at which the PowerPoint service processes broadcast requests, administrators can add the behaviorConfiguration="MaxPendingRecieves" setting for endpoint elements for the BroadcastStateService service and add the MaxPendingReceives behavior to the endpointBehaviors element in the Web.config file on middle-tier servers.

Important:
Adding this setting will increase the number of attendees who can attend broadcasts. However, this may increase resources that are used by middle-tier servers. This setting is recommended only for a dedicated broadcast server farm, or for a server farm where significant broadcast usage is expected.

Adding this setting may require modifying multiple Web.config files on each farm server. For more information about how to work with Web.config files, see Working with Web.config Files (*http://go.microsoft.com/fwlink/?LinkID=183741&clcid=0x409*) at http://msdn.microsoft.com/en-us/library/ms460914(office.14).aspx.

To increase broadcast capacity, use Notepad to edit the Web.config file at C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\14\WebServices\PowerPoint\Web.config

The following example increases broadcast service capacity by adding the behaviorConfiguration="MaxPendingRecieves" setting and adding the MaxPendingReceives behavior to the endpointBehaviors element in the Web.config file:

```
<service name="Microsoft.Office.Server.Powerpoint.Pipe.Core.BroadcastStateService"

behaviorConfiguration="RenderingServiceBehavior">

<endpoint name="BinaryHttpRenderingService" address="fast"

   binding="customBinding" bindingConfiguration="FastClaimsBinding"

   contract="Microsoft.Office.Server.Powerpoint.Interface.Shared.IBroadcastStateService"

behaviorConfiguration="MaxPendingReceives"/>

<endpoint name="BinaryHttpRenderingService" address="secure"

   binding="customBinding" bindingConfiguration="SecureClaimsBinding"

   contract="Microsoft.Office.Server.Powerpoint.Interface.Shared.IBroadcastStateService"

behaviorConfiguration="MaxPendingReceives"/>

</service>

<endpointBehaviors>

  <behavior name="MaxPendingReceives">

    <dispatcherSynchronization maxPendingReceives="10" />

  </behavior>

</endpointBehaviors>
```

Recommended values

To increase capacity, the recommended value specified for the maxPendingRecieves setting is 10. By default, when the MaxPendingRecieves behavior is not present, the effective value is 1.

Behavior	Setting	Value
MaxPendingReceives	maxPendingReceives	10

Caution:
Changes that you make to Web.config may be overwritten when you install updates or service packs for SharePoint 2010 Products, or when you upgrade an installation to the next product version.

Configure the polling interval

In organizations that will use PowerPoint Broadcast Slide Show for many attendees on a regular basis, but cannot deploy a dedicated SharePoint server farm for PowerPoint Broadcast Slide Show, administrators can change the polling interval value, which determines how frequently attendees poll the Broadcast Slide Show server to check for updates in the presentation. Increasing the polling interval value will decrease polling traffic, but will increase time elapsed before attendees see changes in presentation. The polling interval can be specified in milliseconds (1000 milliseconds = 1 second) by using Windows PowerShell.

To configure the polling interval by using Windows PowerShell

1. Verify that you meet the following minimum requirements: See Add-SPShellAdmin.
2. On the **Start** menu, click **All Programs**.
3. Click **Microsoft SharePoint 2010 Products**.
4. Click **SharePoint 2010 Management Shell**.
5. At the Windows PowerShell command prompt (PS C:\>), type the following command, and then press ENTER:

   Get-SPPowerPointServiceApplicationProxy | Set-SPPowerPointServiceApplicationProxy -BroadcastBrowserHealthyStatePollIntervalInMilliSeconds 2000

For more information, see Get-SPPowerPointServiceApplicationProxy, and Set-SPPowerPointServiceApplicationProxy.

Modify throttling to limit Web front-end resource usage

Administrators can modify the maxConcurrentCalls setting for the serviceThrottling element for ViewServiceTypeBehaviors and ViewServiceTypeBehaviorsHttps behaviors in the PowerPoint viewing Web.config file. This setting limits Web front-end resources that are used for views by the PowerPoint Web App and PowerPoint Broadcast Slide Show. Modifying this setting requires modifying the Web.config files on each front-end farm server. For more information about how to work with Web.config files, see Working with Web.config Files (*http://go.microsoft.com/fwlink/?LinkID=183741&clcid=0x409*) at http://msdn.microsoft.com/en-us/library/ms460914(office.14).aspx.

To modify throttling to limit Web front-end usage, use notepad to edit the Web.config file in IIS at C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\14\isapi\ppt\Web.config

The following example specifies four serviceThrottling maxConcurrentCalls for each behavior on a server that has four CPUs:

```
<behavior name="ViewServiceTypeBehaviors" >
  <serviceMetadata httpGetEnabled="true" />
  <serviceThrottling maxConcurrentCalls="16" />
  <AttemptAuthFallback />
</behavior>
  <behavior name="ViewServiceTypeBehaviorsHttps" >
  <serviceMetadata httpGetEnabled="false" />
  <serviceThrottling maxConcurrentCalls="16" />
  <AttemptAuthFallback />
</behavior>
```

Recommended values

The values specified should be determined by the number of CPUs on the server.

Behavior	Setting	Value
ViewServiceTypeBehaviors	serviceThrottling maxConcurrentCalls	2-5
ViewServiceTypeBehaviorsHttps	serviceThrottling maxConcurrentCalls	2-5

Caution:
Changes that you make to Web.config may be overwritten when you install updates or service packs for SharePoint 2010 Products, or when you upgrade an installation to the next product version.

Change the maximum duration of a broadcast

By default, a broadcast will expire after 12 hours. This supports long meetings and presenters who create a broadcast and send invitations several hours before they start a meeting. However, a long broadcast duration can cause continued server load if a presenter neglects to end the broadcast after the meeting and attendees keep their browser open after the meeting. The maximum duration of a broadcast can be specified (in seconds) by using Windows PowerShell.

To change the maximum duration of a broadcast by using Windows PowerShell

1. Verify that you meet the following minimum requirements: See Add-SPShellAdmin.
2. On the **Start** menu, click **All Programs**.
3. Click **Microsoft SharePoint 2010 Products**.
4. Click **SharePoint 2010 Management Shell**.
5. At the Windows PowerShell command prompt (PS C:\>), type the following command, and then press ENTER:

   Get-SPPowerPointServiceApplicationProxy | Set-SPPowerPointServiceApplicationProxy -BroadcastSessionTimeoutInSeconds 21600.
6. At the Windows PowerShell command prompt (PS C:\>), type the following command, and then press ENTER:

   Get-SPPowerPointServiceApplication | Set-SPPowerPointServiceApplication -BroadcastStateIdleTimeoutInSeconds 21600.

For more information, see Get-SPPowerPointServiceApplicationProxy, and Set-SPPowerPointServiceApplicationProxy.

Set a time limit for idle broadcasts

An additional setting can be enabled to force PowerPoint to end a broadcast after a specified length of time being idle. By default, this setting is 0 indicating that there is no idle limit. Care should be taken when you change this setting as valid broadcasts may be idle if the presenter creates the broadcast several hours before a meeting or if there are breaks during a meeting. The broadcast idle time-out (in seconds) can be specified by using Windows PowerShell.

To set a limit for idle broadcasts by using Windows PowerShell

1. Verify that you meet the following minimum requirements: See Add-SPShellAdmin.
2. On the **Start** menu, click **All Programs**.
3. Click **Microsoft SharePoint 2010 Products**.
4. Click **SharePoint 2010 Management Shell**.
5. At the Windows PowerShell command prompt (PS C:\>), type the following command, and then press ENTER:

   Get-SPPowerPointServiceApplicationProxy | Set-SPPowerPointServiceApplicationProxy -BroadcastPresenterIdleTimeoutInSeconds 14400

For more information, see Get-SPPowerPointServiceApplicationProxy, and Set-SPPowerPointServiceApplicationProxy.

Configure Group Policy settings (Broadcast Slide Show)

The information provided in this article about configuring Microsoft PowerPoint 2010 Broadcast Slide Show Group Policy settings is aimed at IT pros who maintain Microsoft Office Web Apps installed on Microsoft SharePoint 2010 Products on a stand-alone server or on a server farm on premises in their organization. SharePoint 2010 Products in this article refers to Microsoft SharePoint Server 2010 and Microsoft SharePoint Foundation 2010 unless otherwise noted.

Manage Broadcast Slide Show by using Group Policy

To use Group Policy to manage Broadcast Slide Show requires the Microsoft PowerPoint 2010 Administrative Template. To configure Broadcast Slide Show Group Policy settings in the administrative template, use Group Policy Object Editor.

The Broadcast object in the Microsoft PowerPoint 2010 Administrative Template includes the following policy settings:

- **Disable default service** Enable the Disable default service policy to remove the default PowerPoint Broadcast Service from the Broadcast Slide Show dialog box.
- **Disable programmatic access** Enable the Disable programmatic access policy to restrict the ability to create a broadcast programmatically.
- **Prevent access to user-specified services** Enable this policy to prevent users from adding their own services or accessing services they previously added through the Broadcast Slide Show dialog box

In the Broadcast Services object, you can configure up to 10 broadcast services. Broadcast service policy settings include the following properties:

- **Name:** The name of the service as it will appear to users in the Broadcast Slide Show dialog bog.
- **Service URL:** The URL of the broadcast SharePoint site you created earlier.
- **Description:** An optional description of the service.
- **Information URL:** An optional URL for users to get more information about the service.
- **Service agreement URL:** An optional URL for users to find a service agreement for the service.

Disable Broadcast Slide Show by using Group Policy

You can disable the Broadcast Slide Show button on the Slide Tab of PowerPoint 2010 by using Group Policy.

To disable the Broadcast Slide Show button on the Slide Tab

1. In Group Policy Management Editor, expand User Configuration, expand Administrative Templates, expand **Microsoft PowerPoint 2010**, expand **Disable Items in User Interface**, and then select **Predefined**.
2. In the right pane, double-click **Disable Commands**.
3. Select **Enabled**, and then select **Slide Show tab | Start Slide Show | Broadcast Slide Show**..

Manage Office Web Apps (Installed on SharePoint 2010 Products)

This section explains how to perform common management tasks for Office Web Apps installed on Microsoft SharePoint 2010 Products.

In this section:

Article	Description
Activate the Office Web Apps Feature on site collections	Provides information about how to activate or deactivate the Office Web Apps Feature on existing site collections.
Manage the Office Web Apps cache	Provides information about how to manage the Office Web Apps cache that is part of a SharePoint content database.
Configure the default open behavior for browser-enabled documents (Office Web Apps)	Provides information about how to configure how documents in SharePoint are opened after Office Web Apps is installed.
Configure PowerPoint service application settings	Provides information about how to configure settings for the PowerPoint service application.
Configure Word Viewing service settings	Provides information about how to configure settings for the Word Viewing service.
Configure Excel Services Application settings (Office Web Apps)	Provides information about how to configure settings for the Excel Services Application.

Activate the Office Web Apps Feature on site collections

The information in this article on how to activate the Microsoft Office Web Apps Feature on site collections is intended for IT pros who maintain Office Web Apps installed on Microsoft SharePoint 2010 Products on a stand-alone server or a server farm. SharePoint 2010 Products in this article refers to Microsoft SharePoint Server 2010 and Microsoft SharePoint Foundation 2010 unless otherwise noted.

Activate the Office Web Apps Feature for a site collection

The Office Web Apps Feature should be activated on every site collection for which any of the Web apps will be available. Typically, you activate the Office Web Apps Feature on a site collection during the deployment phase after you run setup and activate the services; however, in some cases, you can decide to later activate only on particular site collections and deactivate on other site collections.

Important:
Office Web Apps will not work for documents in Information Rights Management (IRM) protected libraries. For IRM-protected document libraries, we recommend that the **Default open behavior for browser enabled documents** setting be configured to open in the client application. For more information about how to configure the default open behavior, see Configure the default open behavior for browser-enabled documents (Office Web Apps).

Procedures in this task:

- To activate the Office Web Apps Feature on all site collections by using Windows PowerShell
- To activate the Office Web Apps Feature on a site collection on the Site collection features page
- To activate the Office Web Apps Feature on a single site collection by using Windows PowerShell
- To deactivate the Office Web Apps Feature on a single site collection by using Windows PowerShell

Some procedures in this task require using Windows PowerShell or SharePoint 2010 Management Shell to run cmdlets. To run SharePoint 2010 cmdlets in Windows PowerShell, you must add the Microsoft.SharePoint.PowerShell snap-in by using the **Add-PSSnapin** cmdlet, or you can run the SharePoint 2010 cmdlets by using SharePoint 2010 Management Shell. By default, the Microsoft.SharePoint.PowerShell snap-in is already registered, and the snap-in is added to the SharePoint 2010 Management Shell.

To run SharePoint 2010 cmdlets, you must be a member of the SharePoint_Shell_Access role on the configuration database and a member of the WSS_ADMIN_WPG local group on the computer where SharePoint 2010 Products is installed. To run scripts in Windows PowerShell or SharePoint 2010 Management Shell, you must set the execution policy by using the **set-executionpolicy** cmdlet, using the **unrestricted** parameter. For more information about the **Add-PSSnapin** cmdlet, see Add-PSSnapin (*http://go.microsoft.com/fwlink/?LinkId=188450&clcid=0x409*). For more information about how to use Windows PowerShell with SharePoint 2010 Products, see SharePoint 2010 Products administration by using Windows PowerShell.

To activate the Office Web Apps Feature on all site collections by using Windows PowerShell

1. Using Notepad, open a new a text file and then copy and paste the following script into the file.

```
$webAppsFeatureId = $(Get-SPFeature -limit all | where {$_.displayname -eq
"OfficeWebApps"}).Id

Get-SPSite -limit ALL |foreach{Enable-SPFeature $webAppsFeatureId -url $_.URL }
```

2. Save the file with a **.ps1** file name extension to a folder where you run scripts (typically C:\scripts).
3. From the Windows PowerShell command prompt (that is, PS C:\>), type the following command and press ENTER:

   C:\<path>\<filename>.ps1

To activate the Office Web Apps Feature on a site collection on the Site collection features page

1. In a browser, in the SharePoint site, click **Site Actions**, and then click **Site Settings**.
2. On the Site Settings page, in **Site Collection Administration**, click **Site Collection Features**.
3. On the Features page, for **Office Web Apps**, click **Activate**.

To activate the Office Web Apps Feature on a single site collection by using Windows PowerShell

1. Using Notepad, open a new text file and then copy and paste the following script into the file.

```
$webAppsFeatureId = $(Get-SPFeature -limit all | where {$_.displayname -eq
"OfficeWebApps"}).Id

$singleSiteCollection = Get-SPSite -Identity http://<site_name> Enable-SPFeature
$webAppsFeatureId -Url $singleSiteCollection.URL
```

2. Specify the following parameters:

Parameter	Value
–Identity	URL

3. Save the file with a **.ps1** file name extension to a folder where you run scripts (typically C:\scripts).
4. In the Windows PowerShell console, at the command prompt (that is, PS C:\>), type the following command and press ENTER:

   C:\<path>\<filename>.ps1

To deactivate the Office Web Apps Feature on a single site collection by using Windows PowerShell

1. Using Notepad, open a new text file and then copy and paste the following script into the file.

```
$webAppsFeatureId = $(Get-SPFeature -limit all | where {$_.displayname -eq
"OfficeWebApps"}).Id

$singleSiteCollection = Get-SPSite -Identity http://<site_name>

Disable-SPFeature $webAppsFeatureId -Url $singleSiteCollection.URL
```

2. Specify the following parameters:

Parameter	Value
–Identity	URL

3. Save the file with a **.ps1** file name extension to a folder where you run scripts (typically C:\scripts).
4. In the Windows PowerShell console, at the command prompt (that is, PS C:\>), type the following command and press ENTER:

   C:\<path>\<filename>.ps1

See Also

Get-SPSite (*http://technet.microsoft.com/library/f3422bf4-0f9b-4f22-94c8-2a0606a31b16(Office.14).aspx*)

Get-SPFeature (*http://technet.microsoft.com/library/f00cddab-fc38-42fb-8bb7-c9aec2e99a28(Office.14).aspx*)

Enable-SPFeature (*http://technet.microsoft.com/library/9b68c192-b640-4cb8-8a92-a98008169b27(Office.14).aspx*)

Disable-SPFeature (*http://technet.microsoft.com/library/c10fbc69-088c-4e49-9005-fde54c035f23(Office.14).aspx*)

Manage the Office Web Apps cache

The information that is provided in this article about how to manage the Microsoft Office Web Apps cache is intended for IT pros who maintain Office Web Apps installed on Microsoft SharePoint 2010 Products on a stand-alone server or a server farm. SharePoint 2010 Products in this article refers to Microsoft SharePoint Server 2010 and Microsoft SharePoint Foundation 2010 unless otherwise noted.

Manage the Office Web Apps cache

The Microsoft Word Web App and Microsoft PowerPoint Web App generate a series of images to create a rendition of a document that is viewable in the browser. If Microsoft Silverlight 3 is installed, XAML is used to create the rendition. Creating the rendition can consume large amounts of computer resources. To reduce resource consumption, the Word Web App and PowerPoint Web App store the renditions in a cache, created as part of a SharePoint content database. Renditions in the cache are then used for future requests of a view of the same document. In an environment where most documents change infrequently, but are viewed regularly, maximizing the space dedicated to the cache or the expiration period, can improve performance and reduce resource consumption. In an environment where most documents frequently change, you can optimize performance by reducing the space that is dedicated to the cache, or by reducing the time documents are stored in the cache.

Both the Word Web App and PowerPoint Web App will run in the context of a single SharePoint Web application. Only one cache is created per SharePoint Web application. If multiple site collections that are stored in different content databases have the Office Web Apps Feature activated, they will all use the same cache.

You can configure the cache by using the following settings:

Cache size Space available in bytes to render documents. By default, when you install Office Web Apps on a stand-alone server or a server farm, the cache available to render documents is 100 GB (1073741824 in bytes).

Expiration period Number of days that document renditions can remain in the cache before they are purged and rendered again. By default, the cache expiration period is 30 days.

Location Database location for the cache (a SharePoint content database). You may want to isolate the Office Web Apps cache from other content so that you can exclude it from backups. To do so, you can create a separate content database, set it to contain only one site collection, and then configure the Office Web Apps cache to use that database.

Timer jobs Office Web Apps includes two timer jobs associated with the cache. You can view timer job definitions, job status, and edit timer job settings in Central Administration.

Timer job title	Description
Office Web Apps Cache Creation	Ensures that every SharePoint Web application, on which the Office Web Apps are activated, has an Office Web Apps cache associated with it. By default, this timer job runs every 5 minutes.
Office Web Apps Expiration	Expires older documents, and ensures that the Office Web Apps cache does not run out of space. By default, this timer job runs nightly.

Procedures in this task:

- To set the Office Web Apps cache size and expiration period by using Windows PowerShell
- To move the Office Web Apps cache database location by using Windows PowerShell

Some procedures in this task require using Windows PowerShell or SharePoint 2010 Management Shell to run cmdlets. To run SharePoint 2010 cmdlets in Windows PowerShell, you must add the Microsoft.SharePoint.PowerShell snapin by using the **Add-PSSnapin** cmdlet, or you can run the SharePoint 2010 cmdlets by using SharePoint 2010 Management Shell. By default, the Microsoft.SharePoint.PowerShell snapin is already registered, and the snapin is added to the SharePoint 2010 Management Shell. To run SharePoint 2010 cmdlets, you must be a member of the SharePoint_Shell_Access role on the configuration database and a member of the WSS_ADMIN_WPG local group on the computer where SharePoint 2010 Products is installed.

To run scripts in Windows PowerShell or SharePoint 2010 Management Shell, you must set the execution policy by using the **set-executionpolicy** cmdlet, using the **unrestricted** parameter. For more information about the **Add-PSSnapin** cmdlet, see Add-PSSnapin (*http://go.microsoft.com/fwlink/?LinkId=188450&clcid=0x409*). For more information about how to use Windows PowerShell with SharePoint 2010 Products, see SharePoint 2010 Products administration by using Windows PowerShell.

To set the Office Web Apps cache size and expiration period by using Windows PowerShell

1. Using Notepad, create a text file and then copy and paste the following script into the file. This example sets a cache size of 200 in bytes and an expiration period of 30 days.

```
$200gbInBytes = 1024 * 1024 * 1024 * 200

Get-SPWebApplication | Set-SPOfficeWebAppsCache -ExpirationPeriodInDays 30 -
MaxSizeInBytes $200gbInBytes
```

2. Specify the following parameters:

Parameter	Value
–ExpirationPeriodInDays	Number (in days)
–MaxSizeInBytes	Number (in bytes)

3. Save the file with a **.ps1** file name extension to a folder where you run scripts (typically C:\scripts).
4. To run the script, in the Windows PowerShell console, at the command prompt (that is, PS C:\>), type the following command and press ENTER:

   C:\<path>\<filename>.ps1

To move the Office Web Apps cache database location by using Windows PowerShell

1. Using Notepad, create a text file and then copy and paste the following script into the file.

   This example moves the cache associated with the http://contoso Web application to the ContosoDB database.

```
$webapp = "http://contoso"

$newDB = New-SPContentDatabase -Name ContosoDB -WebApplication $webapp -
DatabaseServer ContosoServer1

Get-SPOfficeWebAppsCache -WebApplication $webapp | Move-SPSite -
DestinationDatabase $newDB
```

2. Specify the following parameters:

Parameter	Value
-WebApplication	Web application URL
-Name	Database name
–DatabaseServer	Server name

3. Save the file with a **.ps1** file name extension to a folder where you run scripts (typically C:\scripts).
4. To run the script, in the Windows PowerShell console, at the command prompt (that is, PS C:\>), type the following command and press ENTER:

   C:\<path>\<filename>.ps1

See Also

Get-SPOfficeWebAppsCache

Set-SPOfficeWebAppsCache

Get-SPWebApplication (*http://technet.microsoft.com/library/11d6521f-f99c-433e-9ab5-7cf9e953457a(Office.14).aspx*)

Get-SPContentDatabase (*http://technet.microsoft.com/library/a4a83bb0-0bab-4cad-9b59-0fd89a16f57b(Office.14).aspx*)

Configure the default open behavior for browser-enabled documents (Office Web Apps)

Information provided in this article about how to configure the default open behavior for documents is intended for IT pros who maintain Microsoft Office Web Apps installed on Microsoft SharePoint 2010 Products on a stand-alone server or on a server farm. SharePoint 2010 Products in this article refers to Microsoft SharePoint Server 2010 and Microsoft SharePoint Foundation 2010 unless otherwise noted.

Configure the default open behavior for documents

In SharePoint, you can configure whether browser-enabled documents are opened in a client application or in the browser. By default, when Office Web Apps is installed, Office documents will open in the browser. You can override this setting using the SharePoint OpenInClient feature. The OpenInClient feature can be configured in Central Administration or by using the SPFeature cmdlet in Windows PowerShell.

How documents open in SharePoint varies depending on whether the OpenInClient feature is present, and either enabled or disabled:

- If the OpenInClient feature is not present and Office Web Apps is not installed, documents will open in the client application (SharePoint default).
- If the OpenInClient feature is not present, Office Web Apps is installed and Office Web Apps service applications are activated, documents will open in the browser (Office Web Apps default).
- If the OpenInClient Feature is present and enabled, and Office Web Apps service applications are activated, documents will open in the client application.
- If the OpenInClient Feature is present and disabled, and Office Web Apps service applications are activated, documents in will open in the browser.

Caution:
When you run Setup.exe to install Office Web Apps, setup will take control of the default open behavior in SharePoint to register Word, PowerPoint, Excel, and OneNote documents to be opened in their associated Web app. If a user clicks on a document in SharePoint after Setup.exe is run, but before the Office Web Apps Services and Feature have been activated, the user can get a broken link in the browser. When installing Office Web Apps in a live production environment, we strongly recommend that you enable the OpenInClient Feature before running Office Web Apps setup.

Note:
Configuring the default open behavior for browser-enabled documents will not affect whether users can use Check Out and Send To features in SharePoint to download documents. For information about configuring check out, download, and view permissions in SharePoint, see Security planning for sites and content (SharePoint Server 2010) (*http://technet.microsoft.com/library/85a1866e-2743-4f98-a1ac-9ea61905c6d4(Office.14).aspx*) and Security planning for sites and content (SharePoint Foundation 2010) (*http://technet.microsoft.com/library/8921fbaf-d445-4058-8c10-4f52184357af(Office.14).aspx*).

Procedures in this task:

- To set the default open behavior for site collections by using Central Administration
- To set the default open behavior for site collections by using Windows PowerShell
- To set the default open behavior for a document library by using the document library settings page
- To set the default open behavior for IRM protected document libraries by using Windows PowerShell

Some procedures in this task require using Windows PowerShell or SharePoint 2010 Management Shell to run cmdlets. To run SharePoint 2010 cmdlets in Windows PowerShell, you must add the Microsoft.SharePoint.PowerShell snap-in by using the **Add-PSSnapin** cmdlet, or you can run the SharePoint 2010 cmdlets by using SharePoint 2010 Management Shell. By default, the Microsoft.SharePoint.PowerShell snap-in is already registered, and the snap-in is added to the SharePoint 2010 Management Shell. To run SharePoint 2010 cmdlets, you must be a member of the SharePoint_Shell_Access role on the configuration database and a member of the WSS_ADMIN_WPG local group on the computer where SharePoint 2010 Products is installed. To run scripts in Windows PowerShell or SharePoint 2010 Management Shell, you must set the execution policy by using the **set-executionpolicy** cmdlet, using the **unrestricted** parameter. For more information about the **Add-PSSnapin** cmdlet, see Add-PSSnapin (*http://go.microsoft.com/fwlink/?LinkId=183975&clcid=0x409*). For more information about how to use Windows PowerShell with SharePoint 2010 Products, see SharePoint 2010 Products administration by using Windows PowerShell.

To set the default open behavior for site collections by using Central Administration

1. In SharePoint Central Administration, click **Site Actions**, and then click **Site Settings**.
2. On the Site Settings page, under **Site Collection Administration**, click **Site Collection Features**.
3. On the Features page, for the **Open Documents in Client Applications by Default** feature, click **Activate** (OpenInClient Feature is enabled) to open documents in the client application. Click **Deactivate** (OpenInClient Feature is disabled) to open documents in the browser.

To set the default open behavior for site collections by using Windows PowerShell

1. Using Notepad, open a new text file and then copy and paste the following script into the file.

   The following example sets the default open behavior for all documents in all sites to open in their associated client application (if available).

   ```
   $defaultOpenBehaviorFeatureId = $(Get-SPFeature -limit all | where {$_.displayname -eq "OpenInClient"}).Id
   Get-SPSite -limit ALL |foreach{ Enable-SPFeature $defaultOpenBehaviorFeatureId -url $_.URL }
   ```

   The following example sets the default open behavior for all documents in all sites to open in the browser.

   ```
   $defaultOpenBehaviorFeatureId = $(Get-SPFeature -limit all | where {$_.displayname -eq "OpenInClient"}).Id
   Get-SPSite -limit ALL |foreach{ Disable-SPFeature $defaultOpenBehaviorFeatureId -url $_.URL }
   ```

2. Specify the following parameters:

Parameter	Value
–limit	The type must be a valid non-negative number. Specify ALL to return all site collections for the given scope.

3. Save the file with a **.ps1** file name extension to a folder where you run scripts (typically C:\scripts).
4. In the Windows PowerShell console, at the command prompt (that is, PS C:\>), type the following command and press ENTER:

   C:\<path>\<filename>.ps1

To set the default open behavior for a document library by using the document library settings page

1. On the **<document library>: All Documents** page, under **Library Tools**, click **Library**.
2. In the library toolbar, click **Library Settings**.
3. On the **Document Library Settings** page, click **Advanced Settings**.
4. On the **Advanced Settings** page, in **Opening Document in the Browser**, select one of the following options:

   Open in the client application When a user clicks on a document in this library, the document will open in the corresponding client application (if available).

Open in the browser When a user clicks on a document in this library, the document will open in the Web browser Web app for that document type. When the document is opened in the Web app, the user can then decide to open the document in the client application

Use the server default When a user clicks on a document in this library, the document will open by using the default open behavior specified for the server on which SharePoint 2010 Products is installed.

To set the default open behavior for IRM protected document libraries by using Windows PowerShell

1. Using Notepad, open a new text file and then copy and paste the following script into the file.

   The following example sets the default open behavior for IRM protected document libraries in all sites to open in their associated client application (if available).

```
Get-SPWeb -site http://contoso | % {$_.Lists} | where {$_.IrmEnabled -eq $true} |
% {$_.DefaultItemOpen =[Microsoft.Sharepoint.DefaultItemOpen]::PreferClient;
$_.Update()}
```

2. Specify the following parameters:

Parameter	Value
–site	URL

3. Save the file with a **.ps1** file name extension to a folder where you run scripts (typically C:\scripts).
4. In the Windows PowerShell console, at the command prompt (that is, PS C:\>), type the following command and press ENTER:

   C:\<path>\<filename>.ps1

See Also

Get-SPWeb (*http://technet.microsoft.com/library/9bf9284f-e3b9-439d-8a5f-74020e1eccaf(Office.14).aspx*)

Get-SPSite (*http://technet.microsoft.com/library/f3422bf4-0f9b-4f22-94c8-2a0606a31b16(Office.14).aspx*)

Get-SPFeature (*http://technet.microsoft.com/library/f00cddab-fc38-42fb-8bb7-c9aec2e99a28(Office.14).aspx*)

Configure PowerPoint service application settings

The information provided in this article about configuring the PowerPoint service application is aimed at IT pros who maintain Microsoft Office Web Apps installed on Microsoft SharePoint 2010 Products on a stand-alone server or on a server farm. SharePoint 2010 Products in this article refers to Microsoft SharePoint Server 2010 and Microsoft SharePoint Foundation 2010 unless otherwise noted.

This article contains the following sections:

- Configure the PowerPoint service application settings
- Set maximum worker processes by using Windows PowerShell

Configure the PowerPoint service application settings

The PowerPoint service application settings page allows administrators to set supported file formats, determine or specify a broadcast site URL, and disable presentation scanning.

To configure the PowerPoint service application settings

1. Verify that you have the following administrative credentials:
   - To change settings for the Office Web Apps service applications, you must be a member of the Farm Administrators SharePoint group on the computer that is running the SharePoint Central Administration Web site.
2. On the Central Administration Web site, in the **Application Management** section, click **Manage service applications**.
3. On the **Service Applications** page, click **PowerPoint Service Application**.
4. On the **PowerPoint Service Application** page, configure the following settings:

   Supported File Formats Specify which presentation types users will be able to view by using this PowerPoint service application. If the file format is not selected, consumers of this service will receive an error when they attempt to view this file type in the Web browser.

   - **Office Open XML Presentation** Select this setting to support .pptx, .pptm, .potx, and .potm file formats.
   - **PowerPoint 97-2003 Presentation** Select this setting to support .ppt and .pot file formats.

   Broadcast Site Users can connect to a broadcast site from Microsoft PowerPoint 2010 and broadcast a slide show to remote attendees who can watch by using a Web browser. When the PowerPoint service application is created, a default site will be created and its URL displayed.

You can create another broadcast site at another location by creating a new site collection using the PowerPoint Broadcast Site template.

PowerPoint 97-2003 Presentation Scanning To provide added security when you are loading Microsoft PowerPoint 97 through PowerPoint 2003 documents, the PowerPoint service application performs extra checks for malicious content before opening the document in the browser. These checks can have an effect on overall server performance. Disable this setting only if you trust all documents that are loaded by this service instance. By default, document scanning is enabled. To disable scanning, select **Disable PowerPoint 97-2003 presentation scanning**.

Set maximum worker processes by using Windows PowerShell

You can set the maximum number of worker processes that are permitted to service requests for the PowerPoint service application. Set the value for maximum worker processes high enough to allow for a maximum number of service requests without diminishing worker processes from other applications. A low number of worker processes can reduce performance and resource consumption of the PowerPoint service application. Other factors such as cache size and computer specifications might suggest a different number of maximum worker processes. For typical use, it is recommended that you set two worker processes for each service application per core processor.

Some procedures in this task require using Windows PowerShell or SharePoint 2010 Management Shell to run cmdlets. To run SharePoint 2010 cmdlets in Windows PowerShell, you must add the Microsoft.SharePoint.PowerShell snapin by using the **Add-PSSnapin** cmdlet, or you can run the SharePoint 2010 cmdlets by using SharePoint 2010 Management Shell. By default, the Microsoft.SharePoint.PowerShell snapin is already registered, and the snapin is added to the SharePoint 2010 Management Shell. To run SharePoint 2010 cmdlets, you must be a member of the SharePoint_Shell_Access role on the configuration database and a member of the WSS_ADMIN_WPG local group on the computer where SharePoint 2010 Products is installed. To run scripts in Windows PowerShell or SharePoint 2010 Management Shell, you must set the execution policy by using the **set-executionpolicy** cmdlet, using the **unrestricted** parameter.

For more information about the **Add-PSSnapin** cmdlet, see Add-PSSnapin (*http://go.microsoft.com/fwlink/?LinkId=188450&clcid=0x409*). For more information about using Windows PowerShell with SharePoint 2010 Products, see SharePoint 2010 Products administration by using Windows PowerShell.

To set maximum worker processes for the PowerPoint Viewing service application by using Windows PowerShell

1. Using Notepad, create a text file and then copy and paste the following script into the file.

This example specifies two worker processes for each core processor on a server with four core processors (2 worker processes x 4 core processors):

```
Get-SPPowerPointServiceApplication | Set-SPPowerPointServiceApplication -
ViewingWorkerProcessCount 8
```

2. Specify the following parameters:

Parameter	Value
-ViewingWorkerProcessCount	Number (total worker processes)

3. Save the file with a **.ps1** file name extension to a folder where you run scripts (typically C:\scripts).
4. In the Windows PowerShell console, at the command prompt (that is, PS C:\>), type the following command and press ENTER:

   C:\<path>\<filename>.ps1

See Also

Get-SPPowerPointServiceApplication

Set-SPPowerPointServiceApplication

Configure Word Viewing service settings

The information provided in this article about configuring the Word Viewing service application is aimed at IT pros who maintain Microsoft Office Web Apps installed on Microsoft SharePoint 2010 Products on a stand-alone server or on a server farm. SharePoint 2010 Products in this article refers to Microsoft SharePoint Server 2010 and Microsoft SharePoint Foundation 2010 unless otherwise noted.

This article contains the following sections:

- Configure the Word Viewing service application settings
- Set maximum worker processes by using Windows PowerShell
- Set maximum rendering time by using Windows Powershell

Note:
Some procedures in this article require using Windows PowerShell or SharePoint 2010 Management Shell to run cmdlets. To run SharePoint 2010 cmdlets in Windows PowerShell, you must add the Microsoft.SharePoint.PowerShell snapin by using the **Add-PSSnapin** cmdlet, or you can run the SharePoint 2010 cmdlets by using SharePoint 2010 Management Shell. By default, the Microsoft.SharePoint.PowerShell snapin is already registered, and the snapin is added to the SharePoint 2010 Management Shell. To run SharePoint 2010 cmdlets, you must be a member of the SharePoint_Shell_Access role on the configuration database and a member of the WSS_ADMIN_WPG local group on the computer where SharePoint 2010 Products is installed. To run scripts in Windows PowerShell or SharePoint 2010 Management Shell, you must set the execution policy by using the **set-executionpolicy** cmdlet, using the **unrestricted** parameter. For more information about the **Add-PSSnapin** cmdlet, see Add-PSSnapin (*http://go.microsoft.com/fwlink/?LinkId=188450&clcid=0x409*). For more information about using Windows PowerShell with SharePoint 2010 Products, see SharePoint 2010 Products administration by using Windows PowerShell.

Configure the Word Viewing service application settings

The Word Viewing service application settings page allows administrators to set supported file formats, embedded font support, document scanning, and process settings for users viewing Word documents in a Web browser by using the Word Viewing Web App.

To configure the Word Viewing service application settings

1. Verify that you have the following administrative credentials:
   - To change settings for the Office Web Apps service applications, you must be a member of

the Farm Administrators SharePoint group on the computer that is running the SharePoint Central Administration Web site.

2. On the Central Administration Web site, in the **Application Management** section, click **Manage service applications**.
3. On the **Service Applications** page, click **Word Viewing Service**.
4. On the **Word Viewing Service** page, configure the following settings:

   Supported File Formats Viewing Specify which document types users can view by using this Word Viewing service application. If the file format is not selected, consumers of this service will receive an error message when they attempt to view this file type in the Web browser. By default, all file formats are supported.

   - **Open XML Document** Select this setting to support .docx, .dotx, .docm, and .dotm file formats.
   - **Word 97-2003 Document** Select this setting to support .doc and .dot file formats.

   Embedded Font Support To preserve visual fidelity across different computers, a user can choose to embed a font within the document. You can use this setting to determine whether embedded fonts are used when viewing or converting documents and presentations. By default, embedded fonts are enabled.

   Word 97-2003 Document Scanning To provide added security when loading Microsoft Word 97 through Word 2003 documents, the Word Viewing service application performs extra checks for malicious content before opening the document in the browser. These checks can have an effect on overall server performance. Disable this setting only if you trust all documents that are loaded by this service instance. By default, document scanning is enabled. To disable scanning, under **Disable Word 97-2003 presentation scanning** select **Yes**.

   Recycle Threshold Specify the number of files which a process should be allowed to render before the process is recycled. Whenever this threshold is reached, the worker processes will be recycled, which prevents an application from indefinitely increasing memory consumption. A change to this setting requires using the IISReset command to restart IIS. By default, the recycle threshold is 5.

   Total Active Processes Specify the number of worker processes that are dedicated to viewing Word documents. Set the value for total worker processes high enough to allow for a maximum number of service requests without diminishing worker processes from other applications. A low number of worker processes can reduce the performance and resource consumption of the Word Viewing service application. Other factors such as cache size and computer specifications might suggest a different number of total worker processes. For typical use, it is recommended that you set two worker processes for each service application per core processor. By default, three total worker processes are set. You can also configure this setting by using Windows PowerShell. For more information, see Set maximum worker processes by using Windows PowerShell later in this article.

Set maximum worker processes by using Windows PowerShell

You can set the maximum number of worker processes that are permitted to service requests for the Word Viewing service application. Set the value for maximum worker processes high enough to allow for a maximum number of service requests without diminishing worker processes from other applications. A low number of worker processes can reduce performance and resource consumption of the Word Viewing service application. Other factors such as cache size and computer specifications might suggest a different number of maximum worker processes. For typical use, it is recommended you set two worker processes for each service application per core processor.

To set maximum worker processes for the Word Viewing service application by using Windows PowerShell

1. Using Notepad, create a text file and then copy and paste the following script into the file.

   This example specifies two worker processes for each core processor on a server with four core processors:

   ```
   $serviceAppId = (Get-SPServiceApplication | where {$_.DisplayName -eq "Word Viewing Service"}).ID
   Set-SPWordViewingServiceApplication -Identity $serviceAppId -MaxActiveProcessCount 8
   ```

2. Specify the following parameters:

Parameter	Value
-MaxActiveProcessCount	Number (total worker processes)
-eq	Name of the word viewing service application whose value you wish to change

3. Save the file with a **.ps1** file name extension to a folder where you run scripts (typically C:\scripts).
4. In the Windows PowerShell console, at the command prompt (that is, PS C:\>), type the following command and press ENTER:

   C:\<path>\<filename>.ps1

Note:

Changes to the MaxActiveProcessCount setting require an **IISReset** to take effect.

Set maximum rendering time by using Windows Powershell

Note:
Numbers provided in this section are general estimations. The number of pages per megabyte for a Word document can vary depending on page elements, graphics, etc. The number of pages rendered per minute by a worker process can vary depending on performance factors specific to your environment. Be sure to test your configuration to make sure any changes made are suitable for your environment.

By default, SharePoint supports uploading documents up to 50 megabytes (approximately 3100 pages for a Word document). This limit can be changed by administrators using the Maximum Upload Size setting for the Web application (SharePoint Web application General Settings) to accommodate extremely large documents.

For more information about setting the Maximum Upload Size setting for a Web application, see Configure settings for a Web application (SharePoint Server 2010) (*http://technet.microsoft.com/library/2687d49f-b114-4f16-a4c0-5f2bfe496c92(Office.14).aspx*) or Configure settings for a Web application (SharePoint Foundation 2010) (*http://technet.microsoft.com/library/0e4c6149-58c4-416e-aaeb-81c9c79f0322(Office.14).aspx*).

The Word Web App supports rendering documents limited only by the Maximum Upload Size specified in SharePoint. However, the number of pages rendered by a single worker process may limit rendering time. To support rendering extremely large Word documents, you can set the maximum file rendering time (MaxRenderingLifetimeInSeconds) setting for the Word Viewing service application by using Windows PowerShell. Once fully rendered, a document will remain in the cache, and will not need to be rendered again unless it has been purged from the cache.

A Word document may contain approximately 63 pages per megabyte. Typically, up to 700 pages per minute can be rendered by a worker process. To calculate the maximum rendering time required, divide the maximum number of pages of the largest documents by 700. This will determine the number of minutes required to render the documents. Then multiply the number of minutes by 60 to determine the maximum rendering time in seconds.

For example, to support rendering a document with 7000 pages (approximately 111 megabytes): **7000/700 = 10 minutes. 10 x 60 = 600 seconds.**

To set maximum rendering time for the Word Viewing service application by using Windows PowerShell

1. Using Notepad, create a text file and then copy and paste the following script into the file.

This example specifies a MaxRenderingLifetimeInSeconds setting of 600 seconds (10 mins):

```
$serviceAppId = (Get-SPServiceApplication | where {$_.DisplayName -eq "Word
Viewing Service"}).ID
Set-SPWordViewingServiceApplication -Identity $serviceAppId -
MaxRenderingLifetimeInSeconds 600
```

2. Specify the following parameters:

Parameter	Value
-MaxRenderingLifetimeInSeconds	Number (time, in seconds, that a worker process is given to process a particular rendition)
-eq	Name of the word viewing service application whose value that you want to change

3. Save the file with a **.ps1** file name extension to a folder where you run scripts (typically C:\scripts).
4. In the Windows PowerShell console, at the command prompt (that is, PS C:\>), type the following command and press ENTER:

   C:\<path>\<filename>.ps1

Note:

Changes to the MaxRenderingLifetimeInSeconds setting require an **IISReset** to take effect.

See Also

Set-SPWordViewingServiceApplication

Configure Excel Services Application settings (Office Web Apps)

The information that is provided in this article about how to configure Excel Services Application settings is intended for IT pros who maintain Microsoft Office Web Apps installed on Microsoft SharePoint 2010 Products on a stand-alone server or on a server farm.

Excel Services Application is included in Microsoft SharePoint Server 2010 and Office Web Apps installed on Microsoft SharePoint Foundation 2010. When you install Office Web Apps on SharePoint Server 2010, the Microsoft Excel Web App will use Excel Services Application in SharePoint Server 2010. Excel Services Application is a shared service that you can use to view and edit Excel workbooks. Configuration settings on the Excel Services Application Web service application page will affect both Excel Services Application within the context of SharePoint Server 2010 and how Excel Services Application is used by the Excel Web App. For more information about how to configure Excel Services Application, see Excel Services administration (SharePoint Server 2010) (*http://technet.microsoft.com/library/a96cfa2d-87f8-4991-801c-fd5482d49d0b(Office.14).aspx*)

Technical reference (Office Web Apps)

Technical references includes information about the Office Web Apps provider for Windows PowerShell and other useful reference information about general settings, security, and tools.

In this section:

- Windows PowerShell for Office Web Apps

Windows PowerShell for Office Web Apps

Windows PowerShell can be used with Office Web Apps to perform both simple and complex administrative tasks. Information about cmdlets in Office Web Apps is currently accessible using command-line Help. For information about Windows PowerShell in general, visit the Windows PowerShell Technology Center (*http://go.microsoft.com/fwlink/?LinkId=167152*).

In this section:

- SharePoint 2010 Products administration by using Windows PowerShell
- Windows PowerShell for Office Web Apps reference

SharePoint 2010 Products administration by using Windows PowerShell

This article describes how to use Windows PowerShell with Microsoft SharePoint 2010 Products and provides an overview of basic Windows PowerShell cmdlets and concepts necessary to get the most benefit from this powerful functionality.

In this article:

- Overview
- Accessing Windows PowerShell for SharePoint 2010 Products
- Permissions
- Learning Windows PowerShell

Overview

Windows PowerShell is a command-line scripting tool that provides an administrator full access to applicable application programming interfaces (APIs), along with the ability to unlock the capability to interact directly with SharePoint 2010 Products to manipulate Web applications, site collections, sites, lists and much more. In addition, the administrator has the ability to script cmdlets (pronounced "command-lets"), which makes for an improved experience from past product versions.

Windows PowerShell 2.0 is a pre-requisite for installing SharePoint 2010 Products. It will be installed, if necessary, when you run the Microsoft SharePoint Products Preparation Tool. By default, Windows PowerShell is located at the following path: **<%SystemRoot%>\System32\WindowsPowerShell\v1.0\PowerShell.exe**.

Note:
Windows PowerShell 2.0 is backward compatible with Windows PowerShell 1.0 and subsequently installs to the v1.0 folder.

For a list of new features for Windows PowerShell version 2.0, see About Windows PowerShell 2.0 (*http://go.microsoft.com/fwlink/?LinkId=113247*).

We recommend that you use Windows PowerShell when performing command-line administrative tasks. The Stsadm command-line tool has been deprecated, but is included to support compatibility with previous product versions.

Accessing Windows PowerShell for SharePoint 2010 Products

After installing SharePoint 2010 Products, the applicable Windows PowerShell cmdlets are available by using the SharePoint 2010 Management Shell, or by using the Windows PowerShell console. With the management shell, you can manage every aspect of SharePoint 2010 Products. You can create new site collections, Web applications, user accounts, service applications, proxies, and more. The commands from the management shell output SharePoint objects based on the Microsoft .NET Platform. These objects can be applied as input to subsequent commands or stored in local variables for later use.

With the management shell, you do not have to register the snap-in that contains the cmdlets. Registration of the Microsoft.SharePoint.PowerShell.dll module for SharePoint 2010 cmdlets is automatic, as a result of the line `Add-PSSnapin Microsoft.SharePoint.PowerShell` in the SharePoint.ps1 file located in *%CommonProgramFiles%*\Microsoft Shared\Web Server Extensions\14\Config\PowerShell\Registration. If you choose to use the Windows PowerShell console, you must register this snap-in manually.

Whether you are using the management shell or the Windows PowerShell console, you can also load additional snap-ins. For more information, see Customizing Profiles (*http://go.microsoft.com/fwlink/?LinkId=183166*).

To access the SharePoint 2010 Management Shell

1. On the **Start** menu, click **All Programs**.
2. Click **Microsoft SharePoint 2010 Products**.
3. Click **SharePoint 2010 Management Shell**.

Note:

The SharePoint 2010 Management Shell and the Windows PowerShell console also differ in the use of the `ReuseThread` option, which defines how the threading model is used. The management shell's use is defined by this line, `{Host.Runspace.ThreadOptions = "ReuseThread"}`, which is in the SharePoint.ps1 file. For more information, see PS Thread Options (*http://go.microsoft.com/fwlink/?LinkId=183145*).

Permissions

Before you can use the management shell and the Windows PowerShell cmdlets, verify that you meet the following minimum requirements: See Add-SPShellAdmin.

If you do not have membership in the SharePoint_Shell_Access role or WSS_Admin_WPG local group, use the Add-SPShellAdmin cmdlet. When the **Add-SPShellAdmin** cmdlet is used, the user is added to the **WSS_Admin_WPG** group in all front-end Web servers and is added to the

SharePoint_Shell_Access role. If the target database does not have a **SharePoint_Shell_Access** role, the role is automatically created. Once the **Add-SPShellAdmin** cmdlet has been run, the user can run SharePoint 2010 Windows PowerShell cmdlets in a multiple-server farm environment.

When you run the **Add-SPShellAdmin** cmdlet to add a user to the **SharePoint_Shell_Access** role, you must have the following security permissions:

- **Securityadmin** server role access on the SQL instance and the **db_owner** role in a database.
- Administrative permission on the local computer.

Note:
Typically, the person that will use the **Add-SPShellAdmin** cmdlet must be associated with the user account that was used for Setup.

You must run the **Add-SPShellAdmin** cmdlet for all databases to which you want to grant access. If no database is specified, the farm configuration database is used. If you do specify a database, the farm content database will be included in addition to the farm configuration database you specify.

To see a list of all of the ***SPShellAdmin** cmdlets, from a Windows PowerShell command prompt, type `Get-Command -Noun SPShellAdmin`.

Learning Windows PowerShell

There are several Windows PowerShell learning resources for SharePoint IT professionals who are not familiar with Windows PowerShell.

TechNet Scripting Center

The TechNet Scripting Center includes many resources for learning the basics of using Windows PowerShell. It also contains script repositories with samples of scripts commonly used with various Microsoft products. The following table shows the main learning resources.

Page	Description
Windows PowerShell Documentation on TechNet (*http://go.microsoft.com/fwlink/?LinkId=187813*)	This section of the TechNet Library contains Web copies of the core Windows PowerShell Get-Help topics. The section also has Web copies of the Windows PowerShell Getting Started document, the PowerShell.exe help, and a Windows PowerShell primer.
Scripting With Windows PowerShell (*http://go.microsoft.com/fwlink/?LinkId=187815*)	The home page for Windows PowerShell scripting learning resources.
Windows PowerShell Owner's Manual (*http://go.microsoft.com/fwlink/?LinkId=187817*)	Web-based guide for getting started with Windows PowerShell.

Page	Description
Windows PowerShell Quick Reference (*http://go.microsoft.com/fwlink/?LinkId=187819*)	Downloadable copy of the Quick Reference document installed with Windows PowerShell.

Windows PowerShell documents

Windows PowerShell installs the following documents in the Windows PowerShell 1.0 program group. They can also be installed after downloading the Windows PowerShell 1.0 Documentation Pack (*http://go.microsoft.com/fwlink/?LinkId=187822*).

File	Description
GettingStarted.rtf	Describes how to start using Windows PowerShell.
UsersGuide.rtf	A primer for the Windows PowerShell environment and language.
QuadFold.rtf	A printable quick reference document for the commonly used Windows PowerShell syntax and commands. Also available as a download from the TechNet Scripting Center.

As you read these resources, consider that the following concepts and cmdlets are useful ones to learn before using Windows PowerShell for SharePoint 2010 Products:

- Get-Command (*http://go.microsoft.com/fwlink/?LinkId=171069*)
- Get-Member (*http://go.microsoft.com/fwlink/?LinkId=171070*)
- Get-Help (*http://go.microsoft.com/fwlink/?LinkId=171068*).
- Aliasing (*http://go.microsoft.com/fwlink/?LinkId=113207*)
- Piping and the Pipeline in Windows PowerShell (*http://go.microsoft.com/fwlink/?LinkId=187808*)
- Cmdlet Parameter Sets (*http://go.microsoft.com/fwlink/?LinkId=187810*)
- Foreach-Object (*http://go.microsoft.com/fwlink/?LinkId=187812*)
- Where-Object (*http://go.microsoft.com/fwlink/?LinkId=187811*)

Windows PowerShell for Office Web Apps reference

The following articles list Windows PowerShell cmdlets for Office Web Apps.

Category article
Excel Web App cmdlets (Office Web Apps)
Office Web Apps cmdlets

Excel Web App cmdlets (Office Web Apps)

Excel Services Application is a service application that you can use to share, secure, and manage Microsoft Excel 2010 workbooks (.xlsx, .xlsb, and .xlsm extensions) as interactive reports in a consistent way throughout the enterprise.

All Excel Services Application settings will support backup and recovery regardless of whether there is a UI setting in the Central Administration Web site. This means that all global settings, trusted locations, user-defined-functions, data connection libraries, and trusted data provider settings will support backup and recovery. However, the following items will not be backed up as part of the backup and recovery process:

- User-defined function assemblies. These assemblies will not be backed up or restored because they are binary files and not simply settings; however, the data in the list of trusted user-defined functions is backed up.
- Trusted data provider binaries. (The full list of trusted data providers and all associated properties, however, are backed up.)
- Any Excel Web Access (EWA) Web Part properties.
- Office data connection (.odc extension) files and workbook files.

Note:
URLs of any kind (that is, URLs to trusted locations, user-defined functions, or data connection libraries) must be manually fixed after a restore operation because they may no longer be valid.

In addition to using Windows PowerShell for Excel Services Application backup and recovery functionality, you can use Windows PowerShell to provision the service application from scratch or provision a new instance that uses settings from a previous backup, and manage and configure trusted file locations, data providers, data connection libraries, and settings for user-defined function assemblies. You can also change global settings for Excel Services Application.

Cmdlet Name	Description
Get-SPExcelBlockedFileType	Returns a file type or list of file types that are prevented from being loaded.
Get-SPExcelDataConnectionLibrary	Returns a trusted data connection library or a list of trusted data connection libraries.
Get-SPExcelDataProvider	Returns a safe data provider or a list of safe data providers.

Get-SPExcelFileLocation	Returns a trusted file location or a list of trusted file locations.
Get-SPExcelServiceApplication	Returns an **SPExcelServiceApplication** object.
Get-SPExcelUserDefinedFunction	Returns a user-defined function or a collection of user-defined functions.
New-SPExcelBlockedFileType	Adds a file type to the list of file types that Excel Services Application prevents from being loaded.
New-SPExcelDataConnectionLibrary	Adds a new data connection library to Excel Services Application.
New-SPExcelDataProvider	Adds a new safe data provider to Excel Services Application.
New-SPExcelFileLocation	Adds a new trusted location to Excel Services Application.
New-SPExcelServiceApplication	Creates a new instance of Excel Services Application.
New-SPExcelUserDefinedFunction	Adds a new user-defined function to Excel Services Application.
Remove-SPExcelBlockedFileType	Removes an entry from the list of file types that are prevented from being loaded on Excel Services Application.
Remove-SPExcelDataConnectionLibrary	Removes a data connection library from Excel Services Application.
Remove-SPExcelDataProvider	Removes a data provider from Excel Services Application.
Remove-SPExcelFileLocation	Removes a trusted file location from Excel Services Application.
Remove-SPExcelUserDefinedFunction	Removes a user-defined function from Excel Services Application.
Set-SPExcelDataConnectionLibrary	Sets properties of a data connection library for Excel Services Application.
Set-SPExcelDataProvider	Sets properties of a safe data provider for Excel Services Application.
Set-SPExcelFileLocation	Sets properties of a trusted file location for Excel Services Application.

Set-SPExcelServiceApplication	Sets global properties for Excel Services Application.
Set-SPExcelUserDefinedFunction	Sets properties of a user-defined function in Excel Services Application.

See Also

Introduction to Excel Services (*http://go.microsoft.com/fwlink/?LinkId=188453&clcid=0x409*)

Get-SPExcelBlockedFileType

Returns a file type or list of file types that are prevented from loading.

Syntax

Get-SPExcelBlockedFileType [[-Identity] <SPExcelBlockedFileTypePipeBind>] -ExcelServiceApplication <SPExcelServiceApplicationPipeBind> [-AssignmentCollection <SPAssignmentCollection>]

Detailed Description

The **Get-SPExcelBlockedFileType** cmdlet reads a file type or a list of file types that cannot be loaded in Excel Services Application. Excel Services Application does not load any files that correspond to a file type in this list.

Parameters

Parameter	Required	Type	Description
Identity	Optional	Microsoft.Office.Excel.Server.Cmdlet.SPExcelBlockedFileTypePipeBind	Specifies the **BlockedFileType** object to return. If the **Identity** parameter is not specified, this cmdlet returns the list of file extensions that are currently prevented from loading on Excel Services Application. The type must be a valid GUID, in the form 12345678-90ab-cdef-1234-567890bcdefgh; a valid name of a file

Parameter	Required	Type	Description
			type (for example, FileType1); one of the following file types: XLSX, XLSB, XLSM; or an instance of a valid **SPExcelBlockedFileType** object.
ExcelServiceApplication	Yes	Microsoft.Office.Excel.Server.Cmdlet.SPExcelServiceApplicationPipeBind	Specifies the Excel Services Application Web service application that contains the **SPBlockedExcelFileType** list object. The type must be a valid GUID, in the form 12345678-90ab-cdef-1234-567890bcdefgh; a valid name of an Excel Services Application Web service application in the farm (for example, MyExcelService1); or an instance of a valid **SPExcelServiceApplication** object.
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	Manages objects for the purpose of proper disposal. Use of objects, such as **SPWeb** or **SPSite**, can use large amounts of memory and use of these

Parameter	Required	Type	Description
			objects in Windows PowerShell scripts requires proper memory management. Using the **SPAssignment** object, you can assign objects to a variable and dispose of the objects after they are needed to free up memory. When **SPWeb**, **SPSite**, or **SPSiteAdministration** objects are used, the objects are automatically disposed of if an assignment collection or the **Global** parameter is not used. **Note:** When the **Global** parameter is used, all objects are contained in the global store. If objects are not immediately used, or disposed of by using the **Stop-SPAssignm**

Parameter	Required	Type	Description
			ent command, an out-of-memory scenario can occur.

AutoGenParams

Parameter	Required	Type	Description
Identity	Optional	Microsoft.Office.Excel.Server.Cmdlet.SPExcelBlockedFileTypePipeBind	
ExcelServiceApplication	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelServiceApplicationPipeBind	
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	

Input Types

Return Types

Example

Description

----------EXAMPLE 1----------

Code

```
Get-SPExcelServiceApplication –Identity "MyExcelService" | Get-SPExcelBlockedFileType | format-table
```

Comments

This example returns a list of file extensions that currently cannot be loaded by using Excel Services Application, and then formats the result in a table.

See Also

New-SPExcelBlockedFileType

Remove-SPExcelBlockedFileType

Get-SPExcelDataConnectionLibrary

Returns a trusted data connection library or a list of trusted data connection libraries.

Syntax

Get-SPExcelDataConnectionLibrary [[-Identity] <SPExcelDCLPipeBind>] -ExcelServiceApplication <SPExcelServiceApplicationPipeBind> [-AssignmentCollection <SPAssignmentCollection>]

Detailed Description

The **Get-SPExcelDataConnectionLibrary** cmdlet gets a library of data connection files that Excel Services Application and the Excel client use to connect to an external data source. Excel Services Application loads data connection files only if they are stored in a data connection library that is in the list of trusted data connection libraries.

Parameters

Parameter	Required	Type	Description
Identity	Optional	Microsoft.Office.Excel.Server.Cmdlet.SPExcelDCLPipeBind	Specifies the **DataConnectionLibrary** object to return. If the **identity** parameter is not specified, this cmdlet returns all the data connection libraries that have connection files that can be loaded by Excel Services Application. The type must be a valid GUID, in the form 12345678-90ab-cdef-1234-567890bcdefgh; a valid name of a data connection library (for example,

Parameter	Required	Type	Description
			DataConnectionLib1); a valid URL, in the form http://server_name; or an instance of a valid **SPExcelDCL** object.
ExcelServiceApplication	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelServiceApplicationPipeBind	Specifies the Excel Services Application Web service application that contains the **SPExcelDataConnectionLibrary** list object. The type must be a valid GUID, in the form 12345678-90ab-cdef-1234-567890bcdefgh; a valid name of an Excel Services Application Web service application in the farm (for example, MyExcelService1); or an instance of a valid **SPExcelServiceApplication** object.
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	Manages objects for the purpose of proper disposal. Use of objects, such as **SPWeb** or **SPSite**, can use large amounts of memory and use of these objects in Windows PowerShell scripts requires proper memory management. Using the **SPAssignment**

Parameter	Required	Type	Description
			object, you can assign objects to a variable and dispose of the objects after they are needed to free up memory. When **SPWeb**, **SPSite**, or **SPSiteAdministration** objects are used, the objects are automatically disposed of if an assignment collection or the **Global** parameter is not used. **Note:** When the **Global** parameter is used, all objects are contained in the global store. If objects are not immediately used, or disposed of by using the **Stop-SPAssignment** command, an out-of-memory scenario can occur.

AutoGenParams

Parameter	Required	Type	Description
Identity	Optional	Microsoft.Office.Excel.Server.Cmdlet.SPExcelDCLPipeBind	
ExcelServiceApplication	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelServiceApplicationPipeBind	
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	

Input Types

Return Types

Example

Description

----------EXAMPLE 1----------

Code

```
Get-SPExcelserviceapplication –Identity "MyExcelService" | Get-SPExcelDataConnectionLibrary
```

Comments

This example displays all the data connection libraries that connection files can be loaded by an Excel Services Application Web service application.

Example

Description

----------EXAMPLE 2----------

Code

```
Get-SPExcelserviceapplication -identity "MyExcelService" | Get-SPExcelDataConnectionLibrary |
where {$_.Description.Contains("Sales Cube")}
```

Comments

This example displays all the data connection libraries that have descriptions containing the words `Sales Cube` from the specified Excel Services Application Web service application.

See Also

New-SPExcelDataConnectionLibrary

Remove-SPExcelDataConnectionLibrary

Set-SPExcelDataConnectionLibrary

Get-SPExcelDataProvider

Returns a safe data provider or a list of safe data providers.

Syntax

Get-SPExcelDataProvider -ExcelServiceApplication <SPExcelServiceApplicationPipeBind> [-AssignmentCollection <SPAssignmentCollection>]

Detailed Description

The **Get-SPExcelDataProvider** cmdlet reads a safe data provider or a list of safe data providers for an Excel Services Application. Excel Services Application refreshes external data connections only if the data provider for the connection is on the list of safe data providers. Excel Services Application includes a set of common data providers on this list. Therefore, using cmdlets to modify safe data providers is typically done only in custom data connection scenarios.

Parameters

Parameter	Required	Type	Description
ExcelService Application	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelServiceApplicationPipeBind	Specifies the Excel Services Application Web service application that contains the **SPExcelDataProvider** list object. The type must be a valid GUID, in the form 12345678-90ab-cdef-1234-567890bcdefgh; a valid name of an Excel Services Application Web service application in the farm (for example,

Parameter	Required	Type	Description
			MyExcelService1); or an instance of a valid **SPExcelServiceApplication** object.
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	Manages objects for the purpose of proper disposal. Use of objects, such as **SPWeb** or **SPSite**, can use large amounts of memory and use of these objects in Windows PowerShell scripts requires proper memory management. Using the **SPAssignment** object, you can assign objects to a variable and dispose of the objects after they are needed to free up memory. When **SPWeb**, **SPSite**, or **SPSiteAdministration** objects are used, the objects are automatically disposed of if an assignment collection or the **Global** parameter is not used. **Note:** When the

Parameter	Required	Type	Description
			Global parameter is used, all objects are contained in the global store. If objects are not immediately used, or disposed of by using the **Stop-SPAssignment** command, an out-of-memory scenario can occur.

AutoGenParams

Parameter	Required	Type	Description
ExcelServiceApplication	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelServiceApplicationPipeBind	
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	

Input Types

Return Types

Example

Description

----------EXAMPLE----------

Code

```
Get-SPExcelServiceApplication -Identity "ExcelServices" | Get-SPExcelDataProvider | where
{$_.ProviderID -eq "SQLOLEDB"}
```

Comments

This example displays the data provider with the provider identifier (ID) `SQLOLEDB` from the list of data providers that is on the `MyExcelService` application. To select a specific provider from the list of safe providers, you must pipe a where clause that can filter the list of providers.

Example

Description

----------EXAMPLE----------

Code

```
Get-SPExcelDataProvider –ExcelServiceApplication "Excel Services" | format-table
```

Comments

This example displays a list of safe data providers for the specific Excel Services Application Web service application, and then formats the results of the data providers in a table.

See Also

New-SPExcelDataProvider

Remove-SPExcelDataProvider

Set-SPExcelDataProvider

Get-SPExcelFileLocation

Returns a trusted file location or a list of trusted file locations.

Syntax

Get-SPExcelFileLocation [[-Identity] <SPExcelFileLocationPipeBind>] -ExcelServiceApplication <SPExcelServiceApplicationPipeBind> [-AssignmentCollection <SPAssignmentCollection>]

Detailed Description

The **Get-SPExcelFileLocation** cmdlet gets locations from which Excel Services Application loads workbooks. Excel Services Application loads only workbooks that are stored in a trusted file location. Properties of trusted file locations control how workbooks can be used when loaded by Excel Services Application. Excel Services Application always enforces the properties that are defined by the trusted file location from which a workbook was loaded. The properties of the trusted file location to use are determined through comparison of the file path for the workbook with the address of the trusted file location.

Parameters

Parameter	Required	Type	Description
Identity	Optional	Microsoft.Office.Excel.Server.Cmdlet.SPExcelFileLocationPipeBind	Specifies the **SPExcelFileLocation** object to return. If the **Identity** parameter is not specified, the cmdlet returns all the file locations in the specified. Excel Services Application Web service application. The type must be a valid string that identifies the file location, in the form

Parameter	Required	Type	Description
			http://myPortal/myTeam; a valid string that identifies the path, in the form C:\folder_name; a valid UNC path to a trusted location in the form, \\computer_name\share; a valid GUID, in the form 12345678-90ab-cdef-1234-567890bcdefgh; or an instance of a valid **SPExcelFileLocation** object.
ExcelServiceApplication	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelServiceApplicationPipeBind	Specifies the Excel Services Application Web service application that contains the **SPExcelFileLocation** list object. The type must be a valid GUID, in the form 12345678-90ab-cdef-1234-567890bcdefgh; a valid name of an Excel Services Application Web service application in the farm (for example, MyExcelService1); or an instance of a valid **SPExcelServiceAp**

Parameter	Required	Type	Description
			plication object.
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	Manages objects for the purpose of proper disposal. Use of objects, such as **SPWeb** or **SPSite**, can use large amounts of memory and use of these objects in Windows PowerShell scripts requires proper memory management. Using the **SPAssignment** object, you can assign objects to a variable and dispose of the objects after they are needed to free up memory. When **SPWeb**, **SPSite**, or **SPSiteAdministration** objects are used, the objects are automatically disposed of if an assignment collection or the **Global** parameter is not used. **Note:** When the **Global** parameter is used, all objects are

Parameter	Required	Type	Description
			contained in the global store. If objects are not immediately used, or disposed of by using the **Stop-SPAssignment** command, an out-of-memory scenario can occur.

AutoGenParams

Parameter	Required	Type	Description
Identity	Optional	Microsoft.Office.Excel.Server.Cmdlet.SPExcelFileLocationPipeBind	
ExcelServiceApplication	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelServiceApplicationPipeBind	
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	

Input Types

Return Types

Example

Description

----------EXAMPLE 1----------

Code

```
Get-SPExcelServiceApplication -Identity "MyExcelService" | Get-SPExcelFileLocation
```

Comments

This example displays a list of trusted file locations for the Excel Services Application Web service application named `MyExcelService`.

Example

Description

----------EXAMPLE 2----------

Code

```
Get-SPExcelServiceApplication -Identity "MyExcelService" | Get-SPExcelFileLocation | Format-list *
```

Comments

This example displays a list of trusted file locations for the Excel Services Application Web service application named `MyExcelService`, and then formats the list of properties into a table.

Example

Description

----------EXAMPLE 3----------

Code

```
Get-SPExcelServiceApplication -Identity "MyExcelService" | Get-SPExcelFileLocation | where
{$_.ExternalDataAllowed -ne "None"}
```

Comments

This example displays a list of trusted file locations that allow external data connectivity for the Excel Services Application Web service application named `MyExcelService`.

See Also

New-SPExcelFileLocation

Remove-SPExcelFileLocation

Set-SPExcelFileLocation

Get-SPExcelServiceApplication

Returns an **SPExcelServiceApplication** object.

Syntax

Get-SPExcelServiceApplication [[-Identity] <SPExcelServiceApplicationPipeBind>] [-AssignmentCollection <SPAssignmentCollection>]

Detailed Description

The **Get-SPExcelServiceApplication** cmdlet retrieves an **SPExcelServiceApplication** object. If the object returned by this cmdlet is changed, the change affects all instances of the specified Excel Services Application Web service application in the farm.

Parameters

Parameter	Required	Type	Description
Identity	Optional	Microsoft.Office.Excel.Server.Cmdlet.SPExcelServiceApplicationPipeBind	Specifies the **ExcelServiceApplication** object to get. The type must be a valid GUID, in the form 12345678-90ab-cdef-1234-567890bcdefgh; a valid name of an Excel Services Application Web service application in the farm (for example, MyExcelService1); or an instance of a valid **SPExcelServiceApplication** object.

Parameter	Required	Type	Description
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	Manages objects for the purpose of proper disposal. Use of objects, such as **SPWeb** or **SPSite**, can use large amounts of memory and use of these objects in Windows PowerShell scripts requires proper memory management. Using the **SPAssignment** object, you can assign objects to a variable and dispose of the objects after they are needed to free up memory. When **SPWeb**, **SPSite**, or **SPSiteAdministration** objects are used, the objects are automatically disposed of if an assignment collection or the **Global** parameter is not used. **Note:** When the **Global** parameter is used, all objects are contained in the global

Parameter	Required	Type	Description
			store. If objects are not immediately used, or disposed of by using the **Stop-SPAssignment** command, an out-of-memory scenario can occur.

AutoGenParams

Parameter	Required	Type	Description
Identity	Optional	Microsoft.Office.Excel.Server.Cmdlet.SPExcelServiceApplicationPipeBind	
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	

Input Types

Return Types

Example

Description

----------EXAMPLE 1----------

Code

```
Get-SPExcelServiceApplication "MyExcelServices"
```

Comments

This example displays the Excel Services Application Web service application named `MyExcelServices`.

Example

Description

----------EXAMPLE 2----------

Code

```
Get-SPExcelServiceApplication
```

Comments

This example displays every Excel Services Application Web service application that runs in the farm.

Example

Description

----------EXAMPLE 3----------

Code

```
Get-SPExcelServiceApplication –Identity "MyExcelService" | format-list *
```

Comments

This example displays Excel Services Application Web service application named `MyExcelService`, and then displays all of its settings in a list format.

See Also

New-SPExcelServiceApplication

Set-SPExcelServiceApplication

Get-SPExcelUserDefinedFunction

Returns a user-defined function or a collection of user-defined functions.

Syntax

Get-SPExcelUserDefinedFunction [[-Identity] <SPExcelUDFPipeBind>] -ExcelServiceApplication <SPExcelServiceApplicationPipeBind> [-AssignmentCollection <SPAssignmentCollection>]

Detailed Description

The **Get-SPExcelUserDefinedFunction** cmdlet reads a user-defined function and its properties, or returns the collection of user-defined functions for an Excel Services Application Web service application. Excel Services Application uses only user-defined functions that are in the list of user-defined functions. User-defined functions are managed code assemblies that can be called from an Excel Services Application workbook by using standard Excel Services Application formula syntax. The assemblies can perform custom logic or other actions, such as refreshing data. Windows PowerShell cmdlets operate on the entries in the user-defined function list, but do not operate on the assemblies.

Parameters

Parameter	Required	Type	Description
Identity	Optional	Microsoft.Office.Excel.Server.Cmdlet.SPExcelUDFPipeBind	Specifies the **UserDefinedFunction** object to get. If the **Identity** parameter is not specified, the cmdlet returns all of the functions in the user defined functions list. The type must be a valid name of the code assembly (for example, SampleCompany.SampleApplication.SampleUdf); or an instance of a valid **SPExcelUDF** object.
ExcelServiceA	Required	Microsoft.Office.Excel.Server.Cmdlet.S	Specifies the Excel Services

Parameter	Required	Type	Description
pplication		PExcelServiceApplicationPipeBind	Application Web service application that contains the **SPExcelUserDefinedFunctio n** list object. The type must be a valid GUID, in the form 12345678-90ab-cdef-1234-567890bcdefgh; a valid name of an Excel Services Application Web service application in the farm (for example, MyExcelService1); or an instance of a valid **SPExcelServiceApplication** object.
AssignmentCo llection	Optional	Microsoft.SharePoint.PowerShell.SPAs signmentCollection	Manages objects for the purpose of proper disposal. Use of objects, such as **SPWeb** or **SPSite**, can use large amounts of memory and use of these objects in Windows PowerShell scripts requires proper memory management. Using the **SPAssignment** object, you can assign objects to a variable and dispose of the objects after they are needed to free up memory. When **SPWeb**, **SPSite**, or **SPSiteAdministration** objects are used, the objects are automatically disposed of if an assignment collection or the **Global** parameter is not used. **Note:** When the **Global** parameter is used, all

Parameter	Required	Type	Description
			objects are contained in the global store. If objects are not immediately used, or disposed of by using the **Stop-SPAssignment** command, an out-of-memory scenario can occur.

AutoGenParams

Parameter	Required	Type	Description
Identity	Optional	Microsoft.Office.Excel.Server.Cmdlet.SPExcelUDFPipeBind	
ExcelServiceApplication	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelServiceApplicationPipeBind	
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	

Input Types

Return Types

Example

Description

----------EXAMPLE 1----------

Code

```
Get-SPExcelServiceApplication -Identity "MyExcelService" | Get-SPExcelUserDefinedFunction -
Identity SampleCompany.SampleApplication.SampleUdf
```

Comments

This example displays the user-defined function named `SampleCompany.SampleApplication.SampleUdf` from the list of user-defined functions that is in the Excel Services Application Web service application named `MyExcelService`.

Example

Description

----------EXAMPLE 2----------

Code

```
Get-SPExcelServiceApplication | Get-SPExcelUserDefinedFunction | format-list
```

Comments

This example displays every user-defined function for every Excel Services Application Web service application in the farm, and then displays the results in a list.

See Also

New-SPExcelUserDefinedFunction

Remove-SPExcelUserDefinedFunction

Set-SPExcelUserDefinedFunction

New-SPExcelBlockedFileType

Adds a file type to the list of file types that Excel Services Application prevents from loading.

Syntax

New-SPExcelBlockedFileType -ExcelServiceApplication <SPExcelServiceApplicationPipeBind> -FileType <XLSX | XLSB | XLSM> [-AssignmentCollection <SPAssignmentCollection>] [-Confirm [<SwitchParameter>]] [-WhatIf [<SwitchParameter>]]

Detailed Description

The **New-SPExcelBlockedExcelFileType** cmdlet adds an entry to the list of file types that Excel Services Application prevents from loading. Excel Services Application does not load any files that correspond to a file type that is on this list. The specified file type is prevented from loading by all instances of Excel Services Application for the specified Excel Services Application Web service application.

Parameters

Parameter	Required	Type	Description
ExcelServiceApplication	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelServiceApplicationPipeBind	Specifies the Excel Services Application Web service application that contains the **SPBlockedExcelFileType** list object. The type must be a valid GUID, in the form 12345678-90ab-cdef-1234-567890bcdefgh; a valid name of an Excel Services Application Web service application in the farm (for

Parameter	Required	Type	Description
			example, MyExcelService1); or an instance of a valid **SPExcelServiceApplication** object.
FileType	Required	Microsoft.Office.Excel.Server.FileType	Specifies the file types that are blocked from loading. The type must be one of the following file extensions: XLSX, XLSB, or XLSM.
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	Manages objects for the purpose of proper disposal. Use of objects, such as **SPWeb** or **SPSite**, can use large amounts of memory and use of these objects in Windows PowerShell scripts requires proper memory management. Using the **SPAssignment** object, you can assign objects to a variable and dispose of the objects after they are needed to free up memory. When **SPWeb**, **SPSite**, or **SPSiteAdministrati**

Parameter	Required	Type	Description
			on objects are used, the objects are automatically disposed of if an assignment collection or the **Global** parameter is not used. **Note:** When the **Global** parameter is used, all objects are contained in the global store. If objects are not immediately used, or disposed of by using the **Stop-SPAssignment** command, an out-of-memory scenario can occur.
Confirm	Optional	System.Management.Automation.SwitchParameter	Prompts you for confirmation before executing the command. For more information, type the following command: **get-help about_commonpar**

Parameter	Required	Type	Description
			ameters
WhatIf	Optional	System.Management.Automation.SwitchParameter	Displays a message that describes the effect of the command instead of executing the command. For more information, type the following command: **get-help about_commonparameters**

AutoGenParams

Parameter	Required	Type	Description
ExcelServiceApplication	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelServiceApplicationPipeBind	
FileType	Required	Microsoft.Office.Excel.Server.FileType	
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	
Confirm	Optional	System.Management.Automation.SwitchParameter	
WhatIf	Optional	System.Management.Automation.SwitchParameter	

Input Types

Return Types

Example

Description

----------EXAMPLE----------

Code

```
Get-SPExcelServiceApplication –identity "MyExcelService" | New-SPExcelBlockedExcelFileType –
FileType XLSM
```

Comments

This example adds an entry for XLSM to the list of file types that Excel Services Application currently prevents from loading.

See Also

Get-SPExcelBlockedFileType

Remove-SPExcelBlockedFileType

New-SPExcelDataConnectionLibrary

Adds a new data connection library to Excel Services Application.

Syntax

New-SPExcelDataConnectionLibrary -Address <String> -ExcelServiceApplication <SPExcelServiceApplicationPipeBind> [-AssignmentCollection <SPAssignmentCollection>] [-Confirm [<SwitchParameter>]] [-Description <String>] [-WhatIf [<SwitchParameter>]]

Detailed Description

The **New-SPExcelDataConnectionLibrary** cmdlet adds a data connection library to the list of trusted data connection libraries for an Excel Services Application Web service application. Excel Services Application loads only data connection files that are stored in a data connection library that is on the list of trusted data connection libraries. Data connection libraries contain data connection files, which contain all the resources that Excel Services Application and the Excel client require to connect to an external data source. Data connection libraries enable you to reuse and share data connections. Data connections can be loaded by using information from the workbook file; however, using a data connection library to load data connections allows data connections to be managed separately from workbooks.

Parameters

Parameter	Required	Type	Description
Address	Required	System.String	Specifies the address of the data connection library. The address must be unique in the list of trusted data connection libraries. The type must be a valid URL, in the form http://server_name.
ExcelServiceApplication	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelServiceApplicationPipeBind	Specifies the Excel Services Application Web service

Parameter	Required	Type	Description
			application that contains the **SPExcelDataConnectionLibrary** list object. The type must be a valid GUID, in the form 12345678-90ab-cdef-1234-567890bcdefgh; a valid name of an Excel Services Application Web service application in the farm (for example, MyExcelService1); or an instance of a valid **SPExcelServiceApplication** object.
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	Manages objects for the purpose of proper disposal. Use of objects, such as **SPWeb** or **SPSite**, can use large amounts of memory and use of these objects in Windows PowerShell scripts requires proper memory management. Using the **SPAssignment** object, you can assign objects to a variable and dispose of the objects after they are needed to free up memory. When **SPWeb**, **SPSite**, or **SPSiteAdministration** objects are used, the objects are

Parameter	Required	Type	Description
			automatically disposed of if an assignment collection or the **Global** parameter is not used. **Note:** When the **Global** parameter is used, all objects are contained in the global store. If objects are not immediately used, or disposed of by using the **Stop-SPAssignment** command, an out-of-memory scenario can occur.
Confirm	Optional	System.Management.Automation.SwitchParameter	Prompts you for confirmation before executing the command. For more information, type the following command: **get-help about_commonparameters**
Description	Optional	System.String	Specifies a friendly description for the data connection library.

Parameter	Required	Type	Description
			The type must be a valid string; for example, this is a custom OLEDB provider.
WhatIf	Optional	System.Management.Automation.SwitchParameter	Displays a message that describes the effect of the command instead of executing the command. For more information, type the following command: **get-help about_commonparameters**

AutoGenParams

Parameter	Required	Type	Description
Address	Required	System.String	
ExcelServiceApplication	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelServiceApplicationPipeBind	
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	
Confirm	Optional	System.Management.Automation.SwitchParameter	
Description	Optional	System.String	
WhatIf	Optional	System.Management.Automation.SwitchParameter	

Input Types

Return Types

Example

Description

----------EXAMPLE----------

Code

```
Get-SPExcelServiceApplication -Identity "MyExcelService" | New-SPExcelDataConnectionLibrary -address "http://portal/site/salesDCL" -description "This is the DCL where all the connections to the sales data are stored."
```

Comments

This example adds a new data connection library (`DCL`), and then sets its description to the list of trusted data connection libraries that is in the Excel Services Application Web service application named `MyExcelService`.

Workbooks can use connection files stored in that library to refresh data on the Excel Services Application Web service application.

See Also

Get-SPExcelDataConnectionLibrary

Remove-SPExcelDataConnectionLibrary

Set-SPExcelDataConnectionLibrary

New-SPExcelDataProvider

Adds a new safe data provider to Excel Services Application.

Syntax

New-SPExcelDataProvider -ExcelServiceApplication <SPExcelServiceApplicationPipeBind> -ProviderId <String> -ProviderType <Oledb | Odbc | OdbcDsn> [-AssignmentCollection <SPAssignmentCollection>] [-Confirm [<SwitchParameter>]] [-Description <String>] [-WhatIf [<SwitchParameter>]]

Detailed Description

The **New-SPExcelDataProvider** cmdlet adds a data provider to the list of safe data providers in an Excel Services Application Web service application. Excel Services Application refreshes external data connections only if the data provider for a connection is on the list of safe data providers. Excel Services Application includes a set of common data providers on this list. Therefore, using cmdlets to modify safe data providers typically occurs only in custom data connection scenarios.

Parameters

Parameter	Required	Type	Description
ExcelServiceApplication	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelServiceApplicationPipeBind	Specifies the Excel Services Application Web service application that contains the **SPExcelDataProvider** list object. The type must be a valid GUID, in the form 12345678-90ab-cdef-1234-567890bcdefgh; a valid name of an Excel Services Application Web service application in the farm (for

Parameter	Required	Type	Description
			example, MyExcelService1); or an instance of a valid **SPExcelServiceApplication** object.
ProviderId	Required	System.String	Specifies the name of the provider. This ID should be present in the connection strings generated by this provider and the ID must be unique for the provider type. The type must be a valid name of a provider; for example CustomOLEDB.
ProviderType	Required	Microsoft.Office.Excel.Server.DataProviderType	Specifies the supported type of data provider to add to safe provider list. The type must be one of the following: OleDb, Odbc, or OdbcDsn.
Assignment Collection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	Manages objects for the purpose of proper disposal. Use of objects, such as **SPWeb** or **SPSite**, can use large amounts of memory and use of these objects in Windows

Parameter	Required	Type	Description
			PowerShell scripts requires proper memory management. Using the **SPAssignment** object, you can assign objects to a variable and dispose of the objects after they are needed to free up memory. When **SPWeb**, **SPSite**, or **SPSiteAdministration** objects are used, the objects are automatically disposed of if an assignment collection or the **Global** parameter is not used. **Note:** When the **Global** parameter is used, all objects are contained in the global store. If objects are not immediately used, or disposed of by using the **Stop-SPAssignment**

Parameter	Required	Type	Description
			command, an out-of-memory scenario can occur.
Confirm	Optional	System.Management.Automation.SwitchParameter	Prompts you for confirmation before executing the command. For more information, type the following command: **get-help about_commonparameters**
Description	Optional	System.String	Specifies a friendly description for the new data provider. The type must be a valid string; for example, "This is a custom OLEDB provider".
WhatIf	Optional	System.Management.Automation.SwitchParameter	Displays a message that describes the effect of the command instead of executing the command. For more information, type the following command: **get-help about_commonparameters**

AutoGenParams

Parameter	Required	Type	Description
ExcelServiceApplication	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelServiceApplicationPipeBind	
ProviderId	Required	System.String	
ProviderType	Required	Microsoft.Office.Excel.Server.DataProviderType	
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	
Confirm	Optional	System.Management.Automation.SwitchParameter	
Description	Optional	System.String	
WhatIf	Optional	System.Management.Automation.SwitchParameter	

Input Types

Return Types

Example

Description

----------EXAMPLE 1----------

Code

```
Get-SPExcelServiceApplication –identity "MyExcelService" | New-SPExcelDataProvider –providerID "CustomOLEDB" –ProviderType OleDb -description "This is a custom OLEDB provider"
```

Comments

This example adds a new custom OLEDB data provider to the list of safe data providers for the Excel Services Application Web service application named `MyExcelService`.

Example

Description

----------EXAMPLE 2----------

Code

```
Get-SPExcelServiceApplication | New-SPExcelDataProvider -providerID "CustomOLEDB" -ProviderType
OleDb -description "This is a custom OLEDB provider"
```

Comments

This example adds a new custom OLEDB data provider to the list of safe data providers for all Excel Services Application Web service application in the farm.

See Also

Get-SPExcelDataProvider

Remove-SPExcelDataProvider

Set-SPExcelDataProvider

New-SPExcelFileLocation

Adds a new trusted location to Excel Services Application.

Syntax

New-SPExcelFileLocation -Address <String> -ExcelServiceApplication <SPExcelServiceApplicationPipeBind> [-AbortOnRefreshOnOpenFail <SwitchParameter>] [-AssignmentCollection <SPAssignmentCollection>] [-AutomaticVolatileFunctionCacheLifetime <Int32>] [-ChartAndImageSizeMax <Int32>] [-ChartRenderDurationMax <Int32>] [-ConcurrentDataRequestsPerSessionMax <Int32>] [-Confirm [<SwitchParameter>]] [-DefaultWorkbookCalcMode <File | Manual | Auto | AutoDataTables>] [-Description <String>] [-DisplayGranularExtDataErrors <SwitchParameter>] [-ExternalDataAllowed <None | Dcl | DclAndEmbedded>] [-IncludeChildren <SwitchParameter>] [-LocationType <SharePoint | Unc | Http>] [-ManualExtDataCacheLifetime <Int32>] [-NewWorkbookSessionTimeout <Int32>] [-PeriodicExtDataCacheLifetime <Int32>] [-RequestDurationMax <Int32>] [-RESTExternalDataAllowed <SwitchParameter>] [-SessionTimeout <Int32>] [-ShortSessionTimeout <Int32>] [-UdfsAllowed <SwitchParameter>] [-WarnOnDataRefresh <SwitchParameter>] [-WhatIf [<SwitchParameter>]] [-WorkbookSizeMax <Int32>]

Detailed Description

The **New-SPExcelFileLocation** cmdlet adds a new location to the list of trusted file locations in an Excel Services Application Web service application. Excel Services Application loads only workbooks that are stored in a trusted file location. Properties of trusted file locations control how workbooks can be used when loaded on Excel Services Application. Excel Services Application always enforces the properties defined by the trusted file location from which a workbook was loaded. The properties that the trusted file location uses are determined through comparison of the file path for the workbook with the **Address** parameter of the trusted file location.

The longest match takes precedence. For example, if http://portal/site/ and http://portal/site/subsite are trusted locations, and you load a workbook from the subsite, the application uses the properties from the subsite trusted location because it has the longest matching address.

Parameters

Parameter	Required	Type	Description
Address	Required	System.String	Specifies a Web folder, or file

Parameter	Required	Type	Description
			directory from which Excel Services Application can load workbooks. The type must be a valid URL, in the form http://myPortal/myTeam; or a valid UNC path in form, \\server_name\folder_name
ExcelServiceApplication	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelServiceApplicationPipeBind	Specifies the Excel Services Application Web service application that contains the **SPExcelFileLocation** list object. The type must be a valid GUID, in the form 12345678-90ab-cdef-1234-567890bcdefgh; a valid name of an Excel Services Application Web service application in the farm (for example, MyExcelService1); or an instance of a valid **SPExcelServiceApplication**

Parameter	Required	Type	Description
			object.
AbortOnRefreshOnOpenFail	Optional	System.Management.Automation.SwitchParameter	Specifies that the loading of a Excel Services Application file automatically fails if an automatic data refresh operation fails when the file is opened. The load is failed only if the following conditions are true: - The user has SharePoint Product and Technologies read-only permissions to the file. - The workbook file has data connections that are refreshed automatically when the Excel Services Application file is opened.
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	Manages objects for the purpose of proper disposal. Use of objects, such as **SPWeb** or **SPSite**, can use large amounts of

Parameter	Required	Type	Description
			memory and use of these objects in Windows PowerShell scripts requires proper memory management. Using the **SPAssignment** object, you can assign objects to a variable and dispose of the objects after they are needed to free up memory. When **SPWeb**, **SPSite**, or **SPSiteAdministration** objects are used, the objects are automatically disposed of if an assignment collection or the **Global** parameter is not used. **Note:** When the **Global** parameter is used, all objects are contained in the global store. If objects are not

Parameter	Required	Type	Description
			immediately used, or disposed of by using the **Stop-SPAssignment** command, an out-of-memory scenario can occur.
AutomaticVolatileFunctionCacheLifetime	Optional	System.Int32	Specifies the maximum time, in seconds, that a computed value for a volatile function is cached for automatic recalculations. Valid values include: **-1** Calculates once when the workbook loads. **0** Always calculates. **1** to **2073600** Caches 1 second to 24 days. The default integer value is **300**. The type must be the integers -1 or

Parameter	Required	Type	Description
			0, or an integer value in the range of 1 to 2073600.
ChartAndImageSizeMax	Optional	System.Int32	Specifies the maximum size, in megabytes, of a chart or image that can be opened. The default value is **1**. The type must be a valid integer greater than 0.
ConcurrentDataRequestsPerSessionMax	Optional	System.Int32	Specifies the maximum number of concurrent external data requests allowed in each session. If a session must issue more than this number of requests, additional requests must be queued. The scope of this setting is the logical server. The default value is **5**. The type must be a positive integer.
Confirm	Optional	System.Management.Automation.SwitchParameter	Prompts you for confirmation before executing the command. For more

Parameter	Required	Type	Description
			information, type the following command: **get-help about_commonparameters**
DefaultWorkbookCalcMode	Optional	Microsoft.Office.Excel.Server.DefaultWorkbookCalcMode	Specifies the calculation mode of workbooks. Settings other than **File** override the workbook settings. The default setting is **File**. The type must be one of the following: **File**, **Manual**, **Auto**, or **AutoDataTables**.
Description	Optional	System.String	Specifies a friendly description for the new file location. The type must be a valid string; for example, this is the file location of the monthly sales reports.
DisplayGranularExtDataErrors	Optional	System.Management.Automation.SwitchParameter	Displays granular error messages for external data failures for files in this location.
ExternalDataAllowed	Optional	Microsoft.Office.Excel.Server.AllowExternalData	Specifies the type of external data access allowed

Parameter	Required	Type	Description
			for workbooks. **None** specifies that data refresh is disabled. **Dcl** specifies that data is refreshed only when it uses a connection file in a trusted data connection library. **DclandEmbedded** specifies that a data connection library can be used but is not required. The scope of this setting is the trusted location. The default value is **None**. The type must be any of the following settings: None Dcl DclAndEmbedded
IncludeChildren	Optional	System.Management.Automation.SwitchParameter	Indicates that subordinate URLs, directories and libraries are trusted. For example, if the trusted file location is http://portal, and **IncludeChildren** is true,

Parameter	Required	Type	Description
			http://portal/subSite is also trusted, as is any URL subordinate to the URL specified in **Address**.
LocationType	Optional	Microsoft.Office.Excel.Server.LocationType	Specifies the type of location. Location type indicates how Excel Services Application retrieves files from the location specified in **Address**. The default setting is the **SharePoint** type. The default value is **SharePoint**. The type must be one of the following: SharePoint, UNC, or HTTP.
ManualExtDataCacheLifetime	Optional	System.Int32	Specifies the time, in seconds, that Excel Services Application waits before it re-issues a manual, or user-initiated, external data request. During this time period, the results from the previous

Parameter	Required	Type	Description
			external data request can be returned from an external data cache. The default value is **300**. The valid values are any one of the following items: **-1** Never refresh after first query.**1 to 2073600**-Caches 1 second to 24 days.
NewWorkbookSessionTimeout	Optional	System.Int32	Specifies the time, in seconds, that a session for a new, unsaved, workbook remains active on Excel Services Application with no user activity. Valid values include the following: **-1** Session never expires. **0** Session expires at the end of a single request. **1** to **2073600** Remains active 1 second to 24 days. The default value is **1800**.

Parameter	Required	Type	Description
			The type must be -1, 0, or an integer in the range of 1 to 2073600.
PeriodicExtDataCacheLifetime	Optional	System.Int32	Specifies the time, in seconds, that Excel Services Application waits before it re-issues an *on-open* or *periodic* (that is, automatic) external data request. During this time period, the results from the previous external data request can be returned from an external data cache. A value of **-1** specifies never to refresh after the first query. The default value is **300**. The type must be -1, 0, or integer in the range of 1 to 2073600.
RequestDurationMax	Optional	System.Int32	Specifies the maximum duration, in seconds, for a single request in a session. After

Parameter	Required	Type	Description
			this time elapses, the request times out. The default value is **300**. The type must be -1 (no limit) or an integer in the range of 1 to 2073600.
RESTExternalDataAllowed	Optional	System.Management.Automation.SwitchParameter	Specifies whether requests from the Representational State Transfer (REST) Application Programming Interface (API) are permitted to refresh external data connections. The **RESTExternalDataAllowed** parameter has no effect if ExternalDataAllowed parameter is set to **None**.
SessionTimeout	Optional	System.Int32	Specifies the time, in seconds, that a session remains active on Excel Services Application with no user activity. Valid values include the following: **-1** Session never

Parameter	Required	Type	Description
			expires. **0** session expires at the end of a single request. **1** to **2073600** Remains active 1 second to 24 days. The default value is **300**. The type must be -1, 0, or an integer in the range of 1 to 2073600.
ShortSessionTimeout	Optional	System.Int32	Specifies the time, in seconds, that a user has to make the initial interaction with a spreadsheet. After this time has elapsed, if the user is not active, the session is closed. The value must always be less than **SessionTimeout**. If the **ShortSessionTimeout** value is greater than or equal to **SessionTimeout**, an error is returned. The default value is

Parameter	Required	Type	Description
			75. The type must be -1 or an integer in the range of 1 to 2073600.
UdfsAllowed	Optional	System.Management.Automation.SwitchParameter	Specifies that user-defined functions can be called by workbooks that are loaded from the trusted file location that is specified in **Address**. The default value is **False**.
WarnOnDataRefresh	Optional	System.Management.Automation.SwitchParameter	Specifies that a warning is displayed to the user on the first refresh of data for the workbook. The default value is **True**.
WhatIf	Optional	System.Management.Automation.SwitchParameter	Displays a message that describes the effect of the command instead of executing the command. For more information, type the following command: **get-help about_commonparameters**

Parameter	Required	Type	Description
WorkbookSizeMax	Optional	System.Int32	Specifies the maximum size, in megabytes, of a workbook that can be loaded. The default value is **10**. The type must be an integer value in the range of 1 to 2000.

AutoGenParams

Parameter	Required	Type	Description
Address	Required	System.String	
ExcelServiceApplication	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelServiceApplicationPipeBind	
AbortOnRefreshOnOpenFail	Optional	System.Management.Automation.SwitchParameter	
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	
AutomaticVolatileFunctionCacheLifetime	Optional	System.Int32	
ChartAndImageSizeMax	Optional	System.Int32	
ChartRenderDurationMax	Optional	System.Int32	
ConcurrentDataRequestsPerSessionMax	Optional	System.Int32	
Confirm	Optional	System.Management.Automation.SwitchParameter	
DefaultWorkbookCalcMode	Optional	Microsoft.Office.Excel.Server.DefaultWorkbook	

Parameter	Required	Type	Description
		CalcMode	
Description	Optional	System.String	
DisplayGranularExtDataErrors	Optional	System.Management.Automation.SwitchParameter	
ExternalDataAllowed	Optional	Microsoft.Office.Excel.Server.AllowExternalData	
IncludeChildren	Optional	System.Management.Automation.SwitchParameter	
LocationType	Optional	Microsoft.Office.Excel.Server.LocationType	
ManualExtDataCacheLifetime	Optional	System.Int32	
NewWorkbookSessionTimeout	Optional	System.Int32	
PeriodicExtDataCacheLifetime	Optional	System.Int32	
RequestDurationMax	Optional	System.Int32	
RESTExternalDataAllowed	Optional	System.Management.Automation.SwitchParameter	
SessionTimeout	Optional	System.Int32	
ShortSessionTimeout	Optional	System.Int32	
UdfsAllowed	Optional	System.Management.Automation.SwitchParameter	
WarnOnDataRefresh	Optional	System.Management.Automation.SwitchParameter	
WhatIf	Optional	System.Management.Automation.SwitchParameter	
WorkbookSizeMax	Optional	System.Int32	

Input Types

Return Types

Example

Description

----------EXAMPLE 1----------

Code

```
Get-SPExcelServiceApplication –identity "MyExcelService" | New-SPExcelFileLocation –address "http://myPortal/myTeam" –includechildren –locationType SharePoint –description "This is my team’s site on myPortal." –workbooksizemax 50 –externaldataallowed DclAndEmbedded –WarnOnDataRefresh:$false
```

Comments

This example adds a new trusted file location to the list of trusted file locations that is in the Excel Services Application Web service application named `MyExcelService`.

This example enables workbooks to load from any child or subfolder that starts with `http://myPortal/myTeam`. All workbooks loaded from this location must be smaller than 50 megabytes (MB) in size. The workbooks can refresh external data by using connections that are embedded in the workbook file or stored in a data connection library. A warning message that the workbook is attempting to refresh external data will not be displayed.

Example

Description

----------EXAMPLE 2----------

Code

```
Get-SPExcelServiceApplication –identity "MyExcelService" | New-SPExcelFileLocation –address http://myPortal/myTeam/managers –locationType SharePoint –description "This is the manager’s subsite for myTeam."
```

Comments

This example adds a new trusted file location to the list of trusted file locations that is in the Excel Services Application Web service application named `MyExcelService` with a given address and description.

See Also

Get-SPExcelFileLocation

Remove-SPExcelFileLocation

Set-SPExcelFileLocation

New-SPExcelServiceApplication

Creates a new instance of Excel Services Application.

Syntax

New-SPExcelServiceApplication -ApplicationPool <SPIisWebServiceApplicationPoolPipeBind> [-AssignmentCollection <SPAssignmentCollection>] [-CachingOfUnusedFilesEnable <SwitchParameter>] [-Confirm [<SwitchParameter>]] [-CrossDomainAccessAllowed <SwitchParameter>] [-Default <SwitchParameter>] [-EncryptedUserConnectionRequired <None | Connection>] [-ExternalDataConnectionLifetime <Int32>] [-FileAccessMethod <UseImpersonation | UseFileAccessAccount>] [-LoadBalancingScheme <WorkbookUrl | RoundRobin | Local>] [-MemoryCacheThreshold <Int32>] [-Name <String>] [-PrivateBytesMax <Int32>] [-SessionsPerUserMax <Int32>] [-SiteCollectionAnonymousSessionsMax <Int32>] [-TerminateProcessOnAccessViolation <SwitchParameter>] [-ThrottleAccessViolationsPerSiteCollection <Int32>] [-UnattendedAccountApplicationId <String>] [-UnusedObjectAgeMax <Int32>] [-WhatIf [<SwitchParameter>]] [-WorkbookCache <String>] [-WorkbookCacheSizeMax <Int32>]

Detailed Description

The **New-SPExcelServiceApplication** cmdlet creates a new Excel Services Application Web service application with new settings. After you create a new Excel Services Application Web service application, you can use the **Set-SPExcelServiceApplication** cmdlet to modify the global settings of the new application.

Parameters

Parameter	Required	Type	Description
ApplicationPool	Required	Microsoft.SharePoint.PowerShell.SPIisWebServiceApplicationPoolPipeBind	Specifies the existing IIS application pool to run the Web service in for the new Excel Service application. The type must be a valid **SPIisWebServiceApplicationPool** object.
Name	Required	System.String	Specifies a friendly name

Parameter	Required	Type	Description
			for the new Excel Services Application Web service application. The name that you use must be a unique name of an Excel Services Application Web service application that is in this farm. The name can be a maximum of 128 characters. The default value is *Service_numofapplications+1*. The type must be a valid string; for example, MyExcelServiceApp1.
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	Manages objects for the purpose of proper disposal. Use of objects, such as **SPWeb** or **SPSite**, can use large amounts of memory and use of these objects in Windows PowerShell scripts requires proper memory management. Using the **SPAssignment** object, you can assign objects to a variable and dispose of the objects after they are needed to free up memory. When **SPWeb**, **SPSite**, or **SPSiteAdministration** objects are used, the objects are automatically disposed of if an assignment collection or the **Global** parameter is

Parameter	Required	Type	Description
			not used. **Note:** When the **Global** parameter is used, all objects are contained in the global store. If objects are not immediately used, or disposed of by using the **Stop-SPAssignment** command, an out-of-memory scenario can occur.
CachingOfUnusedFilesEnable	Optional	System.Management.Automation.SwitchParameter	Specifies that files that are no longer used by Excel Services Application can remain in the cache for later use. The default is true when the parameter is not specified on the command.
Confirm	Optional	System.Management.Automation.SwitchParameter	Prompts you for confirmation before executing the command. For more information, type the following command: **get-help about_commonparameters**
CrossDomainAccessAllowed	Optional	System.Management.Automation.SwitchParameter	Specifies that trusted workbooks and data connection files can be requested and rendered by Web Parts or pages

Parameter	Required	Type	Description
			that reside in other HTTP domains.
Default	Optional	System.Management.Automation.Switch Parameter	Specifies that the proxy for this service application is added to the default proxy group of the farm.
EncryptedUserConnectionRequired	Optional	Microsoft.Office.Excel.Server.Encrypted ConnectionMode	Requires that encryption is used between the end user and the Excel Services Application server. The scope of this setting is the logical server. The default value is **None**. The type must be one of the following: None or Connection.
ExternalDataConnectionLifetime	Optional	System.Int32	Specifies the maximum number of seconds that an external data connection can remain open in the connection pool. Valid values include: **-1** Never recycle **1 to 2073600** Remain active 1 second to 24 days. The default value is **1800**.
FileAccessMethod	Optional	Microsoft.Office.Excel.Server.FileAccess Method	Specifies the authentication method that Excel Services Application uses to retrieve files. **UseFileAccessAccount** specifies that all workbooks always use

Parameter	Required	Type	Description
			the credentials of the service account running Excel Services Application. **UseImpersonation** specifies that Excel Services Application retrieves the workbook files as the user who is making the request. The default value is **UseImpersonation**. The type must be one of the following: UseImpersonation or UseFileAccessAccount.
LoadBalancingScheme	Optional	Microsoft.Office.Excel.Server.LoadBalancingScheme	Specifies the load-balancing schema that is used by the Excel Services Application Web service application for sending requests to different back-end computers that are running Excel Calculation Services. The default value is **WorkbookURL**. The type must be one of the following: RoundRobin, Local, or WorkbookURL.
MemoryCacheThreshold	Optional	System.Int32	Specifies the percentage of the maximum private bytes that can be allocated to inactive objects. When the memory cache threshold is exceeded, cached objects that are not

Parameter	Required	Type	Description
			currently in use are released. Setting this value to **0** disables caching of inactive objects. The default value is **90**. The type must be an integer in the range of 0 to 95.
PrivateBytesMax	Optional	System.Int32	Specifies the maximum private bytes, in megabytes, used by Excel Services Application. When set to -1, the maximum private bytes defaults to 50 percent of physical memory on the computer. Valid values include -1 for no limit, and any positive integer. The default value is **-1**. The type must be -1 or any positive integer.
SessionsPerUserMax	Optional	System.Int32	Specifies the maximum number of sessions allowed for a user. If a user reaches this maximum, the oldest session will be deleted when a new session is started. Valid values include -1 for no limit, and any positive integer. The default value is **25**. The type must be -1 or any positive integer.
SiteCollectionAno	Optional	System.Int32	Specifies the maximum

Parameter	Required	Type	Description
nymousSessionsMax			number of anonymous sessions allowed per site collection. A value of 0 specifies that no anonymous sessions are allowed. Valid values include -1 for no limit, 0 to disable the setting, or any positive integer. The default value is **25**. The type must be -1, 0, or any positive integer.
TerminateProcessOnAccessViolation	Optional	System.Management.Automation.SwitchParameter	Terminates Excel Services Application when an access violation occurs in the process.
ThrottleAccessViolationsPerSiteCollection	Optional	System.Int32	Specifies that if a workbook causes an access violation error on Excel Services Application, all files that originate from that workbook's site collection are blocked from loading for the specified period (in seconds). Existing sessions for files on that site collection are unaffected. Valid values include 0 to disable the setting, and the range of 1 to 2073600 seconds (24 days). The default value is **0**. The type must be an integer value in the range of 0 to 2073600.
UnattendedAccou	Optional	System.String	Specifies that the

Parameter	Required	Type	Description
ntApplicationId			application ID that is used to look up the unattended service account credentials from the secure store service specified by **UnattendedAccountSecureServiceAppName**. The secure store service is contacted when this parameter is set and specific error cases occur. Valid values are alphanumeric strings with a maximum of 256 characters. The default value is an empty string. The type must be a valid string of alphanumeric characters; for example, UnattendedServiceAccountID. **Note:** The unattended account is a single account that all workbooks can use to refresh data. An unattended account is required when workbook connections specify *None* for authentication, or when any non-Windows credentials are

Parameter	Required	Type	Description
			used to refresh data.
UnusedObjectAgeMax	Optional	System.Int32	Specifies the maximum amount of time, in minutes, that objects not currently used in a session are kept in the memory cache. Valid values include -1 for no maximum setting, and a positive integer in the range of 1 to 34560 minutes (24 days). The default value is **-1**. The type must be -1, or an integer in the range of 1 to 34560.
WhatIf	Optional	System.Management.Automation.SwitchParameter	Displays a message that describes the effect of the command instead of executing the command. For more information, type the following command: **get-help about_commonparameters**
WorkbookCache	Optional	System.String	Specifies the local file system location of the cache that is used for storing workbooks used by Excel Services Application. If no value is specified, then the default value is a subdirectory in the system temporary directory. The type must be a valid path in either of the

Parameter	Required	Type	Description
			following forms: -- C:\folder_name -- \\server_name\folder_name
WorkbookCacheSizeMax	Optional	System.Int32	Specifies the maximum allowable size, in megabytes, of an individual session. The default value is **0**. The type must be any positive integer.

AutoGenParams

Parameter	Required	Type	Description
ApplicationPool	Required	Microsoft.SharePoint.PowerShell.SPIisWebServiceApplicationPoolPipeBind	
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	
CachingOfUnusedFilesEnable	Optional	System.Management.Automation.SwitchParameter	
Confirm	Optional	System.Management.Automation.SwitchParameter	
CrossDomainAccessAllowed	Optional	System.Management.Automation.SwitchParameter	
Default	Optional	System.Management.Automation.SwitchParameter	
EncryptedUserConnectionRequired	Optional	Microsoft.Office.Excel.Server.EncryptedConnectionMode	
ExternalDataConnectionLifetime	Optional	System.Int32	
FileAccessMethod	Optional	Microsoft.Office.Excel.Server.FileAccessMethod	

Parameter	Required	Type	Description
LoadBalancingScheme	Optional	Microsoft.Office.Excel.Server.LoadBalancingScheme	
MemoryCacheThreshold	Optional	System.Int32	
Name	Optional	System.String	
PrivateBytesMax	Optional	System.Int32	
SessionsPerUserMax	Optional	System.Int32	
SiteCollectionAnonymousSessionsMax	Optional	System.Int32	
TerminateProcessOnAccessViolation	Optional	System.Management.Automation.SwitchParameter	
ThrottleAccessViolationsPerSiteCollection	Optional	System.Int32	
UnattendedAccountApplicationId	Optional	System.String	
UnusedObjectAgeMax	Optional	System.Int32	
WhatIf	Optional	System.Management.Automation.SwitchParameter	
WorkbookCache	Optional	System.String	
WorkbookCacheSizeMax	Optional	System.Int32	

Input Types

Return Types

Example

Description

----------EXAMPLE 1----------

Code

```
New-SPExcelServiceApplication -Name "MyExcelService" -ApplicationPool "SharePoint Services"
```

Comments

This example creates a new instance of Excel Services Application with the display name `MyExcelService`.

Example

Description

----------EXAMPLE 2----------

Code

```
New-SPExcelServiceApplication -Name "My Excel Service" -ApplicationPool "SharePoint Services" -
SessionsPerUserMax 5
```

Comments

This example creates a new instance of Excel Services Application with the display name `My Excel Service` and allows only five sessions per user for each back-end application server computer that is running Excel Services Application.

See Also

Get-SPExcelServiceApplication

Set-SPExcelServiceApplication

New-SPExcelUserDefinedFunction

Adds a new user-defined function to Excel Services Application.

Syntax

New-SPExcelUserDefinedFunction -Assembly <String> -ExcelServiceApplication <SPExcelServiceApplicationPipeBind> [-AssemblyLocation <GAC | LocalFile>] [-AssignmentCollection <SPAssignmentCollection>] [-Confirm [<SwitchParameter>]] [-Description <String>] [-Enable <SwitchParameter>] [-WhatIf [<SwitchParameter>]]

Detailed Description

The **New-SPExcelUserDefinedFunction** cmdlet adds a user-defined function to an Excel Services Application Web service application. Excel Services Application uses only user-defined functions that have an entry in the list of user-defined functions. User-defined functions are managed code assemblies that can be called from an Excel Services Application workbook by using standard Excel Services Application formula syntax. The assemblies can perform custom logic or other actions, such as refresh data. Windows PowerShell cmdlets operate on the entries that are in the list of user-defined functions, but they do not operate on the assemblies.

Parameters

Parameter	Required	Type	Description
Assembly	Required	System.String	Specifies the strongly typed name of the assembly. The name must be unique in the list of user-defined functions. The name can be a maximum of 4096 alphanumeric characters. The type must be a string of alphanumeric characters; for example:-- SampleCompany.SampleApplication.SampleUdf -- C:\UDFs\SampleUdf.dll

Parameter	Required	Type	Description
			-- \\MyNetworkServer\UDFs\SampleUdf.dll
ExcelServiceApplication	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelServiceApplicationPipeBind	Specifies the Excel Services Application Web service application that contains the list of user-defined functions. The type must be a valid GUID, in the form 12345678-90ab-cdef-1234-567890bcdefgh; a valid name of an Excel Services Application Web service application in the farm (for example, MyExcelService1); or an instance of a valid **SPExcelServiceApplication** object.
AssemblyLocation	Optional	Microsoft.Office.Excel.Server.AssemblyLocation	Specifies the location where the assembly is stored. The default value is **GAC**. The type must be one of the following: GAC or LocalFile.
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	Manages objects for the purpose of proper disposal. Use of objects, such as **SPWeb** or **SPSite**, can use large amounts of memory and use of these objects in Windows PowerShell scripts requires proper memory management. Using the **SPAssignment** object, you can assign objects to a variable and dispose of the objects after they are needed to free up memory. When **SPWeb**, **SPSite**, or

Parameter	Required	Type	Description
			SPSiteAdministration objects are used, the objects are automatically disposed of if an assignment collection or the **Global** parameter is not used. **Note:** When the **Global** parameter is used, all objects are contained in the global store. If objects are not immediately used, or disposed of by using the **Stop-SPAssignment** command, an out-of-memory scenario can occur.
Confirm	Optional	System.Management.Automation.SwitchParameter	Prompts you for confirmation before executing the command. For more information, type the following command: **get-help about_commonparameters**
Description	Optional	System.String	Specifies a friendly description for the user-defined function. The name can be a maximum of 4096 alphanumeric characters. The default value is an empty string. The type must be a valid string of alphanumeric characters; for example, this is the user-defined function for consolidation reports.
Enable	Optional	System.Management.Automation.Switch	Activates the user-defined

Parameter	Required	Type	Description
		Parameter	function for use with Excel Services Application. The default value is **True**.
WhatIf	Optional	System.Management.Automation.SwitchParameter	Displays a message that describes the effect of the command instead of executing the command. For more information, type the following command: **get-help about_commonparameters**

AutoGenParams

Parameter	Required	Type	Description
Assembly	Required	System.String	
ExcelServiceApplication	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelServiceApplicationPipeBind	
AssemblyLocation	Optional	Microsoft.Office.Excel.Server.AssemblyLocation	
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	
Confirm	Optional	System.Management.Automation.SwitchParameter	
Description	Optional	System.String	
Enable	Optional	System.Management.Automation.SwitchParameter	
WhatIf	Optional	System.Management.Automation.SwitchParameter	

Input Types

Return Types

Example

Description

----------EXAMPLE 1----------

Code

```
New-SPExcelUserDefinedFunction -ExcelServiceApplication "MyExcelService" -Assembly
"SampleCompany.SampleApplication.SampleUdf" -AssemblyLocation GAC
```

Comments

This example adds the `SampleComany.SampleApplication.SampleUdf` file to the Excel Services Application Web service application named `MyExcelService` as a new user-defined function.

The user-defined function assembly is deployed to the global assembly cache (GAC).

Example

Description

----------EXAMPLE 2----------

Code

```
Get-SPExcelServiceApplication | New-SPExcelUserDefinedFunction -Assembly
"SampleCompany.SampleApplication.SampleUdf" -AssemblyLocation GAC
```

Comments

This example adds the `SampleComany.SampleApplication.SampleUdf` file to every Excel Services Application Web service application in the farm.

The user-defined function assembly is deployed to the global assembly cache (GAC).

See Also

Get-SPExcelUserDefinedFunction

Remove-SPExcelUserDefinedFunction

Set-SPExcelUserDefinedFunction

Remove-SPExcelBlockedFileType

Removes an entry from the list of file types that are prevented from loading on Excel Services Application.

Syntax

Remove-SPExcelBlockedFileType [-Identity] <SPExcelBlockedFileTypePipeBind> -ExcelServiceApplication <SPExcelServiceApplicationPipeBind> [-AssignmentCollection <SPAssignmentCollection>] [-Confirm [<SwitchParameter>]] [-WhatIf [<SwitchParameter>]]

Detailed Description

The **Remove-SPExcelBlockedExcelFileType** cmdlet removes an entry from the list of file types that are prevented from loading on an Excel Services Application Web service application. Excel Services Application does not load any file types that correspond to a file type that is on this list.

Parameters

Parameter	Required	Type	Description
Identity	Optional	Microsoft.Office.Excel.Server.Cmdlet.SPExcelBlockedFileTypePipeBind	Specifies the file type to remove from the list of blocked file types. The type must be a valid GUID, in the form 12345678-90ab-cdef-1234-567890bcdefgh; one of the following file types: XLSX, XLSB, XLSM; or an instance of a valid **SPExcelBlockedFileType** object.
ExcelService Application	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelServiceApplicationPipeBind	Specifies the Excel Services Application

Parameter	Required	Type	Description
			Web service application that contains the **SPBlockedExcelFileType** list object. The type must be a valid GUID, in the form 12345678-90ab-cdef-1234-567890bcdefgh; a valid name of an Excel Services Application Web service application in the farm (for example, MyExcelService1); or an instance of a valid **SPExcelServiceApplication** object.
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	Manages objects for the purpose of proper disposal. Use of objects, such as **SPWeb** or **SPSite**, can use large amounts of memory and use of these objects in Windows PowerShell scripts requires proper memory management. Using the **SPAssignment** object, you can assign objects to a variable and dispose of the

Parameter	Required	Type	Description
			objects after they are needed to free up memory. When **SPWeb**, **SPSite**, or **SPSiteAdministration** objects are used, the objects are automatically disposed of if an assignment collection or the **Global** parameter is not used. **Note:** When the **Global** parameter is used, all objects are contained in the global store. If objects are not immediately used, or disposed of by using the **Stop-SPAssignment** command, an out-of-memory scenario can occur.
Confirm	Optional	System.Management.Automation.SwitchParameter	Prompts you for confirmation before executing the

Parameter	Required	Type	Description
			command. For more information, type the following command: **get-help about_commonparameters**
WhatIf	Optional	System.Management.Automation.SwitchParameter	Displays a message that describes the effect of the command instead of executing the command. For more information, type the following command: **get-help about_commonparameters**

AutoGenParams

Parameter	Required	Type	Description
Identity	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelBlockedFileTypePipeBind	
ExcelServiceApplication	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelServiceApplicationPipeBind	
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	
Confirm	Optional	System.Management.Automation.SwitchParameter	
WhatIf	Optional	System.Management.Automation.SwitchParameter	

Input Types

Return Types

Example

Description

----------EXAMPLE----------

Code

```
Get-SPExcelServiceApplication -Identity "MyExcelService" | Remove-SPExcelBlockedExcelFileType -
FileType XLSX
```

Comments

This example removes the .xlsx file name extensions from the list of file types that are prevented from loading by Excel Services Application.

After the .xlsx file name extension is removed from the list, these extensions are allowed to load in Excel Services Application.

See Also

Get-SPExcelBlockedFileType

New-SPExcelBlockedFileType

Remove-SPExcelDataConnectionLibrary

Removes a data connection library from Excel Services Application.

Syntax

Remove-SPExcelDataConnectionLibrary [-Identity] <SPExcelDCLPipeBind> -ExcelServiceApplication <SPExcelServiceApplicationPipeBind> [-AssignmentCollection <SPAssignmentCollection>] [-Confirm [<SwitchParameter>]] [-WhatIf [<SwitchParameter>]]

Detailed Description

The **Remove-SPExcelDataConnectionLibrary** cmdlet removes a library from the Excel Services Application trusted data connection libraries list. Excel Services Application loads data connection files only if they are stored in a data connection library that is on the trusted data connection libraries list.

Parameters

Parameter	Required	Type	Description
Identity	Optional	Microsoft.Office.Excel.Server.Cmdlet.SPExcelDCLPipeBind	Specifies the data connection library to remove. The type must be a valid GUID, in the form 12345678-90ab-cdef-1234-567890bcdefgh; a valid name of a data connection library (for example, DataConnectionLib1); a valid URL, in the form http://server_name; or an instance of a valid **SPExcelDCL** object.
ExcelServiceApplication	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelServiceApplicationPipeBind	Specifies the Excel Services Application

Parameter	Required	Type	Description
			Web service application that contains the **SPExcelDataConnectionLibrary** list object. The type must be a valid GUID, in the form 12345678-90ab-cdef-1234-567890bcdefgh; a valid name of an Excel Services Application Web service application in the farm (for example, MyExcelService1); or an instance of a valid **SPExcelServiceApplication** object.
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	Manages objects for the purpose of proper disposal. Use of objects, such as **SPWeb** or **SPSite**, can use large amounts of memory and use of these objects in Windows PowerShell scripts requires proper memory management. Using the **SPAssignment** object, you can assign objects to a variable and dispose of the objects after they are needed to free up memory. When **SPWeb**, **SPSite**, or **SPSiteAdministration** objects are used, the

Parameter	Required	Type	Description
			objects are automatically disposed of if an assignment collection or the **Global** parameter is not used. **Note:** When the **Global** parameter is used, all objects are contained in the global store. If objects are not immediately used, or disposed of by using the **Stop-SPAssignment** command, an out-of-memory scenario can occur.
Confirm	Optional	System.Management.Automation.SwitchParameter	Prompts you for confirmation before executing the command. For more information, type the following command: **get-help about_commonparameters**
WhatIf	Optional	System.Management.Automation.SwitchParameter	Displays a message that describes the effect of the command

Parameter	Required	Type	Description
			instead of executing the command. For more information, type the following command: **get-help about_commonparameters**

AutoGenParams

Parameter	Required	Type	Description
Identity	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelDCLPipeBind	
ExcelServiceApplication	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelServiceApplicationPipeBind	
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	
Confirm	Optional	System.Management.Automation.SwitchParameter	
WhatIf	Optional	System.Management.Automation.SwitchParameter	

Input Types

Return Types

Example

Description

----------EXAMPLE 1----------

Code

```
Get-SPExcelServiceApplication -Identity "MyExcelService" | Remove- SPExcelDataConnectionLibrary
-Identity "http://portal/site/salesDCL"
```

Comments

This example removes the `http://portal/site/salesDCL` library from the list of trusted data connection libraries, which is on the Excel Services Application Web service application named `MyExcelService`.

Connection files are no longer loaded from this library. Workbooks that depend on this connection file might not refresh data.

Example

Description

----------EXAMPLE 2----------

Code

```
Get-SPExcelServiceApplication | Get-SPExcelDataConnectionLibrary | Remove-
SPExcelDataConnectionLibrary
```

Comments

This example removes all data connection libraries from all Excel Services Application running in the farm.

Connection files are no longer loaded from any libraries. Workbooks that depend on any connection files might not refresh data.

See Also

Get-SPExcelDataConnectionLibrary

New-SPExcelDataConnectionLibrary

Set-SPExcelDataConnectionLibrary

Remove-SPExcelDataProvider

Removes a data provider from Excel Services Application.

Syntax

Remove-SPExcelDataProvider [-Identity] <SPExcelDataProviderPipeBind> [-AssignmentCollection <SPAssignmentCollection>] [-Confirm [<SwitchParameter>]] [-WhatIf [<SwitchParameter>]]

Detailed Description

The **Remove-SPExcelDataProvider** cmdlet removes a data provider that is stored on the list of safe data providers. Excel Services Application refreshes external data connections only if the data provider for that connection is on the list of safe data providers. Excel Services Application includes a set of common data providers on this list, so using cmdlets to modify safe data providers is typically done only in custom data connection scenarios.

Parameters

Parameter	Required	Type	Description
Identity	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelDataProviderPipeBind	Specifies the **SPExcelDataProvider** object to remove. The type must be a valid GUID, in the form 12345678-90ab-cdef-1234-567890bcdefgh, or an instance of a valid **SPExcelDataProvider** object.
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	Manages objects for the purpose of proper disposal. Use of objects, such as **SPWeb** or **SPSite**, can use large

Parameter	Required	Type	Description
			amounts of memory and use of these objects in Windows PowerShell scripts requires proper memory management. Using the **SPAssignment** object, you can assign objects to a variable and dispose of the objects after they are needed to free up memory. When **SPWeb**, **SPSite**, or **SPSiteAdministration** objects are used, the objects are automatically disposed of if an assignment collection or the **Global** parameter is not used. **Note:** When the **Global** parameter is used, all objects are contained in the global store. If objects are not immediately used, or disposed of by using the

Parameter	Required	Type	Description
			Stop-SPAssignment command, an out-of-memory scenario can occur.
Confirm	Optional	System.Management.Automation.SwitchParameter	Prompts you for confirmation before executing the command. For more information, type the following command: **get-help about_commonparameters**
WhatIf	Optional	System.Management.Automation.SwitchParameter	Displays a message that describes the effect of the command instead of executing the command. For more information, type the following command: **get-help about_commonparameters**

AutoGenParams

Parameter	Required	Type	Description
Identity	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelDataProviderPipeBind	

Parameter	Required	Type	Description
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	
Confirm	Optional	System.Management.Automation.SwitchParameter	
WhatIf	Optional	System.Management.Automation.SwitchParameter	

Input Types

Return Types

Example

Description

----------EXAMPLE 1----------

Code

```
Get-SPExcelServiceApplication -Identity "MyExcelService" | Get-SPExcelDataProvider | where
{$_.providerID -eq "SQLOLEDB" } | Remove-SPExcelDataProvider
```

Comments

This example removes the default SQL OLEDB data provider from the list of safe data providers that is on the Excel Services Application Web service application named `MyExcelService`.

Excel Services Application will no longer refresh any data connections that use this data provider.

Example

Description

----------EXAMPLE 2----------

Code

```
$provider = Get-SPExcelServiceApplication -Identity "MyExcelService" | Get-SPExcelDataProvider
| { where {$_.providerID -eq "MyCustomProvider"}
```

Code

```
Remove-SPExcelDataProvider $provider
```

Comments

This example removes a custom data provider from the list of safe data providers that is on the Excel Services Application Web service application named `MyExcelService`.

Excel Services Application will no longer refresh any data connections that use this data provider.

See Also

Get-SPExcelDataProvider

New-SPExcelDataProvider

Set-SPExcelDataProvider

Remove-SPExcelFileLocation

Removes a trusted file location from Excel Services Application.

Syntax

Remove-SPExcelFileLocation [-Identity] <SPExcelFileLocationPipeBind> -ExcelServiceApplication <SPExcelServiceApplicationPipeBind> [-AssignmentCollection <SPAssignmentCollection>] [-Confirm [<SwitchParameter>]] [-WhatIf [<SwitchParameter>]]

Detailed Description

The **Remove-SPExcelFileLocation** cmdlet removes a location from the list of trusted file locations. Excel Services Application loads only workbooks that are stored in a trusted file location. Properties of trusted file locations control how workbooks can be used when loaded on Excel Services Application. Excel Services Application always enforces the properties defined by the trusted file location from which a workbook was loaded. The properties used by the trusted file location are determined by comparing the file path for the workbook with the **Address** parameter of the trusted file location.

Parameters

Parameter	Required	Type	Description
Identity	Optional	Microsoft.Office.Excel.Server.Cmdlet.SPExcelFileLocationPipeBind	Specifies the **FileLocation** object to remove. The type must be a valid GUID, in the form 12345678-90ab-cdef-1234-567890bcdefgh; a valid string that identifies the file location, in the form http://myPortal/myTeam; a valid string that identifies the path, in the form C:\folder_name; or

Parameter	Required	Type	Description
			an instance of a valid **SPExcelFileLocation** object.
ExcelServiceApplication	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelServiceApplicationPipeBind	Specifies the Excel Services Application Web service application that contains the **SPExcelFileLocation** list object. The type must be a valid GUID, in the form 12345678-90ab-cdef-1234-567890bcdefgh; a valid name of an Excel Services Application Web service application in the farm (for example, MyExcelService1); or an instance of a valid **SPExcelServiceApplication** object.
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	Manages objects for the purpose of proper disposal. Use of objects, such as **SPWeb** or **SPSite**, can use large amounts of memory and use of these objects in Windows PowerShell scripts requires proper

Parameter	Required	Type	Description
			memory management. Using the **SPAssignment** object, you can assign objects to a variable and dispose of the objects after they are needed to free up memory. When **SPWeb**, **SPSite**, or **SPSiteAdministration** objects are used, the objects are automatically disposed of if an assignment collection or the **Global** parameter is not used. **Note:** When the **Global** parameter is used, all objects are contained in the global store. If objects are not immediately used, or disposed of by using the **Stop-SPAssignment** command, an out-of-

Parameter	Required	Type	Description
			memory scenario can occur.
Confirm	Optional	System.Management.Automation.SwitchParameter	Prompts you for confirmation before executing the command. For more information, type the following command: **get-help about_commonparameters**
WhatIf	Optional	System.Management.Automation.SwitchParameter	Displays a message that describes the effect of the command instead of executing the command. For more information, type the following command: **get-help about_commonparameters**

AutoGenParams

Parameter	Required	Type	Description
Identity	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelFileLocationPipeBind	
ExcelServiceApplication	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelServiceApplicationPipeBind	
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	

Parameter	Required	Type	Description
Confirm	Optional	System.Management.Automation.SwitchParameter	
WhatIf	Optional	System.Management.Automation.SwitchParameter	

Input Types

Return Types

Example

Description

----------EXAMPLE 1----------

Code

```
Remove-SPExcelFileLocation -ExcelServiceApplication "My Excel Service" -Identity
http://portal/badTeamSite
```

Comments

This example removes the `http://portal/badTeamSite` from the list of trusted file locations for the Excel Services Application Web service application named `My Excel Service`.

Example

Description

----------EXAMPLE 2----------

Code

```
Get-SPExcelServiceApplication | Get-SPExcelFileLocation | where {$_.Address -ne "http://"} |
Remove-SPExcelFileLocation
```

Comments

This example removes all nondefault trusted file locations from every Excel Services Application Web service application in the farm.

See Also

Get-SPExcelFileLocation

New-SPExcelFileLocation

Set-SPExcelFileLocation

Remove-SPExcelUserDefinedFunction

Removes a user-defined function from Excel Services Application.

Syntax

Remove-SPExcelUserDefinedFunction [-Identity] <SPExcelUDFPipeBind> -ExcelServiceApplication <SPExcelServiceApplicationPipeBind> [-AssignmentCollection <SPAssignmentCollection>] [-Confirm [<SwitchParameter>]] [-WhatIf [<SwitchParameter>]]

Detailed Description

The **Remove-SPExcelUserDefinedFunction** cmdlet removes a user-defined function from the list of user-defined functions. User-defined functions are managed code assemblies that can be called from an Excel Services Application workbook by using standard Excel Services Application formula syntax. The assemblies can perform custom logic or other actions, such as refresh data. The Windows PowerShell cmdlets operate on the entries in the user-defined function list, but do not operate on the assemblies.

Parameters

Parameter	Required	Type	Description
Identity	Optional	Microsoft.Office.Excel.Server.Cmdlet.SPExcelUDFPipeBind	Specifies the **SPExcelUserDefinedFunction** object that you want to remove. The type must be a valid name of the code assembly (for example, SampleCompany.SampleApplication.SampleUdf); or an instance of a valid **SPExcelUDF** object.
ExcelServiceApplication	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelServiceApplicationPipeBind	Specifies the Excel Services Application Web service application that contains the list of user-defined functions.

Parameter	Required	Type	Description
			The type must be a valid GUID, in the form 12345678-90ab-cdef-1234-567890bcdefgh; a valid name of an Excel Services Application Web service application in the farm (for example, MyExcelService1); or an instance of a valid **SPExcelServiceApplication** object.
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	Manages objects for the purpose of proper disposal. Use of objects, such as **SPWeb** or **SPSite**, can use large amounts of memory and use of these objects in Windows PowerShell scripts requires proper memory management. Using the **SPAssignment** object, you can assign objects to a variable and dispose of the objects after they are needed to free up memory. When **SPWeb**, **SPSite**, or **SPSiteAdministration** objects are used, the objects are automatically disposed of if an assignment collection or the **Global** parameter is not used. **Note:** When the **Global** parameter is used, all objects are contained in the global store. If objects are not immediately used, or

Parameter	Required	Type	Description
			disposed of by using the **Stop-SPAssignment** command, an out-of-memory scenario can occur.
Confirm	Optional	System.Management.Automation.SwitchParameter	Prompts you for confirmation before executing the command. For more information, type the following command: **get-help about_commonparameters**
WhatIf	Optional	System.Management.Automation.SwitchParameter	Displays a message that describes the effect of the command instead of executing the command. For more information, type the following command: **get-help about_commonparameters**

AutoGenParams

Parameter	Required	Type	Description
Identity	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelUDFPipeBind	
ExcelServiceApplication	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelServiceApplicationPipeBind	
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	
Confirm	Optional	System.Management.Automation.SwitchParameter	
WhatIf	Optional	System.Management.Automation.SwitchParameter	

Input Types

Return Types

Example

Description

----------EXAMPLE 1----------

Code

```
Remove-SPExcelUserDefinedFunction -ExcelServiceApplication "My Excel Service" -Identity
"SampleCompany.SampleApplication.SampleUdf"
```

Comments

This example removes the `SampleCompany.SampleApplication.SampleUdf` user-defined function from the list of user-defined functions that is in the Excel Services Application Web service application named `MyExcelService`.

Example

Description

----------EXAMPLE 2----------

Code

```
Get-SPExcelServiceApplication | Remove-SPExcelUserDefinedFunction -Identity
"SampleCompany.SampleApplication.SampleUdf"
```

Comments

This example removes the `SampleCompany.SampleApplication.SampleUdf` user-defined function from the list of user-defined functions for every Excel Services Application Web service application in the farm.

See Also

Get-SPExcelUserDefinedFunction

New-SPExcelUserDefinedFunction

Set-SPExcelUserDefinedFunction

Set-SPExcelDataConnectionLibrary

Sets properties of a data connection library for Excel Services Application.

Syntax

Set-SPExcelDataConnectionLibrary [-Identity] <SPExcelDCLPipeBind> -ExcelServiceApplication <SPExcelServiceApplicationPipeBind> [-Address <String>] [-AssignmentCollection <SPAssignmentCollection>] [-Confirm [<SwitchParameter>]] [-Description <String>] [-WhatIf [<SwitchParameter>]]

Detailed Description

The **Set-SPExcelDataConnectionLibrary** cmdlet sets the properties of a library from which Excel Services Application loads data connection files. Excel Services Application loads data connection files only if they are stored in a data connection library that is on the list of trusted data connection libraries. Data connection libraries constitute a kind of list that contains data connection files. These files contain everything that Excel Services Application and the Excel client require to connect to an external data source. Data connection libraries enable you to reuse and share data connections. Data connections can be loaded by using information from the workbook file. However, using a data connection library to load data connections provides an additional layer for data connections so that they can be managed separately from workbooks.

Parameters

Parameter	Required	Type	Description
Identity	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelDCLPipeBind	Specifies the **DataConnectionLibrary** object to update. The type must be a valid GUID, in the form 12345678-90ab-cdef-1234-567890bcdefgh; a valid name of a data connection library (for example, DataConnectionLib1); a valid URL, in the

Parameter	Required	Type	Description
			form http://server_name; or an instance of a valid **SPExcelDCL** object.
ExcelServiceApplication	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelServiceApplicationPipeBind	Specifies the Excel Services Application Web service application that contains **SPExcelDataConnectionLibrary** list object. The type must be a valid GUID, in the form 12345678-90ab-cdef-1234-567890bcdefgh; a valid name of an Excel Services Application Web service application in the farm (for example, MyExcelService1); or an instance of a valid **SPExcelServiceApplication** object.
Address	Optional	System.String	Specifies the address of the data connection library. The address must be unique in the list of trusted data connection libraries. The type must be a valid URL, in the form http://server_name.
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	Manages objects for the purpose of proper disposal. Use of objects, such as **SPWeb** or **SPSite**, can

Parameter	Required	Type	Description
			use large amounts of memory and use of these objects in Windows PowerShell scripts requires proper memory management. Using the **SPAssignment** object, you can assign objects to a variable and dispose of the objects after they are needed to free up memory. When **SPWeb**, **SPSite**, or **SPSiteAdministration** objects are used, the objects are automatically disposed of if an assignment collection or the **Global** parameter is not used. **Note:** When the **Global** parameter is used, all objects are contained in the global store. If objects are not immediately used, or disposed of by using the **Stop-SPAssignment** command,

Parameter	Required	Type	Description
			an out-of-memory scenario can occur.
Confirm	Optional	System.Management.Automation.SwitchParameter	Prompts you for confirmation before executing the command. For more information, type the following command: **get-help about_commonparameters**
Description	Optional	System.String	Specifies a friendly description for the data connection library. The type must a valid a string; for example, "This is the data connection library where all the connections to the sales data are stored".
WhatIf	Optional	System.Management.Automation.SwitchParameter	Displays a message that describes the effect of the command instead of executing the command. For more information, type the following command: **get-help about_commonparameters**

AutoGenParams

Parameter	Required	Type	Description
Identity	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelDCLPipeBind	
ExcelServiceApplication	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelServiceApplicationPipeBind	
Address	Optional	System.String	
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	
Confirm	Optional	System.Management.Automation.SwitchParameter	
Description	Optional	System.String	
WhatIf	Optional	System.Management.Automation.SwitchParameter	

Input Types

Return Types

Example

Description

----------EXAMPLE 1----------

Code

```
Get-SPExcelServiceApplication -Identity "MyExcelService" | Get-SPExcelDataConnectionLibrary |
where {$_.address -eq "http://portal/site/salesDCL" } | Set-SPExcelDataConnectionLibrary -
Description "This is the DCL where all the connections to the sales data are stored
```

Comments

This example sets a display description for the entry `http://portal/site/salesDCL` in the list of trusted data connection libraries (`DCL`s) that is on the Excel Services Application Web service application named `MyExcelService`.

Example

Description

----------EXAMPLE 2----------

Code

```
Get-SPExcelServiceApplication -Identity "MyExcelService" | Get-SPExcelDataConnectionLibrary |
where {$_.address -eq "http://portal/site/salesDCL" } | Set-SPExcelDataConnectionLibrary -
Address "http://portal/site/salesDCL
```

Comments

This example updates the address of the data connection library for the entry `http://portal/test/site/salesDCL` to `http://portal/site/salesDCL` in the list of trusted data connection libraries, which is on the Excel Services Application Web service application named `MyExcelService`.

See Also

Get-SPExcelDataConnectionLibrary

New-SPExcelDataConnectionLibrary

Remove-SPExcelDataConnectionLibrary

Set-SPExcelDataProvider

Sets properties of a safe data provider for Excel Services Application.

Syntax

Set-SPExcelDataProvider [-Identity] <SPExcelDataProviderPipeBind> [-AssignmentCollection <SPAssignmentCollection>] [-Confirm [<SwitchParameter>]] [-Description <String>] [-ProviderId <String>] [-ProviderType <Oledb | Odbc | OdbcDsn>] [-WhatIf [<SwitchParameter>]]

Detailed Description

The **Set-SPExcelDataProvider** cmdlet sets the properties of a data provider that is stored in the list of safe data providers. Excel Services Application refreshes external data connections only if the data provider for that connection is in the list of safe data providers. Excel Services Application include a set of common data providers in this list. Therefore, using cmdlets to modify safe data providers is typically an operation of custom data connection scenarios.

Parameters

Parameter	Required	Type	Description
Identity	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelDataProviderPipeBind	Specifies the **SPExcelDataProvider** object to update. The type must be a valid GUID, in the form 12345678-90ab-cdef-1234-567890bcdefgh, or an instance of a valid **SPExcelDataProvider** object.
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	Manages objects for the purpose of proper disposal. Use of objects, such as **SPWeb** or **SPSite**,

Parameter	Required	Type	Description
			can use large amounts of memory and use of these objects in Windows PowerShell scripts requires proper memory management. Using the **SPAssignment** object, you can assign objects to a variable and dispose of the objects after they are needed to free up memory. When **SPWeb**, **SPSite**, or **SPSiteAdministration** objects are used, the objects are automatically disposed of if an assignment collection or the **Global** parameter is not used. **Note:** When the **Global** parameter is used, all objects are contained in the global store. If objects are not immediately used, or disposed of

Parameter	Required	Type	Description
			by using the **Stop-SPAssignment** command, an out-of-memory scenario can occur.
Confirm	Optional	System.Management.Automation.SwitchParameter	Prompts you for confirmation before executing the command. For more information, type the following command: **get-help about_commonparameters**
Description	Optional	System.String	Specifies a display description for the data provider. The type must a valid a string; for example, this is a custom OLEDB provider.
ProviderId	Optional	System.String	Specifies the name of the data provider. This ID should be present in the connection strings generated by this data provider, and the ID must be unique for the data provider type. The type must a valid name of a provider; for example,

Parameter	Required	Type	Description
			CustomOLEDB.
ProviderType	Optional	Microsoft.Office.Excel.Server.DataProviderType	Specifies the supported type of the data provider to set. The type must be one of the following: **OleDb**, **Odbc**, or **OdbcDsn**.
WhatIf	Optional	System.Management.Automation.SwitchParameter	Displays a message that describes the effect of the command instead of executing the command. For more information, type the following command: **get-help about_commonparameters**

AutoGenParams

Parameter	Required	Type	Description
Identity	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelDataProviderPipeBind	
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	
Confirm	Optional	System.Management.Automation.SwitchParameter	
Description	Optional	System.String	
ProviderId	Optional	System.String	
ProviderType	Optional	Microsoft.Office.Excel.Server.DataProviderType	
WhatIf	Optional	System.Management.Automation.SwitchParameter	

Input Types

Return Types

Example

Description

----------EXAMPLE 1----------

Code

```
Get-SPExcelServiceApplication -Identity "MyExcelService" | Get-SPExcelDataProvider | where {$_.providerID -eq "SQLOLEDB" } | Set-SPExcelDataProvider -Description "This is the Microsoft SQL OLEDB driver (MDAC)"
```

Comments

This example retrieves the `SQLOLEDB` data provider from the list of safe data providers that is on the Excel Services Application Web service application named `MyExcelService`, and then adds a display description.

Example

Description

----------EXAMPLE 2----------

Code

```
$provider = Get-SPExcelServiceApplication -Identity "MyExcelService" | Get-SPExcelDataProvider | where {$_.providerID -eq "SQLOLEDB" }
```

Code

```
Set-SPExcelDataProvider $provider –Description "The SQLOLEDB provider description has changed."
```

Code

```
$provider | format-table
```

Comments

This example retrieves the `SQLOLEDB` data provider from the list of safe data providers that is on the Excel Services Application Web service application named `MyExcelService`, as an object, adds a new display description, and then displays the provider output in a Command Prompt window, formatted as a table.

See Also

Get-SPExcelDataProvider

Remove-SPExcelDataProvider

Set-SPExcelDataProvider

Set-SPExcelFileLocation

Sets properties of a trusted file location for Excel Services Application.

Syntax

Set-SPExcelFileLocation [-Identity] <SPExcelFileLocationPipeBind> -ExcelServiceApplication <SPExcelServiceApplicationPipeBind> [-AbortOnRefreshOnOpenFail <SwitchParameter>] [-Address <String>] [-AssignmentCollection <SPAssignmentCollection>] [-AutomaticVolatileFunctionCacheLifetime <Int32>] [-ChartAndImageSizeMax <Int32>] [-ChartRenderDurationMax <Int32>] [-ConcurrentDataRequestsPerSessionMax <Int32>] [-Confirm [<SwitchParameter>]] [-DefaultWorkbookCalcMode <File | Manual | Auto | AutoDataTables>] [-Description <String>] [-DisplayGranularExtDataErrors <SwitchParameter>] [-ExternalDataAllowed <None | Dcl | DclAndEmbedded>] [-IncludeChildren <SwitchParameter>] [-LocationType <SharePoint | Unc | Http>] [-ManualExtDataCacheLifetime <Int32>] [-NewWorkbookSessionTimeout <Int32>] [-PeriodicExtDataCacheLifetime <Int32>] [-RequestDurationMax <Int32>] [-RESTExternalDataAllowed <SwitchParameter>] [-SessionTimeout <Int32>] [-ShortSessionTimeout <Int32>] [-UdfsAllowed <SwitchParameter>] [-WarnOnDataRefresh <SwitchParameter>] [-WhatIf [<SwitchParameter>]] [-WorkbookSizeMax <Int32>]

Detailed Description

The **Set-SPExcelFileLocation** cmdlet sets properties of a location from which Excel Services Application loads workbooks. Excel Services Application loads only workbooks that are stored in a trusted file location. Properties of trusted file locations control how workbooks can be used when they are loaded on Excel Services Application. Excel Services Application always enforces the properties that are defined by the trusted file location from which a workbook was loaded. The properties that are used by the trusted file location are determined by comparison of the file path for the workbook with the **Address** parameter of the trusted file location.

Parameters

Parameter	Required	Type	Description
Identity	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelFileLocationPipeBind	Specifies the **FileLocation** object to update. The name must be unique in the

Parameter	Required	Type	Description
			list of file locations. The type must be a valid GUID, in the form 12345678-90ab-cdef-1234-567890bcdefgh; a valid string identifying the file location, in the form http://myPortal/myTeam; a valid string identifying the path, in the form C:\folder_name; or an instance of a valid **SPExcelFileLocation** object.
ExcelServiceApplication	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelServiceApplicationPipeBind	Specifies the Excel Services Application Web service application that contains the **SPExcelFileLocation** list object. The type must be a valid GUID, in the form 12345678-90ab-cdef-1234-567890bcdefgh; a valid name of an Excel Services Application Web

Parameter	Required	Type	Description
			service application in the farm (for example, MyExcelService1) ; or an instance of a valid **SPExcelServiceApplication** object.
AbortOnRefreshOnOpenFail	Optional	System.Management.Automation.SwitchParameter	Specifies that the load of a Excel Services Application file automatically fails if an automatic data refresh operation fails when the file is opened. The load is failed only if the following conditions are true: -- User has read-only permissions to the Excel Services Application file. -- Workbook file has data connections that refresh automatically when the Excel file is opened.
Address	Optional		System.String Specifies a Web folder, or file

Parameter	Required	Type	Description
			directory from which Excel Services Application can load workbooks. The type must be a valid URL, in the form http://myPortal/myTeam; or a valid UNC path in form, \\server_name\folder_name
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	Manages objects for the purpose of proper disposal. Use of objects, such as **SPWeb** or **SPSite**, can use large amounts of memory and use of these objects in Windows PowerShell scripts requires proper memory management. Using the **SPAssignment** object, you can assign objects to a variable and dispose of the objects after they are needed to free up memory. When **SPWeb**, **SPSite**, or **SPSiteAdministr**

Parameter	Required	Type	Description
			ation objects are used, the objects are automatically disposed of if an assignment collection or the **Global** parameter is not used. **Note:** When the **Global** parameter is used, all objects are contained in the global store. If objects are not immediately used, or disposed of by using the **Stop-SPAssignment** command, an out-of-memory scenario can occur.
AutomaticVolatileFunctionCacheLifetime	Optional	System.Int32	Specifies the maximum time, in

Parameter	Required	Type	Description
			seconds, that a computed value for a volatile function is cached for automatic recalculations. Valid values include: **-1**, calculate once when the workbook loads **0**, always calculate **1** to **2073600**, cache 1 second to 24 days The type must be the integers -1 or 0, or an integer value in the range of 1 to 2073600.
ChartAndImageSizeMax	Optional	System.Int32	Specifies the maximum size, in megabytes, of a chart or image that can be opened. The default value is **1**. The type must be a valid integer greater than 0.
ConcurrentDataRequestsPerSessionMax	Optional	System.Int32	Specifies the maximum number of concurrent external data requests allowed in each session. If a session must

Parameter	Required	Type	Description
			issue more than this number of requests, additional requests must be queued. The scope of this setting is the logical server. The default value is **5**. The type must be a positive integer.
Confirm	Optional	System.Management.Automation.SwitchParameter	Prompts you for confirmation before executing the command. For more information, type the following command: **get-help about_commonparameters**
DefaultWorkbookCalcMode	Optional	Microsoft.Office.Excel.Server.DefaultWorkbookCalcMode	Specifies the calculation mode of workbooks. All of the settings except the **File** setting override the workbook settings. If this parameter is changed, it is applied to all new sessions on Excel Services Application when the ConfigDB

Parameter	Required	Type	Description
			properties cache is refreshed. The default setting is **File**. The type must be one of the following: **File**, **Manual**, **Auto**, or **AutoDataTables**.
Description	Optional	System.String	Specifies a friendly description for the new file location. The type must be a valid string; for example, this is the file location of the monthly sales reports.
DisplayGranularExtDataErrors	Optional	System.Management.Automation.SwitchParameter	Displays granular error messages for external data failures for files in this location.
ExternalDataAllowed	Optional	Microsoft.Office.Excel.Server.AllowExternalData	Specifies the type of external data access allowed for workbooks: **None** specifies that data refresh is disabled. **Dcl** specifies that data is refreshed only when it uses a connection file in a trusted data connection library.

Parameter	Required	Type	Description
			DclandEmbedded specifies that a data connection library can be used but is not required. The scope of this setting is the trusted location. The default value is **None**. The type must be one of the following: **None**, **Dcl**, or **DclAndEmbedded**.
IncludeChildren	Optional	System.Management.Automation.SwitchParameter	Indicates that subordinate URLs, directories and libraries are trusted. For example, if the trusted file location is http://portal, and **IncludeChildren** is true, then http://portal/subSite is also trusted, as is any subURL under the URL specified in **Address**.
LocationType	Optional	Microsoft.Office.Excel.Server.LocationType	The type of location. **LocationType** indicates how Excel Services

Parameter	Required	Type	Description
			Application retrieves files from that location specified in **Address**. The default value is **SharePoint**. The type must be one of the following: **SharePoint**, **UNC**, or **HTTP**.
ManualExtDataCacheLifetime	Optional	System.Int32	Specifies the time, in seconds, that Excel Services Application waits before re-issuing a manual, or user-initiated, external data request. During this period, the results from the previous external data request may be returned from an external data cache. Valid values are: **-1** Never refresh after first query **1 to 2073600** Caches 1 second to 24 days. The default value is **300**.

Parameter	Required	Type	Description
NewWorkbookSessionTimeout	Optional	System.Int32	Specifies the time, in seconds, that a session for a new, unsaved, workbook will remain active on Excel Services Application with no user activity. The default value **1800**. Valid values include the following: **-1**, session never expires **0**, session expires at the end of a single request **1** to **2073600**, remain active 1 second to 24 days. The type must be -1, 0, or an integer in the range of 1 to 2073600.
PeriodicExtDataCacheLifetime	Optional	System.Int32	Specifies the time, in seconds, that Excel Services Application waits before re-issuing an *on-open* or *periodic* (that is, automatic) external data request. During this period, the

Parameter	Required	Type	Description
			results from the previous external data request may be returned from an external data cache. A value -1 specifies to never refresh after the first query. The type must be -1, 0, or an integer in the range of 1 to 2073600. The default value is **300**.
RequestDurationMax	Optional	System.Int32	Specifies the maximum duration, in seconds, for a single request in a session. After this time elapses the request is timed out. The type must be -1 (no limit) or an integer in the range of 1 to 2073600. The default value is **300**.
RESTExternalDataAllowed	Optional	System.Management.Automation.SwitchParameter	Specifies whether requests from the Representational State Transfer (REST) Application Programming

Parameter	Required	Type	Description
			Interface (API) are permitted to refresh external data connections. The **RESTExternalDataAllowed** parameter has no effect if the **External DataAllowed** parameter is set to **None**.
SessionTimeout	Optional	System.Int32	Specifies the time, in seconds, that a session will remain active on Excel Services Application with no user activity. The default is 300 seconds. The default value is **300**. Valid values include: **-1**, session never expires **0**, session expires at the end of a single request **1** to **2073600**, remain active 1 second to 24 days The type must be -1, 0, or an integer in the range of 1 to 2073600.

Parameter	Required	Type	Description
ShortSessionTimeout	Optional	System.Int32	Specifies the time, in seconds, that a user has to make the initial interaction with a spreadsheet. After this time has elapsed, if the user is not active, then the session is closed. The value must always be less than **SessionTimeout**. If the **ShortSessionTimeout** value is greater than or equal to **SessionTimeout**, an error is returned. The default value is **75**. The type must be -1 or an integer in the range of 1 to 2073600.
UdfsAllowed	Optional	System.Management.Automation.SwitchParameter	Specifies that user-defined functions can be called by workbooks loaded from the trusted file location that is specified in **Address**.
WarnOnDataRefresh	Optional	System.Management.Automation.SwitchPara	Specifies that a

Parameter	Required	Type	Description
		meter	warning is displayed to the user on the first refresh of data for the workbook. The default value is **True**.
WhatIf	Optional	System.Management.Automation.SwitchParameter	Displays a message that describes the effect of the command instead of executing the command. For more information, type the following command: **get-help about_commonparameters**
WorkbookSizeMax	Optional	System.Int32	Specifies the maximum size, in megabytes, of a workbook that can be loaded. The default value is **10**. The type must be an integer value in the range of 1 to 2000.

AutoGenParams

Parameter	Required	Type	Description
Identity	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelFileLocationPipeBind	
ExcelServiceApplication	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelServiceApplicationPipeBind	
AbortOnRefreshOnOpenFail	Optional	System.Management.Automation.SwitchParameter	
Address	Optional	System.String	
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	
AutomaticVolatileFunctionCacheLifetime	Optional	System.Int32	
ChartAndImageSizeMax	Optional	System.Int32	
ChartRenderDurationMax	Optional	System.Int32	
ConcurrentDataRequestsPerSessionMax	Optional	System.Int32	
Confirm	Optional	System.Management.Automation.SwitchParameter	
DefaultWorkbookCalcMode	Optional	Microsoft.Office.Excel.Server.DefaultWorkbookCalcMode	
Description	Optional	System.String	
DisplayGranularExtDataErrors	Optional	System.Management.Automation.SwitchParameter	
ExternalDataAllowed	Optional	Microsoft.Office.Excel.Server.AllowExternalData	
IncludeChildren	Optional	System.Management.Automation.SwitchParameter	
LocationType	Optional	Microsoft.Office.Excel.Server.LocationType	
ManualExtDataCacheLifetime	Optional	System.Int32	
NewWorkbookSessionTimeout	Optional	System.Int32	

Parameter	Required	Type	Description
PeriodicExtDataCacheLifetime	Optional	System.Int32	
RequestDurationMax	Optional	System.Int32	
RESTExternalDataAllowed	Optional	System.Management.Automation.SwitchParameter	
SessionTimeout	Optional	System.Int32	
ShortSessionTimeout	Optional	System.Int32	
UdfsAllowed	Optional	System.Management.Automation.SwitchParameter	
WarnOnDataRefresh	Optional	System.Management.Automation.SwitchParameter	
WhatIf	Optional	System.Management.Automation.SwitchParameter	
WorkbookSizeMax	Optional	System.Int32	

Input Types

Return Types

Example

Description

----------EXAMPLE 1----------

Code

```
Get-SPExcelFileLocation –ExcelServiceApplication "My Excel Service" | where {
$_.externaldataallowed –eq "DclAndEmbedded"} | Set-SPExcelFileLocation –ExernalDataAllowed Dcl
```

Comments

This example gets every trusted file location for the Excel Services Application Web service application named `My Excel Service` that allows data connections to load from workbook files and specifies that the locations must use a data connection library to load data connections.

Example

Description

----------EXAMPLE 2----------

Code

```
Get-SPExcelServiceApplication | Get-SPExcelFileLocation | where {$_.Address -eq "http://"} |
Set-SPExcelFileLocation -Description "This is the default trusted file location for Excel
Services Application. It allows any file from anywhere in SharePoint to load."
```

Comments

This example changes the description of the default file location for every Excel Services Application Web service application in the farm.

See Also

Get-SPExcelFileLocation

New-SPExcelFileLocation

Remove-SPExcelFileLocation

Set-SPExcelServiceApplication

Sets global properties for Excel Services Application.

Syntax

Set-SPExcelServiceApplication [[-Identity] <SPExcelServiceApplicationPipeBind>] [-AssignmentCollection <SPAssignmentCollection>] [-CachingOfUnusedFilesEnable <SwitchParameter>] [-Confirm [<SwitchParameter>]] [-CrossDomainAccessAllowed <SwitchParameter>] [-EncryptedUserConnectionRequired <None | Connection>] [-ExternalDataConnectionLifetime <Int32>] [-FileAccessMethod <UseImpersonation | UseFileAccessAccount>] [-LoadBalancingScheme <WorkbookUrl | RoundRobin | Local>] [-MemoryCacheThreshold <Int32>] [-PrivateBytesMax <Int32>] [-SessionsPerUserMax <Int32>] [-SiteCollectionAnonymousSessionsMax <Int32>] [-TerminateProcessOnAccessViolation <SwitchParameter>] [-ThrottleAccessViolationsPerSiteCollection <Int32>] [-UnattendedAccountApplicationId <String>] [-UnusedObjectAgeMax <Int32>] [-WhatIf [<SwitchParameter>]] [-WorkbookCache <String>] [-WorkbookCacheSizeMax <Int32>]

Detailed Description

The **Set-ExcelServiceApplication** cmdlet changes global runtime properties for Excel Services Application. Changes to properties that are made by using this cmdlet affect all machines in the farm that are running an instance of the specified Excel Services Application Web service application.

Parameters

Parameter	Required	Type	Description
Identity	Optional	Microsoft.Office.Excel.Server.Cmdlet.SPExcelServiceApplicationPipeBind	Specifies the **SPExcelServiceApplication** object to update. The type must be a valid GUID, in the form 12345678-90ab-cdef-1234-567890bcdefgh; a valid name of an Excel Services Application Web service application in the farm (for example,

Parameter	Required	Type	Description
			MyExcelService1); or an instance of a valid **SPExcelServiceApplication** object.
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	Manages objects for the purpose of proper disposal. Use of objects, such as **SPWeb** or **SPSite**, can use large amounts of memory and use of these objects in Windows PowerShell scripts requires proper memory management. Using the **SPAssignment** object, you can assign objects to a variable and dispose of the objects after they are needed to free up memory. When **SPWeb**, **SPSite**, or **SPSiteAdministration** objects are used, the objects are automatically disposed of if an assignment collection or the **Global** parameter is not used. **Note:** When the **Global** parameter is used, all objects are contained in the global store. If objects are not immediately used, or disposed of by using the **Stop-SPAssignment**

Parameter	Required	Type	Description
			command, an out-of-memory scenario can occur.
CachingOfUnusedFilesEnable	Optional	System.Management.Automation.SwitchParameter	Specifies that files that are no longer used by Excel Services Application can remain in the cache for later use. The default is true when the parameter is not specified on the command.
Confirm	Optional	System.Management.Automation.SwitchParameter	Prompts you for confirmation before executing the command. For more information, type the following command: **get-help about_commonparameters**
CrossDomainAccessAllowed	Optional	System.Management.Automation.SwitchParameter	Specifies that trusted workbooks and data connection files can be requested and rendered by Web Parts or pages that reside in other HTTP domains.
EncryptedUserConnectionRequired	Optional	Microsoft.Office.Excel.Server.EncryptedConnectionMode	Requires that encryption is used between the end-user and the server running Excel Services Application. The scope of this setting is the logical server. The default value is **None**. The type must be one of the following: **None** or

Parameter	Required	Type	Description
			Connection.
ExternalDataConnectionLifetime	Optional	System.Int32	Specifies the maximum number of seconds that an external data connection can remain open in the connection pool. Valid values are: **-1** Never recycle **1 to 2073600** Remain active 1 second to 24 days. The default value is **1800**.
FileAccessMethod	Optional	Microsoft.Office.Excel.Server.FileAccessMethod	Specifies the authentication method that Excel Services Application uses to retrieve files. **UseFileAccessAccount** specifies that all workbooks always use the credentials of the service account running Excel Services Application. **UseImpersonation** specifies that Excel Services Application retrieves the workbook files as the user who makes the request. The default value is **UseImpersonation**. The type must be one of the following: **UseImpersonation** or **UseFileAccessAccount**.

Parameter	Required	Type	Description
LoadBalancingScheme	Optional	Microsoft.Office.Excel.Server.LoadBalancingScheme	Specifies the load-balancing schema that is used by the Excel Services Application Web service application to send requests to different back-end Excel Services Application computers. The default value is **WorkbookURL**. The type must be one of the following: RoundRobin, Local, or WorkbookURL.
MemoryCacheThreshold	Optional	System.Int32	Specifies the percentage of the maximum private bytes that can be allocated to inactive objects. When the memory cache threshold is exceeded, cached objects that are not currently in use are released. Setting this value to **0** disables caching of inactive objects. The default value is **90**. The type must be an integer in the range of 0 to 95.
PrivateBytesMax	Optional	System.Int32	Specifies the maximum private bytes, in megabytes, that are used by Excel Services Application. When set to -1, the maximum private bytes defaults to 50 percent of physical

Parameter	Required	Type	Description
			memory on the computer. Valid values include -1 for no limit, and any positive integer. The default value is **-1**. The type must be -1 or any positive integer.
SessionsPerUserMax	Optional	System.Int32	Specifies the maximum number of sessions allowed for a user. If a user reaches this maximum the oldest session is deleted when a new session is started. Valid values include -1 for no limit, and any positive integer. The default value is **25**. The type must be -1 or any positive integer.
SiteCollectionAnonymousSessionsMax	Optional	System.Int32	Specifies the maximum number of anonymous sessions allowed per site collection. A value of 0 specifies that no anonymous sessions are allowed. Valid values include -1 for no limit, 0 to disable the setting, or any positive integer. The default value is **25**. The type must be -1, 0, or any positive integer.
TerminateProcessOnAccessViolation	Optional	System.Management.Automation.SwitchParameter	Terminates Excel Services Application when an access violation occurs in the process.

Parameter	Required	Type	Description
ThrottleAccessViolationsPerSiteCollection	Optional	System.Int32	Specifies that if a workbook causes an access violation error on Excel Services Application, all files originating from that workbook's site collection are blocked from loading for the specified period (in seconds). Existing sessions for files on that site collection are unaffected. Valid values include 0 to disable the setting, and the range of 1 to 2073600 seconds (24 days). The default value is **0**. The type must be an integer value in the range of 0 to 2073600.
UnattendedAccountApplicationId	Optional	System.String	Specifies that the application ID that is used to look up the unattended service account credentials from the secure storage service that is specified by the **UnattendedAccountSecureServiceAppName** parameter. The secure storage service is contacted when this parameter is set and specific error cases occur. Valid values are alphanumeric strings with a maximum of 256 characters. The default

Parameter	Required	Type	Description
			value is an empty string. The type must be a valid string of alphanumeric characters; for example, UnattendedServiceAccountID. **Note:** The unattended account is a single account that all workbooks can use to refresh data. An unattended account is required when workbook connections specify *None* for authentication, or when any non-Windows credentials are used to refresh data.
UnusedObjectAgeMax	Optional	System.Int32	Specifies the maximum amount of time, in minutes, that objects not currently used in a session are kept in the memory cache. Valid values include -1 for no maximum setting, and a positive integer in the range of 1 to 34560 minutes (24 days). The default value is **-1**. The type must be -1, or an integer in the range of

Parameter	Required	Type	Description
			1 to 34560.
WhatIf	Optional	System.Management.Automation.SwitchParameter	Displays a message that describes the effect of the command instead of executing the command. For more information, type the following command: **get-help about_commonparameters**
WorkbookCache	Optional	System.String	Specifies the local file system location of the cache that is used to store workbooks that are used by Excel Services Application. If no value is specified, the default value is a subdirectory in the system temporary directory. The type must be a valid path in either of the following forms: -- C:\folder_name -- \\server_name\folder_name
WorkbookCacheSizeMax	Optional	System.Int32	Specifies the maximum allowable size, in megabytes, of an individual session. The default value is **0**. The type must be any positive integer.

AutoGenParams

Parameter	Required	Type	Description
Identity	Optional	Microsoft.Office.Excel.Server.Cmdlet.SPExcelServiceApplicationPipeBind	
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	
CachingOfUnusedFilesEnable	Optional	System.Management.Automation.SwitchParameter	
Confirm	Optional	System.Management.Automation.SwitchParameter	
CrossDomainAccessAllowed	Optional	System.Management.Automation.SwitchParameter	
EncryptedUserConnectionRequired	Optional	Microsoft.Office.Excel.Server.EncryptedConnectionMode	
ExternalDataConnectionLifetime	Optional	System.Int32	
FileAccessMethod	Optional	Microsoft.Office.Excel.Server.FileAccessMethod	
LoadBalancingScheme	Optional	Microsoft.Office.Excel.Server.LoadBalancingScheme	
MemoryCacheThreshold	Optional	System.Int32	
PrivateBytesMax	Optional	System.Int32	
SessionsPerUserMax	Optional	System.Int32	
SiteCollectionAnonymousSessionsMax	Optional	System.Int32	
TerminateProcessOnAccessViolation	Optional	System.Management.Automation.SwitchParameter	
ThrottleAccessViolationsPerSiteCollection	Optional	System.Int32	
UnattendedAccountApplicationId	Optional	System.String	
UnusedObjectAgeMax	Optional	System.Int32	

Parameter	Required	Type	Description
WhatIf	Optional	System.Management.Automation.SwitchParameter	
WorkbookCache	Optional	System.String	
WorkbookCacheSizeMax	Optional	System.Int32	

Input Types

Return Types

Example

Description

----------EXAMPLE 1----------

Code

```
Set-SPExcelServiceApplication ExcelServiceTestApplication –sessionsperusermax 10
```

Comments

This example sets the Excel Services Application Web service application named `ExcelServiceTestApplication` to enable each user to open no more than `10` sessions at one time on a back-end application server that runs Excel Services Application.

Example

Description

----------EXAMPLE 2----------

Code

```
Get-SPExcelServiceApplication | where{ $_.privatebytesmax –ne -1 } | Set-
SPExcelServiceApplication –PrivateBytesMax -1
```

Comments

This example displays every Excel Services Application application that runs in the farm that is using a user-specified value for the maximum number of private bytes of memory on the computer, and then sets that value to **-1**, which indicates that Excel Services Application will consume 50 percent of the available memory resources.

See Also

Get-SPExcelServiceApplication

New-SPExcelServiceApplication

Set-SPExcelUserDefinedFunction

Sets properties of a user-defined function in Excel Services Application.

Syntax

Set-SPExcelUserDefinedFunction [-Identity] <SPExcelUDFPipeBind> -ExcelServiceApplication <SPExcelServiceApplicationPipeBind> [-Assembly <String>] [-AssemblyLocation <GAC | LocalFile>] [-AssignmentCollection <SPAssignmentCollection>] [-Confirm [<SwitchParameter>]] [-Description <String>] [-Enable <SwitchParameter>] [-WhatIf [<SwitchParameter>]]

Detailed Description

The **Set-SPExcelUserDefinedFunction** cmdlet sets properties of a user-defined function assembly. Excel Services Application uses only user-defined functions that have an entry in the user-defined function list. User-defined functions are managed code assemblies that can be called from a Excel Services Application workbook by using standard Excel Services Application formula syntax. The assemblies can perform custom logic or other actions, such as refresh data. Windows PowerShell cmdlets operate on the entries in the user-defined function list and do not operate on the assemblies.

Parameters

Parameter	Required	Type	Description
Identity	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelUDFPipeBind	Specifies the **SPExcelUserDefinedFunction** object to update. The type must be a valid name of the code assembly; for example, SampleCompany.SampleApplication.SampleUdf; or an instance of a valid **SPExcelUDF** object.
ExcelServiceApplication	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelServiceApplicationPipeBind	Specifies the Excel Services Application Web service application that contains the **SPExcelUserDefinedFuncti**

Parameter	Required	Type	Description
			on list object. The type must be a valid GUID, in the form 12345678-90ab-cdef-1234-567890bcdefgh; a valid name of an Excel Services Application Web service application in the farm (for example, MyExcelService1); or an instance of a valid **SPExcelServiceApplication** object.
Assembly	Optional	System.String	Specifies the strongly-typed name of the assembly. The name must be unique in the list of user-defined functions, and cannot be more than 4096 alphanumeric characters long. The type must be a string of alphanumeric characters; for example, SampleCompany.SampleApplication.SampleUdf, C:\UDFs\SampleUdf.dll, or \\MyNetworkServer\UDFs\SampleUdf.dll.
AssemblyLocation	Optional	Microsoft.Office.Excel.Server.AssemblyLocation	Specifies the location where the assembly is stored. The type must be one of the following: GAC or LocalFile.
Assignment Collection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	Manages objects for the purpose of proper disposal. Use of objects, such as **SPWeb** or **SPSite**, can use large amounts of memory and use of these objects in Windows PowerShell scripts

Parameter	Required	Type	Description
			requires proper memory management. Using the **SPAssignment** object, you can assign objects to a variable and dispose of the objects after they are needed to free up memory. When **SPWeb**, **SPSite**, or **SPSiteAdministration** objects are used, the objects are automatically disposed of if an assignment collection or the **Global** parameter is not used. **Note:** When the **Global** parameter is used, all objects are contained in the global store. If objects are not immediately used, or disposed of by using the **Stop-SPAssignment** command, an out-of-memory scenario can occur.
Confirm	Optional	System.Management.Automation.SwitchParameter	Prompts you for confirmation before executing the command. For more information, type the following command: **get-help about_commonparameters**
Description	Optional	System.String	Specifies a display description for the user-defined function. The name cannot be more than 4096 alphanumeric characters long.

Parameter	Required	Type	Description
			The type must be a string of alphanumeric characters; for example, this is the user-defined function for consolidation reports.
Enable	Optional	System.Management.Automation.SwitchParameter	Activates the user-defined function for use with Excel Services Application.
WhatIf	Optional	System.Management.Automation.SwitchParameter	Displays a message that describes the effect of the command instead of executing the command. For more information, type the following command: **get-help about_commonparameters**

AutoGenParams

Parameter	Required	Type	Description
Identity	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelUDFPipeBind	
ExcelServiceApplication	Required	Microsoft.Office.Excel.Server.Cmdlet.SPExcelServiceApplicationPipeBind	
Assembly	Optional	System.String	
AssemblyLocation	Optional	Microsoft.Office.Excel.Server.AssemblyLocation	
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	
Confirm	Optional	System.Management.Automation.SwitchParameter	
Description	Optional	System.String	
Enable	Optional	System.Management.Automation.SwitchParameter	
WhatIf	Optional	System.Management.Automation.SwitchParameter	

Input Types

Return Types

Example

Description

----------EXAMPLE 1----------

Code

```
Set-SPExcelUserDefinedFunction -ExcelServiceApplication "My Excel Service" -Identity
SampleCompany.SampleApplication.SampleUdf -Description "This is the new description for the
sample UDF"
```

Comments

This example changes the description of the user-defined function (UDF) to the UDF `SampleCompany.SampleApplication.SampleUdf` file for the Excel Services Application Web service application named `My Excel Service`.

Example

Description

----------EXAMPLE 2----------

Code

```
Get-SPExcelServiceApplication "My Excel Service" | Set-SPExcelUserDefinedFunction -Identity
SampleCompany.SampleApplication.SampleUdf -Enable: $false
```

Comments

This example disables the sample UDF `SampleCompany.SampleApplication.SampleUdf` file from the Excel Services Application named `My Excel Service`.

See Also

Get-SPExcelUserDefinedFunction

New-SPExcelUserDefinedFunction

Remove-SPExcelUserDefinedFunction

Office Web Apps cmdlets

Office Web Apps is a new Web-based productivity offering from Microsoft Office 2010 suites. Office Web Apps include companions to Microsoft Word 2010, Microsoft Excel 2010, Microsoft PowerPoint 2010, and Microsoft OneNote 2010. These Web-based applications are stand-alone applications focused on offering access to Word 2010, PowerPoint 2010, Excel 2010, and OneNote 2010 documents through any browser across multiple platforms; lightweight creation and editing capabilities in standard formats; sharing and collaboration on those documents through the browser; and a variety of Web-enabled scenarios. Documents created by using Office Web Apps are no different than documents created by using the corresponding desktop applications. The associated services are used to prepare documents for viewing and editing in a Web browser

All Office Web Apps will support backup and recovery regardless of whether there is a UI setting. However, backup and recovery only applies to the service-level and administrative-level settings; Web Part properties are not backed up as part of this process. Also, URLs must be fixed after a restore process.

In addition to providing backup and restore functionality, Office Web Apps provide Windows PowerShell functionality that you can use to provision the service from scratch or provision a new instance that uses settings from a previous backup, configure and manage application proxies and service applications, and configure the global settings of the service.

Cmdlet name	Description
Get-SPOfficeWebAppsCache	Gets the **SPSite** object that represents the Microsoft Office Web Apps cache associated with a particular Web application.
Get-SPOneNoteWebAppConfig	Gets the **SPOneNoteWebAppConfig** object.
New-SPOfficeWebAppsCache	Creates a new Microsoft Office Web Apps cache for a given Web application.
New-SPWordViewingServiceApplication	Creates a new instance of a Word Viewing Service application.
New-SPWordViewingServiceApplicationProxy	Creates a new service application proxy for use with the Word Viewing Service application.
Remove-SPOfficeWebAppsCache	Deletes the Microsoft Office Web Apps cache.
Set-SPOfficeWebAppsCache	Modifies properties on the Microsoft Office Web Apps cache.

Cmdlet name	Description
Set-SPOneNoteWebAppConfig	Modifies properties on the **SPOneNoteWebAppConfig** object that is activated on the specified **SPSite** object.
Set-SPWordViewingServiceApplication	Sets a global property for a Word Viewing Service application.
Set-SPWordViewingServiceApplicationProxy	Sets a global value for a Word Viewing Service application proxy.
Get-SPPowerPointServiceApplication	Displays a list of instances of the PowerPoint Service application.
Get-SPPowerPointServiceApplicationProxy	Displays one or more objects for a PowerPoint Service application proxy.
New-SPPowerPointServiceApplication	Creates a PowerPoint Service application.
New-SPPowerPointServiceApplicationProxy	Creates a PowerPoint Service application proxy.
Remove-SPPowerPointServiceApplication	Removes an instance of the PowerPoint Service application.
Remove-SPPowerPointServiceApplicationProxy	Removes an instance of a PowerPoint Service application proxy.
Set-SPPowerPointServiceApplication	Configures global settings for a PowerPoint Service application.
Set-SPPowerPointServiceApplicationProxy	Configures global settings for a PowerPoint Service application proxy object.

Get-SPOfficeWebAppsCache

Gets the **SPSite** object that represents the Microsoft Office Web Apps cache associated with a particular Web application.

Syntax

Get-SPOfficeWebAppsCache [[-WebApplication] <SPWebApplicationPipeBind>] [-AssignmentCollection <SPAssignmentCollection>]

Get-SPOfficeWebAppsCache [-AssignmentCollection <SPAssignmentCollection>] [-Site <SPSitePipeBind>]

Detailed Description

The **Get-SPOfficeWebAppsCache** cmdlet retrieves the site collection that stores each of the cached renditions for the Word Viewing Service and PowerPoint Service applications. The **SPSite** object can be used to adjust the expiration of documents cached within that site collection, and to change the maximum size to which the companion cache can grow.

Parameters

Parameter	Required	Type	Description
WebApplication	Optional	Microsoft.SharePoint.PowerShell.SPWebApplicationPipeBind	Specifies the Web application in which the companion cache is located.
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	Manages objects for the purpose of proper disposal. Use of objects, such as **SPWeb** or **SPSite**, can use large amounts of memory and use of these objects in Windows

Parameter	Required	Type	Description
			PowerShell scripts requires proper memory management. Using the **SPAssignment** object, you can assign objects to a variable and dispose of the objects after they are needed to free up memory. When **SPWeb**, **SPSite**, or **SPSiteAdministration** objects are used, the objects are automatically disposed of if an assignment collection or the **Global** parameter is not used. **Note:** When the **Global** parameter is used, all objects are contained in the global store. If objects are not immediately used, or disposed of by using

Parameter	Required	Type	Description
			the **Stop-SPAssignment** command, an out-of-memory scenario can occur.
Site	Optional	Microsoft.SharePoint.PowerShell.SPSitePipeBind	Specifies the identity of the Office Web Apps cache site collection.

AutoGenParams

Parameter	Required	Type	Description
WebApplication	Optional	Microsoft.SharePoint.PowerShell.SPWebApplicationPipeBind	
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	
Site	Optional	Microsoft.SharePoint.PowerShell.SPSitePipeBind	

Input Types

Return Types

Example

Description

----------EXAMPLE 1----------

Code

```
$a = Get-SPOfficeWebAppsCache -WebApplication "SharePoint - 80"
```

Code

```
Get-Member -InputObject $a
```

Code

```
$a.Quota
```

Code

```
$a.Owner
```

Comments

This example returns the current quota and owner for the Office Web Apps cache that is associated with the Web application named `SharePoint - 80`.

Example

Description

----------EXAMPLE 2----------

Code

```
$cache = Get-SPOfficeWebAppsCache -WebApplication <WebAppURL>
```

Code

```
$cache.Quota.StorageMaximumLevel
```

Code

```
$cache.Quota.StorageWarningLevel
```

Comments

This example returns the current values for the MaxSizeInBytes and WarningSizeInBytes properties.

Example

Description

----------EXAMPLE 3----------

Code

```
$cache = Get-SPOfficeWebAppsCache -WebApplication <WebAppURL>
```

Code

```
$web = Get-SPWeb -Site $cache.URL
```

Code

```
$web.Properties
```

Comments

This example returns the current values for the ExpirationPeriodInDays property.

See Also

New-SPOfficeWebAppsCache

Remove-SPOfficeWebAppsCache

Set-SPOfficeWebAppsCache

Get-SPOneNoteWebAppConfig

Gets the **SPOneNoteWebAppConfig** object.

Syntax

Get-SPOneNoteWebAppConfig [-AssignmentCollection <SPAssignmentCollection>] [-Site <SPSitePipeBind>]

Detailed Description

The settings for OneNote Web App are stored as part of the **SPFeature** object that is associated with OneNote Web App. Use the **Get-SPOneNoteWebAppConfig** cmdlet to retrieve the **SPFeature** object associated with OneNote Web App on a particular **SPSite** object. Also use the **Get-SPOneNoteWebAppConfig** cmdlet to provide a way by which an administrator can access the appropriate properties for OneNote Web App.

Parameters

Parameter	Required	Type	Description
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	Manages objects for the purpose of proper disposal. Use of objects, such as **SPWeb** or **SPSite**, can use large amounts of memory and use of these objects in Windows PowerShell scripts requires proper memory management. Using the **SPAssignment** object, you can assign objects to a

Parameter	Required	Type	Description
			variable and dispose of the objects after they are needed to free up memory. When **SPWeb**, **SPSite**, or **SPSiteAdministration** objects are used, the objects are automatically disposed of if an assignment collection or the **Global** parameter is not used. **Note:** When the **Global** parameter is used, all objects are contained in the global store. If objects are not immediately used, or disposed of by using the **Stop-SPAssignment** command, an out-of-memory scenario can occur.

Parameter	Required	Type	Description
Site	Optional	Microsoft.SharePoint.PowerShell.SPSitePipeBind	Specifies the **SPSite** object on which Microsoft OneNote Web App is activated.

AutoGenParams

Parameter	Required	Type	Description
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	
Site	Optional	Microsoft.SharePoint.PowerShell.SPSitePipeBind	

Input Types

Return Types

Example

Description

----------EXAMPLE----------

Code

```
$b = Get-SPOneNoteWebAppConfig -Site "http://sitename"
```

Code

```
Get-Member -InputObject $b
```

Code

```
$b.Properties
```

Comments

This example displays all of the properties for OneNote Web App at the site located at `http://sitename`.

See Also

Set-SPOneNoteWebAppConfig

Get-SPPowerPointServiceApplication

Displays a list of instances.

Syntax

Get-SPPowerPointServiceApplication [[-Identity] <SPPowerPointServiceApplicationPipeBind>] [-AssignmentCollection <SPAssignmentCollection>]

Detailed Description

The **Get-SPPowerPointServiceApplication** cmdlet displays properties for an instance of the PowerPoint Service application.

Parameters

Parameter	Required	Type	Description
Identity	Optional	Microsoft.Office.Server.Powerpoint.SharePoint.PowerShell.SPPowerPointServiceApplicationPipeBind	Specifies the name of the PowerPoint Service application. The type must be a valid GUID, in the form 12345678-90ab-cdef-1234-567890bcdefgh.
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	Manages objects for the purpose of proper disposal. Use

Parameter	Required	Type	Description
			of objects, such as **SPWeb** or **SPSite**, can use large amounts of memory and use of these objects in Windows PowerShell scripts requires proper memory management. Using the **SPAssignment** object, you can assign objects to a variable and dispose of the objects after they are needed to free up memory. When **SPWeb**, **SPSite**, or **SPSiteAdministration** objects are used, the objects are automatically disposed of if an assignment collection or

Parameter	Required	Type	Description
			the **Global** parameter is not used. **Note:** When the **Global** parameter is used, all objects are contained in the global store. If objects are not immediately used, or disposed of by using the **Stop-SPAssignment** command, an

Parameter	Required	Type	Description
			out-of-memory scenario can occur.

AutoGenParams

Parameter	Required	Type	Description
Identity	Optional	Microsoft.Office.Server.Powerpoint.SharePoint.PowerShell.SPPowerPointServiceApplicationPipeBind	
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	

Input Types

Return Types

Example

Description

----------EXAMPLE 1----------

Code

```
Get-SPPowerPointServiceApplication | Select *
```

Comments

This example lists all parameters with their current values.

Example

Description

----------EXAMPLE 2----------

Code

```
Get-SPPowerPointServiceApplication | Select viewing*
```

Comments

This example lists only the `viewing` parameters.

See Also

New-SPPowerPointServiceApplication

Remove-SPPowerPointServiceApplication

Set-SPPowerPointServiceApplication

Get-SPPowerPointServiceApplicationProxy

Displays one or more objects for a PowerPoint Service application proxy.

Syntax

Get-SPPowerPointServiceApplicationProxy [[-Identity] <SPPowerPointServiceApplicationProxyPipeBind>] [-AssignmentCollection <SPAssignmentCollection>]

Detailed Description

The **Get-SPPowerPointServiceApplicationProxy** cmdlet displays one or more objects for a PowerPoint Service application proxy. If a value is specified, all objects for the PowerPoint Service application proxy are displayed.

Parameters

Parameter	Required	Type	Description
Identity	Optional	Microsoft.Office.Server.Powerpoint.SharePoint.PowerShell.SPPowerPointServiceApplicationProxyPipeBind	Specifies the proxy for the PowerPoint Service application.
Assignment Collection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	Manages objects for the purpose of proper disposal. Use of objects, such as **SPWeb** or **SPSite**, can use large amounts of memory and use of these objects in

Parameter	Required	Type	Description
			Windows PowerShell scripts requires proper memory management. Using the **SPAssignment** object, you can assign objects to a variable and dispose of the objects after they are needed to free up memory. When **SPWeb**, **SPSite**, or **SPSiteAdministration** objects are used, the objects are automatically disposed of if an assignment collection or the **Global** parameter is not used. **Note:** When the **Global**

Parameter	Required	Type	Description
			parameter is used, all objects are contained in the global store. If objects are not immediately used, or disposed of by using the **Stop-SPAssignment** command, an out-of-memory

Parameter	Required	Type	Description
			scen ario can occur .

AutoGenParams

Parameter	Required	Type	Des crip tion
Identity	Optional	Microsoft.Office.Server.Powerpoint.SharePoint.PowerShell.SPPowerPointServiceApplicationProxyPipeBind	
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	

Input Types

Return Types

Example

Description

----------EXAMPLE 1----------

Code

```
Get-SPPowerPointServiceApplicationProxy | Select *
```

Comments

This example lists all current proxies with their values.

Example

Description

----------EXAMPLE 2----------

Code

```
Get-SPPowerPointServiceApplicationProxy | Select Broadcast*
```

Comments

This example lists only the `Broadcast` parameter and values.

See Also

New-SPPowerPointServiceApplicationProxy

Remove-SPPowerPointServiceApplicationProxy

Set-SPPowerPointServiceApplicationProxy

New-SPOfficeWebAppsCache

Creates a new Microsoft Office Web Apps cache for a given Web application.

Syntax

New-SPOfficeWebAppsCache [-WebApplication] <SPWebApplicationPipeBind> -MaxSizeInBytes <Int64> -Owner <SPUser> [-AssignmentCollection <SPAssignmentCollection>] [-Confirm [<SwitchParameter>]] [-ExpirationPeriodInDays <Int32>] [-WarningSizeInBytes <Int64>] [-WhatIf [<SwitchParameter>]]

Detailed Description

The **New-SPOfficeWebAppsCache** cmdlet creates a new rendering cache for a particular Web application. There can be at most one cache per Web application. Every Web application must have a rendering cache associated with it to guarantee optimal performance of the Word Viewing Service and PowerPoint Service applications.

Parameters

Parameter	Required	Type	Description
WebApplication	Required	Microsoft.SharePoint.PowerShell.SPWebApplicationPipeBind	Specifies the Web application in which the Office Web Apps cache are created.
MaxSizeInBytes	Required	System.Int64	Specifies the size in bytes of the cache. After the limit is exceeded, the next expiration timer job will remove the oldest items in the cache until the cache is below quota. The default value is 107374182400.

Parameter	Required	Type	Description
Owner	Required	Microsoft.SharePoint.SPUser	Specifies the owner of the Office Web Apps cache.
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	Manages objects for the purpose of proper disposal. Use of objects, such as **SPWeb** or **SPSite**, can use large amounts of memory and use of these objects in Windows PowerShell scripts requires proper memory management. Using the **SPAssignment** object, you can assign objects to a variable and dispose of the objects after they are needed to free up memory. When **SPWeb**, **SPSite**, or **SPSiteAdministration** objects are used, the objects are automatically disposed of if an assignment collection or the **Global** parameter is not used. **Note:** When the **Global** parameter is used, all objects are contained in

Parameter	Required	Type	Description
			the global store. If objects are not immediately used, or disposed of by using the **Stop-SPAssignment** command, an out-of-memory scenario can occur.
Confirm	Optional	System.Management.Automation.SwitchParameter	Prompts you for confirmation before executing the command. For more information, type the following command: **get-help about_commonparameters**
ExpirationPeriodInDays	Optional	System.Int32	Specifies the amount of time since creation that cached renditions are to be stored. The default value is **30**.
WarningSizeInBytes	Optional	System.Int64	Specifies the threshold, in bytes, at which the owner starts to receive alerts that the companion cache is nearing its quota limit.

Parameter	Required	Type	Description
WhatIf	Optional	System.Management.Automation.SwitchParameter	Displays a message that describes the effect of the command instead of executing the command. For more information, type the following command: **get-help about_commonparameters**

AutoGenParams

Parameter	Required	Type	Description
WebApplication	Required	Microsoft.SharePoint.PowerShell.SPWebApplicationPipeBind	
MaxSizeInBytes	Required	System.Int64	
Owner	Required	Microsoft.SharePoint.SPUser	
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	
Confirm	Optional	System.Management.Automation.SwitchParameter	
ExpirationPeriodInDays	Optional	System.Int32	
WarningSizeInBytes	Optional	System.Int64	
WhatIf	Optional	System.Management.Automation.SwitchParameter	

Input Types

Return Types

Example

Description

----------EXAMPLE----------

Code

```
>$owner = New-SPUser -UserAlias domain\user -Web "http://sitename"
```

Code

```
New-SPOfficeWebAppsCache -WebApplication "SharePoint - 80" -Owner $owner -MaxSizeInBytes
214748364800
```

Comments

This example creates a new Office Web Apps cache on `SharePoint - 80` with a maximum capacity of 200 gigabytes (GB).

See Also

Get-SPOfficeWebAppsCache

Remove-SPOfficeWebAppsCache

Set-SPOfficeWebAppsCache

New-SPPowerPointServiceApplication

Creates a PowerPoint Service application.

Syntax

New-SPPowerPointServiceApplication [-Name] <String> -ApplicationPool <SPIisWebServiceApplicationPoolPipeBind> [-AssignmentCollection <SPAssignmentCollection>] [-CachePath <String>] [-Confirm [<SwitchParameter>]] [-WhatIf [<SwitchParameter>]]

Detailed Description

The **New-SPPowerPointServiceApplication** cmdlet creates a new instance of a PowerPoint Service application by using the **Name** parameter.

After the PowerPoint Service application is created, you can view, edit, and broadcast PowerPoint Service files using a Web browser.

Parameters

Parameter	Required	Type	Description
Name	Required	System.String	Specifies the name of the Web application.
ApplicationPool	Required	Microsoft.SharePoint.PowerShell.SPIisWebServiceApplicationPoolPipeBind	Assigns an application pool that will be used by IIS for this service application.
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	Manages objects for the purpose of proper disposal. Use of objects, such as **SPWeb** or **SPSite**, can use large amounts of

Parameter	Required	Type	Description
			memory and use of these objects in Windows PowerShell scripts requires proper memory management. Using the **SPAssignment** object, you can assign objects to a variable and dispose of the objects after they are needed to free up memory. When **SPWeb**, **SPSite**, or **SPSiteAdministration** objects are used, the objects are automatically disposed of if an assignment collection or the **Global** parameter is not used. **Note:** When the **Global** parameter is used, all objects are contained in the global store. If

Parameter	Required	Type	Description
			objects are not immediately used, or disposed of by using the **Stop-SPAssignment** command, an out-of-memory scenario can occur.
CachePath	Optional	System.String	Specifies the base folder path to a folder on the local machine where files will be cached. If not specified, the base folder path will be %TEMP%\powerpointcache
Confirm	Optional	System.Management.Automation.SwitchParameter	Prompts you for confirmation before executing the command. For more information, type the following command: **get-help about_common**

Parameter	Required	Type	Description
			parameters
WhatIf	Optional	System.Management.Automation.SwitchParameter	Displays a message that describes the effect of the command instead of executing the command. For more information, type the following command: **get-help about_common parameters**

AutoGenParams

Parameter	Required	Type	Description
Name	Required	System.String	
ApplicationPool	Required	Microsoft.SharePoint.PowerShell.SPIisWebServiceApplicationPoolPipeBind	
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	
CachePath	Optional	System.String	
Confirm	Optional	System.Management.Automation.SwitchParameter	
WhatIf	Optional	System.Management.Automation.SwitchParameter	

Input Types

Return Types

Example

Description

----------EXAMPLE----------

Code

```
New-SPPowerPointServiceApplication -name "MyWorkgroupPPTApp" -ApplicationPool "SharePoint Web Services Default" -cachepath d:\program files\custom cache"
```

Comments

This example creates a new instance of the PowerPoint Service application named `MyWorkgroupPPTApp`, assigns it to the default application pool, and establishes custom cache location under the Program Files folder.

See Also

Get-SPPowerPointServiceApplication

Remove-SPPowerPointServiceApplication

Set-SPPowerPointServiceApplication

New-SPPowerPointServiceApplicationProxy

Creates a PowerPoint Service application proxy.

Syntax

New-SPPowerPointServiceApplicationProxy [-Name] <String> -ServiceApplication <SPPowerPointServiceApplicationPipeBind> [-AddToDefaultGroup <SwitchParameter>] [-AssignmentCollection <SPAssignmentCollection>] [-Confirm [<SwitchParameter>]] [-WhatIf [<SwitchParameter>]]

Detailed Description

The **New-SPPowerPointServiceApplicationProxy** cmdlet creates a PowerPoint Service application proxy. The service application proxy is instantiated on the front-end Web server and acts as an intermediary between the client computer and the service application back end.

Parameters

Parameter	Required	Type	Description
Name	Required	System.String	Specifies the name of the Web application proxy.
ServiceApplication	Required	Microsoft.Office.Server.Powerpoint.SharePoint.PowerShell.SPPowerPointServiceApplicationPipeBind	Specifies the name of the back-end service to bind.
AddToDefaultGroup	Optional	System.Management.Automation.SwitchParameter	Adds the newly created proxy to the default PowerPoint Service application proxy group. If not specified,

Parameter	Required	Type	Description
			the proxy is not added to any group.
Assignment Collection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	Manages objects for the purpose of proper disposal. Use of objects, such as **SPWeb** or **SPSite**, can use large amounts of memory and use of these objects in Windows PowerShell scripts requires proper memory management. Using the **SPAssignment** object, you can assign objects to a variable and dispose of the objects after they are needed to free up memory. When **SPWeb**, **SPSite**, or **SPSiteAdministration** objects are used, the objects are

Parameter	Required	Type	Description
			automatically disposed of if an assignment collection or the **Global** parameter is not used. **Note:** When the **Global** parameter is used, all objects are contained in the global store. If objects are not immediately used, or disposed of by using the **Stop-SPAssignment** command, an out-of-

Parameter	Required	Type	Description
			memory scenario can occur.
Confirm	Optional	System.Management.Automation.SwitchParameter	Prompts you for confirmation before executing the command. For more information, type the following command: **get-help about_commonparameters**
WhatIf	Optional	System.Management.Automation.SwitchParameter	Displays a message that describes the effect of the command instead of executing the command. For more information, type the following command: **get-help about_commonparameters**

AutoGenParams

Parameter	Required	Type	Description
Name	Required	System.String	
ServiceApplication	Required	Microsoft.Office.Server.Powerpoint.SharePoint.PowerShell.SPPowerPointServiceApplicationPipeBind	
AddToDefaultGroup	Optional	System.Management.Automation.SwitchParameter	
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	
Confirm	Optional	System.Management.Automation.SwitchParameter	
WhatIf	Optional	System.Management.Automation.SwitchParameter	

Input Types

Return Types

Example

Description

----------EXAMPLE----------

Code

```
New-SPPowerPointServiceApplicationProxy -name "MyWorkgroupPPTAppProxy" -ServiceApplication
"MyWorkgroupPPTApp" -AddtoDefaultGroup
```

Comments

This example creates a new instance of the PowerPoint Service application proxy named `MyWorkgroupPPTAppProxy`, binds it to the `MyWorkgroupPPTApp` service application, and then adds it to the default PowerPoint Service application proxy group.

See Also

Get-SPPowerPointServiceApplicationProxy

Remove-SPPowerPointServiceApplicationProxy

Set-SPPowerPointServiceApplicationProxy

New-SPWordViewingServiceApplication

Creates a new instance of a Word Viewing Service application.

Syntax

```
New-SPWordViewingServiceApplication -ApplicationPool <SPIisWebServiceApplicationPoolPipeBind>
[-AssignmentCollection <SPAssignmentCollection>] [-Confirm [<SwitchParameter>]] [-Name <String>]
[-WhatIf [<SwitchParameter>]]
```

Detailed Description

The **New-SPWordViewingServiceApplication** cmdlet creates a new Word Viewing Service application. Once created, the Word Viewing Service application can be used in conjunction with the Word Viewing Service application proxy to enable Microsoft Word document viewing.

Parameters

Parameter	Required	Type	Description
Name	Required	System.String	Specifies the name of the new Word Viewing Service application. The default value is **Word Viewing ServiceApplication_ <GUID>.**.
ApplicationPool	Optional	Microsoft.SharePoint.PowerShell.SPIisWebServiceApplicationPoolPipeBind	Specifies the application pool to be used by Internet Information Services (IIS) for the service application.
Confirm	Optional	System.Management.Automation.SwitchParameter	Prompts you for confirmation before executing the command. For more

Parameter	Required	Type	Description
			information, type the following command: **get-help about_commonparameters**
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	Manages objects for the purpose of proper disposal. Use of objects, such as **SPWeb** or **SPSite**, can use large amounts of memory and use of these objects in Windows PowerShell scripts requires proper memory management. Using the **SPAssignment** object, you can assign objects to a variable and dispose of the objects after they are needed to free up memory. When **SPWeb**, **SPSite**, or **SPSiteAdministration** objects are used, the objects are automatically disposed of if an assignment collection or the **Global** parameter is not used. **Note:** When the **Global**

Parameter	Required	Type	Description
			parameter is used, all objects are contained in the global store. If objects are not immediately used, or disposed of by using the **Stop-SPAssignment** command, an out-of-memory scenario can occur.
WhatIf	Optional	System.Management.Automation.SwitchParameter	Displays a message that describes the effect of the command instead of executing the command. For more information, type the following command: **get-help about_commonparameters**

AutoGenParams

Parameter	Required	Type	Description
ApplicationPool	Required	Microsoft.SharePoint.PowerShell.SPIisWebServiceApplicationPoolPipeBind	
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	
Confirm	Optional	System.Management.Automation.SwitchParameter	
Name	Optional	System.String	
WhatIf	Optional	System.Management.Automation.SwitchParameter	

Input Types

Return Types

Example

Description

----------EXAMPLE 1----------

Code

```
New-SPIisWebServiceApplicationPool -Name "WordViewingAppPool" -Account domain\user | New-SPWordViewingServiceApplication -Name "My Service Application"
```

Comments

This example creates a Word Viewing Service application named `My Service Application`.

See Also

New-SPWordViewingServiceApplicationProxy

Set-SPWordViewingServiceApplication

Set-SPWordViewingServiceApplicationProxy

Get-SPServiceApplication (*http://technet.microsoft.com/library/71a467dc-3b95-4b65-af93-0d0d6ebb8326(Office.14).aspx*)

Get-SPServiceApplicationProxy (*http://technet.microsoft.com/library/77f3b00a-ae45-4b51-b9dc-8a2faa5dd358(Office.14).aspx*)

New-SPWordViewingServiceApplicationProxy

Creates a new service application proxy for use with the Word Viewing Service application.

Syntax

New-SPWordViewingServiceApplicationProxy -URI <Uri> [-AssignmentCollection <SPAssignmentCollection>] [-Confirm [<SwitchParameter>]] [-Name <String>] [-NotInDefaultProxyGroup <SwitchParameter>] [-WhatIf [<SwitchParameter>]]

New-SPWordViewingServiceApplicationProxy -ServiceApplication <SPServiceApplicationPipeBind> [-AssignmentCollection <SPAssignmentCollection>] [-Confirm [<SwitchParameter>]] [-Name <String>] [-NotInDefaultProxyGroup <SwitchParameter>] [-WhatIf [<SwitchParameter>]]

Detailed Description

The **New-SPWordViewingServiceApplicationProxy** cmdlet creates a new service application proxy for the Word Viewing Service application. Administrators use this object to associate Word Viewing Service applications with those Web applications for which the viewing functionality of the Word Web App should be available.

Parameters

	Required	Type	Description
Name	Required	System.String	Specifies a unique name of the service application proxy to create. The default value is **WordViewingServiceApplicationProxy_<GUID>.**.
ServiceApplication	Required	Microsoft.SharePoint.PowerShell.SPServiceApplicationPipeBind	Specifies the identifier for the service application to which this proxy connects.
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	Manages objects for the purpose of proper disposal. Use of objects, such as **SPWeb** or **SPSite**, can use large amounts of memory and use of these

	Required	Type	Description
			objects in Windows PowerShell scripts requires proper memory management. Using the **SPAssignment** object, you can assign objects to a variable and dispose of the objects after they are needed to free up memory. When **SPWeb**, **SPSite**, or **SPSiteAdministration** objects are used, the objects are automatically disposed of if an assignment collection or the **Global** parameter is not used. **Note:** When the **Global** parameter is used, all objects are contained in the global store. If objects are not immediately used, or disposed of by using the **Stop-SPAssignment** command, an out-of-memory scenario can occur.
Confirm	Optional	System.Management.Automation.SwitchParameter	Prompts you for confirmation before executing the command. For more information, type the following command: **get-help about_commonparameters**
NotInDefaultProxyGroup	Optional	System.Management.Automation.SwitchParameter	Prevents the addition of the newly provisioned proxy to the default proxy group. Valid values are **True** or **False**. The default value is **False**.
URI	Required	System.Uri	Specifies the URI of a remote service application to which to connect.

	Required	Type	Description
WhatIf	Optional	System.Management.Automation.SwitchParameter	Displays a message that describes the effect of the command instead of executing the command. For more information, type the following command: **get-help about_commonparameters**

AutoGenParams

Parameter	Required	Type	Description
ServiceApplication	Required	Microsoft.SharePoint.PowerShell.SPServiceApplicationPipeBind	
URI	Required	System.Uri	
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	
Confirm	Optional	System.Management.Automation.SwitchParameter	
Name	Optional	System.String	
NotInDefaultProxyGroup	Optional	System.Management.Automation.SwitchParameter	
WhatIf	Optional	System.Management.Automation.SwitchParameter	

Input Types

Return Types

Example

Description

----------EXAMPLE 1----------

Code

```
New-SPWordViewingServiceApplication -Name "My Service Application" -ApplicationPool "SharePoint Web Services Default" | New-SPWordViewingServiceApplicationProxy -Name "My Service Proxy"
```

Comments

This example creates new Word Viewing Service Application, `My Service Application`, and uses the newly created Word Viewing Service Application proxy, `My Service Application Proxy`.

See Also

New-SPWordViewingServiceApplication

Set-SPWordViewingServiceApplicationProxy

Get-SPServiceApplicationProxy (*http://technet.microsoft.com/library/77f3b00a-ae45-4b51-b9dc-8a2faa5dd358(Office.14).aspx*)

Remove-SPOfficeWebAppsCache

Deletes the Microsoft Office Web Apps cache.

Syntax

Remove-SPOfficeWebAppsCache -WebApplication <SPWebApplicationPipeBind> [-AssignmentCollection <SPAssignmentCollection>] [-Confirm [<SwitchParameter>]] [-WhatIf [<SwitchParameter>]]

Remove-SPOfficeWebAppsCache -Site <SPSitePipeBind> [-AssignmentCollection <SPAssignmentCollection>] [-Confirm [<SwitchParameter>]] [-WhatIf [<SwitchParameter>]]

Detailed Description

The **Remove-SPOfficeWebAppsCache** cmdlet removes the cache that is associated with a given Web application. The deletion of the Office Web Apps cache results in degraded functionality for both the Word Viewing Service application and the PowerPoint Service application. Therefore, if the cache is not re-created by using the **New-SPOfficeWebAppsCache** cmdlet, Office Web Apps attempts to re-create a default cache by using an Office Web Apps Cache Creation timer job.

Parameters

Parameter	Required	Type	Description
Site	Required	Microsoft.SharePoint.PowerShell.SPSitePipeBind	Specifies the identity of the Office Web Apps cache site collection.
WebApplication	Required	Microsoft.SharePoint.PowerShell.SPWebApplicationPipeBind	Specifies the Web application that contains the Office Web Apps cache.
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	Manages objects for the purpose of proper disposal. Use of objects, such as **SPWeb** or **SPSite**, can use large amounts

Parameter	Required	Type	Description
			of memory and use of these objects in Windows PowerShell scripts requires proper memory management. Using the **SPAssignment** object, you can assign objects to a variable and dispose of the objects after they are needed to free up memory. When **SPWeb**, **SPSite**, or **SPSiteAdministration** objects are used, the objects are automatically disposed of if an assignment collection or the **Global** parameter is not used. **Note:** When the **Global** parameter is used, all objects are contained in the global store. If objects are not immediately used, or disposed of by using the **Stop-SPAssignment** command, an out-of-

Parameter	Required	Type	Description
			memory scenario can occur.
Confirm	Optional	System.Management.Automation.SwitchParameter	Prompts you for confirmation before executing the command. For more information, type the following command: **get-help about_commonparameters**
WhatIf	Optional	System.Management.Automation.SwitchParameter	Displays a message that describes the effect of the command instead of executing the command. For more information, type the following command: **get-help about_commonparameters**

AutoGenParams

Parameter	Required	Type	Description
Site	Required	Microsoft.SharePoint.PowerShell.SPSitePipeBind	
WebApplication	Required	Microsoft.SharePoint.PowerShell.SPWebApplicationPipeBind	
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	
Confirm	Optional	System.Management.Automation.SwitchParameter	
WhatIf	Optional	System.Management.Automation.SwitchParameter	

Input Types

Return Types

Example

Description

----------EXAMPLE----------

Code

```
Get-SPOfficeWebAppsCache -WebApplication "SharePoint - 80" | Remove-SPOfficeWebAppsCache
```

Comments

This example removes the Office Web Apps cache that is associated with the SharePoint Web application at `SharePoint - 80`.

See Also

Get-SPOfficeWebAppsCache

New-SPOfficeWebAppsCache

Set-SPOfficeWebAppsCache

Remove-SPPowerPointServiceApplication

Removes an instance of the PowerPoint Service application.

Syntax

Remove-SPPowerPointServiceApplication [-Identity] <SPPowerPointServiceApplicationPipeBind> [-AssignmentCollection <SPAssignmentCollection>] [-Confirm [<SwitchParameter>]] [-WhatIf [<SwitchParameter>]]

Detailed Description

The **Remove-SPPowerPointServiceApplication** cmdlet removes a PowerPoint Service application instance. If no parameters are specified, all service instances will be removed.

Parameters

Parameter	Required	Type	Description
Identity	Required	Microsoft.Office.Server.Powerpoint.SharePoint.PowerShell.SPPowerPointServiceApplicationPipeBind	Specifies the unique name for the PowerPoint Service application.
Assignment Collection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	Manages objects for the purpose of proper disposal. Use of objects, such as **SPWeb** or **SPSite**, can use large amounts of memory and use of these objects in Windows PowerShell scripts requires proper memory management. Using the

Parameter	Required	Type	Description
			SPAssignment object, you can assign objects to a variable and dispose of the objects after they are needed to free up memory. When **SPWeb**, **SPSite**, or **SPSiteAdministration** objects are used, the objects are automatically disposed of if an assignment collection or the **Global** parameter is not used. **Note:** When the **Global** parameter is used, all objects are contained in the global store. If objects are not immediately used, or disposed of by using the **Stop-SPAssignment** command, an out-of-memory

Parameter	Required	Type	Description
			scenario can occur.
Confirm	Optional	System.Management.Automation.SwitchParameter	Prompts you for confirmation before executing the command. For more information, type the following command: **get-help about_commonparameters**
WhatIf	Optional	System.Management.Automation.SwitchParameter	Displays a message that describes the effect of the command instead of executing the command. For more information, type the following command: **get-help about_commonparameters**

AutoGenParams

Parameter	Required	Type	Description
Identity	Required	Microsoft.Office.Server.Powerpoint.SharePoint.PowerShell.SPPowerPointServiceApplicationPipeBind	
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	
Confirm	Optional	System.Management.Automation.SwitchParameter	
WhatIf	Optional	System.Management.Automation.SwitchParameter	

Input Types

Return Types

Example

Description

----------EXAMPLE 1----------

Code

```
Remove-SPPowerPointServiceApplication
```

Comments

This example removes the current PowerPoint Service application if there is only one instance. If there is more than one instance, use the **Identity** parameter to specify which instance to remove.

Example

Description

----------EXAMPLE 2----------

Code

```
Remove-SPPowerPointServiceApplication -Identity "MyWorkgroupPPTApp"
```

Comments

This example uses the **Identity** parameter to specify which PowerPoint Service application to remove.

See Also

Get-SPPowerPointServiceApplication

New-SPPowerPointServiceApplication

Set-SPPowerPointServiceApplication

Remove-SPPowerPointServiceApplicationProxy

Removes an instance of a PowerPoint Service application proxy.

Syntax

Remove-SPPowerPointServiceApplicationProxy [-Identity] <SPPowerPointServiceApplicationProxyPipeBind> [-AssignmentCollection <SPAssignmentCollection>] [-Confirm [<SwitchParameter>]] [-WhatIf [<SwitchParameter>]]

Detailed Description

The **Remove-SPPowerPointServiceApplicationProxy** cmdlet removes an instance of a PowerPoint Service application proxy. If no parameters are specified, all proxies are removed. Use the **Identity** parameter to remove a specific proxy.

Parameters

Parameter	Required	Type	Description
Identity	Required	Microsoft.Office.Server.Powerpoint.SharePoint.PowerShell.SPPowerPointServiceApplicationProxyPipeBind	Specifies the unique name of the service application proxy to remove.
Assignment Collection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	Manages objects for the purpose of proper disposal. Use of objects, such as **SPWeb** or **SPSite**, can use large amounts of memory and use of these objects in Windows PowerShell scripts requires

Parameter	Required	Type	Description
			proper memory management. Using the **SPAssignment** object, you can assign objects to a variable and dispose of the objects after they are needed to free up memory. When **SPWeb**, **SPSite**, or **SPSiteAdministration** objects are used, the objects are automatically disposed of if an assignment collection or the **Global** parameter is not used. **Note:** When the **Global** parameter is used, all objects are contained in the global store. If objects are not immediately used, or

Parameter	Required	Type	Description
			disposed of by using the **Stop-SPAssignment** command, an out-of-memory scenario can occur.
Confirm	Optional	System.Management.Automation.SwitchParameter	Prompts you for confirmation before executing the command. For more information, type the following command: **get-help about_common parameters**
WhatIf	Optional	System.Management.Automation.SwitchParameter	Displays a message that describes the effect of the command instead of executing the command. For more information, type the following command: **get-help about_common parameters**

AutoGenParams

Parameter	Required	Type	Description
Identity	Required	Microsoft.Office.Server.Powerpoint.SharePoint.PowerShell.SPPowerPointServiceApplicationProxyPipeBind	
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	
Confirm	Optional	System.Management.Automation.SwitchParameter	
WhatIf	Optional	System.Management.Automation.SwitchParameter	

Input Types

Return Types

Example

Description

----------EXAMPLE 1----------

Code

```
Remove-SPPowerPointServiceApplicationProxy
```

Comments

This example removes the current PowerPoint Service application proxy if there is only one instance. If there is more than one instance, use the **Identity** parameter to specify the instance to remove.

Example

Description

----------EXAMPLE 2----------

Code

```
Remove-SPPowerPointServiceApplicationProxy -Identity "MyWorkgroupPPTAppProxy"
```

Comments

This example uses the **Identity** parameter to specify the PowerPoint Service application proxy to remove.

See Also

Get-SPPowerPointServiceApplicationProxy

New-SPPowerPointServiceApplicationProxy

Set-SPPowerPointServiceApplicationProxy

Set-SPOfficeWebAppsCache

Modifies properties on the Microsoft Office Web Apps cache.

Syntax

```
Set-SPOfficeWebAppsCache [-WebApplication] <SPWebApplicationPipeBind> [-AssignmentCollection <SPAssignmentCollection>] [-Confirm [<SwitchParameter>]] [-ExpirationPeriodInDays <Int32>] [-MaxFrontEndCacheSizeInMB <Int32>] [-MaxFrontEndDocumentInfoCacheItems <UInt32>] [-MaxSizeInBytes <Int64>] [-Owner <SPUser>] [-WarningSizeInBytes <Int64>] [-WhatIf [<SwitchParameter>]]
```

```
Set-SPOfficeWebAppsCache -Site <SPSitePipeBind> [-AssignmentCollection <SPAssignmentCollection>] [-Confirm [<SwitchParameter>]] [-ExpirationPeriodInDays <Int32>] [-MaxFrontEndCacheSizeInMB <Int32>] [-MaxFrontEndDocumentInfoCacheItems <UInt32>] [-MaxSizeInBytes <Int64>] [-Owner <SPUser>] [-WarningSizeInBytes <Int64>] [-WhatIf [<SwitchParameter>]]
```

Detailed Description

The **Set-SPOfficeWebAppsCache** cmdlet changes properties of the site collection associated with the cache for the specified Web application. In addition to enabling the modification of all properties exposed on the **SPSitePipeBind** object, the **Set-SPOfficeWebAppsCache** cmdlet can also be used to modify the expiration period for stored renditions.

Parameters

Parameter	Required	Type	Description
WebApplication	Required	Microsoft.SharePoint.PowerShell.SPWebApplicationPipeBind	Specifies the Web application that contains the Office Web Apps cache.
Site	Required	Microsoft.SharePoint.PowerShell.SPSitePipeBind	Specifies the Office Web Apps cache site collection.
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	Manages objects for the purpose of proper disposal.

Parameter	Required	Type	Description
			Use of objects, such as **SPWeb** or **SPSite**, can use large amounts of memory and use of these objects in Windows PowerShell scripts requires proper memory management. Using the **SPAssignment** object, you can assign objects to a variable and dispose of the objects after they are needed to free up memory. When **SPWeb**, **SPSite**, or **SPSiteAdministration** objects are used, the objects are automatically disposed of if an assignment collection or the **Global** parameter is not used. **Note:** When the **Global** parameter is used, all objects are contained in the global store. If

Parameter	Required	Type	Description
			objects are not immediately used, or disposed of by using the **Stop-SPAssignment** command, an out-of-memory scenario can occur.
Confirm	Optional	System.Management.Automation.SwitchParameter	Prompts you for confirmation before executing the command. For more information, type the following command: **get-help about_commonparameters**
WarningSizeInBytes	Optional	System.Int64	Specifies the threshold, in bytes, at which the owner starts to receive alerts that the Office Web Apps cache is nearing its quota limitation. The default value is **0**.
ExpirationPeriodInDays	Optional	System.Nullable	Specifies the amount of time since creation that cached renditions will be stored. The default value is **30**.

Parameter	Required	Type	Description
MaxFrontEndCacheSizeInMB	Optional	System.Int32	Specifies, in megabytes, the maximum amount of memory the cache is allowed to consume. The default value is 75.
MaxFrontEndDocumentInfoCacheItems	Optional	System.UInt32	Specifies the maximum number of items that can be added to the cache. The default value is 1,000.
MaxSizeInBytes	Optional	System.Int64	Specifies the size in bytes of the Office Web Apps cache. After the limit is exceeded, the next expiration timer job removes the oldest items in the cache until the cache is below quota. The default value is **107374182400**.
Owner	Optional	Microsoft.SharePoint.SPUser	Specifies the owner of the Office Web Apps cache.
WarningSizeInBytes	Optional	System.Int64	When the Office Web Apps cache size reaches the size specified by this parameter, a notification is sent to the site collection owner that the cache is

Parameter	Required	Type	Description
			approaching its quota.
WhatIf	Optional	System.Management.Automation.SwitchParameter	Displays a message that describes the effect of the command instead of executing the command. For more information, type the following command: **get-help about_commonparameters**

AutoGenParams

Parameter	Required	Type	Description
WebApplication	Required	Microsoft.SharePoint.PowerShell.SPWebApplicationPipeBind	
Site	Required	Microsoft.SharePoint.PowerShell.SPSitePipeBind	
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	
Confirm	Optional	System.Management.Automation.SwitchParameter	
ExpirationPeriodInDays	Optional	System.Int32	
MaxFrontEndCacheSizeInMB	Optional	System.Int32	
MaxFrontEndDocumentInfoCacheItems	Optional	System.UInt32	
MaxSizeInBytes	Optional	System.Int64	

Parameter	Required	Type	Description
Owner	Optional	Microsoft.SharePoint.SPUser	
WarningSizeInBytes	Optional	System.Int64	
WhatIf	Optional	System.Management.Automation.SwitchParameter	

Input Types

Return Types

Example

Description

----------EXAMPLE----------

Code

```
Get-SPWebApplication | Set-SPOfficeWebAppsCache -ExpirationPeriod 30
```

Comments

This example enumerates all Web applications in the farm and sets the expiration on each Web application's Office Web Apps cache to `30` days.

See Also

Get-SPOfficeWebAppsCache

New-SPOfficeWebAppsCache

Remove-SPOfficeWebAppsCache

Set-SPOneNoteWebAppConfig

Modifies properties on the SPOneNoteWebAppConfig object that is activated on the specified **SPSite** object.

Syntax

Set-SPOneNoteWebAppConfig [-Site] <SPSitePipeBind> [-AssignmentCollection <SPAssignmentCollection>] [-ClipartEnabled <SwitchParameter>] [-Confirm [<SwitchParameter>]] [-NotebookVersioningEnabled <SwitchParameter>] [-Proxy <String>] [-SyncIntervalMaxInSeconds <Nullable>] [-SyncIntervalMinInSeconds <Nullable>] [-SyncIntervalMinOnHeavyLoadInSeconds <Nullable>] [-SyncIntervalStepInSeconds <Nullable>] [-WhatIf [<SwitchParameter>]]

Detailed Description

The **Set-SPOneNoteWebAppConfig** cmdlet changes properties associated with OneNote Web App that are activated on the specified **SPSite** object.

Parameters

Parameter	Required	Type	Description
Site	Required	Microsoft.SharePoint.PowerShell.SPSitePipeBind	Specifies the site on which Microsoft OneNote Web App is activated.
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	Manages objects for the purpose of proper disposal. Use of objects, such as **SPWeb** or **SPSite**, can use large amounts of memory and use of these objects in Windows PowerShell scripts requires proper

Parameter	Required	Type	Description
			memory management. Using the **SPAssignment** object, you can assign objects to a variable and dispose of the objects after they are needed to free up memory. When **SPWeb**, **SPSite**, or **SPSiteAdministration** objects are used, the objects are automatically disposed of if an assignment collection or the **Global** parameter is not used. **Note:** When the **Global** parameter is used, all objects are contained in the global store. If objects are not immediately used, or disposed of by using the **Stop-SPAssignment**

Parameter	Required	Type	Description
			command, an out-of-memory scenario can occur.
ClipartEnabled	Optional	System.Management.Automation.SwitchParameter	Enables support for inserting clipart in Microsoft Word Web App and OneNote Web App.
Confirm	Optional	System.Management.Automation.SwitchParameter	Prompts you for confirmation before executing the command. For more information, type the following command: **get-help about_commonparameters**
NotebookVersioningEnabled	Optional	System.Management.Automation.SwitchParameter	Specifies that the notebook use its internal versioning system to track versions. Valid values are **True** or **False**. The default value is **True**.
Proxy	Optional	System.String	Specifies the URL of the proxy server configured to allow HTTP requests to Office.com to succeed. Used so that the server is able to retrieve and insert clipart in the Word Web App and

Parameter	Required	Type	Description
			OneNote Web App. Changes to this setting require an **IISReset** command to take effect. To reset this value back to its default setting, use "".
SyncIntervalMaxInSeconds	Optional	System.Nullable	Specifies the minimum interval, in seconds, for replicating when the user is idle. The default value is **30**.
SyncIntervalMinInSeconds	Optional	System.Nullable	Specifies the minimum level, in seconds, for replication to occur when a user is idle. The default value is **5**.
SyncIntervalMinOnHeavyLoadInSeconds	Optional	System.Nullable	Specifies the sync interval level, in seconds, for the OneNote Web App when the server is experiencing heavy load. Valid values are 1 to 1000. The default value is 30.
SyncIntervalStepInSeconds	Optional	System.Nullable	Specifies the amount, in seconds, to increase the interval of

Parameter	Required	Type	Description
			replicating when the user is idle. The default value is **5**.
WhatIf	Optional	System.Management.Automation.SwitchParameter	Displays a message that describes the effect of the command instead of executing the command. For more information, type the following command: **get-help about_commonparameters**

AutoGenParams

Parameter	Required	Type	Description
Site	Required	Microsoft.SharePoint.PowerShell.SPSitePipeBind	
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	
ClipartEnabled	Optional	System.Management.Automation.SwitchParameter	
Confirm	Optional	System.Management.Automation.SwitchParameter	
NotebookVersioningEnabled	Optional	System.Management.Automation.SwitchParameter	
Proxy	Optional	System.String	
SyncIntervalMaxInSeconds	Optional	System.Nullable	
SyncIntervalMinInSeconds	Optional	System.Nullable	

Parameter	Required	Type	Description
SyncIntervalMinOnHeavyLoadInSeconds	Optional	System.Nullable	
SyncIntervalStepInSeconds	Optional	System.Nullable	
WhatIf	Optional	System.Management.Automation.SwitchParameter	

Input Types

Return Types

Example

Description

----------EXAMPLE 1----------

Code

```
Set-SPOneNoteWebAppConfig -Site "http://sitename" -SyncIntervalMinInSeconds 50
```

Comments

This example changes the sync interval for the OneNote Web App that is activated on the site located at `http://sitename`.

Example

Description

----------EXAMPLE 2----------

Code

```
Get-SPSite | Set-SPOneNoteWebAppConfig -ClipartEnabled
```

Comments

This example enables clipart on all instances of OneNote Web App in the farm.

See Also

Get-SPOneNoteWebAppConfig

Set-SPPowerPointServiceApplication

Configures global settings for a PowerPoint Service application.

Syntax

Set-SPPowerPointServiceApplication [-Identity] <SPPowerPointServiceApplicationPipeBind> [-AssignmentCollection <SPAssignmentCollection>] [-BroadcastStateIdleTimeoutInSeconds <UInt32>] [-BroadcastStateSessionTimeoutInSeconds <UInt32>] [-DisableBinaryScan <$true | $false>] [-EditingWorkerProcessCount <UInt32>] [-EditingWorkerTimeoutInSeconds <UInt32>] [-EnableSandboxedEditing <$true | $false>] [-EnableSandboxedViewing <$true | $false>] [-EnableViewing97To2003Formats <$true | $false>] [-EnableViewingOpenXmlFormats <$true | $false>] [-ViewingCacheExpirationPeriodInSeconds <UInt32>] [-ViewingMaximumConversionsPerWorker <UInt32>] [-ViewingWorkerKeepAliveTimeoutInSeconds <UInt32>] [-ViewingWorkerProcessCount <UInt32>] [-ViewingWorkerTimeoutInSeconds <UInt32>]

Detailed Description

The **Set-SPPowerPointServiceApplication** cmdlet sets global properties and settings for an instance of a PowerPoint Service application that is in a farm.

Parameters

Parameter	Required	Type	Description
Identity	Required	Microsoft.Office.Server.Powerpoint.SharePoint.PowerShell.SPPowerPointServiceApplicationPipeBind	Specifies the unique name of this PowerPoint Service application.
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	Manages objects for the purpose of proper disposal. Use of objects, such as **SPWeb** or **SPSite**, can use large amounts of memory and use of these objects in

Parameter	Required	Type	Description
			Windows PowerShell scripts requires proper memory management. Using the **SPAssignment** object, you can assign objects to a variable and dispose of the objects after they are needed to free up memory. When **SPWeb**, **SPSite**, or **SPSiteAdministration** objects are used, the objects are automatically disposed of if an assignment collection or the **Global** parameter is not used. **Note:** When the **Global** parameter is used, all objects are contained in the global store. If objects are not immediately used, or disposed of by using

Parameter	Required	Type	Description
			the **Stop-SPAssignment** command, an out-of-memory scenario can occur.
BroadcastStateIdleTimeoutInSeconds	Optional	System.UInt32	Specifies the maximum total time, in seconds, that a broadcast connection is allowed to be idle before being automatically disconnected. The default value is 5400 seconds (90 minutes).
BroadcastStateSessionTimeoutInSeconds	Optional	System.UInt32	Specifies the maximum total time, in seconds, that a broadcast session is allowed to be active before being automatically disconnected. Set this parameter to the same value as **BroadcastSessionTimeoutInSeconds**. The default value is 43200 seconds (12 hours).
DisableBinaryScan	Optional	System.Boolean	Specifies if binary files are scanned by using the GateKeeper -

Parameter	Required	Type	Description
			Validator feature before a Microsoft PowerPoint file is opened. The default value is **$True**.
EditingWorkerProcessCount	Optional	System.UInt32	Specifies the number of active instances of the named application. This value must be less than the Windows Communication Foundation (WCF) connection limit for this computer. The default value is **100**.
EditingWorkerTimeoutInSeconds	Optional	System.UInt32	Specifies the time-out in seconds that the editing service is given for a request sent to the editing service. The default value is 60 seconds.
EnableSandboxedEditing	Optional	System.Boolean	Specifies if the Microsoft PowerPoint Web App editing code should run in a sandboxed (low-permission mode) environment. The default value is **$True**.
EnableSandboxe	Optional	System.Boolean	Specifies if the

Parameter	Required	Type	Description
dViewing			Microsoft PowerPoint Web App viewing code should run in a sandboxed (low-permission mode) environment. The default value is **$True**.
EnableViewing97To2003Formats	Optional	System.Boolean	Specifies if the Microsoft PowerPoint Web App can open presentation (*.ppt) files written in Office PowerPoint 1997 to 2003 versions. If this parameter is disabled, the PowerPoint Web App is prevented from viewing that file type. The default value is **$True**.
EnableViewingOpenXmlFormats	Optional	System.Boolean	Specifies if the Microsoft PowerPoint Web App can open Office Open XML files (*.pptx) in PowerPoint. If this parameter is disabled, the PowerPoint Web App is prevented from viewing that file type. The

Parameter	Required	Type	Description
			default value is **$True**.
ViewingCacheExpirationPeriodInSeconds	Optional	System.UInt32	Specifies the maximum time, in seconds, that items are kept in the back-end server cache. The default value is 600 seconds (10 minutes).
ViewingMaximumConversionsPerWorker	Optional	System.UInt32	Specifies the number of conversion jobs that a given instance of the worker process is allowed to convert before recycling. The default value is **5**.
ViewingWorkerKeepAliveTimeoutInSeconds	Optional	System.UInt32	Specifies the time-out in seconds, for the keep alive signal interval between the start and end of a process on the viewing back-end. The default value is 120 seconds.
ViewingWorkerProcessCount	Optional	System.UInt32	Specifies the number of active instances of the named application. This value must be less than the Windows

Parameter	Required	Type	Description
			Communication Foundation (WCF) connection limit for this computer. The default value is **3**.
ViewingWorkerTimeoutInSeconds	Optional	System.UInt32	Specifies the time-out in seconds that the viewing service is given for any job submitted by the application manager. The default is 300 seconds (5 minutes).

AutoGenParams

Parameter	Required	Type	Description
Identity	Required	Microsoft.Office.Server.Powerpoint.SharePoint.PowerShell.SPPowerPointServiceApplicationPipeBind	
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	
BroadcastStateIdleTimeoutInSeconds	Optional	System.UInt32	
BroadcastStateSessionTimeoutInSeconds	Optional	System.UInt32	
DisableBinaryScan	Optional	System.Boolean	
EditingWorkerProcessCount	Optional	System.UInt32	
EditingWorkerTimeoutInSeconds	Optional	System.UInt32	

Parameter	Required	Type	Description
EnableSandboxedEditing	Optional	System.Boolean	
EnableSandboxedViewing	Optional	System.Boolean	
EnableViewing97To2003Formats	Optional	System.Boolean	
EnableViewingOpenXmlFormats	Optional	System.Boolean	
ViewingCacheExpirationPeriodInSeconds	Optional	System.UInt32	
ViewingMaximumConversionsPerWorker	Optional	System.UInt32	
ViewingWorkerKeepAliveTimeoutInSeconds	Optional	System.UInt32	
ViewingWorkerProcessCount	Optional	System.UInt32	
ViewingWorkerTimeoutInSeconds	Optional	System.UInt32	

Input Types

Return Types

Example

Description

----------EXAMPLE 1----------

Code

```
Set-SPPowerPointServiceApplication -Identity "MyWorkgroupPPTApp" -
ViewingCacheExpirationPeriodInSeconds 1200
```

Comments

This example establishes new operational defaults for the viewing cache expiration.

Example

Description

----------EXAMPLE 2----------

Code

```
Set-SPPowerPointServiceApplication -Identity "MyWorkgroupPPTApp" -DisableBinaryScan:$false
```

Comments

This example disables binary scanning of documents.

See Also

Get-SPPowerPointServiceApplication

New-SPPowerPointServiceApplication

Remove-SPPowerPointServiceApplication

Set-SPPowerPointServiceApplicationProxy

Configures global settings for a PowerPoint Service application proxy object.

Syntax

```
Set-SPPowerPointServiceApplicationProxy [-Identity]
<SPPowerPointServiceApplicationProxyPipeBind> [-AssignmentCollection <SPAssignmentCollection>]
[-BroadcastBrowserDisconnectedStateThresholdInSeconds <UInt32>] [-
BroadcastBrowserGetDataTimeoutInSeconds <UInt32>] [-
BroadcastBrowserHealthyStatePollIntervalInMilliSeconds <UInt32>] [-
BroadcastBrowserReconnectingStatePollIntervalInSeconds <UInt32>] [-
BroadcastBrowserReconnectingStateThresholdInSeconds <UInt32>] [-
BroadcastDataCacheCleanupPeriodInMilliSeconds <UInt32>] [-
BroadcastDataCacheExpirationPeriodInMilliSeconds <UInt32>] [-BroadcastDataCacheMaxSessions
<UInt32>] [-BroadcastPresenterIdleTimeoutInSeconds <UInt32>] [-
BroadcastSessionTimeoutInSeconds <UInt32>] [-SilverlightCookieExpirationPeriodInDays <Int32>]
```

Detailed Description

The **Set-SPPowerPointServiceApplicationProxy** cmdlet configures global settings of an instance of a PowerPoint Service application proxy object within a farm.

Parameters

Parameter	Required	Type	Description
Identity	Required	Microsoft.Office.Server.Powerpoint.SharePoint.PowerShell.SPPowerPointServiceApplicationProxyPipeBind	Specifies the unique name of this service application.
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	Manages objects for the purpose of proper disposal. Use of objects, such as **SPWeb** or **SPSite**, can use large amounts of memory and use of

Parameter	Required	Type	Description
			these objects in Windows PowerShell scripts requires proper memory management. Using the **SPAssignment** object, you can assign objects to a variable and dispose of the objects after they are needed to free up memory. When **SPWeb**, **SPSite**, or **SPSiteAdministration** objects are used, the objects are automatically disposed of if an assignment collection or the **Global** parameter is not used. **Note:** When the **Global** parameter is used, all objects are contained in the global store. If objects are not immediately used, or disposed of

Parameter	Required	Type	Description
			by using the **Stop-SPAssignment** command, an out-of-memory scenario can occur.
BroadcastBrowserDisconnectedStateThresholdInSeconds	Optional	System.UInt32	Specifies the total time, in milliseconds, for which a broadcast attendee will tolerate failing GetData poll requests before changing to disconnected state. The default value is 60000 milliseconds.
BroadcastBrowserGetDataTimeoutInSeconds	Optional	System.UInt32	Specifies the timeout value, in milliseconds, for broadcast attendee GetData requests. The default value is 20000.
BroadcastBrowserHealthyStatePollIntervalInMilliSeconds	Optional	System.UInt32	Specifies the time, in milliseconds, between scheduled GetData poll requests for broadcast attendees when in healthy state. The default value is 1000 milliseconds.

Parameter	Required	Type	Description
BroadcastBrowserReconnectingStatePollIntervalInSeconds	Optional	System.UInt32	Specifies the time, in seconds, between scheduled GetData poll requests for broadcast attendees when in a reconnecting state. The default value is **5**.
BroadcastBrowserReconnectingStateThresholdInSeconds	Optional	System.UInt32	Specifies the total time, in milliseconds, for which a broadcast attendee will tolerate failing GetData poll requests, before changing to the reconnecting state. The default value is 20000 milliseconds.
BroadcastDataCacheCleanupPeriodInMilliSeconds	Optional	System.UInt32	Specifies the periodic time interval, in milliseconds, when the front-end BroadcastData cache cleanup operation is run. The default value is 3,600,000 milliseconds (60 mins).
BroadcastDataCacheExpirationPeriodInMilliSeconds	Optional	System.UInt32	Specifies the time, in milliseconds, after which the

Parameter	Required	Type	Description
			cached BroadcastData entry expires. The default value is 300 milliseconds.
BroadcastDataCacheMaxSessions	Optional	System.UInt32	Specifies the maximum number of sessions that will be cached in the front-end BroadcastData cache. The default value is 20,000.
BroadcastPresenterIdleTimeoutInSeconds	Optional	System.UInt32	Specifies the presenter session idle time-out value in seconds. The default value is 0 seconds. A value of **0** indicates no time-out.
BroadcastSessionTimeoutInSeconds	Optional	System.UInt32	Specifies the maximum total time, in seconds, that a broadcast session is allowed to be active before it is automatically disconnected. Set this parameter to the same value as **BroadcastStateSessionTimeoutInSeconds**. A value of **0** indicates no timeout. The default is 43200 seconds (12 hours).

Parameter	Required	Type	Description
SilverlightCookieExpirationPeriodInDays	Optional	System.Int32	Specifies the number of days after the Microsoft Silverlight 3 cookie will expire. The default value is **7**.

AutoGenParams

Parameter	Required	Type	Description
Identity	Required	Microsoft.Office.Server.Powerpoint.SharePoint.PowerShell.SPPowerPointServiceApplicationProxyPipeBind	
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	
BroadcastBrowserDisconnectedStateThresholdInSeconds	Optional	System.UInt32	
BroadcastBrowserGetDataTimeoutInSeconds	Optional	System.UInt32	
BroadcastBrowserHealthyStatePollIntervalInMilliSeconds	Optional	System.UInt32	
BroadcastBrowserReconnectingStatePollIntervalInSeconds	Optional	System.UInt32	
BroadcastBrowserReconnectingStateThresholdInSeconds	Optional	System.UInt32	
BroadcastDataCacheCleanupPeriodInMilliSeconds	Optional	System.UInt32	
BroadcastDataCacheExpirationPeriodInMilliSeconds	Optional	System.UInt32	
BroadcastDataCacheMaxSessions	Optional	System.UInt32	

Parameter	Required	Type	Description
BroadcastPresenterIdleTimeoutInSeconds	Optional	System.UInt32	
BroadcastSessionTimeoutInSeconds	Optional	System.UInt32	
SilverlightCookieExpirationPeriodInDays	Optional	System.Int32	

Input Types

Return Types

Example

Description

----------EXAMPLE 1----------

Code

```
Set-SPPowerPointServiceApplicationProxy -Identity "MyWorkgroupPPTAppProxy" -
SilverlightCookieExpirationPeriodInDays 365 -
BroadcastBrowserDisconnectedStateThresholdInSeconds 15000
```

Comments

This example sets the `MyWorkgroupPPTAppProxy` instance for the Silverlight 3 cookie to expire in 365 days, and sets the Broadcast attendee time-to-disconnect threshold for user interface (UI) messages to 15 seconds.

Example

Description

----------EXAMPLE 2----------

Code

```
Set-SPPowerPointServiceApplication -Identity "MyWorkgroupPPTApp" -
BroadcastStateSessionTimeoutInSeconds 21600
```

Code

```
Set-SPPowerPointServiceApplicationProxy -Identity "MyWorkgroupPPTAppProxy"
BroadcastSessionTimeoutInSeconds 21600
```

Comments

This example sets the Broadcast time-out to 6 hours for both the service application and the proxy. These two parameters must always be set to the same value.

See Also

Get-SPPowerPointServiceApplicationProxy

Remove-SPPowerPointServiceApplicationProxy

New-SPPowerPointServiceApplicationProxy

Set-SPWordViewingServiceApplication

Sets a global property for a Word Viewing Service application.

Syntax

Set-SPWordViewingServiceApplication [-Identity] <SPServiceApplicationPipeBind> [-AssignmentCollection <SPAssignmentCollection>] [-Confirm [<SwitchParameter>]] [-DisableBinaryScan <SwitchParameter>] [-DisableEmbeddedFonts <SwitchParameter>] [-EnableBinaryFormatSupport <SwitchParameter>] [-EnableOpenXmlFormatSupport <SwitchParameter>] [-KeepAliveTimeoutInSeconds <Int32>] [-LocalCacheStoreDirectory <String>] [-LocalCacheTimeoutInSeconds <Int32>] [-MaxActiveProcessCount <Int32>] [-MaxRenderingLifetimeInSeconds <Int32>] [-RecycleActiveProcessCount <Int32>] [-WhatIf [<SwitchParameter>]]

Detailed Description

The **Set-SPWordViewingServiceApplication** cmdlet changes global run-time properties for a Word Viewing Service application.

Parameters

Parameter	Required	Type	Description
Identity	Required	Microsoft.SharePoint.PowerShell.SPServiceApplicationPipeBind	Specifies the unique identity of a service application.
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	Manages objects for the purpose of proper disposal. Use of objects, such as **SPWeb** or **SPSite**, can use large amounts of memory and use of these objects in Windows PowerShell scripts

Parameter	Required	Type	Description
			requires proper memory management. Using the **SPAssignment** object, you can assign objects to a variable and dispose of the objects after they are needed to free up memory. When **SPWeb**, **SPSite**, or **SPSiteAdministration** objects are used, the objects are automatically disposed of if an assignment collection or the **Global** parameter is not used. **Note:** When the **Global** parameter is used, all objects are contained in the global store. If objects are not immediately used, or disposed of by using the **Stop-SPAssignment** command,

Parameter	Required	Type	Description
			an out-of-memory scenario can occur.
Confirm	Optional	System.Management.Automation.SwitchParameter	Prompts you for confirmation before executing the command. For more information, type the following command: **get-help about_commonparameters**
DisableBinaryScan	Optional	System.Management.Automation.SwitchParameter	Disables binary document scanning for corruption or other malicious content. The valid values are **True** or **False**. The default value is **False**.
DisableEmbeddedFonts	Optional	System.Management.Automation.SwitchParameter	Disables support for rendering embedded fonts in Microsoft Word documents. The valid values are **True** or **False**. The default value is **False**.
EnableBinaryFormatSupport	Optional	System.Management.Automation.SwitchParameter	Enables support for binary file formats in the Word Viewing Service application. The value values are **True** or **False**. The default value is **True**.

Parameter	Required	Type	Description
EnableOpenXmlFormatSupport	Optional	System.Management.Automation.SwitchParameter	Enables support for XML-based file formats in the Word Viewing Service application. The value values are **True** or **False**. The default value is **True**.
KeepAliveTimeoutInSeconds	Optional	System.Int32	Specifies the amount of time, in seconds, that a worker process is allowed to be nonresponsive before it is terminated. Changes to this setting require an **IISReset** command to take effect. The type must be an integer in the range of 5 to 3600.
LocalCacheStoreDirectory	Optional	System.String	Specifies the location of the temporary cache for use by the Word Viewing Service application. Changes to this setting require an **IISReset** command to take effect. The default value is *%temp%\waccache.* .
LocalCacheTimeoutInS	Optional	System.Int32	Specifies the

Parameter	Required	Type	Description
econds			amount of time, in seconds, that an output file remains on the application server. Changes to this setting require an **IISReset** command to take effect. The default value is **300**.
MaxActiveProcessCount	Optional	System.Int32	Specifies the number of active processes available to render Word documents. The default value is **3**. Changes to this parameter require an **IISReset** command to take effect.
MaxRenderingLifetimeInSeconds	Optional	System.Int32	Specifies the total amount of time, in seconds, that a worker process is allowed to process a particular rendition. Changes to this setting require an **IISReset** to take effect.
RecycleActiveProcessCount	Optional	System.Int32	Specifies the number of files that a single process is allowed to render before the Word Viewing Service

Parameter	Required	Type	Description
			Application recycles the process. Changes to this setting require an **IISReset** command to take effect. The default value is **5**.
WhatIf	Optional	System.Management.Automation.Switch Parameter	Displays a message that describes the effect of the command instead of executing the command. For more information, type the following command: **get-help about_commonparameters**

AutoGenParams

Parameter	Required	Type	Description
Identity	Required	Microsoft.SharePoint.PowerShell.SPServiceApplicationPipeBind	
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	
Confirm	Optional	System.Management.Automation.SwitchParameter	
DisableBinaryScan	Optional	System.Management.Automation.SwitchParameter	
DisableEmbeddedFonts	Optional	System.Management.Automation.SwitchParameter	
EnableBinaryFormatSu	Optional	System.Management.Automation.SwitchParameter	

Parameter	Required	Type	Description
pport			
EnableOpenXmlFormatSupport	Optional	System.Management.Automation.SwitchParameter	
KeepAliveTimeoutInSeconds	Optional	System.Int32	
LocalCacheStoreDirectory	Optional	System.String	
LocalCacheTimeoutInSeconds	Optional	System.Int32	
MaxActiveProcessCount	Optional	System.Int32	
MaxRenderingLifetimeInSeconds	Optional	System.Int32	
RecycleActiveProcessCount	Optional	System.Int32	
WhatIf	Optional	System.Management.Automation.SwitchParameter	

Input Types

Return Types

Example

Description

----------EXAMPLE 1----------

Code

```
Get-SPIisWebServiceApplicationPool -Identity "SharePoint Web Services System" | New-SPWordViewingServiceApplication -Name "My Service Application" | Set-SPWordViewingServiceApplication -MaxActiveProcessCount 10 -LocalCacheStoreDirectory "C:\temp"
```

Comments

This example creates a new Word Viewing Service Application, setting the maximum number of processes available for render to `10`, and the local cache directory to `C:\temp`.

See Also

New-SPWordViewingServiceApplication

New-SPWordViewingServiceApplicationProxy

Set-SPWordViewingServiceApplicationProxy

Get-SPServiceApplication (*http://technet.microsoft.com/library/71a467dc-3b95-4b65-af93-0d0d6ebb8326(Office.14).aspx*)

Get-SPServiceApplicationProxy (*http://technet.microsoft.com/library/77f3b00a-ae45-4b51-b9dc-8a2faa5dd358(Office.14).aspx*)

Set-SPWordViewingServiceApplicationProxy

Sets a global value for a Word Viewing Service application proxy.

Syntax

Set-SPWordViewingServiceApplicationProxy [-Identity] <SPServiceApplicationProxyPipeBind> [-AssignmentCollection <SPAssignmentCollection>] [-Confirm [<SwitchParameter>]] [-SilverlightOnly <SwitchParameter>] [-WhatIf [<SwitchParameter>]]

Detailed Description

The **Set-SPWordViewingServiceApplicationProxy** cmdlet sets a global value for a Word Viewing Service application proxy.

Parameters

Parameter	Required	Type	Description
Identity	Required	Microsoft.SharePoint.PowerShell.SPServiceApplicationProxyPipeBind	Specifies the unique name that identifies the service application proxy.
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	Manages objects for the purpose of proper disposal. Use of objects, such as **SPWeb** or **SPSite**, can use large amounts of memory and use of these objects in Windows PowerShell scripts requires proper memory management. Using the **SPAssignment** object, you can

Parameter	Required	Type	Description
			assign objects to a variable and dispose of the objects after they are needed to free up memory. When **SPWeb**, **SPSite**, or **SPSiteAdministration** objects are used, the objects are automatically disposed of if an assignment collection or the **Global** parameter is not used. **Note:** When the **Global** parameter is used, all objects are contained in the global store. If objects are not immediately used, or disposed of by using the **Stop-SPAssignment** command, an out-of-memory scenario can occur.

Parameter	Required	Type	Description
Confirm	Optional	System.Management.Automation.SwitchParameter	Prompts you for confirmation before executing the command. For more information, type the following command: **get-help about_commonparameters**
SilverlightOnly	Optional	System.Management.Automation.SwitchParameter	Requires Microsoft Silverlight 3 to be installed to view Microsoft Word documents.
WhatIf	Optional	System.Management.Automation.SwitchParameter	Displays a message that describes the effect of the command instead of executing the command. For more information, type the following command: **get-help about_commonparameters**

AutoGenParams

Parameter	Required	Type	Description
Identity	Required	Microsoft.SharePoint.PowerShell.SPServiceApplicationProxyPipeBind	
AssignmentCollection	Optional	Microsoft.SharePoint.PowerShell.SPAssignmentCollection	

Parameter	Required	Type	Description
Confirm	Optional	System.Management.Automation.SwitchParameter	
SilverlightOnly	Optional	System.Management.Automation.SwitchParameter	
WhatIf	Optional	System.Management.Automation.SwitchParameter	

Input Types

Return Types

Example

Description

----------EXAMPLE----------

Code

```
Get-SPServiceApplication -Name "My Service Application" | New-
SPWordViewingServiceApplicationProxy -Name "My Service Proxy" | Set-
SPWordViewingServiceApplicationProxy -SilverlightOnly:$true
```

Comments

This example creates a new Word Viewing Service application proxy, associates the proxy with `My Service Application` and sets all users of that proxy to force the use of Silverlight 3 on all renditions.

See Also

Set-SPWordViewingServiceApplication

Get-SPServiceApplication (*http://technet.microsoft.com/library/71a467dc-3b95-4b65-af93-0d0d6ebb8326(Office.14).aspx*)

Get-SPServiceApplicationProxy (*http://technet.microsoft.com/library/77f3b00a-ae45-4b51-b9dc-8a2faa5dd358(Office.14).aspx*)

New-SPWordViewingServiceApplication

New-SPWordViewingServiceApplicationProxy

System Center Operations Manager knowledge articles (Office Web Apps)

The articles in this section are knowledge articles for the Product Full Name 2010 management pack for Microsoft System Center Operations Manager 2007. Typically, you would see these articles after clicking a link in an alert in the Operations Manager console. You can use these articles to help you troubleshoot and resolve problems in Product Short Name 2010.

In this section:

- Merge failures high (Microsoft OneNote Web App)
- Request processing slow (Microsoft OneNote Web App)
- PowerPoint document high download time (Microsoft PowerPoint Web App)
- PowerPoint high number of queued conversion requests (Microsoft PowerPoint Web App)
- PowerPoint high rate of "No Available Server" errors (Microsoft PowerPoint Web App)
- PowerPoint high rate of cache write operations (Microsoft PowerPoint Web App)
- PowerPoint rate of BroadcastGetData requests too high (Microsoft PowerPoint Web App)

Merge failures high (Microsoft OneNote Web App)

Alert Name: Merge failures high

Event ID: None

Summary: With Microsoft OneNote Web App, Microsoft SharePoint Server 2010 users can modify their notebooks by using a Web browser. The front end Web server portion of the feature is responsible for ensuring that all updates to the notebook that were made in the Web browser are saved to the server and reflected across all client computers. The front-end Web server portion of this feature is also responsible for merging and reconciling changes that were made to the same content by multiple users.

Symptoms: Should the front-end Web server portion of the feature fail to synchronize simultaneous merge requests, edits may take a long time to replicate across multiple users who are editing the same document.

Cause: This might be caused by OneNote Web App competing for scarce resources.

Resolution: Restart Internet Information Services

1. Verify that the user account that is performing this procedure is a member of the local Administrators group.
2. At the command prompt, type **iisreset –force** and press ENTER.

Resolution: Add another server to the farm

1. If this failure keeps occurring, you should add a server to the farm to help load-balance the workload.

Request processing slow (Microsoft OneNote Web App)

Alert Name: Request processing slow

Event ID: None

Summary: With Microsoft OneNote Web App, Microsoft SharePoint Server 2010 users can modify their notebooks by using a Web browser. The front end Web server portion of the feature is responsible for ensuring that all updates to the notebook that were made in the Web browser are saved to the server and reflected across all client computers.

Symptoms: If the front-end Web server starts taking a long time to process requests, you may observe the following symptoms:

- Edits that are made by multiple authors who edit the document at the same time take a long time to replicate across multiple browsers.
- OneNote Web App takes a long time to exit.

Cause: This might be caused by OneNote Web App competing for scarce resources.

Resolution: Restart Internet Information Services

1. Verify that the user account that is performing this procedure is a member of the local Administrators group.
2. At the command prompt, type **iisreset -force** and press ENTER.

Resolution: Add another server to the farm

1. If this failure keeps occurring, you should add a server to the farm to help load-balance the workload.

PowerPoint document high download time (Microsoft PowerPoint Web App)

Alert Name: PowerPoint document high download time

Event ID: No event ID

Summary: With Microsoft PowerPoint Web App, Microsoft SharePoint Server 2010 users can view and modify their presentations by using a Web browser. If the average time it takes to download the document is too high, it may mean something is wrong with the application server or the database server.

Symptoms: One or more of the following might be the cause:

- Users who use PowerPoint Web App to view documents experience very poor performance.
- In worst-case scenarios, users cannot use PowerPoint Web App to view documents because they do not receive responses from the application server.

Cause: One or more of the following might be the cause:

- The database server is malfunctioning or is overloaded.
- The document being downloaded is very large.

Resolution: Verify availability of database and application servers

1. Verify that the database servers are consistently available.
2. Verify that the farm application servers are consistently available.

Resolution: Add another node to the SQL Server cluster

1. If this failure keeps occurring, you should add a server to the SQL Server cluster to help load-balance the workload.

PowerPoint high number of queued conversion requests (Microsoft PowerPoint Web App)

Alert Name: PowerPoint high number of queued conversion requests

Event ID: No event ID

Summary: With Microsoft PowerPoint Web App, Microsoft SharePoint Server 2010 users view and modify their presentations by using a Web browser. When a high number of viewing conversion requests are queued, something might be wrong with the application server, or PowerPoint Web App might be unable to serve all requests from users.

Symptoms: One or more of the following symptoms might appear:

- Users who use PowerPoint Web App to view documents experience very poor performance.
- Users cannot use PowerPoint Web App to view documents because they do not receive responses from the application server.

Cause: One or more of the following might be the cause:

- The application server is malfunctioning.
- There are not enough PowerPoint Web App viewing worker processes.
- There are not enough application servers in the farm.

Resolution: Check the general availability of farm servers

1. Check the availability of the farm servers by performing a **PING** command to each server and checking the response times.

Resolution: Increase the maximum number of worker processes

1. Verify that you meet the following minimum requirements: See Add-SPShellAdmin.
2. On the **Start** menu, click **All Programs**.
3. Click **Microsoft SharePoint 2010 Products**.
4. Click **SharePoint 2010 Management Shell**.
5. At the Windows PowerShell command prompt, type the following command:

```
Set-SPPowerPointServiceApplication -Identity <ID> –ViewingWorkerProcessCount <N>
```

Where:

- *<ID>* is the name or ID of the PowerPoint Web App service application.
- *<N>* is the number of viewing worker threads.

Resolution: Reset Internet Information Services on all front-end Web servers

1. At the command prompt, type the following command, and then press ENTER:

   iisreset

Resolution: Add another server to the farm

1. If this failure keeps occurring even after you have followed the steps in all other resolutions, you should add a server to the farm to help load-balance the workload.

PowerPoint high rate of "No Available Server" errors (Microsoft PowerPoint Web App)

Alert Name: PowerPoint high rate of "No Available Server" errors

Event ID: No event ID

Summary: With Microsoft PowerPoint Web App, Microsoft SharePoint 2010 Products users can view and modify their presentations by using a Web browser. A high rate of "No Available Server" responses to editing requests may mean there are issues with the application server, or that PowerPoint Web App is unable to serve all user requests.

Symptoms: One or more of the following symptoms might appear:

- Users who use PowerPoint Web App to edit documents experience very poor performance.
- Users who use PowerPoint Web App are unable to edit documents because they do not receive responses from the application server.

Cause: One or more of the following might be the cause:

- The application server is malfunctioning.
- There are not enough PowerPoint Web App editing worker processes.
- There are not enough application servers.

Resolution: Increase the maximum number of worker processes

1. Verify that you meet the following minimum requirements: See Add-SPShellAdmin.
2. On the **Start** menu, click **All Programs**.
3. Click **Microsoft SharePoint 2010 Products**.
4. Click **SharePoint 2010 Management Shell**.
5. At the Windows PowerShell command prompt, type the following command:

   ```
   Set-SPPowerPointServiceApplication -Identity <ID> -EditingWorkerProcessCount <N>
   ```

   Where:

   - *<ID>* is the name or ID of the PowerPoint Web App service application.
   - *<N>* is the number of editor worker processes.

   For more information, see Set-SPPowerPointServiceApplication.

Resolution: Reset Internet Information Services on all front-end Web servers

1. At the command prompt, type the following command, and then press ENTER:

   iisreset

Resolution: Add another server to the farm

1. If this failure keeps occurring even after you have followed the steps in all other resolutions, you should add a server to the farm to help load-balance the workload.

PowerPoint high rate of cache write operations (Microsoft PowerPoint Web App)

Alert Name: PowerPoint high rate of cache write operations

Event ID: No event ID

Summary: With Microsoft PowerPoint Web App, Microsoft SharePoint Server 2010 users can view and modify their presentations by using a Web browser. A high number of writes to the cache on the front-end Web server may mean that something is wrong with the front-end Web server, or that PowerPoint Web App is unable to serve all requests from users.

Symptoms: One or more of the following symptoms might appear:

- Users who use PowerPoint Web App to view documents experience very poor performance.
- In some cases, users cannot view documents by using PowerPoint Web App because they do not receive responses from the application server.

Cause: One or more of the following might be the cause:

- The servers do have affinity set to the front-end Web servers.
- The front-end Web server cache is too small.

Resolution: Increase the size of the front-end Web server cache

1. Verify that you meet the following minimum requirements: See Add-SPShellAdmin.
2. One any farm server, on the **Start** menu, click **All Programs**.
3. Click **Microsoft SharePoint 2010 Products**.
4. Click **SharePoint 2010 Management Shell**.
5. At the Windows PowerShell command prompt, type the following command:

```
Set-SPOfficeWebAppsCache -WebApplication <ID> -MaxFrontEndCacheSizeInMB <S> -
MaxFrontEndDocumentInfoCacheItems <N>
```

Where:

- *<ID>* is name or ID of PowerPoint service application.
- *<S>* is the size, in megabytes (MB), that you want to set the cache to.
- *<N>* is the maximum number of items that you want to store in the front-end Web server cache. Each item will take up 1 MB (or less) of data.
  MaxFrontEndDocumentInfoCacheItems should always be equal to, or greater than, **MaxFrontEndCacheSizeInMB**.

6. At a command prompt on each front-end Web server, type the following command, and then press ENTER:**iisreset**

Resolution: Set affinity in the load balancer for the Web servers

1. For the steps to set affinity, see the load balancer software or hardware documentation.

PowerPoint rate of BroadcastGetData requests too high (Microsoft PowerPoint Web App)

Alert Name: PowerPoint rate of BroadcastGetData requests too high

Event ID: No event ID

Summary: A high rate of BroadcastGetData requests indicates that a large number of users are viewing PowerPoint Broadcasts. This means PowerPoint Broadcast viewers may be using more CPU time in front-end Web servers than is optimal. This can reduce overall server performance if total CPU usage is high.

Symptoms: Users of this Microsoft SharePoint Server 2010 farm may experience very poor performance. If this is the case you should take steps to correct the problems that might be causing poor performance. Otherwise, this monitor only serves as a warning and you do not need to take further action.

Cause: One or more of the following might be the cause:

- A large number of users are viewing PowerPoint Broadcasts.
- PowerPoint Broadcast users poll the server too often.

  If users of this SharePoint farm experience poor performance, compare CPU usage on the front-end Web servers to the rate of BroadcastGetData requests. If spikes in CPU usage occur only with high rates of BroadcastGetData requests, you should increase the PowerPoint Broadcast polling interval. This will decrease the amount of traffic generated by PowerPoint Broadcast viewers.

Resolution: Check the general availability of farm servers

1. Check the availability of the farm servers by performing a **PING** command to each server and checking the response times.

Resolution: Increase the PowerPoint Broadcast polling interval

1. Verify that you meet the following minimum requirements: See Add-SPShellAdmin.
2. On the **Start** menu, click **All Programs**.
3. Click **Microsoft SharePoint 2010 Products**.
4. Click **SharePoint 2010 Management Shell**.
5. At the Windows PowerShell command prompt, type the following command:

```
Set-SPPowerPointServiceApplication -Identity <ID> -
BroadcastBrowserHealthyStatePollIntervalInMilliSeconds <N>
```

Where:

- <*ID*> is name or ID of the Power Point service application.
- <*N*> is number of milliseconds. The default interval is 1 second, and the next recommended value is 2 seconds. Increasing this interval beyond 2 seconds may negatively affect the end-user experience while viewing PowerPoint Broadcasts. This change will only affect new viewers of PowerPoint Broadcasts.

For more information, see Set-SPPowerPointServiceApplication.

Resolution: Reset Internet Information Services on all front-end Web servers

1. At the command prompt, type the following command, and then press ENTER:

   iisreset

Resolution: Add another server to the farm

1. If this failure keeps occurring even after you have followed the steps in all other resolutions, you should add a server to the farm to help load-balance the workload.

Conformance statement A-level (Office Web Apps)

Conformance statement for Product Full Name in regards to Web Content Accessibility Guidelines 2.0 (*http://www.w3.org/TR/WCAG20/*).

Conformance statement for configuration and administration experience at A-level

Claim	Details
Date of the claim	May 12th, 2010
Guidelines title, version, and URI	Web Content Accessibility Guidelines 2.0 (*http://www.w3.org/TR/2008/REC-WCAG20-20081211/*)
Conformance level satisfied: (Level A, AA or AAA)	A
A concise description of the Web pages, such as a list of URIs for which the claim is made, including whether subdomains are included in the claim.	Default content included with Product Short Name and Administration and Configuration features of Product Short Name. For other product features, see Conformance statement AA-level (Office Web Apps).
A list of the Web content technologies relied upon.	HTML JavaScript CSS
Additional Web content technologies used, but not relied upon.	ARIA Silverlight RSS PNG GIF JPEG
A list of success criteria beyond the level of conformance claimed that have been met. This information should be provided in a form that users can use,	Conforms to 1.4.9 Images of Text (No Exception) - AAA. Conforms to 2.1.3 Keyboard (No Exception) - AAA. Conforms to 2.2.3 No Timing (No Exception) - AAA. Conforms to 2.3.2 Three Flashes - AAA.

Claim	Details
preferably machine-readable metadata.	Conforms to 2.4.8 Location - AAA. Conforms to 3.3.6 Error Prevention (All) - AAA.

This conformance statement pertains to Product Short Name.

Customization of the product voids this conformance statement from Microsoft. Customers may make independent conformance statements if they have conducted due diligence to meet all relevant requirements for their customization.

Please consult with Assistive Technology (AT) vendors for compatibility specifications of specific AT products. AT products that are compatible with all the rely-upon-technologies are expected to be functional with the product.

This document is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS DOCUMENT.

Revised 07/22/2010.

Microsoft regularly updates its websites and provides new information about the accessibility of products as that information becomes available.

Conformance statement AA-level (Office Web Apps)

Conformance statement for Product Full Name in regards to Web Content Accessibility Guidelines 2.0 (*http://www.w3.org/TR/WCAG20/*).

Conformance statement for core reader, author, and contributor experience at AA-level

Claim	Details
Date of the claim	May 12th, 2010
Guidelines title, version, and URI	Web Content Accessibility Guidelines 2.0 (*http://www.w3.org/TR/2008/REC-WCAG20-20081211/*)
Conformance level satisfied: (Level A, AA or AAA)	AA
A concise description of the Web pages, such as a list of URIs for which the claim is made, including whether subdomains are included in the claim.	All default content included withProduct Short Name and Web application features of Product Short Name except where limited by Conformance statement A-level (Office Web Apps).
A list of the Web content technologies relied upon.	HTML JavaScript CSS
Additional Web content technologies used, but not relied upon.	ARIA Silverlight RSS PNG GIF JPEG
A list of success criteria beyond the level of conformance claimed that have been met. This information should be provided in a form that users can use, preferably	Conforms to 1.4.9 Images of Text (No Exception) - AAA. Conforms to 2.1.3 Keyboard (No Exception) - AAA. Conforms to 2.2.3 No Timing (No Exception) - AAA. Conforms to 2.3.2 Three Flashes - AAA.

Claim	Details
machine-readable metadata.	Conforms to 2.4.8 Location - AAA. Conforms to 3.3.6 Error Prevention (All) - AAA.

This conformance statement pertains to Product Short Name.

Customization of the product voids this conformance statement from Microsoft. Customers may make independent conformance statements if they have conducted due diligence to meet all relevant requirements for their customization.

Please consult with Assistive Technology (AT) vendors for compatibility specifications of specific AT products. AT products that are compatible with all the rely-upon-technologies are expected to be functional with the product.

This document is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS DOCUMENT.

Revised 07/22/2010.

Microsoft regularly updates its websites and provides new information about the accessibility of products as that information becomes available.