



Metalogix StoragePoint PowerShell Reference

Copyright © 2011 Metalogix International. All Rights Reserved.

This software is protected by copyright law and international treaties. Unauthorized reproduction or distribution of this software, or any portion of it, may result in severe civil and/or criminal penalties and will be prosecuted to the maximum extent possible under the law.

Table of Contents

StoragePoint PowerShell Overview	1
Getting Started.....	1
Profile and Endpoint Management Cmdlets	4
Add-Endpoint	4
Add-Profile	5
Add-EndpointToProfile	7
Get-EndpointExists.....	8
Get-Endpoint.....	9
Get-AllEndpoints	9
Get-Profile	10
Get-AllProfiles	11
Get-AllEndpointsByProfile	11
FarmProfileSetup.ps1 (PowerShell Script File)	12
Timer Job Scheduling Cmdlets	13
Set-OrphanBLOBCleanupJob.....	13
Set-BLOBExternalizationJob	14
Set-BLOBRecallJob	14
Set-BLOBMigrateJob	15
Set-ArchivingAgingJob	15
Set-ArchivingMetadataJob.....	16
Set-ArchivingVersioningJob	17
BLOB Information and Migration Cmdlets.....	18
Get-BLOB.....	18
Get-AllBLOBs	18
Move-BLOB	19
Miscellaneous SharePoint Utility Cmdlets.....	20
Get-SiteCollectionId	20
Get-ContentDBId.....	20
Get-WebApplicationId	20
Find-SharePointItem	20

PowerShell Script Examples	22
Creating a Site Collection Profile with No Encryption or Compression	22
Creating a Content Database (EBS) Profile with Encryption and Compression	22
Creating a Content Database Profile Using RBS (SharePoint 2010 only)	23
Creating Multiple Profiles Using the Same Endpoint	23
Creating a Profile with an Asynchronous Endpoint	24
Creating a Profile with Asynchronous Endpoint Selection	24
Displaying All Endpoints Meeting Certain Criteria	24
Displaying All BLOB Files Associated with a SharePoint Document	24

StoragePoint PowerShell Overview

StoragePoint provides a number of PowerShell cmdlets to automate setup and configuration tasks in StoragePoint 3.1 and greater. While it is also possible to use the StoragePoint API directly in PowerShell, these cmdlets simplify common tasks and provide a basis for more complex API-oriented scripts.

The StoragePoint cmdlets can be grouped into the following categories:

- Profile and endpoint management
- Timer job scheduling
- BLOB information and migration
- Miscellaneous SharePoint utility commands

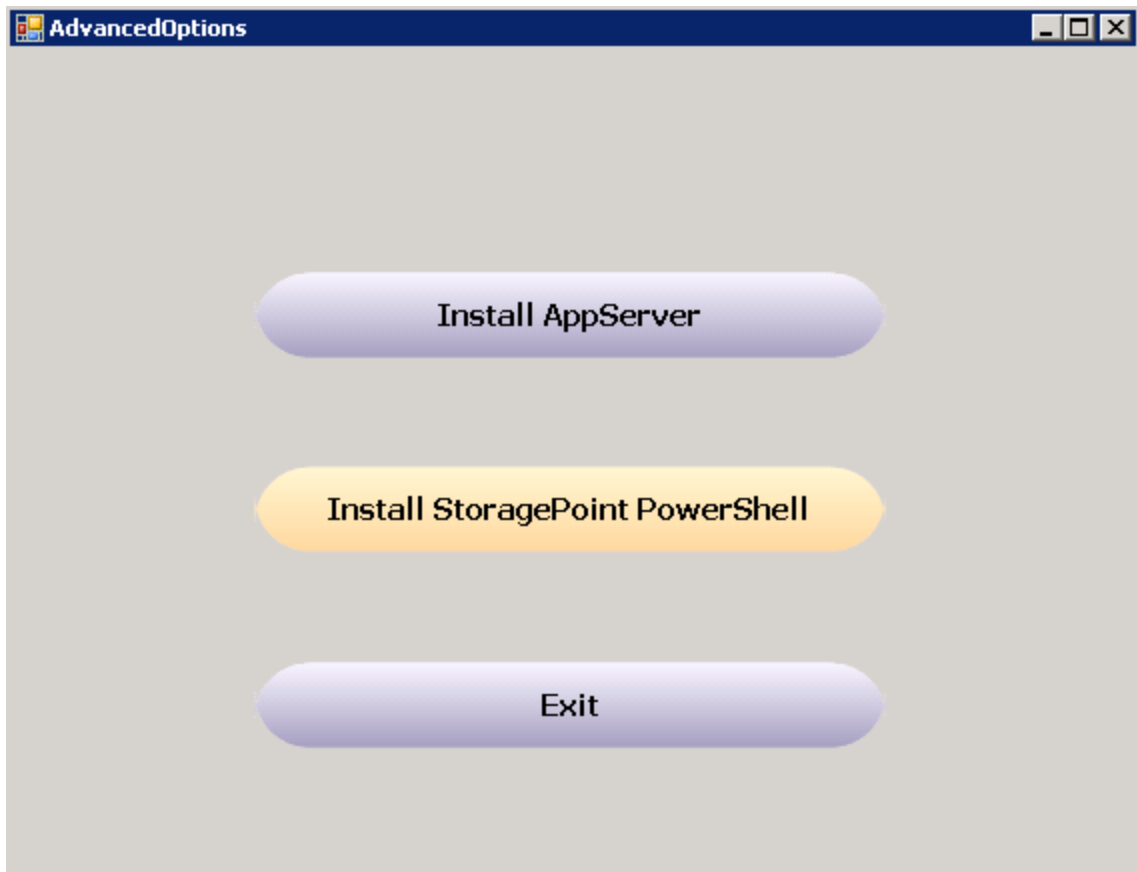
Getting Started

The StoragePoint PowerShell cmdlets require PowerShell version 2.0 or higher. Windows 2008 R2 includes PowerShell 2.0. For other operating systems, PowerShell version 2.0 is available in the Windows Management Framework download (see the Microsoft downloads site). For more information on the cmdlets, please see this [link](#) on Metalogix.com.

This procedure should be done after installing Metalogix StoragePoint on the farm server.

1. Once downloaded or copied to a WFE within your farm or single-server environment, simply double-click the executable to launch the Metalogix StoragePoint installer.
2. Click the **Advanced Options** button.

- Click the **Install StoragePoint PowerShell** button.



- A new command prompt window will open and show the progress of the bits being installed. When complete, the screen will display a message that the install has been registered successfully. Press any key to close the window.

```
C:\Windows\system32\cmd.exe
nt.PowershellCmdlets\1.0.0.0_141fe4b547d7494f\StoragePoint.PowershellCmdlets.dll
assembly's progress.
The file is located at C:\Windows\assembly\gac_msil\StoragePoint.PowershellCmdlets\1.0.0.0_141fe4b547d7494f\StoragePoint.PowershellCmdlets.InstallLog.
Committing assembly 'C:\Windows\assembly\gac_msil\StoragePoint.PowershellCmdlets\1.0.0.0_141fe4b547d7494f\StoragePoint.PowershellCmdlets.dll'.
Affected parameters are:
  assemblypath = C:\Windows\assembly\gac_msil\StoragePoint.PowershellCmdlets\1.0.0.0_141fe4b547d7494f\StoragePoint.PowershellCmdlets.dll
  logfile = C:\Windows\assembly\gac_msil\StoragePoint.PowershellCmdlets\1.0.0.0_141fe4b547d7494f\StoragePoint.PowershellCmdlets.InstallLog
  logtoconsole =

The Commit phase completed successfully.
The transacted install has completed.
C:\Users\ADMINI~1\AppData\Local\Temp\1\StoragePoint\appserver>GOTO BYE
C:\Users\ADMINI~1\AppData\Local\Temp\1\StoragePoint\appserver>echo Setup Powershell for "StoragePoint Build <1723>" on AppServer(x64) Complete
Setup Powershell for "StoragePoint Build <1723>" on AppServer(x64) Complete
C:\Users\ADMINI~1\AppData\Local\Temp\1\StoragePoint\appserver>PAUSE
Press any key to continue . . .
```

5. Before using the cmdlets in any script or PowerShell session, the following PowerShell command must be run to load the StoragePoint PowerShell snap-in:

Add-PSSnapIn StoragePoint.PowershellCmdlets

Profile and Endpoint Management Cmdlets

The StoragePoint PowerShell cmdlet library contains a number of cmdlets to simplify the creation and management of profiles and endpoints. In many cases these cmdlets return StoragePoint API objects (i.e. ProfileAPI, EndpointAPI, ProfileEndpointAPI). Please refer to the StoragePoint API Reference for more information on properties available in these objects.

Add-Endpoint

This command will create and save a new storage endpoint in StoragePoint.

Parameters

- EndpointName (epn): The name of the new Endpoint. **REQUIRED.**
- Connection (conn): The connection string for the Endpoint. **REQUIRED.**
- AdapterName (adn): The adapter name; e.g. FileSystem, Azure, etc. **REQUIRED.**
- Foldering: When specified (-Foldering), folders will be created within the endpoint BLOB store.
- FolderingLevel: A number associated with the needed level of foldering within the BLOB store:
 - 1 = Year
 - 2 = Year and Month
 - 3 = Year, Month, and Day
 - 4 = Year, Month, Day, and Hour
 - 5 (default) = Year, Month, Day, Hour, and Minute
 - 6 = Year, Month, Day, Hour, Minute, and Second
- UseCompression: When specified (-UseCompression), indicates the endpoint should compress content written to it.
- UseEncryption: When specified (-UseEncryption), indicates the endpoint should encrypt content.
- EncryptionPassphrase: Encryption passphrase used to generate an encryption key. **REQUIRED IF -UseEncryption is specified.**
- EncryptionProviderId: The Guid Id associated with the encryption provider that will be used to encrypt content. The default is AES 256-bit encryption which is appropriate for most scenarios.
- Status: Endpoint status (Online = 1, Offline = 0).
- EndpointType: Endpoint Type (0=Normal, 1=System Cache, 2=Default Template).
- ErrorCountThreshold: Number of consecutive write errors before endpoint is automatically taken offline.
- ErrorCountWarningThreshold: Number of consecutive write errors before a warning email is sent.
- FreespaceThreshold: Amount of freespace (in MB) below which endpoint is automatically taken offline.
- FreespaceWarningThreshold: Amount of freespace (in MB) below which a warning email is sent.

- FreespacePercentThreshold: Percentage of freespace below which endpoint is automatically taken offline.
- FreespacePercentWarningThreshold: Percentage of freespace below which a warning email is sent.
- UseDefaultContacts: Send freespace warning emails to default contact list (maintained in General Settings).
- OtherNotify: Other email addresses (separate multiple with ;) to send freespace warnings to.

Examples

The following example will create a new StoragePoint endpoint with only the required parameters applied. All other values will be assigned default values when the endpoint is created. This endpoint will be on the file system and use folder \\server\blobShare\EP2.

```
Add-Endpoint -epn MainEndpoint2 -conn "  
PATH=\\Server\blobShare\EP2;AUDITDELETE=False;BUFFERSIZE=8192;SHREDONDELETE=False;"  
-adn FileSystem
```

This example creates an endpoint with compression and encryption turned on.

```
Add-Endpoint -epn MainEndpoint2 -conn "  
PATH=\\Server\blobShare\EP2;AUDITDELETE=False;BUFFERSIZE=8192;SHREDONDELETE=False;"  
-adn FileSystem -UseCompression -UseEncryption -EncryptionPassphrase "encryptionpwd"
```

Add-Profile

This command will create a new StoragePoint profile. It creates one endpoint but others can be added afterwards by the Add-EndpointToProfile command (see below).

Parameters

- ProfileName (pn): The name of the new Profile. **REQUIRED.**
- ProfileType (ptype): The type of profile to create, e.g. SiteCollection, WebApplication, or ContentDB. **REQUIRED.**
- ScopeId (sid): The GUID Id value of the scope of the profile. For ProfileType of SiteCollection, this is the SiteId. For a ProfileType of ContentDb, this is the Id of the content database. For a ProfileType of WebApplication, this is the Id of the web application. **REQUIRED.**
- EndpointId (eid): The Id or Name of the first endpoint to create on the profile. **REQUIRED.**
- BlobRetentionDays: Number of days to retain BLOB files after they have been removed from SharePoint. Default is 30 days.
- UseRBS: If specified (-UseRBS) then RBS will be used if a Content Database profile is being created. By default EBS is used. (SharePoint 2010 only.)
- SelectEndpointAsync: If specified (-SelectEndpointAsync) then the profile will evaluate endpoints asynchronously. This allows the use of file type and scope filters.

- EndpointWriteMode: Write mode of the endpoint (Synchronous or Asynchronous). (Default is Synchronous.)
- EndpointStartFolder: Specifies a start sub-folder to use on the endpoint to write content from the profile.
- EndpointAsyncPromoteFilename: If specified (-EndpointAsyncPromoteFilename) then the SharePoint filename will be used as the basis for the filename of the externalized BLOB file. (Requires WriteMode to be set to Asynchronous.)
- EndpointAsyncPromoteFolder: If specified (-EndpointAsyncPromoteFolder) then the SharePoint folder structure will be used as the foldering on the BLOB store for a given externalized BLOB. (Requires WriteMode to be set to Asynchronous.)
- EndpointAsyncPromoteExtension: If specified (-EndpointAsyncPromoteExtension) then the extension of the SharePoint filename will be used as the extension on the externalized BLOB file. This is ignored if AsyncPromoteFilename is also true. (Requires WriteMode to be set to Asynchronous.)
- EndpointFileSizeOp: Specifies the operator to use if a file size filter is specified. 0 for <= or 1 for >=.
- EndpointFileSize: The file size in KB if a file size filter is specified. Default is -1 (no size filter).
- EndpointFileTypeOp: Specifies the operator to use if a file type filter is specified. 0 for Exclude or 1 for Include.
- EndpointFileTypes: A comma separated list of file extensions to filter on (DOCX, PPTX, TIF, etc.). If the SharePoint filename contains the extension, then it will be included. (Requires use of -SelectEndpointAsync option.)
- EndpointScopeOp: Specifies the operator to use if a Scope filter is used. 0 for Include and 1 for Exclude.
- EndpointScopes: Specifies a comma separated list of scope filter values. See the StoragePoint API Reference for more information on the syntax of scope filters. (Requires use of -SelectEndpointAsync option.)

Examples

This simple example creates a basic profile with an endpoint. The endpoint must have been created previously (see the Add-Endpoint command).

```
$scid = Get-SiteCollectionId -s http://sharepoint
```

```
Add-Profile -pn MainProfile2 -ptype SiteCollection -sid $scid.ToString() -eid SANEndpoint
```

This next example creates an RBS ContentDb profile with an asynchronous endpoint:

```
$scid = Get-ContentDbId -s http://moss
```

```
$newprofile = Add-Profile -pn MainProfile2 -ptype ContentDb -sid $cid.ToString() -UseRBS -eid  
NASEndpoint -EndpointWriteMode Asynchronous -EndpointAsyncPromoteFilename -  
EndpointAsyncPromoteFolder
```

This final example creates an EBS ContentDb profile and sets it for asynchronous endpoint selection so a file type filter can be applied to the endpoint. It also sets the BLOB retention period to 180 days and sets the SharePoint filename and folder promotion options.

```
$cid = Get-ContentDbId -s http://moss
```

```
$newprofile = Add-Profile -pn MainProfile2 -ptype ContentDb -sid $cid.ToString() -  
BlobRetentionDays 180 -eid NASEndpoint -SelectEndpointAsync -  
EndpointAsyncPromoteFilename -EndpointAsyncPromoteFolder -EndpointFileTypeOp 1 -  
EndpointFileTypes "DOC,DOCX,XLS,XLSX"
```

Add-EndpointToProfile

The command allows you to add an endpoint to an existing profile.

Parameters

- Endpoint (ep): The Id or name of the endpoint.
- Profile (p): The Id or name of the profile.
- WriteMode: Write mode of the endpoint (Synchronous or Asynchronous). (Default is Synchronous.)
- StartFolder: Specifies a start sub-folder to use on the endpoint to write content from the profile.
- AsyncPromoteFilename: If specified (-AsyncPromoteFilename) then the SharePoint filename will be used as the basis for the filename of the externalized BLOB file. (Requires WriteMode to be set to Asynchronous.)
- AsyncPromoteFolder: If specified (-AsyncPromoteFolder) then the SharePoint folder structure will be used as the foldering on the BLOB store for a given externalized BLOB. (Requires WriteMode to be set to Asynchronous.)
- AsyncPromoteExtension: If specified (-AsyncPromoteExtension) then the extension of the SharePoint filename will be used as the extension on the externalized BLOB file. This is ignored if AsyncPromoteFilename is also true. (Requires WriteMode to be set to Asynchronous.)
- FileSizeOp: Specifies the operator to use if a file size filter is specified. 0 for <= or 1 for >=.
- FileSize: The file size in KB if a file size filter is specified. Default is -1 (no size filter).
- FileTypeOp: Specifies the operator to use if a file type filter is specified. 0 for Exclude or 1 for Include.
- FileTypes: A comma separated list of file extensions to filter on (DOCX, PPTX, TIF, etc.). If the SharePoint filename contains the extension, then it will be included. (Only works on profiles created with the -SelectEndpointAsync option.)

- ScopeOp: Specifies the operator to use if a Scope filter is used. 0 for Include and 1 for Exclude.
- Scopes: Specifies a comma separated list of scope filter values. See the StoragePoint API Reference for more information on the syntax of scope filters. (Only works on profiles created with the -SelectEndpointAsync option.)

Examples

The following example shows how to add an endpoint to a profile. This example adds the endpoint using all of the default values (synchronous write mode, no filename/folder promotion, no endpoint filters).

```
Add-EndpointToProfile -ep MainEndpoint1 -p MainProfile1
```

The next example looks up the profile using the Get-Profile cmdlet and then adds the endpoint to it using the ProfileId property on the Profile object returned. This example shows the use of Profile objects in PowerShell scripts.

```
$profile = Get-Profile -ProfileType SiteCollection -SiteUrl http://sharepoint  
Add-EndpointToProfile -ep MainEndpoint1 -p $profile.ProfileId.ToString()
```

The final example shows adding an endpoint with several advanced features including asynchronous write mode, SharePoint filename/folder promotion and a file type filter (to include only Word and Excel documents). (Note that the file type filter in this example requires that the profile was created with the SelectEndpointAsync option.)

```
Add-EndpointToProfile -ep MainEndpoint1 -p MainProfile1 -WriteMode Asynchronous -  
AsyncPromoteFilename -AsyncPromoteFolder -FileTypeOp 1 -FileTypes "DOC,DOCX,XLS,XLSX"
```

Get-EndpointExists

Determines whether the specified endpoint exists.

Parameters

- EndpointId (ep): The endpoint name or Id

Example

```
Get-EndpointExists -ep MainEndpoint1
```

Result

True if endpoint exists. False if not.

Get-Endpoint

This command will retrieve an endpoint object. An EndpointAPI object is returned. See the StoragePoint API Reference for more information on properties of this object.

Parameters

- Endpoint (ep): The name or Id of the endpoint to retrieve

Examples

This example will retrieve the endpoint object and list all properties of the object for viewing.

```
Get-Endpoint -ep MainEndpoint1
```

This final example shows how to assign an Endpoint object to a variable within the PowerShell pipeline and then use this object later on in the script.

```
$endpoint = Get-Endpoint -ep MainEndpoint1
```

```
Add-EndpointToProfile -ep $endpoint.EndpointId.ToString() -p MainProfile1
```

Get-AllEndpoints

This command will retrieve all StoragePoint endpoints into a list and display all the properties of each endpoint found.

Example

```
Get-AllEndpoints
```

Result

```
EndpointId      : <GUID>
Name            : MainEndpoint1
AdapterName     : FileSystem
Connection      : :PATH=\\Server\blobShare;BUFFERSIZE=8192;SHREDONDELETE=False;
Foldering       : True
FolderingLevel  : 5
StoreSourceMetadata :
HandleMissingMetadata :
IsNew           : False
UseCompression  : False
UseEncryption   : False
EncryptionProviderId :
EncryptionPassphrase :
ProfileState    : 1
```

Get-Profile

This command will retrieve a profile object.

Parameters

If looking up a profile for a given scope (site coll, content db, web app):

- ProfileType (ptype): The profile type, e.g. SiteCollection, WebApplication, ContentDB to find. *Must also specify either SiteUrl or Scopeld*
- SiteUrl (s): The URL of the SharePoint site in which a profile is associated with. If ProfileType is SiteCollection, this will cause the command to find the profile for the exact site collection. If ProfileType is ContentDb or WebApplication, then the command will find the profile covering the content database or web application, respectively, of the site collection url specified.
- Scopeld (sid): The scope id of the siteCollection, WebApplication, or ContentDB to find the profile for.

If looking up a profile for an externalized document:

- DocUrl (doc): Url of an externalized document in SharePoint to retrieve the profile for.

If looking up a specific profile by name or profile id:

- ProfileId (id): The Id (GUID) of the StoragePoint profile.
- ProfileName (name): The Name of the StoragePoint profile.

Example

This example retrieves the profile object and list all properties of the object for view.

```
Get-Profile -name MainProfile1
```

The following example shows to assign a Profile object to a variable within the PowerShell pipeline for later use.

```
$profile = Get-Profile -name MainProfile1
```

The following example retrieves the profile object that a document was externalized under.

```
Get-Profile -DocUrl http://moss/docs/documents/123.tif
```

The following example looks up a site collection's profile.

```
Get-Profile -ProfileType SiteCollection -SiteUrl http://sharepoint
```

The final example shows how to retrieve a profile using a scope id value (site id, content db id or web app id). The scope id can be determined in a number of ways but this example uses the StoragePoint cmdlet helper command Get-ContentDbId.

```
$cid = Get-ContentDbId -s http://sharepoint  
Get-Profile -ProfileType ContentDb -ScopelId $cid.Id.ToString()
```

Get-AllProfiles

This command will retrieve all StoragePoint profiles into a list and display all properties for the profiles.

Example

```
Get-AllProfiles
```

Result

```
Type                : SiteCollection  
ProfileId           : <GUID>  
IsNew               : False  
IsActive            : True  
ScopelId            : <GUID>  
BlobRetentionDays  : 30  
CabinetOn           : False  
CabinetLevel        : 5  
ProfileState        : 1  
ProfileEndpoints    : {MainEndpoint1, MainEndpoint2}  
Name                : MainProfile1  
AdapterName         : FileSystem
```

Get-AllEndpointsByProfile

This command will show all endpoints that are associated with the specified profile.

Parameters

- Profile (p): The name or Id of the profile

Example

The following example will list all endpoints that are associated with the profile 'MainProfile1'

```
Get-AllEndpointsByProfile -p MainProfile1
```

Result

```
EndPointType        : Synchronous  
Name                : MainEndpoint1  
EndPointId          : <GUID>
```

IsActive : True

FarmProfileSetup.ps1 (PowerShell Script File)

This script file is included in the StoragePoint cmdlet library installation folder. This script file uses many of the cmdlets documented previously to create profiles for an entire farm. Profiles can be created for all site collections, all content databases or all web applications (except CA) in a farm.

This file is an example of building more complex scripts using the StoragePoint cmdlets. It can be freely modified for other scenarios (for example, creating asynchronous endpoints).

Parameters

- ProfileScope: The scope of the profiles to create. Possible values are SiteCollection, ContentDb, or WebApplication. **REQUIRED.**
- EndpointPath: The file share or path to store externalized blob files to. **REQUIRED.**
- EndpointName: The name of the common endpoint to create. **REQUIRED.**
- UseRBS: If specified (-UseRBS), RBS profiles will be created if on SharePoint 2010 and ProfileScope of ContentDb is selected.

Example

The following example will list all endpoints that are associated with the profile 'MainProfile1'

```
Installation_folder\FarmProfileSetup.ps1 -ProfileScope SiteCollection -EndpointPath  
\\NAS\FILESHARE -EndpointName "Common NAS Endpoint"
```

Timer Job Scheduling Cmdlets

The StoragePoint cmdlet library includes a number of commands to schedule the various StoragePoint timer jobs. The commands cover the Orphan BLOB Cleanup, Externalization, Recall, (Bulk) Migrate, Aging, Versioning and Metadata jobs.

Set-OrphanBLOBCleanupJob

This command schedules an Orphan BLOB clean-up job for a specified profile.

Parameters

- Profile (p): The name or Id of the profile in which to run the job under. **REQUIRED.**
- Type (st): The type of schedule to use when setting up this job. Valid values are OneTime, Daily, and Weekly.
- DayOfWeek (dow): The day of the week to run the job when the schedule type is set to Weekly.
- JobStartDate (sdate): The start date/time. **The time portion is used to set the start time for Daily and Weekly jobs.**
- JobEndDate (edate): The end date/time. Not used for OneTime jobs. **The time portion is used to set the end time for Daily and Weekly jobs.**
- NumberOfThreads (threads): Number of threads to run the job with.
- RunNow: If specified (-RunNow), then run the job immediately. Ignores any scheduling options provided.
- EmailDefault: If specified (-EmailDefault), send status email to the default notification group setup in General Settings.
- NotificationEmailOther (email): Other email addresses (not in default notification group) to send status emails to.
- EmailOnErrorOnly: If specified (-EmailOnErrorOnly), then send status email only if an error occurs when running the job.

Examples

The following example will run the Orphan BLOB Cleanup Job immediately.

```
Set-OrphanBLOBCleanupJob -p MainProfile1 -RunNow
```

The following example schedules the job on a recurring weekly schedule.

```
Set-OrphanBLOBCleanupJob -p MainProfile1 -st Weekly -dow Saturday -sdate "12/26/2009 3:00" -edate "12/26/2009 6:00" -EmailDefault -email someone@example.com -threads 2
```

Set-BLOBExternalizationJob

This command schedules externalization job for a specified profile. The job may be run immediately (-RunNow) or at a specified date/time (-JobStartDate).

Parameters

- Profile (p): The name or Id of the profile in which to run the job under. **REQUIRED.**
- JobStartDate (sdate): The start date/time to run the job unless -RunNow switch is specified.
- NumberOfThreads (threads): Number of threads to run the job with.
- RunNow: If specified (-RunNow), then run the job immediately. Ignores any scheduling options provided.
- EmailDefault: If specified (-EmailDefault), send status email to the default notification group setup in General Settings.
- NotificationEmailOther (email): Other email addresses (not in default notification group) to send status emails to.
- EmailOnErrorOnly: If specified (-EmailOnErrorOnly), then send status email only if an error occurs when running the job.

Example

This example schedules an externalization job to run immediately.

```
Set-BLOBExternalizationJob -p MainProfile1 -threads 20 -EmailDefault -RunNow
```

Set-BLOBRecallJob

This command allows a user to schedule a recall job for a particular profile scope.

Parameters

- Profile (p): The name or Id of the profile in which to run the job under. **REQUIRED.**
- JobStartDate (sdate): The start date/time to run the job unless -RunNow switch is specified.
- NumberOfThreads (threads): Number of threads to run the job with.
- RunNow: If specified (-RunNow), then run the job immediately. Ignores any scheduling options provided.
- EmailDefault: If specified (-EmailDefault), send status email to the default notification group setup in General Settings.
- NotificationEmailOther (email): Other email addresses (not in default notification group) to send status emails to.
- EmailOnErrorOnly: If specified (-EmailOnErrorOnly), then send status email only if an error occurs when running the job.

Example

This example schedules a recall job to run immediately.

```
Set-BLOBRecallJob -p MainProfile1 -threads 20 -EmailDefault -RunNow
```

Set-BLOBMigrateJob

This command allows a user to schedule a migrate job for a specified profile. When executed, the job will migrate BLOBs from the source endpoint to the target endpoint.

Parameters

- Profile (p): The name or Id of the profile in which to run the job under. **REQUIRED.**
- SourceEndpoint: The source endpoint's name/Id
- TargetEndpoint: The destination endpoint's name/Id
- JobStartDate (sdate): The start date/time to run the job unless `-RunNow` switch is specified.
- NumberOfThreads (threads): Number of threads to run the job with.
- RunNow: If specified (`-RunNow`), then run the job immediately. Ignores any scheduling options provided.
- EmailDefault: If specified (`-EmailDefault`), send status email to the default notification group setup in General Settings.
- NotificationEmailOther (email): Other email addresses (not in default notification group) to send status emails to.
- EmailOnErrorOnly: If specified (`-EmailOnErrorOnly`), then send status email only if an error occurs when running the job.

Example

This example schedules an externalization job to run immediately.

```
Set-BLOBMigrateJob -p MainProfile1-SourceEndpoint Endpoint1 -TargetEndpoint Endpoint2 -threads 20 -EmailDefault -RunNow
```

Set-ArchivingAgingJob

This command schedules an Aging job for a specified profile.

Parameters

- Profile (p): The name or Id of the profile in which to run the job under. **REQUIRED.**

- Type (st): The type of schedule to use when setting up this job. Valid values are OneTime, Daily, and Weekly.
- DayOfWeek (dow): The day of the week to run the job when the schedule type is set to Weekly.
- JobStartDate (sdate): The start date/time. **The time portion is used to set the start time for Daily and Weekly jobs.**
- JobEndDate (edate): The end date/time. Not used for OneTime jobs. **The time portion is used to set the end time for Daily and Weekly jobs.**
- NumberOfThreads (threads): Number of threads to run the job with.
- RunNow: If specified (-RunNow), then run the job immediately. Ignores any scheduling options provided.
- EmailDefault: If specified (-EmailDefault), send status email to the default notification group setup in General Settings.
- NotificationEmailOther (email): Other email addresses (not in default notification group) to send status emails to.
- EmailOnErrorOnly: If specified (-EmailOnErrorOnly), then send status email only if an error occurs when running the job.

Example

The following example schedules the job on a recurring weekly schedule.

```
Set-ArchivingAgingJob -p MainProfile1 -st Weekly -dow Saturday -sdate "12/26/2009 3:00" -  
edate "12/26/2009 6:00" -EmailDefault -email someone@example.com -threads 2
```

Set-ArchivingMetadataJob

This command schedules an archiving metadata processing job for a specified profile.

Parameters

- Profile (p): The name or Id of the profile in which to run the job under. **REQUIRED.**
- JobStartDate (sdate): The start date/time to run the job unless -RunNow switch is specified.
- NumberOfThreads (threads): Number of threads to run the job with.
- RunNow: If specified (-RunNow), then run the job immediately. Ignores any scheduling options provided.
- EmailDefault: If specified (-EmailDefault), send status email to the default notification group setup in General Settings.
- NotificationEmailOther (email): Other email addresses (not in default notification group) to send status emails to.
- EmailOnErrorOnly: If specified (-EmailOnErrorOnly), then send status email only if an error occurs when running the job.

Example

This example schedules an archiving metadata processing job to run immediately.

```
Set-ArchivingMetadataJob -p MainProfile1 -threads 20 -EmailDefault -RunNow
```

Set-ArchivingVersioningJob

This command schedules an archiving version processing job for a specified profile..

Parameters

- Profile (p): The name or Id of the profile in which to run the job under. **REQUIRED.**
- JobStartDate (sdate): The start date/time to run the job unless `-RunNow` switch is specified.
- NumberOfThreads (threads): Number of threads to run the job with.
- RunNow: If specified (`-RunNow`), then run the job immediately. Ignores any scheduling options provided.
- EmailDefault: If specified (`-EmailDefault`), send status email to the default notification group setup in General Settings.
- NotificationEmailOther (email): Other email addresses (not in default notification group) to send status emails to.
- EmailOnErrorOnly: If specified (`-EmailOnErrorOnly`), then send status email only if an error occurs when running the job.

Example

This example schedules an archiving metadata processing job to run immediately.

```
Set-ArchivingVersioningJob -p MainProfile1 -threads 20 -EmailDefault -RunNow
```

BLOB Information and Migration Cmdlets

The StoragePoint cmdlet library contains functions to query for BLOB file information and also to migrate individual BLOBs.

Get-BLOB

This command gets the BLOB Reference object for a SharePoint document that is stored via StoragePoint. The BLOB Reference object contains information about the BLOB file including its file system path and filename.

Parameters

- DoctUrl (doc): Full URL of the document you wish to retrieve

Example

This example gets the BLOB Reference for a specified document.

```
Get-BLOB -doc http://moss/docs/documents/123.tif
```

Result

DocId	: <GUID>
DocType	: Doc
Version	: 1.0
DocLevel	: Published
DocUrl	: http://moss/docs/documents/123.tif
ProfileId	: <GUID>
EndpointId	: <GUID>
Endpoint	: Endpoint object
BlobId	: < GUID>
Folder	: \\server/blobShare\2009\12\31\12\01
FilePath	: \\server/blobShare\2009\12\31\12\01\<GUID>
BlobSize	: 61098

Get-AllBLOBs

This command returns a list of BLOB Reference objects based on a SharePoint CAML query executed against a specific SharePoint document library.

Parameters

- SiteUrl (s): The URL of the SharePoint site in which a profile is associated with.
- List (l): The SharePoint document library to query.
- Query (q): The CAML query in order to find the needed BLOB reference(s).

Example

The following runs a CAML query, finding all documents that have a file name which contains "tif". The BLOB Reference objects for the results of the query will be returned in a list.

```
Get-AllBLOBs -s http://sharepoint/docs -l Shared Documents -q "<Where><Contains><FieldRef Name=FileLeafRef /><Value Type=File>tif</Value></Contains></Where>
```

Move-BLOB

This command migrates an externalized SharePoint document's BLOB to another endpoint.

Parameters

- DocUrl (doc): Absolute URL of the document BLOB to migrate.
- EndpointId (ep): The Id (GUID) or name of the target endpoint.
- IncludeVersions (iv): If specified (-IncludeVersions), command will migrate all versions of the document as well.

Example

The following example moves a BLOB to the endpoint MainEndpoint2.

```
Move-BLOB -doc "http://sharepoint/docs/documents/123.tif" -ep MainEndpoint2
```

Miscellaneous SharePoint Utility Cmdlets

The StoragePoint cmdlet library contains some miscellaneous functions to help lookup values in SharePoint that may be needed by the other commands in the library. It is important to note that there are other methods of finding these values, especially in SharePoint 2010 which has an extensive PowerShell cmdlet library of its own.

Get-SiteCollectionId

This command gets the Site Collection Id for the site collection associated with the site URL

Parameters

- SiteUrl (s): The URL of the SharePoint site.

Get-ContentDBId

This command gets the Content DB Id for the content database of the specified site collection.

Parameters

- SiteUrl (s): The URL of the SharePoint site.

Get-WebApplicationId

This command gets the Web Application Id for the web application of the specified site collection.

Parameters

- SiteUrl (s): The URL of the SharePoint site in which a profile is associated with.

Find-SharePointItem

This command will return a list of SharePoint list items based on the query submitted.

Parameters

- SiteUrl (s): The URL of the SharePoint site to query.
- List (l): The SharePoint list to query. If not specified, the query will be run on the site.
- Query (q): The CAML query to run on the site or list. Can be left out if List parameter is specified (to get all items in the list).

PowerShell Script Examples

The following are examples of PowerShell scripts that utilize various StoragePoint cmdlets. These scripts should serve as templates for many common scenarios and can be customized as needed. (NOTE: Be sure to run **Add-PSSnapIn StoragePoint.PowershellCmdlets** before running these scripts or else the cmdlets will not be recognized.)

Creating a Site Collection Profile with No Encryption or Compression

This example creates a basic profile on a site collection. It uses the included FileSystem adapter and does not enable compression and encryption support. It creates a new endpoint which is then added to the profile as a synchronous-write endpoint. Previous versions of StoragePoint (prior to 3.0) used synchronous write mode exclusively.

```
$siteid = Get-SiteCollectionId -s "http://sharepoint/site"

Add-Endpoint -epn "NAS Endpoint 1" -adn "FileSystem" -conn
"path=\\NAS\FILESTORE"

Add-Profile -pn "Site Collection 1 Profile" -ptype SiteCollection -sid
$siteid -eid "NAS Endpoint 1"
```

Creating a Content Database (EBS) Profile with Encryption and Compression

This example creates a basic profile with a content database scope (content database scope is new to version 3.0). Also note that the endpoint is set to enable compression and encryption.

```
$cid = Get-ContentDbId -s "http://sharepoint/site"

Add-Endpoint -epn "NAS Endpoint 1" -adn "FileSystem" -conn
"path=\\NAS\FILESTORE" -UseCompression -UseEncryption -
EncryptionPassphrase "anypassword"

Add-Profile -pn "Content Database Profile" -ptype ContentDb -sid $cid -
eid "NAS Endpoint 1"
```

Creating a Content Database Profile Using RBS (SharePoint 2010 only)

This example creates a basic content database profile using the RBS interface of SharePoint 2010 instead of EBS. RBS is supported only on SharePoint. Also note that RBS requires SQL Server 2008 Enterprise Edition and the profile creation will error out if trying to activate an RBS profile on any other edition of SQL Server. (See the previous example for an example of creating a content database profile using the EBS interface.)

```
$cid = Get-ContentDbId -s "http://sharepoint/site"

Add-Endpoint -epn "NAS Endpoint 1" -adn "FileSystem" -conn
"path=\\NAS\FILESTORE" -UseCompression -UseEncryption -
EncryptionPassphrase "anypassword"

Add-Profile -pn "Content Db Profile" -ptype ContentDb -sid $cid -UseRBS
-eid "NAS Endpoint 1"
```

Creating Multiple Profiles Using the Same Endpoint

This example iterates all of the site collections in a web application and creates a profile for each one. Each profile uses the same endpoint but specifies a different Start Folder. Using different Start Folder values will keep the content separated by profile even though the same endpoint is being used.

```
# Add the common endpoint
Add-Endpoint -epn "Common Endpoint 1" -adn "FileSystem" -conn
"path=\\NAS\COMMON"

# Get web app - there are many other ways to do this.
$site = New-Object Microsoft.SharePoint.SPSite("http://spsite")
$webapp = $site.WebApplication

# Iterate through all site collections in web app
foreach($sc in $webapp.Sites)
{
    $startfolder = $sc.RootWeb.Title
    $profilename = $startfolder + " Profile"
    Write-Output "Creating profile for: $startfolder"

    # Create profile
    Add-Profile -pn $profilename -ptype SiteCollection -sid
    $sc.ID.ToString() -eid "Common Endpoint 1" -EndpointStartFolder
    $startfolder

    # Cleanup
    $sc.Dispose()
}

# Cleanup
$site.Dispose()
```

Creating a Profile with an Asynchronous Endpoint

This example creates a profile with an asynchronous endpoint. The endpoint mapping is set to promote the SharePoint filename and folder to the blob store when writing externalized BLOB files.

```
$siteid = Get-SiteCollectionId -s "http://sharepoint/site"

Add-Endpoint -epn "NAS Endpoint 1" -adn "FileSystem" -conn
"path=\\NAS\FILESTORE"

Add-Profile -pn "Site Collection 1 Profile" -ptype SiteCollection -sid
$siteid -eid "NAS Endpoint 1" -EndpointWriteMode Asynchronous -
EndpointAsyncPromoteFilename -EndpointAsyncPromoteFolder
```

Creating a Profile with Asynchronous Endpoint Selection

This example creates a profile that will select endpoints asynchronously. This allows the profile to support enhanced filters including file extension/type and multiple additional scope options including web, list and content type. This sample sets up a filter to externalize only Microsoft Word and Excel documents.

```
$siteid = Get-SiteCollectionId -s "http://sharepoint/site"

Add-Endpoint -epn "NAS Endpoint 1" -adn "FileSystem" -conn
"path=\\NAS\FILESTORE"

Add-Profile -pn "Site Collection 1 Profile" -ptype SiteCollection -sid
$siteid -SelectEndpointAsync -eid "NAS Endpoint 1" -EndpointWriteMode
Asynchronous -EndpointFileTypeOp 1 -EndpointFileTypes
"DOCX, DOC, XLS, XLSX"
```

Displaying All Endpoints Meeting Certain Criteria

This example displays all endpoints in the system whose name contains the word Test. This illustrates the use of WHERE filters in PowerShell as well as use of StoragePoint API object properties (EndpointAPI.Name and .Connection in this case).

```
$Endpoints = Get-AllEndpoints | Where-Object {$_.Name -Like "*Test*"}

ForEach ($ep in $Endpoints) {$ep.Name + " : " + $ep.Connection}
```

Displaying All BLOB Files Associated with a SharePoint Document

This example displays all BLOB files associated with a given document in SharePoint. This includes versions and drafts/checked out versions, etc.

```
$blobs = Get-BLOB -doc "http://spsite/documents/123.tif"
```

```
ForEach ($b in $blobs) {$b.Endpoint.Name + " : " + $b.FilePath}
```