

ControlPoint

Running ControlPoint Operations Programatically

April 16, 2018

Copyright

© Metalogix International GmbH., 2008-2018
All rights reserved. No part or section of the contents of this material may be reproduced or transmitted in any form or by any means without the written permission of Metalogix International GmbH.

ControlPoint™ is a trademark of Metalogix International GmbH.

Windows SharePoint Services is either a registered trademark or a trademark of Microsoft Corporation in the United States and/or other countries. Other product and company names mentioned herein may be the trademarks of their respective owners.

Technical Support

For information about Metalogix Technical support visit http://metalogix.com/support.

Technical support specialists can be reached by phone at +1-202-609-9100. The level of technical support provided depends upon the support package that you have purchased. Contact us to discuss your support requirements.

Contents

Preface **4**

Executing ControlPoint Actions in PowerShell **5**

Loading the ControlPoint Cmdlets 7
Backup-UserPermissions 10
Set-UserDirectPermissions 12
Duplicate-UserPermissions 15
Delete-UserPermissions 18
Delete-Site 21
Run-Discovery 25

Executing Operations Using ControlPoint Saved Instructions **26**

Executing Instructions for ContrlPoint Actions 26
Executing Instructions for ContrlPoint Analyses 26

Incorporating ControlPoint Actions into .NET Applications as Web Services **28**

Example Code to Call BackupPermissions Web Service 29
Example Code to Call SetUserDirectPermissions Web Service 30
Example Code to Call DuplicateUserPermissions 31
Example Code to Call DeleteSite Web Service 32
Example Code to Call RunDiscovery 33
Creating a Custom Web Service for Use with a ControlPoint Content Creation Policy 33

Preface

This guide describes

- how to use both Windows PowerShell and Web services to carry out the following ControlPoint actions:
  - Backup Permissions
  - Set User Direct Permissions
  - Duplicate User Permissions
  - Delete User Permissions
  - Delete Site
  - Full Discovery

    NOTE: Any ControlPoint settings that have been configured for Full Discovery will also apply to a programmatically-run Full Discovery. Refer to the *ControlPoint Administration Guide* for details.
- how to use PowerShell to execute ControlPoint actions and analyses using Saved Instructions
- how to integrate a Web service containing custom business logic with ControlPoint to create a policy for controlling content creation.

Executing ControlPoint Actions in PowerShell

You can run ControlPoint actions either directly by entering commands and parameters from the PowerShell command line or by using the command line to execute an xml file containing the commands and parameters. Most actions can be run for any level of the SharePoint Hierarchy—farm, Web application (WAP), site collection (SITE), or site (WEB). You can also choose to have a ControlPoint Task Audit generated as a .pdf file and saved to the file system folder on the server of your choice. (Regardless of whether you choose to have a Task Audit saved as a .pdf, actions will be recorded in the Task Audit that is accessible from the ControlPoint application interface.)

NOTE: Parameters, valid values, and examples for each action are covered in this guide.

Account Login Requirements

You can run ControlPoint actions in PowerShell from a remote desktop or console session on any Web front-end server on which ControlPoint is installed. The account you use to log into the server machine must have the appropriate permissions within ControlPoint for the scope of the action being performed.

Refer to the topic “ControlPoint Security” in the ControlPoint User’s Guide for complete detail.

For SharePoint 2010 and 2013, it may be necessary to perform additional steps to grant the appropriate permissions. Refer to the following Microsoft TechNet articles for more details:

- For SharePoint 2010:
  - http://technet.microsoft.com/en-us/library/ee806878(v=office.14).aspx
  - http://technet.microsoft.com/en-us/library/ff607596(v=office.14).aspx
- For SharePoint 2013:
  - http://technet.microsoft.com/en-us/library/ee806878.aspx
  - http://technet.microsoft.com/en-us/library/ff607596.aspx

The PowerShell Command Line

The advantage of using a command line is that you can "pipe" cmdlets together to perform multiple actions. However, each cmdlet must operate on the same scope (url and object type). This information is passed from one cmdlet to the next.

The example below shows a PowerShell command that will back up permissions on the Skunkworks site, then deletes the permissions of Mark Twain, Margaret Meade, and Isaac Asimov from that site.

Because both actions are operating on the same object, the url and type need to be specified only once, in the first cmdlet. After the first action is carried out, these parameters are passed to the next cmdlet.

```
Backup-UserPermissions -Type WEB -url
http://2010sharepoint/sites/alpha/skunkworks | Delete-UserPermissions -
UserLoginNames
2010sharepoint\marktwain,sbellum\margaretmeade,sbellum\isaacasimov
```

xml Files

With an xml file, you can act on multiple objects. However, only one ControlPoint action can be executed per file.

NOTE: xml templates for ControlPoint actions are located in the folder **C:\Program Files\Metalogix\ControlPoint\Support\PowerShell.** When you upgrade from a previous version of ControlPoint, these templates will be overwritten with newer versions. You should always make sure that your xml files use the most up-to-date templates.

The example below shows an xml file formatted to execute the Delete User Permissions action on two sites within the Clients Web application.

```
<?xml version="1.0" encoding="utf-8"?>
<DeleteUserPermissionsContract xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <SharePointObjects>
        <SharePointObject>
            <SPHierarchyLevel>WEB</SPHierarchyLevel>
            <!--Valid values are FARM, WAP, SITE, WEB-->
            <Url>http://2008sharepoint/sites/alpha</Url>
            <!--For FARM, WAP, or SITE, enter any url within the scope of the selection-->
            <IncludeChildren>true</IncludeChildren>
            <!--If SPHierarchyLevel=FARM, WAP, or SITE, true is the only valid value. If SPHierarchyLevel=WEB and
            you do not want the action to also include child sites, change the value to false.-->
        </SharePointObject>
        <SharePointObject>
            <SPHierarchyLevel>WEB</SPHierarchyLevel>
            <!--Valid values are FARM, WAP, SITE, WEB-->
            <Url>http://2008sharepoint/sites/beta</Url>
            <!--For FARM, WAP, or SITE, enter any url within the scope of the selection-->
            <IncludeChildren>true</IncludeChildren>
            <!--If SPHierarchyLevel=FARM, WAP, or SITE, true is the only valid value. If SPHierarchyLevel=WEB and
            you do not want the action to also include child sites, change the value to false.-->
        </SharePointObject>
    </SharePointObjects>
    <UserLoginNames>
        <string>2008sharepoint\marktwain</string>
        <string>2008sharepoint\margaretmeade</string>
        <string>2008sharepoint\isaacasimov</string>
    </UserLoginNames>
    <!--Login name(s)of user(s)whose permissions you want to set.  Enter each name as a separate string.-->
    <ReassignTo>
        <string>2008sharepoint\jamesjoyce</string>
    </ReassignTo>
    <!--(Optional) Login name(s) of user(s) to whom you want to reassign the deleted permissions.  Enter each
    name as a separate string-->
    <DeleteFromAllPeople>false</DeleteFromAllPeople>
    <!--If you want to delete the user(s) from the site collection All People list, change this value to true.-->
    <TaskAuditPath></TaskAuditPath>
    <!--(Optional) Path to the directory where you want to save a .pdf of a ControlPoint Task Audit for the
    action.-->
</DeleteUserPermissionsContract>
```

To execute the xml from the PowerShell command line, enter the Cmdlet, the parameter -FilePath, and the path to the xml file. For this example:

```
Delete-UserPermissions -FilePath C:\Temp\xml\DeleteUserPermissions.xml
```

Loading the ControlPoint Cmdlets

The ControlPoint snap-in contains the cmdlets needed to execute a ControlPoint action.

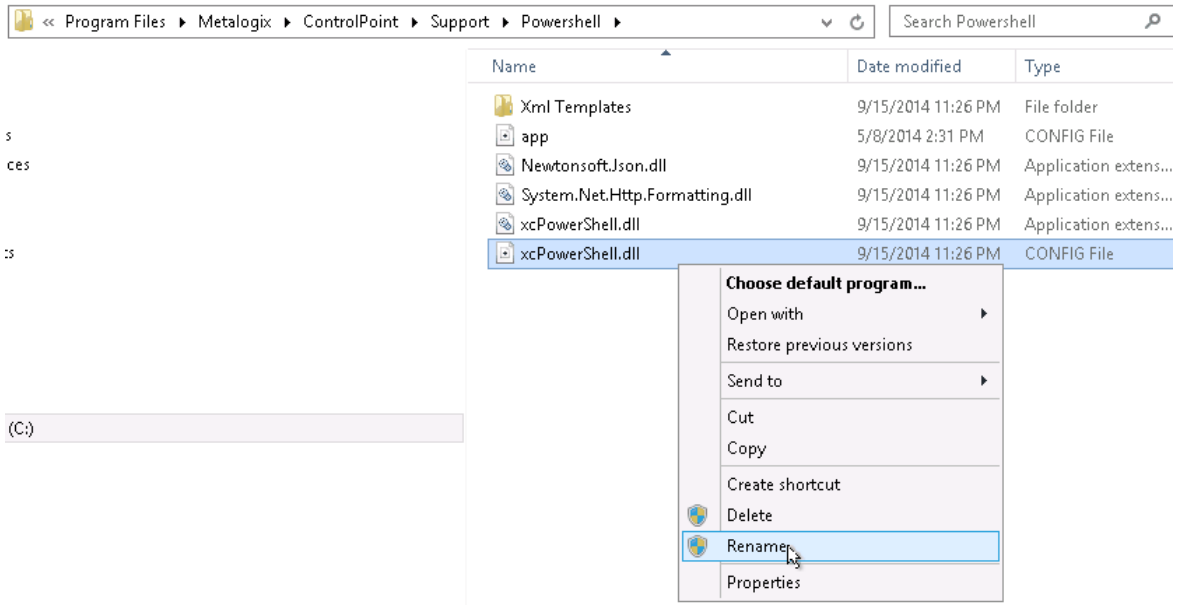
You can either:

- load the snap-in every time you open PowerShell, or
- add the snap-in to your PowerShell profile so it will automatically load the cmdlets whenever PowerShell opens.

  Refer to the msdn article on Windows PowerShell Profiles for details on creating a PowerShell profile.

Before You Begin: Rename xcPowerShell.dll.config to App.config

Before using Powershell to perform ControlPoint operations, navigate to the folder `C:\Program Files\Metalogix\ControlPoint\Support\Powershell`. Rename the file xcPowerShell.dll.config, to **App.config**.



In addition, if you are using a port number other than 1818 or a host header to access ControlPoint, open App.config for editing and locate the string `<endpoint address="http://bcalhost:1818/_layouts/Axceler/ActionsService.svc"`. Update the URL to reflect the port number or host header you are using to access ControlPoint.



Loading the ControlPoint snap-in

To load the ControlPoint snap-in:

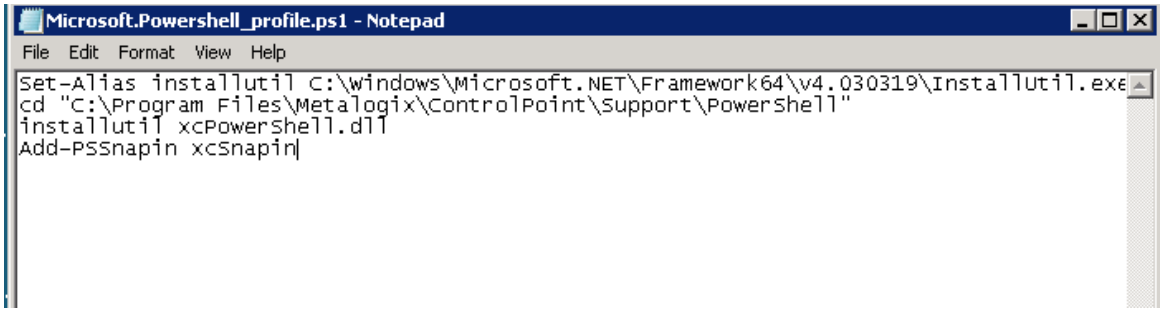
Line	Command	Description

1	**+-SharePoint 2010 32-bit environment:**
	`Set-Alias installutil C:\Windows\Microsoft.NET\Framework\v2.0.50727\InstallUtil.exe`
	SharePoint 2010 64-bit environment:
	`Set-Alias installutil C:\Windows\Microsoft.NET\Framework64\v2.0.50727\InstallUtil.exe`
	SharePoint 2013 environment
	`C:\Windows\Microsoft.NET\Framework64\v4.0.30319\InstallUtil.exe`
	NOTE: These are the Windows default directories. The location of the executable may be different in your environment.
		Maps an alias to an install utility that ships with the .NET Framework.

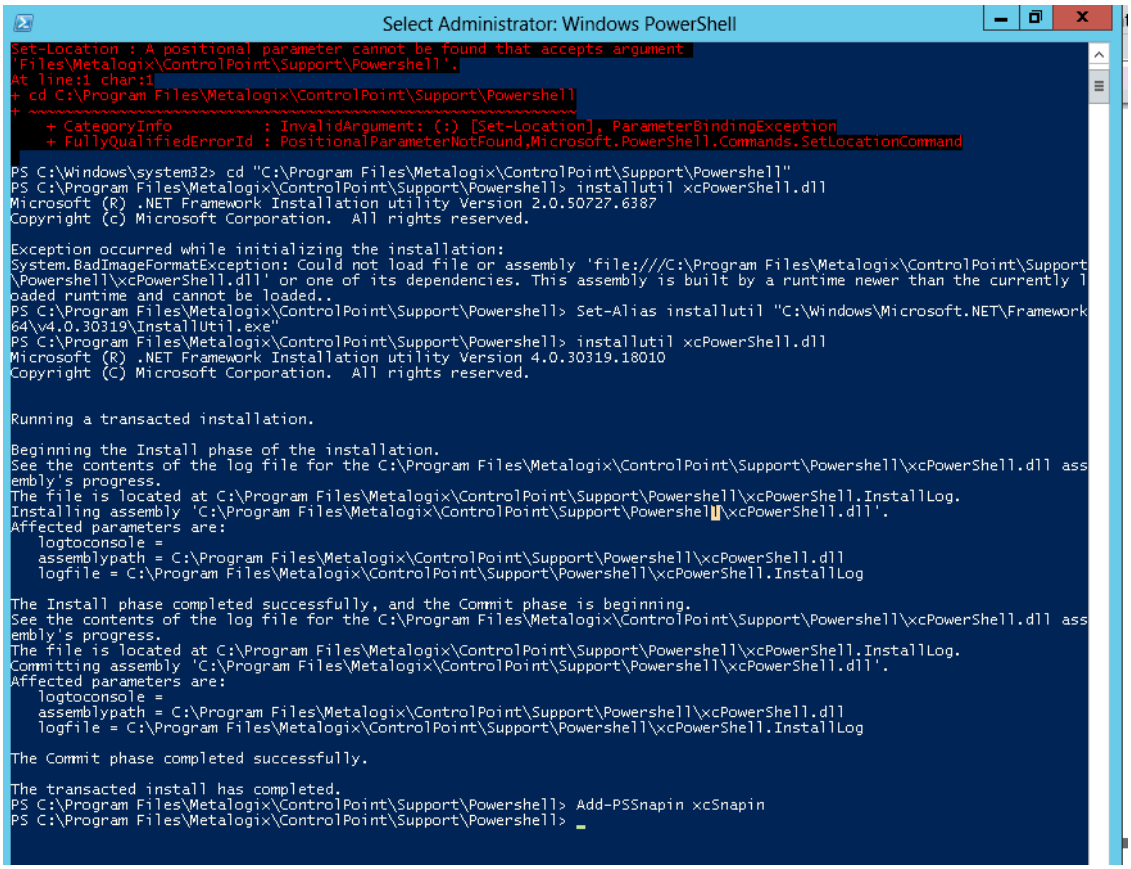
2	`cd C:\Program Files\Metalogix\ControlPoint\Support\PowerShell`	Navigates to the directory containing the xcPowerShell.dll file that contains the ControlPoint cmdlets.

Line	Command	Description
3	`installutil xcPowerShell.dll`	Uses the install utility to load xcPowerShell.dll into Powershell
4	`Add-PSSnapin xcSnapin`	Adds the snap-in containing all of the ControlPoint cmdlets.

Below is an example of the lines of text above in the PowerShell profile.



Below is an example of a PowerShell session with the ControlPoint Cmdlets successfully loaded.



Backup-UserPermissions

Following are the PowerShell cmdlet parameters and xml syntax for the ControlPoint **Backup User Permissions** action.

PowerShell Parameters

Parameter	Description
Backup-UserPermissions	The cmdlet name
-Type	The level of the hierarchy containing the SharePoint object(s) that will be acted on. Valid values are: ▪ **FARM** ▪ **WAP** ▪ **SITE** ▪ **WEB**
-Url	The url of the SharePoint object to act on.

Parameter	Description
	NOTE: If the action is to be performed on a farm, Web application, or site collection, you can use the url for any site within that scope.
-TaskAuditPath (Optional)	If you want to save a ControlPoint Task Audit for the action as a .pdf file, the path to the directory where you want the file to be saved.

Example PowerShell Cmdlet with Parameters

```
Backup-UserPermissions -Type FARM -Url http://2008SharePoint/sites/alpha/
-TaskAuditPath C:\Temp\TaskAudits
```

xml Tags

REMINDER: Xml templates for ControlPoint actions are located in the folder **C:\Program Files\Metalogix\ControlPoint\Support\Powershell.**

Tags	Description
<BackupPermissionsContract...>	The action to be performed
<SharePointObjects>	The container for all of the objects that you want to include in the operation.
<SharePointObject>	The container for the current object.
<SPHierarchyLevel>	The level of the hierarchy for the SharePoint object(s) that will be acted on. Valid values are: ▪ **FARM** ▪ **WAP** ▪ **SITE** ▪ **WEB**
<Url>	The url of the SharePoint object to act on.
<TaskAuditPath> (Optional)	If you want to save a ControlPoint Task Audit for the action as a .pdf file, the path to the directory on the server where you want the file to be saved.

Example xml File Contents

```
<?xml version="1.0" encoding="utf-8"?>
<BackupPermissionsContract xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <SharePointObjects>
        <SharePointObject>
            <SPHierarchyLevel>FARM</SPHierarchyLevel>
            <!--Valid values are FARM, WAP, SITE, WEB-->
            <Url>http://2008sharepoint/sites/alpha</Url>
            <!--For FARM, WAP, or SITE, enter any url within the scope of the
            selection-->
        </SharePointObject>
    </SharePointObjects>
    <TaskAuditPath>C:\Temp\TaskAudits</TaskAuditPath>
    <!--(Optional) Path to the directory where you want to save a .pdf of a
    ControlPoint Task Audit for the action.-->
</BackupPermissionsContract>
```

PowerShell Command to Execute xml

```
Backup-UserPermissions -FilePath C:\Temp\xml\BackupUserPermissions.xml
```

Set-UserDirectPermissions

Following are the PowerShell cmdlet parameters and xml syntax for the ControlPoint **Set User Direct Permissions** action.

PowerShell Parameters

Parameter	Description
Set-UserDirectPermissions	The cmdlet name
-Type	The level of the hierarchy for the SharePoint object(s) that will be acted on. Valid values are: ▪ **FARM** ▪ **WAP** ▪ **SITE** ▪ **WEB** ▪ **LIST**
-Url	The url of the SharePoint object to act on. NOTE: If the action is to be performed on a farm, Web application, or site collection, you can use the url for any site within that scope.
-IncludeChildren	An indication of whether you want the action to include child objects.

Parameter	Description
	▪ If -Type is FARM, WAP, or SITE, **$true** is the only valid value. ▪ If -Type is WEB, valid values are **$true** or **$false**. NOTE: If this parameter is omitted: ▪ for FARM, WAP, or SITE, a value of **true** will be assumed ▪ for WEB or LIST, a value of **false** will be assumed.
-UserLoginNames	Login name(s) of the user(s) whose permissions will be set. Enter multiple login names as a comma-separated list.
-PermissionsLevel	The permissions level to set for the user(s). NOTE: You can specify a custom permissions level as long as it has already been defined for the site collection.
-TaskAuditPath (Optional)	If you want to save a ControlPoint Task Audit for the action as a .pdf file, the path to the directory where you want the file to be saved.

Example PowerShell Cmdlet with Parameters

```
Set-UserDirectPermissions -url http://2008SharePoint/alpha -type SITE -
UserLoginNames 2008SharePoint\jamesjoyce,2008SharePoint\marktwain -
PermissionsLevel Design -IncludeChildren $true -TaskAuditPath C:\temp\TaskAudits
```

xml Tags

REMINDER: Xml templates for ControlPoint actions are located in the **C:\Program Files\Metalogix\ControlPoint\Support\Powershell..**

Tags	Description
<SetUserDirectPermissionsContract...>	The action to be performed
<SharePointObjects>	The container for all of the objects that you want to include in the operation.
<SharePointObject>	The container for the current object.
<SPHierarchyLevel>	The level of the hierarchy for the SharePoint object(s) that will be acted on. Possible values are: ▪ **FARM** ▪ **WAP** ▪ **SITE** ▪ **WEB** ▪ **LIST**

Tags	Description
	NOTE: If the action is to be performed on a farm, Web application, or site collection, you can use the url for any site within that scope.
<Url>	The url of the SharePoint object to act on.
<UserLoginNames> <string>	Login name(s) of the user(s) whose permissions will be set. Enter each login name as a separate string.
<PermissionsType>	The permissions level to set for the user(s). NOTE: You can specify a custom permissions level as long as it has already been defined for the site collection.
<IncludeChildren>	An indication of whether you want the action to include child objects. ▪ If SPHierarchyLevel=FARM, WAP, or SITE, **true** is the only valid value. ▪ If SPHierarchyLevel=WEB or LIST, valid values are **true** or **false**.
<TaskAuditPath> (Optional)	If you want to generate a ControlPoint Task Audit (as a .pdf file) for the action, the path to the directory where you want the file to be saved

Example xml File Contents

```
<?xml version="1.0" encoding="utf-8"?>
<SetUserDirectPermissionsContract xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <SharePointObjects>
        <SharePointObject>
        <SPHierarchyLevel>WEB</SPHierarchyLevel>
        <!--Valid values are FARM, WAP, SITE, WEB, LIST-->
        <Url>http://2008sharepoint/sites/alpha/</Url>
        <!--For FARM, WAP, or SITE, enter any url within the scope of the selection-->
        <IncludeChildren>true</IncludeChildren>
        <!--If SPHierarchyLevel=FARM, WAP, or SITE, true is the only valid value.  If
        SPHierarchyLevel=WEB or LIST and you do NOT want the action to include child
        objects, change the value to false.-->
        </SharePointObject>
    </SharePointObjects>
    <UserLoginNames>
        <string>2008sharepoint\isaacasimov</string>
    </UserLoginNames>
    <!--Login name(s) of user(s)whose permissions you want to set.  Enter each name as a
    separate string.-->
    <PermissionsType>Contribute</PermissionsType>
    <!--Can be any built-in permissions level that is valid for the site template.-->
    <TaskAuditPath>C:\Temp\TaskAudits</TaskAuditPath>
    <!--(Optional) Path to the directory where you want to save a .pdf of a ControlPoint
    Task Audit for the action.-->
</SetUserDirectPermissionsContract>
```

Example PowerShell Command to Execute

```
Set-UserDirectPermissions -FilePath C:\xml\SetUserDirectPermissions.xml
```

Duplicate-UserPermissions

Following are the PowerShell cmdlets and xml syntax for the ControlPoint **Duplicate User Permissions** action.

PowerShell Parameters

Parameter	Description
Duplicate-UserPermissions	The cmdlet name
-Type	The level of the hierarchy containing the SharePoint object(s) that will be acted on. Valid values are: ▪ **FARM** ▪ **WAP** ▪ **SITE** ▪ **WEB**
-Url	The url of the SharePoint object to act on.

Parameter	Description
	NOTE: If the action is to be performed on a farm, Web application, or site collection, you can use the url for any site within that scope.
-IncludeChildren	An indication of whether you want the action to include child sites. ▪ If -Type is FARM, WAP, or SITE, **$true** is the only valid value. ▪ If -Type is WEB, valid values are **$true** or **$false**. NOTE: If this parameter is omitted: ▪ for FARM, WAP, or SITE, a value of **true** will be assumed ▪ for WEB, a value of **false** will be assumed.
-DestinationUserNames	Login name(s) of the user(s) to whom permissions will be duplicated. Enter multiple login names as a comma-separated list.
-ModelUserName	Login name of the user whose permissions you want to duplicate.
-TaskAuditPath (Optional)	If you want to save a ControlPoint Task Audit for the action as a .pdf file, the path to the directory where you want the file to be saved.

Example PowerShell Cmdlet with Parameters

```
Duplicate-UserPermissions -Type WAP -url 2013sharepoint/sites/alpha -
IncludeChildren $true -DestinationUserNames
2013sharepoint\margaretmeade,2013sharepoint\isaacasimov -
ModelUserName2013sharepoint\jamesjoyce -TaskAuditPath C:\temp\TaskAudits
```

xml Tags

REMINDER: Xml templates for ControlPoint actions are located in the folder C:\Program Files\Metalogix\ControlPoint\Support\Powershell.

Tags	Description
<DuplicateUserPermissionsContract...>	The action to be performed
<SharePointObjects>	The container for all of the objects that you want to include in the operation.
<SharePointObject>	The container for the current object.

Tags	Description
<SPHierarchyLevel>	The level of the hierarchy for the SharePoint object(s) that will be acted on. Valid values are: • FARM • WAP • SITE • WEB
<Url>	The url of the SharePoint object to act on.
<IncludeChildren>	An indication of whether you want the action to include child sites. ▪ If SPHierarchyLevel=FARM, WAP, or SITE, true is the only valid value. ▪ If SPHierarchyLevel=WEB, valid values are true or false.
<Url>	The url of the SharePoint object to act on. NOTE: If the action is to be performed on a farm, Web application, or site collection, you can use the url for any site within that scope.
<DestinationUserNames> <string>	Login name(s) of the user(s) to whom permissions will be duplicated. Enter each login name as a separate string.
<ModelUserName>	Login name of the user whose permissions will be duplicated.
<TaskAuditPath> (Optional)	If you want to save a ControlPoint Task Audit for the action as a .pdf file, the path to the directory where you want the file to be saved.

Example xml File Contents

```
<?xml version="1.0" encoding="utf-8"?>
<DuplicateUserPermissionsContract xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <SharePointObjects>
        <SharePointObject>
            <SPHierarchyLevel>WAP</SPHierarchyLevel>
            <!--Valid values are FARM, WAP, SITE, WEB-->
            <Url>http://2008sharepoint/sites/alpha/</Url>
            <!--For FARM, WAP, or SITE, enter any url within the scope of the selection-->
            <IncludeChildren>true</IncludeChildren>
            <!--If SPHierarchyLevel=FARM, WAP, or SITE, true is the only valid value. If
            SPHierarchyLevel=WEB and you do NOT want the action to also include child sites,
            change the value to false.-->
        </SharePointObject>
    </SharePointObjects>
    <DestinationUserNames>
        <string>2008sharepoint\fscottfitzgerald</string>
        <string>2008sharepoint\isaacasimov</string>
        <!--Login name(s) of user(s)whose permissions you want to set.  Enter each name as a
        separate string.-->
    </DestinationUserNames>
    <ModelUserName>2008sharepoint\margaretmeade</ModelUserName>
    <!--Login name of the user whose permissions you want to duplicate-->
    <TaskAuditPath></TaskAuditPath>
    <!--(Optional) Path to the directory where you want to save a .pdf of a ControlPoint Task
    Audit for the action.-->
</DuplicateUserPermissionsContract>
```

PowerShell Comand to Execute xml

```
Duplicate-UserPermissions -FilePath C:\xml\DuplicateUserPermissions.xml
```

Delete-UserPermissions

Following are the PowerShell cmdlet parameters and xml syntax for the ControlPoint Delete User Permissions action.

PowerShell Parameters

Parameter	Description
Delete-UserPermissions	The cmdlet name
-Type	The level of the hierarchy containing the SharePoint object(s) that will be acted on. Valid values are: ▪ **FARM** ▪ **WAP** ▪ **SITE** ▪ **WEB**
-Url	The url of the SharePoint object to act on. NOTE: If the action is to be performed on a farm, Web application, or site collection, you can use the url for any site within that scope.

Parameter	Description
-UserLoginNames	Login name(s) of the user(s) whose permissions will be deleted. Enter multiple login names as a comma-separated list.
-ReassignTo (Optional)	Login name(s) of the user(s) to whom you want to reassign deleted permissions. Enter multiple login names as a comma-separated list.
-IncludeChildren	An indication of whether you want the action to include child sites. ▪ If -Type is FARM, WAP, or SITE, **$true** is the only valid value. ▪ If -Type is WEB, valid values are **$true** or **$false**. NOTE: If this parameter is omitted: ▪ for FARM, WAP, or SITE, a value of **true** will be assumed ▪ for WEB, a value of **false** will be assumed.
-DeleteFromAllPeople	An indication of whether you want the deleted user(s) to be removed from the site collection All People list. Valid values are: ▪ **$true** ▪ **$false** NOTE: If this parameter is omitted, a value of **false** will be assumed.
-DeleteAlerts	An indication of whether you want any alerts that have been created for the user(s) to be deleted. Valid values are: ▪ **$true** ▪ **$false** NOTE: If this parameter is omitted, a value of **false** will be assumed.
-TaskAuditPath (Optional)	If you want to save a ControlPoint Task Audit for the action as a .pdf file, the path to the directory where you want the file to be saved.

Example PowerShell Cmdlet with Parameters

```
Delete-UserPermissions -url http://2008SharePoint/alpha -Type WAP -
UserLoginNames 2008SharePoint\jamesjoyce,2008SharePoint\marktwain -ReassignTo
2008SharePoint\fscottfitzgerald -IncludeChildren $true -DeleteFromAllPeople
$true -DeleteAlerts $true -TaskAuditPath C:\Temp\TaskAudits
```

xml Tags

Tag	Description
<DeleteUserPermissionsContract...>	The action to be performed

Tag	Description
<SharePointObjects>	The container for all of the objects that you want to include in the operation.
<SharePointObject>	The container for the current object.
<SPHierarchyLevel>	The level of the hierarchy for the SharePoint object(s) that will be acted on. Valid values are: ▪ **FARM** ▪ **WAP** ▪ **SITE** ▪ **WEB**
<Url>	The url of the SharePoint object to act on. NOTE: If the action is to be performed on a farm, Web application, or site collection, you can use the url for any site within that scope.
<IncludeChildren>	An indication of whether you want the action to include child sites. ▪ If SPHierarchyLevel=FARM, WAP, or SITE, **true** is the only valid value. If SPHierarchyLevel=WEB, valid values are **true** or **false**.
<UserLoginNames> <string>	List of login names of the users whose permissions will be deleted. Enter each login name as a separate string.
<ReassignTo> <string>	List of login names of the users to whom you want to reassign deleted permissions. Enter each login name as a separate string.
<DeleteFromAllPeople>	An indication of whether you want the user(s) to be deleted from the site collection All People list. Valid values are ▪ **true** ▪ **false**
<DeleteAlerts>	An indication of whether you want any alerts that have been created for the user(s) to be deleted. Valid values are: ▪ **true** ▪ **false**
<TaskAuditPath> (Optional)	If you want to generate a ControlPoint Task Audit (as a .pdf file) for the action, the path to the directory where you want the file to be saved.

Example xml File Contents

```
<?xml version="1.0" encoding="utf-8"?>
<DeleteUserPermissionsContract
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <SharePointObjects>
        <SharePointObject>
            <SPHierarchyLevel></SPHierarchyLevel>
            <!--Valid values are FARM, WAP, SITE, WEB-->
            <Url></Url>
            <!--For FARM, WAP, or SITE, enter any url within the scope of the
            selection-->
            <IncludeChildren>true</IncludeChildren>
            <!--If SPHierarchyLevel=FARM, WAP, or SITE, true is the only valid
            value. If SPHierarchyLevel=WEB and you do NOT want the action to
            also include child sites, change the value to false.-->
        </SharePointObject>
    </SharePointObjects>
    <UserLoginNames>
        <string></string>
    </UserLoginNames>
    <!--Login name(s)of user(s)whose permissions you want to set.  Enter each
    name as a separate string.-->
    <ReassignTo>
        <string></string>
    </ReassignTo>
    <!--(Optional) Login name(s) of user(s) to whom you want to reassign the
    deleted permissions.  Enter each name as a separate string-->
    <DeleteFromAllPeople>false</DeleteFromAllPeople>
    <!--If you want to delete the user(s) from the site collection All People
    list, change this value to true.-->
    <DeleteAlerts>false<\DeleteAlerts>
    <!--If you want to delete any alert(s) that have been created for the user
    change this value to true.-->
    <TaskAuditPath></TaskAuditPath>
    <!--(Optional) Path to the directory where you want to save a .pdf of a
    ControlPoint Task Audit for the action.-->
</DeleteUserPermissionsContract>
```

PowerShell Command to Execute xml

```
Delete-UserPermissions -FilePath C:\xml\DeleteUserPermissions.xml
```

Delete-Site

Following are the PowerShell cmdlet parameters and xml syntax for the ControlPoint Delete Site action.

PowerShell Parameters

Parameter	Description
Delete-Site	The cmdlet name

Parameter	Description
-Type	The level of the hierarchy containing the SharePoint object(s) that will be acted on. Valid values are: ▪ **FARM** ▪ **WAP** ▪ **SITE** ▪ **WEB**
-Url	The url of the SharePoint object to act on. NOTE: For FARM, WAP, or SITE, you can use the url for any site within the scope.
-IncludeChildren	This parameter must be explicitly included in the command with a value of **$true**.
-DeleteAll	If -Type is FARM, this parameter must be included with a value of **$true** for the action to be processed. NOTE: This parameter does not apply when type is WAP, SITE, or WEB.
-DoBackup (Optional)	An indication of whether the site(s) should be backed up before deletion. Valid values are: ▪ **$true** ▪ **$false** NOTE: If you omit this parameter, a value of **false** will be assumed.
-BackupSecurity (Optional)	If -DoBackup is set to true, an indication of whether site security (which includes timestamps, security, and user information) should be backed up as well. Valid values are An indication of whether the site(s) should be backed up before deletion. Valid values are: ▪ **$true** ▪ **$false** NOTE: If you omit this parameter, a value of **false** will be assumed.
-Compress (Optional)	If -DoBackup is set to true, an indication of whether you want each site backup to be compressed into a CAB file (with a .cmp extension). Valid values are: ▪ **$true** ▪ **$false** NOTE: If you omit this parameter, a value of **false** will be assumed.

Parameter	Description
-ExportPath	If DoBackup is set to true, the directory to which you want site content to be exported.
-TaskAuditPath (Optional)	If you want to save a ControlPoint Task Audit for the action as a .pdf file, the path to the directory where you want the file to be saved.

```
Delete-Site -Type WEB -url http://sbellum:8080/sites/alpha/skunkworks/ -
IncludeChildren $true -DoBackup  $true -BackupSecurity $true -ExportPath C:
\Temp\Site_Backups
-TaskAuditPath C:\Temp\Task Audits
```

xml Tags

Tags	Description
<DeleteSiteContract...>	The action to be performed
<SharePointObjects>	The container for all of the objects that you want to include in the operation.
<SharePointObject>	The container for the current object.
<SPHierarchyLevel>	The level of the hierarchy for the SharePoint object(s) that will be acted on. Valid values are: ▪ **FARM** ▪ **WAP** ▪ **SITE** ▪ **WEB**
<Url>	The url of the SharePoint object to act on. NOTE: For FARM, WAP, or SITE, you can use the url for any site within the scope.
<IncludeChildren>	For this action, the only valid value is **true**.
<DoBackup>	An indication of whether the site(s) should be backed up before deletion. Valid values are: ▪ **true** ▪ **false**
<BackupSecurity>	If DoBackup is set to true, an indication of whether site security (which includes timestamps, security, and user information) should be backed up as well. Valid values are ▪ **true** ▪ **false**

Tags	Description
<Compress> (Optional)	If DoBackup is set to true, an indication of whether you want each site backup to be compressed into a CAB file (with a .cmp extension). Valid values are: ▪ **true** ▪ **false**
<ExportPath>	If DoBackup is set to true, an indication of whether you want each site backup to be compressed into a CAB file (with a .cmp extension). Valid values are: ▪ **true** ▪ **false**
<TaskAuditPath> (Optional)	If you want to save a ControlPoint Task Audit for the action as a .pdf file, the path to the directory where you want the file to be saved.

Example xml File Contents

```
<?xml version="1.0" encoding="utf-8"?>
<DeleteSiteContract xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <SharePointObjects>
        <SharePointObject>
            <SPHierarchyLevel>WEB</SPHierarchyLevel>
            <!--Valid values are FARM, WAP, SITE, WEB-->
            <Url>http://2008sharepoint/sites/alpha/newsite</Url>
            <!--For FARM, WAP, OR SITE, enter any url within the scope of the
            selection-->
            <IncludeChildren>true</IncludeChildren>
            <!--The only valid value is true.-->
        </SharePointObject>
    </SharePointObjects>
    <DoBackup>false</DoBackup>
    <!--(Optional) If you want to back up the site(s) before deletion, enter a
    value of true-->
    <BackupSecurity>true</BackupSecurity>
    <!--If DoBackup is set to true and you want to back up site security (which
    includes timestamps, security, and user information), enter a value of true.-->
    <Compress>true</Compress>
        <!--If DoBackup is set to true and you want the site backup to be
        compressed into a CAB file (with a .cmp extension), enter a value of
        true-->
    <ExportPath>\\2008sharepoint\Temp\Backup</ExportPath>
    <!--If DoBackup is set to true, the path to the directory where you want the
    backup to be exported.-->
    <TaskAuditPath>C:\temp\TaskAudits</TaskAuditPath>
    <!--(Optional) Path to the directory where you want to save a .pdf of a
    ControlPoint Task Audit for the action.-->
</DeleteSiteContract>
```

PowerShell Command to Execute xml

```
Delete-Site -FilePath C:\xml\DeleteSite.xml
```

Run-Discovery

In PowerShell, the Run-Discovery cmdlet has no additional parameters. From the PowerShell command line, simply type `Run-Discovery`.

REMINDER: Any ControlPoint settings that have been configured for Full Discovery will also apply to a programmatically-run Full Discovery. Refer to the *ControlPoint User's Guide* for detail.

Executing Operations Using ControlPoint Saved Instructions

You can use PowerShell to execute any ControlPoint action or analysis that includes the **[Save Instructions]** option.

(See also "Saving Instructions" in the *ControlPoint User Guide*.)

Executing Instructions for ContrlPoint Actions

Use the following parameters and syntax to execute saved instructions for ControlPoint actions.

Parameter	Description
Execute-Xml	The cmdlet "name"
-FilePath	The full path to the location where the instructions you want to execute are saved.
-TaskAuditPath (Optional)	If you want to save a ControlPoint Tast Audit for the action as a .pdf file, the path to the directory where you want the file to be saved.

Example PowerShell Cmdlet with Parameters for Executing Instructions for a ControlPoint Action

```
Execute-Xml -FilePath "C:\somedir\Backup Permissions1.xml" -TaskAuditPath "c:
\auditdir"
```

Executing Instructions for ContrlPoint Analyses

Use the following parameters and syntax to execute saved instructions for ControlPoint analyses.

Parameter	Description
Execute-Xml	The cmdlet "name"
-FilePath	The full path to the location where the instructions you want to execute are saved.
-ReoportPath (Optional)	The location and file name (including file type extension) for saved analysis results. Valid file extensions are: • .csvcsv • .pdf. • .xls or xlsx (Excel Format)

Example PowerShell Cmdlet with Parameters for Executing Instructions for a ControlPoint Action

```
Execute-Xml -FilePath "C:\somedir\Site Permissions.xml" -ReportPath "c:
\reportdirectory\MyFavoritePermissions.pdf"
```

Incorporating ControlPoint Actions into .NET Applications as Web Services

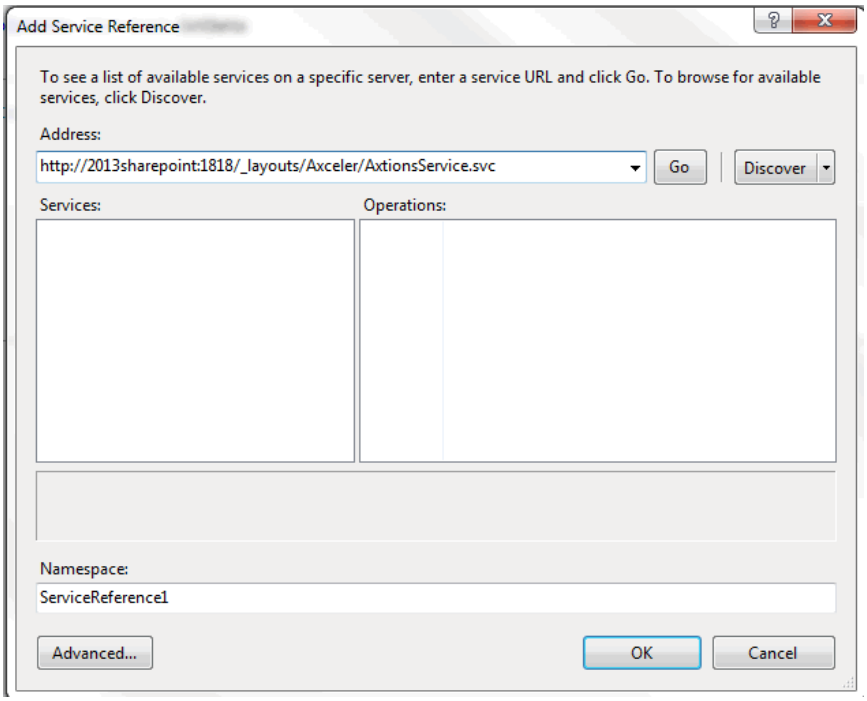
ControlPoint exposes several of its actions via Windows Communication Foundation (WCF) Service methods. You can reference a single WCF service to incorporate these methods in your own .NET applications.

To add the ControlPoint Actions Service Reference from Within Visual Studio:

Type the following url into the Add Service Reference Dialog **Address** field:

http://<machine_name>:<port_number>/_layouts/Axceler/ActionsService.svc

where <*machine_name*> refers to the name of the machine on which ControlPoint was initially installed and <*port_number*> refers to the port used by the ControlPoint application (1818 is the ControlPoint default).



Example Code to Call BackupPermissions Web Service

Parameters and valid values are comparable to those for the PowerShell cmdlet Backup-UserPermissions.

```
// Create the WCF Client
using (ActionsServiceClient client = new ActionsServiceClient())
{
    // Set impersonation levels
    client.ClientCredentials.Windows.AllowedImpersonationLevel =
            System.Security.Principal.TokenImpersonationLevel.Impersonation;
    client.ChannelFactory.Credentials.Windows.ClientCredential =
            System.Net.CredentialCache.DefaultNetworkCredentials;

    BackupPermissionsContract contract =
            new BackupPermissionsContract();

    // Add SharePoint objects to perform the action on
    List<SharePointObject> sharePointObjects = new List<SharePointObject>();
    SharePointObject sharePointObject = new SharePointObject()
    {
        SPHierarchyLevel = SPHierarchyLevel.WEB,
        Url = "http://devsp3:8011/sites/Coll1/Site1",
        IncludeChildren = "true"
    };

    sharePointObjects.Add(sharePointObject);
    contract.SharePointObjects = sharePointObjects;

    // Set the permission type
    contract.AdminLoginName = @"DEVSP3\Administrator";

    // Set the directory where the task audit report will be exported
    // Note: this is optional
    contract.TaskAuditPath = @"C:\TaskAuditReports";

    try
    {
        client.Open();
        client.BackupPermissions(contract);
        client.Close();
    }
    catch
    {
        client.Abort(); // Abort the call if there is an exception
    }
}
```

Example Code to Call SetUserDirectPermissions Web Service

Parameters and valid values are comparable to those for the PowerShell cmdlet Set-UserDirectPermissions.

```
// Create the WCF Client
using (ActionsServiceClient client = new ActionsServiceClient())
{
    // Set impersonation levels
    client.ClientCredentials.Windows.AllowedImpersonationLevel =
            System.Security.Principal.TokenImpersonationLevel.Impersonation;
    client.ChannelFactory.Credentials.Windows.ClientCredential =
            System.Net.CredentialCache.DefaultNetworkCredentials;

    SetUserDirectPermissionsContract contract =
            new SetUserDirectPermissionsContract();

    // Add SharePoint objects to perform the action on
    List<SharePointObject> sharePointObjects = new List<SharePointObject>();
    SharePointObject sharePointObject = new SharePointObject()
    {
        SPHierarchyLevel = SPHierarchyLevel.WEB,
        Url = "http://devsp3:8011/sites/Coll1/Site1",
        IncludeChildren = "true"
    };

    sharePointObjects.Add(sharePointObject);
    contract.SharePointObjects = sharePointObjects;

    // Set the permission type
    contract.AdminLoginName = @"DEVSP3\Administrator";
    contract.PermissionsType = "Read";

    // Set the directory where the task audit report will be exported
    // Note: this is optional
    contract.TaskAuditPath = @"C:\TaskAuditReports";
    contract.UserLoginNames = new List<string>()
    {
        @"DEVSP3\User1"
    };

    try
    {
        client.Open();
        client.SetUserDirectPermissions(contract);
        client.Close();
    }
    catch
    {
        client.Abort(); // Abort the call if there is an exception
    }
}
```

Example Code to Call DuplicateUserPermissions

Parameters and valid values are comparable to those for the PowerShell cmdlet Duplicate-UserPermissions.

```
// Create the WCF Client
using (ActionsServiceClient client = new ActionsServiceClient())
{
    // Set impersonation levels
    client.ClientCredentials.Windows.AllowedImpersonationLevel =
            System.Security.Principal.TokenImpersonationLevel.Impersonation;
    client.ChannelFactory.Credentials.Windows.ClientCredential =
            System.Net.CredentialCache.DefaultNetworkCredentials;

    DuplicateUserPermissionsContract contract =
            new DuplicateUserPermissionsContract();

    // Add SharePoint objects to perform the action on
    List<SharePointObject> sharePointObjects = new List<SharePointObject>();
    SharePointObject sharePointObject = new SharePointObject()
    {
        SPHierarchyLevel = SPHierarchyLevel.WEB,
        Url = "http://devsp3:8011/sites/Coll1/Site1",
        IncludeChildren = "true"
    };

    sharePointObjects.Add(sharePointObject);
    contract.SharePointObjects = sharePointObjects;

    contract.ModelUserName = @"DEVSP3\User1";
    contract.DestinationUserNames = new List<string>()
    {
        @"DEVSP3\User2"
    };

    // Set the permission type
    contract.AdminLoginName = @"DEVSP3\Administrator";

    // Set the directory where the task audit report will be exported
    // Note: this is optional
    contract.TaskAuditPath = @"C:\TaskAuditReports";

    try
    {
        client.Open();
        client.DuplicateUserPermissions(contract);
        client.Close();
    }
    catch
    {
        client.Abort(); // Abort the call if there is an exception
    }
}
```

Example Code to Call DeleteSite Web Service

Parameters and valid values are comparable to those for the PowerShell cmdlet Delete-UserPermissions.

```
// Create the WCF Client
using (ActionsServiceClient client = new ActionsServiceClient())
{
    // Set impersonation levels
    client.ClientCredentials.Windows.AllowedImpersonationLevel =
            System.Security.Principal.TokenImpersonationLevel.Impersonation;
    client.ChannelFactory.Credentials.Windows.ClientCredential =
            System.Net.CredentialCache.DefaultNetworkCredentials;

    DeleteSiteContract contract =
            new DeleteSiteContract();

    // Add SharePoint objects to perform the action on
    List<SharePointObject> sharePointObjects = new List<SharePointObject>();
    SharePointObject sharePointObject = new SharePointObject()
    {
        SPHierarchyLevel = SPHierarchyLevel.WEB,
        Url = "http://devsp3:8011/sites/Coll1/Site3",
        IncludeChildren = "true"
    };

    sharePointObjects.Add(sharePointObject);
    contract.SharePointObjects = sharePointObjects;

    // Set the permission type
    contract.AdminLoginName = @"DEVSP3\Administrator";
    contract.BackupSecurity = true;
    contract.Compress = true;
    contract.DoBackup = true;
    contract.ExportPath = @"C:\Backup";

    // Set the directory where the task audit report will be exported
    // Note: this is optional
    contract.TaskAuditPath = @"C:\TaskAuditReports";

    try
    {
        client.Open();
        client.DeleteSite(contract);
        client.Close();
    }
    catch
    {
        client.Abort(); // Abort the call if there is an exception
    }
}
```

Example Code to Call RunDiscovery

Parameters and valid values are comparable to those for the PowerShell cmdlet Run-Discovery.

Creating a Custom Web Service for Use with a ControlPoint Content Creation Policy

When a ControlPoint policy is created to control content creation, a Web service which includes your own custom logic is called whenever a user adds or updates content or attachments in a list or library covered by the policy. The logic may, for example, scan content for particular words or text strings, and ControlPoint will send a notification to one or more users specified in the policy when a violation occurs.

NOTE: Custom logic included in a Web service is not considered part of the ControlPoint product and therefore is not supported by Metalogix.

In order to integrate with ControlPoint, the Web service must contain the following entry points using parameters entered *exactly* as shown:

```
bool CPPolicyEventReceiverItemAdded(string user, guid siteID, Guid webID, Guid
listItemId, string itemXml, byte[] file)

bool CPPolicyEventReceiverItemAttachmentAdded(string user, Guid siteID, Guid
webID, GuidlistItemId, string attachUrl, string itemXml, byte[] file)

bool CPPolicyEventReceiverItemUpdated(string user, Guid siteID, Guid webID, Guid
listItemId, string itemXml, byte[] file)
```

where:

user is the SharePoint login account of the user triggering the policy

siteID is the guid of the site collection on which the policy is being triggered

webID is the guid of the site on which the policy is being triggered

listID is the guid of the list or library on which the policy is being triggered

listItemID is the guid of the item for which the policy is being triggered

attachURL is the url for the attachment for which the policy is being triggered

itemXml is the xml that contains all of the meta data for the item

file is the content of the document for which the policy is being triggered, in binary format

After your custom logic, the following logic must be added to indicate whether or not an email notification should be sent as specified in the ContolPoint user interface at the time the policy was created:

```
if (condition)
   return true;

else
   return false;
```

EXAMPLE:

```
/// <summary>
/// This method will be invoked when the new item or document is added to List/Library in the existing ControlPoint policy scope.
/// </summary>
/// <param name="user">username(item or document owner)</param>
/// <param name="siteId">Site Collection Id</param>
/// <param name="webId">Site Id</param>
/// <param name="listId">List or Document Library Id</param>
/// <param name="listItemId">Item Id</param>
/// <param name="itemXml">Item Schema(xml)</param>
/// <param name="file">Document content(if it is item, this parameter is null)</param>
/// <returns>return true if you want to notify policy email recievers</returns>
[WebMethod]
public bool CPPolicyEventReceiverItemAdded(string user, System.Guid siteId, System.Guid webId, System.Guid listId, System.Guid listItemId,
string itemXml, byte[] file)
    {
        bool condition = false;
        //Custom logic starts here



        //Custom logic ends here
        if (condition)
            return true;
        else
            return false;
    }
```

The entry points provided above will be called after the content has been added to SharePoint.

The logic within the Web service must be able to access the content that has been added through standard SharePoint methods, such as the SharePoint API.

For ControlPoint to be able to access the Web service:

- the ControlPoint Service account must have access to the Web service from the Web front end server on which ControlPoint is installed

AND

- the ControlPoint Configuration Setting (POLICYSERVICEURL) must contain the full service url to the location of the Web service. (See the *ControlPoint Administration Guide* for information on maintaining ControlPoint Configuration Settings.)

NOTE: In order to simulate the environment in which it will be used, it is recommended that the functionality be tested on the Web front-end server on which ControlPoint is installed while logged in as the ControlPoint Service account.