BeyondTrust

Privilege Management API Guide

Powered By Defendpoint

Table of Contents

Privilege Management Powershell API Reference Guide

The Privilege Management for Windows PowerShell API enables you to configure Privilege Management for Windows using PowerShell. This enables integrations with external systems and provides an alternative to using the Privilege Management Policy Editor.

All operations available using the Privilege Management Policy Editor are also available using the PowerShell API. Syntax help is available via the module by using `Get-Help <cmdlet name>`.

The types properties and values you will be working with are documented in the automatically generated file `PowerShell API.chm`, which resides in the PowerShell directory. This is usually located at `C:/Program Files/Avecto/Privilege Guard Client/PowerShell`.

IMPORTANT!

In order to maintain backwards compatibility, the PowerShell cmdlets have not been renamed along with the product. Therefore, many of the objects and cmdlets reference the former product name Defendpoint.

The examples in this guide will show you how to retrieve, update, and save an existing configuration file.

The typical workflow for using the PowerShellAPI is:

- `Get-DefendpointSettings`.
- Find the relevant section of the config you want to work with.
- Update, add, or delete the information.
- `Set-DefendpointSettings`.

Prerequisites

Before running any other cmdlets, run the following two commands to initialize the cmdlets:

```
Import-Module 'C:/Program Files/Avecto/Privilege Guard
Client/PowerShell/Avecto.Defendpoint.Cmdlets/Avecto.Defendpoint.Cmdlets.dll'
```

```
Import-Module 'C:/Program Files/Avecto/Privilege Guard
Client/PowerShell/Avecto.Defendpoint.Cmdlets/Avecto.Defendpoint.Settings.dll'
```

Get Application Definitions from Files: Get-DefendpointFileInformation

`Get-DefendpointFileInformation` gets properties of one or more applications or files. The properties returned include those Privilege Management for Windows can use as matching criteria for files and applications, such as **Publisher**, **ProductName**, and **FileNam**e. This information can be used to construct Privilege Management for Windows settings.

Overview

Syntax

```
Get-DefendpointFileInformation [-Path '/path/to/file'] [-Directory '/path/to/directory' [-Recurse]]
[-FileType 'ApplicationType.$FileType'] [-COM]
```

Description

The `Get-DefendpointFileInformation` cmdlet retrieves properties of a file or list of files. This information can then be used to construct Privilege Management settings.

Note: *This cmdlet supports Common Parameters.*

Parameters

Parameter	Type	Description	Required
`Path`	String	Define the path of the file to investigate. This parameter supports regular expressions (regex). This parameter can be piped.	Yes if `Directory` is not set.
`Directory`	String	Define the directory containing multiple files to investigate. To search all subfolders and files in the directories, include the `-Recurse` parameter.	Yes if `Path` is not set.
`Recurse`	Boolean	Search all subfolders and files in the directories specified by the `-Directory` parameter.	
`COM`	Boolean	Returns registered COM objects stored on the local machine.	
`EncodeUnicodeChars`	Boolean	Encode any unicode characters in strings within the object.	
`FileType`	String	Define which file type to search for. Only one file type can be supplied for each command. See below for a list of possible file types.	

File Types

Input String	File Type
ActiveXControl	Active X
BatchFile	Batch File
COMClass	COM Class

Input String	File Type
Content	File Resource
ControlPanelApplet	Control Panel
Executable	Executable
InstallerPackage	Installer Package
ManagementConsoleSnapin	MMC Snap In
PowerShellScript	PowerShell Script
RegistrySettings	Registry Settings
RemotePowerShellCommand	Remote PowerShell Command
RemotePowerShellScript	Remote PowerShell Script
Service	Service
Unistaller	Uninstaller
Url	URL
WindowsScript	Windows Script
WindowsStoreApplication	AppX Package

Return Values

```
Get-DefendpointFileInformation returns a list of application definitions.
```

Examples

Get All Executables in a Specified Eath and Reference the FileName of the First

```
$Executables = Get-DefendpointFileInformation –Path “C:\Program Files\Internet Explorer\*.exe”
```

```
$Executables.Item(0).FileName
```

Get All Application Files in a Specified Directory and Return Unique Publishers

```
$Files = Get-DefendpointFileInformation –Directory “C:\Program Files\Internet Explorer\“
```

```
$Files.Publisher
```

Get All Registered COM Objects

```
Get-DefendpointFileInformation –COM
```

Recursively Search for All Batch Files within a Directory

```
Get-DefendpointFileInformation -Directory "C:/Users/admin/Desktop" -FileType "BatchFile" -Recurse
```

Retrieve the Privilege Management for Windows Settings: Get-DefendpointSettings

`Get-DefendpointSettings` retrieves the Privilege Management for Windows settings from local file, local group policy, or domain GPO. Once you have the settings in a PowerShell session or script, you can update relevant sections, and then write them back using **Set-DefendpointSettings**.

Syntax

```
Get-DefendpointSettings [-LocalPolicy] [-LocalFile -FileLocation 'path/to/file'] [-UserPolicy] [-
Domain -LDAP 'path/to/LDAP'] [-XML]
```

Description

The `Get-DefendpointSettings` cmdlet gets the Privilege Management for Windows settings from the Local Group Policy, a specified Group Policy Object (GPO), or from a specified XML file. The output is a **DefendpointSettings** object or an XML- formatted string, depending on the parameters supplied.

> ***Note:*** *This cmdlet supports Common Parameters.*

Parameters

Parameter	Type	Description	Required
`LocalPolicy`	Boolean	Return the Privilege Management for Windows settings object from the local group policy.	One of
`LocalFile`	Boolean	Return the Privilege Management for Windows settings from a local Privilege Management settings file. This cmdlet defaults to `%PROGRAMDATA%\Avecto\Privilege Guard\PrivilegeGuardConfig.xml`. Specify an alternate file using the `-FileLocation` parameter.	One of
`FileLocation`	String	Specify the location of the Privilege Management for Windows settings file. This cmdlet defaults to `%PROGRAMDATA%\Avecto\Privilege Guard\PrivilegeGuardConfig.xml` if a file path is not supplied when the `-LocalFile` parameter is used.	No
`UserPolicy`	Boolean	Return the policy of a user. This cmdlet defaults to a machine policy if this parameter is not used.	No
`Domain`	Boolean	Return the Privilege Management for Windows settings from the Group Policy Object (GPO) specified by the `-LDAP` parameter.	One of
`LDAP`	String	Specify the LDAP path of the Group Policy Object (GPO). This parameter must be used in conjunction with `-Domain`.	Yes if `Domain` is used
`XML`	Boolean	Return the Privilege Management settings as an XML formatted string.	

Return Values

By default, `Get-DefendpointSettings` returns a **DefendpointSettings** object. Using the `-XML` parameter returns the Privilege Management policy as an XML formatted string. The cmdlet returns errors if there are any.

Examples

Get the local Privilege Management Settings as a DefendpontSettings object

```
$settings = Get-DefendpointSettings -LocalPolicy
```

The above example outputs:

```
Version            : 5.2.102.0
ID                 : 1e71ef8e-4ffc-4769-9a5b-11ea102b0f8e
ConfigRevision     : 510
ApplicationGroups : {cmd}
ContentGroups      : {}
URLGroups          : {}
Tokens             : {}
GlobalOptionsSets : {}
Files              : Avecto.Defendpoint.Settings.FileList
Messages           : {Block Message, Allow Message (Elevate)}
Policies           : {New Workstyle}
Licenses           : {Avecto.Defendpoint.Settings.License}
RegistryValues     : {}
```

Save the local Privilege Management Settings into an XML file

```
Get-DefendpointSettings -LocalPolicy -XML > C:/Users/admin/Desktop/DefendpointSettings.xml
```

The above example does not output anything to the terminal. A file called `DefendpointSettings.xml` is created at `C:/Users/admin/Desktop`.

Assign Local Privilege Management Settings to a Variable to Work with PowerShell

```
$settings = Get-DefendpointSettings -LocalPolicy
```

Get the Privilege Management Settings of the Group Policy Object Specified by an LDAP Path

```
Get-DefendpointSettings -Domain -LDAP "LDAP://DC13.Acme.com/CN=
{31B2F340-016D-11D2-945F-00C04FB984F9},CN=Policies,CN=System,DC=Acme,DC=com"
```

```
The above example outputs:
Version            : 5.2.102.0
ID                 : 1e71ef8e-4ffc-4769-9a5b-11ea102b0f8e
ConfigRevision     : 510
ApplicationGroups  : {cmd}
ContentGroups      : {}
URLGroups          : {}
Tokens             : {}
GlobalOptionsSets  : {}
Files              : Avecto.Defendpoint.Settings.FileList
Messages           : {Block Message, Allow Message (Elevate)}
Policies           : {New Workstyle}
Licenses           : {Avecto.Defendpoint.Settings.License}
RegistryValues     : {}
```

Update the Privilege Management for Windows Settings: Set-DefendpointSettings

`Set-DefendpointSettings` saves Privilege Management for Windows settings to either a local Group Policy, a local file, or a specified Group Policy Object (GPO).

Syntax

```
Set-DefendpointSettings [-SettingsObject 'path/to/object'] [-LocalPolicy -XML 'path/to/file'] [-
UserPolicy] [-Merge 'path/to/file']
```

Description

The `Set-DefendpointSettings` cmdlet takes an XML file or Privilege Management for Windows settings object as input and saves it to either a local file, a local Group Policy, or a Group Policy Object (GPO). By default, this function overwrites the existing Privilege Management for Windows settings at the target location unless the `-Merge` parameter is used.

Parameter	Type	Description	Required
`SettingsObject`	Privilege Management Configuration Object	Supply the **DefendpointSettings** object that should be used as input. It can be obtained from `Get-DefendpointSettings`.	Yes
`-LocalPolicy`	Boolean	Set a local policy file as the Privilege Management for Windows settings. This only works with XML files, and it must be used in conjunction with the `-XML` parameter along with the full path to a Privilege Management settings XML file.	
`UserPolicy`	Boolean	Update the user policy. If not set, the machine policy is updated instead.	
`Merge`	Boolean	Merge the input settings with the target file. If this parameter is not set, the target file is overwritten.	
`LocalFile`	String	Save the Privilege Management for Windows settings to a local file. This argument defaults to `%PROGRAMDATA%\Avecto\Privilege Guard\PrivilegeGuardConfig.xml` if `-FileLocation` is not used.	
`TapConfigPath`	String	Define the file save destination. If not set, the file is saved to the local Privilege Management for Windows settings file destination: `%PROGRAMDATA%\Avecto\Privilege Guard\PrivilegeGuardConfig.xml`.	
`Domain`	Boolean	Save to a Group Policy Object (GPO). This is used in conjunction with the `-LDAP` parameter.	
`LDAP`	String	The LDAP path of the GPO. For example: `LDAP://DC13.Acme.com/CN={31B2F340-016D-11D2-945D-00D04CB984F9},CN=Policies,CN=System,DC=Acme,DC=com`	Yes when `-Domain` is supplied
`XML`	String	The path of a Privilege Management for Windows settings XML file that should be used as input.	

Return Values

`Set-DefendpointSettings` returns errors if there are any. If not, the function does not return anything.

Examples

Add License to Privilege Management Settings Configuration

> ***Tip:*** *Although this example is not part of the API, it is useful to know in this context.*

```
$PGLicence = "YOUR_LICENCE_HERE"
$PGConfig = Get-DefendpointSettings -LocalFile
$PGLicence = New-Object Avecto.Defendpoint.Settings.License
$PGLicence.Code = "$PGLicense"
$PGConfig.Licenses.Add($PGLicence)
Set-DefendpointSettings -SettingsObject $PGConfig -LocalFile
```

Set the local Privilege Management for Windows Settings from an XML File

```
Set-DefendpointSettings -LocalPolicy -XML C:/Users/admin/Desktop/PrivilegeGuardConfig.xml
```

The above example does not output anything to the terminal.

Merge a Privilege Management Config with the Domain Machine Policy Privilege Management Config

```
# Set the licence and LDAP.
$Ldap = "LDAP://DC13.Acme.com/CN={31B2F340-016D-11D2-945F-
>> 00C04FB984F9},CN=Policies,CN=System,DC=Acme,DC=com"

# Get the local PG Config file.
$PGConfig = Get-DefendpointSettings -Domain -LDAP $Ldap

# Create a new license object.
$PGLicence = New-Object Avecto.Defendpoint.Settings.License
$PGLicence.Code = "$PGLicense"

# Add the license to the defendpoint config.
$PGConfig.Licenses.Add($PGLicence)

# Merge the Defendpoint config with the existing Domain Machine policy Defendpoint config.
Set-DefendpointSettings -SettingsObject $PGConfig -Domain -Ldap $Ldap
```

The above example does not output anything to the terminal.

Example Use Cases

Introduction

In the following section, we review a few typical use cases. They all assume that your config is a local file in the default location and that you have already loaded the modules required.

For more information, please see "Privilege Management Powershell API Reference Guide" on page 3.

When adapting these examples to your use case, it is useful to reference PowerShell API.chm which resides in the PowerShell directory. This is usually located at:

```
C:/Program Files/Avecto/Privilege Guard Client/PowerShell
```

Application Groups

Add a New Application Definition (FileName contains) to an Existing Application Group with a New Application Definition

```
# Get settings
$PGConfig = Get-DefendpointSettings -LocalFile

# Find target application group
$TargetAppGroup = $PGConfig.ApplicationGroups | Where-Object {$_.name -eq
'YourApplicationGroupName'}
# Create an empty application definition$PGApp = new-object Avecto.Defendpoint.Settings.Application
$PGConfig

# Populate the things you want to$PGApp.Description = "Microsoft Calculator"$PGApp.Type =
[Avecto.Defendpoint.Settings.ApplicationType]::Executable$PGApp.CheckFileName = 1 # 0 = Disabled 1 =
Enabled$PGApp.FileName = "Calc.exe"$PGApp.FileNameStringMatchType = 2 # 2 = Contains (see
StringMatchType in PowerShell API.chm)$PGApp.OpenDlgDropRights = 1
# Add the application definition to the target application group$TargetAppGroup.Applications.Add
($PGApp)

# Save the settingsSet-DefendpointSettings -SettingsObject $PGConfig -LocalFile
```

Add a new Application Definition to an Existing Application Group with Get-DefendpointFileInformation

```
# Get settings$PGConfig = Get-DefendpointSettings -LocalFile

# Find target application group$TargetAppGroup = $PGConfig.ApplicationGroups | Where-Object {$_.name
-eq 'YourApplicationGroupName'}

# Get the details of the file(s) you want to match on$PGApp1 = Get-DefendpointFileInformation -Path
'C:\Windows\System32\cmd.exe'
# Add the list of application definitions to the target app
```

SALES: www.beyondtrust.com/contact SUPPORT: www.beyondtrust.com/support DOCUMENTATION: www.beyondtrust.com/docs 12
©2003-2019 BeyondTrust Corporation. All Rights Reserved. Other trademarks identified on this page are owned by their respective owners. BeyondTrust Corporation is not a chartered bank or trust company, or depository institution. It is not authorized to accept deposits or trust accounts and is not licensed or regulated by any state or federal banking authority. TC: 6/11/2019

```
group$TargetAppGroup.Applications.AddRange($PGApp1)

# Save the settingsSet-DefendpointSettings -SettingsObject $PGConfig -LocalFile
```

Account Filters

Add an Account to an Existing Account Filter

```
# Get the local settings file
$PGConfig = Get-DefendpointSettings -LocalFile

# Find the workstyle you want to work with (workstyles are known as policies here)
$TargetWorkstyle = $PGConfig.Policies | Where-Object {$_.name -eq 'YourWorkstyleName'}

# Get first account filter in list
$TargetAccountFilterCollection = $TargetWorkstyle.Filters.AccountsFilter[0]

# Create an account object and populate it’s values
$Account = New-Object Avecto.Defendpoint.Settings.Account
$Account.Name = Get-WmiObject win32_useraccount | Where-Object {$_.Name -eq 'AccountName' -and
$_.Domain -eq 'DomainName'} | % {return $_.Caption}
$Account.SID = Get-WmiObject win32_useraccount | Where-Object {$_.Name -eq 'AccountName' -and
$_.Domain -eq ‘DomainName’} | % {return $_.SID}

# Add new account to the filter collection
$TargetAccountFilterCollection.Accounts.Add($Account)

# Save Settings
Set-DefendpointSettings -SettingsObject $PGConfig -LocalFile
```