BeyondTrust

Privilege Management 23.9
API Guide

TC:10/30/2023

Table of Contents

Privilege Management Powershell API Reference Guide

Use the PowerShell API to configure Privilege Management for Windows. The API can be used to integrate with external systems and also 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. Use the **Get-Help <cmdlet name>** command to access syntax help.

The types properties and values are documented in the **PowerShell API.chm** help file located in the PowerShell directory: **C:/Program Files/Avecto/Privilege Guard Client/PowerShell**.

IMPORTANT!

To maintain backwards compatibility, the PowerShell cmdlets are not renamed along with the product. Therefore, many of the objects and cmdlets reference the former product name Defendpoint.

The examples in this guide 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 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

Command: Get-DefendpointFileInformation

Description

Use **Get-DefendpointFileInformation** to return properties Privilege Management for Windows can use as matching criteria for files and applications such as **Publisher**, **ProductName**, and **FileName**.

> ***Note:*** *This cmdlet supports Common Parameters. For more information, please see about_CommonParameters at https://learn.microsoft.com/en-us/previous-versions/dd315352(v=technet.10).*

Syntax

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

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 set in 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
Content	File Resource

Input String	File Type
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

The command **Get-DefendpointFileInformation** returns a list of application definitions.

Example: *Get All Executables in a Specified Path and Reference the FileName of the First*

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

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

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

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

```
$Files.Publisher
```

Example: *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

Command: Get-DefendpointSettings

Description

Use the **Get-DefendpointSettings** cmdlet to retrieve 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.

After the settings are in a PowerShell session or script, you can update relevant sections, and then write them back using **Set-DefendpointSettings**.

> ***Note:*** *This cmdlet supports Common Parameters. For more information, please see about_CommonParameters at https://learn.microsoft.com/en-us/previous-versions/dd315352(v=technet.10).*

Syntax

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

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** . Use the **-FileLocation** parameter to set an alternate file.	One of
FileLocation	String	Set 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

Parameter	Type	Description	Required
LDAP	String	Specify the LDAP path of the Group Policy Object (GPO). This parameter must be used 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. Use the **-XML** parameter to return the Privilege Management policy as an XML formatted string. The cmdlet returns errors if there are any.

Example: *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   : {}
```

Example: *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**.*

Example: *Assign Local Privilege Management Settings to a Variable to Work with PowerShell*

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

Example: *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

Command: Set-DefendpointSettings

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, **Set-DefendpointSettings** overwrites the existing Privilege Management for Windows settings at the target location unless the **-Merge** parameter is used.

Syntax

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

Parameters

Parameter	Type	Description	Required
SettingsObject	Privilege Management Configuration Object	Supply the **DefendpointSettings** object to use as input. It can be obtained from **Get-DefendpointSettings**.	Yes
-LocalPolicy	Boolean	Set a local policy file as the Privilege Management for Windowssettings. This only works with XML files, and it must be used 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 with the **-LDAP** parameter.	

Parameter	Type	Description	Required
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.

Example: *Add License to Privilege Management Settings Configuration*

```
$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
```

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

Example: *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.

Example: *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.

Use Cases

You can customize the following use cases for your environment:

- Add a new application definition to an existing Application Group with a new application definition.
- Add a new application definition to an existing Application Group with **Get-DefendpointFileInformation**.
- Add an account to an existing Account filter.

The use cases assume that your config is a local file in the default location and that you already loaded the modules required.

When adapting the examples to your use case, reference the **PowerShell API.chm** help file located in the PowerShell directory: **C:/Program Files/Avecto/Privilege Guard Client/PowerShell**.

Application Groups

Example: *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 settings
Set-DefendpointSettings -SettingsObject $PGConfig -LocalFile
```

Example: *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 group
$TargetAppGroup.Applications.AddRange($PGApp1)

# Save the settings
Set-DefendpointSettings -SettingsObject $PGConfig -LocalFile
```

Account Filters

Example: *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
$TargetAccountFilterCollection.Accounts.WindowsAccounts.Add($Account)
# Save Settings
Set-DefendpointSettings -SettingsObject $PGConfig -LocalFile
```