

# PowerShell Getting Started Guide

## Classifier Administration

April 2021



---

## Copyright Terms and Conditions

---

Copyright Help/Systems LLC and its group of companies.

The content in this document is protected by the Copyright Laws of the United States of America and other countries worldwide. The unauthorized use and/or duplication of this material without express and written permission from HelpSystems is strictly prohibited. Excerpts and links may be used, provided that full and clear credit is given to HelpSystems with appropriate and specific direction to the original content. HelpSystems and its trademarks are properties of the HelpSystems group of companies. All other marks are property of their respective owners.

202104230252

**About Classifier Administration PowerShell ..... 4**

    About this guide ..... 4

    Licensing ..... 4

    Prerequisites ..... 5

**Creating a Configuration ..... 7**

**Testing your Configuration ..... 8**

**Naming Conventions ..... 9**

    Verbs ..... 9

        Common verbs ..... 9

        Data Verbs ..... 9

        Security Verbs ..... 10

    Parameters ..... 10

    Tab Completion, Parameter Completion, and IntelliSense ..... 11

    Pipelining ..... 12

**Selectors, Markings, Policies and Help Cmdlets ..... 13**

    Selectors ..... 13

    Marking Formats ..... 13

    Policies ..... 14

    Help ..... 14

# About Classifier Administration PowerShell

The Classifier Administration PowerShell module allows you to run the Administration Server using PowerShell commands. The module provides over two hundred cmdlets to fetch, create, modify or remove any aspect of the Classifier configuration.

Boldon James recommends the use of Microsoft Visual Studio Code (VSC), including the PowerShell plug-in when getting started with the Classifier Administration cmdlets. The VSC adds syntax colouring, tab completion, IntelliSense, visual debugging, and context-sensitive Help.

PowerShell scripts are installed when you install the Administration Server. See the Getting Started Guide for more information.

## About this guide

This document assumes:

- a single system hosts all three environments. In a live system, Boldon James recommends hosting the Classifier Administration Service on a separate system.
- authorised administrators use the PowerShell interface to connect to this service.
- a network location (or Active Directory) holds the published Configuration and installed Classifier Applications reference that location.

Classifier PowerShell supports a 'Publish Test Configuration' facility whereby the Administrator can publish a 'Test Configuration.' This document makes use of this 'Publish Test Configuration' mechanism to explore the effects of the Configuration on various Classifier-enabled applications (for example, Microsoft Word).

A suitable Microsoft Windows environment is required, along with the following:

**NOTE:** See the Classifier Administration Installation Guide and the Email & Office Classifier release notes for information about the supported versions of Windows and Microsoft Office.

For information on installing the Classifier Administration Server, see the Getting Started Guide.

## Licensing

A PowerShell license can be applied either on a per session basis or to Classifier configuration in the same way as other application licenses. Alternatively, the license can

be used in session variables so that the resulting Classifier configuration cannot use the PowerShell cmdlets.

In contrast, cmdlets that are used only for interrogation of the configuration do not require a licence. This means that any cmdlets that use the verb Get- do not require a licence.

At the start of each session, set a session variable as follows: Set-SessionLicence - LicenceFile 'C:\LicenceFiles\impCPA.lic'.

## Prerequisites

For a list of system requirements to run the Classifier Administration Server, see the Getting Started Guide.

To enable all the prerequisites for your platform, use one of the following PowerShell commands in an elevated command prompt/PowerShell session:

Operating System	PowerShell command
Windows 7 Windows 2008 R2	<pre>dism.exe /online /enable-feature /featurename:IIS-CommonHttpFeatures /featurename:IIS-WebServer /featurename:IIS-WebServerRole /featurename:IIS-StaticContent (recursive snippet) /featurename:IIS-HttpCompressionStatic /featurename:IIS-ASPNET /featurename:IIS-NetFxExtensibility /featurename:IIS-WindowsAuthentication /featurename:IIS-LoggingLibraries</pre>
Windows 8.1 Windows 10	<pre>dism.exe /online /enable-feature /all /featurename:IIS-ManagementConsole /featurename:IIS-ASPNET45 /featurename:IIS-DefaultDocument /featurename:IIS-ISAPIExtensions /featurename:IIS-ISAPIFilter /featurename:IIS-HttpErrors /featurename:IIS-NetFxExtensibility45 /featurename:IIS-RequestFiltering /featurename:NetFx4Extended-ASPNET45 /featurename:IIS-StaticContent /featurename:IIS-HttpCompressionDynamic /featurename:IIS-HttpCompressionStatic /featurename:IIS-WindowsAuthentication /featurename:IIS-LoggingLibraries</pre>

Operating System	PowerShell command
Windows Server 2012	dism.exe /online /enable-feature /all
Windows Server 2012 R2	/featurename:NetFx4Extended-ASPNET45
Windows Server 2016	/featurename:NetFx4 /featurename:IIS-NetFxExtensibility45
Windows Server 2019	/featurename:IIS- WebServerManagementTools /featurename:IIS-DefaultDocument /featurename:IIS-ASPNET45 /featurename:IIS-ISAPIExtensions /featurename:IIS-ISAPIFilter /featurename:IIS-RequestFiltering /featurename:IIS-StaticContent /featurename:IIS-HttpCompressionDynamic  /featurename:IIS-HttpCompressionStatic /featurename:IIS-HttpErrors /featurename:IIS- LoggingLibraries /featurename:IIS-WindowsAuthentication

# Creating a Configuration

This section describes how to create a configuration using PowerShell cmdlets.

**NOTE:** If you are using an existing configuration, use `Get-ServerConfiguration -Location FileSystem` to copy it into the Configuration Import Folder, then go to [Use test mode](#).

1. Use `Set-ServerConfiguration - Location FileSystem` to set the Configuration Import Folder if the Folder is other than the default `%programdata%\Baldon James\Config Import Folder`.
2. Use `Get-Module -ListAvailable -Refresh` to ensure all cmdlets are available.
3. Use `net start BoldeonJamesClassifierManagementWcfService.exe` to ensure the service is running.
4. Use `Add-Licence -FileName Location` to license your new label configuration as in the following example:

```
PS C:\Users\Administrator> Add-Licence -LicenceFile c:\Licenses\impCEM.lic

ProductName : Email Classifier
ProductCode : CEM
LicenceType : Full
ExpiryDate  : Perpetual
UserCount   : 0
LicenceLevel : Enterprise
Value       : {91, 76, 105, 99...}
Components  : {[ELB, Enhanced Labelling], [MSG, Office Mail]}
FileName    : impCEM.lic
```

5. Use `New-LabelConfiguration -Name 'new name' -TemplateName 'template name'` to create a new Label Configuration as in the following example:

```
PS C:\Users\Administrator> New-LabelConfiguration -Name 'Quick Start' -TemplateName 'Corporate Commercial'

Id                                     Name                SecurityPolicyId
--                                     -
f5c063a1-a663-4871-a078-7f09ff34ac08 Quick Start a6cc33f4-3a7a-48d6-a446-30d72d6e1b74
```

# Testing your Configuration

Classifier Administration provides a Test Mode Administration feature so you can preview the effects of the configuration on the end-user experience prior to publication of the configuration to a user community. For more information on the Test Mode Administration feature, see the Getting Started Guide.

1. Enter **Set-ConfigurationTestFolder -Path**, where path is the location of your test configuration. By default, this location is %programdata%\Baldon James\TestMode Folder.
2. Enter **Publish-TestConfiguration -TestName 'name of test configuration'**.
3. When you have finished your testing, enter **-RemoveLabelConfiguration -Name 'name of test configuration -DeleteLicenses:\$true** to remove your test configurations and any licenses in use.



# Naming Conventions

## Verbs

The Classifier Administration PowerShell Module follows Microsoft's guidelines for the verbs of cmdlet names. The following verbs appear in cmdlet names:

### Common verbs

Verb	Action
Add	Adds a resource to a container, or attaches an item to another item
Copy	Copies a resource to another name or to another container
Get	Specifies an action that retrieves a resource. See <i>Selectors, Markings, Policies and Help Cmdlets</i> for examples of Get.
Join	Combines resources into one resource.
Lock	Secures a resource. This verb is paired with Unlock.
New	Creates a resource.
Remove	Deletes a resource from a container.
Rename	Changes the name of a resource.
Set	Replaces data on an existing resource or creates a resource that contains some data
Unlock	Releases a locked resource. This verb is paired with Lock.

### Data Verbs

Verb	Action
ConvertFrom	Converts one primary type of input (the cmdlet noun indicates the input) to one or more supported output types.
ConvertTo	Converts from one or more types of input to a primary output type (the cmdlet noun indicates the output type).
Publish	Makes a resource available to others.
Restore	Sets a resource to a predefined state

## Security Verbs

Verb	Action
Block	Restricts access to a resource.
Grant	Allows access to a resource. This verb is paired with Revoke
Revoke	Specifies an action that does not allow access to a resource. This verb is paired with Grant.
Unprotect	Removes safeguards from a resource. This verb is paired with Unlock.

The nouns of cmdlet names refer to the entity that the action affects.

**NOTE:** When the entity is part of a Classifier policy, the noun will have a prefix of Policy. For example, the cmdlet `Get-SelectorValue` obtains a Classifier selector value from the Selector Library whereas `Get-PolicySelectorValue` obtains the Classifier selector value associated with a specific Classifier policy.

## Parameters

Use either a friendly name or a unique identifier (a GUID) parameter to describe the entity on which the action affects.

When the cmdlet requires the identification of multiple entities, then the entity type is used as a prefix to the parameters. For example, the cmdlet `Get-PolicySelectorValue` requires the identification of a Classifier policy, selector, and value using parameters (in this instance, `-PolicyName`, `-SelectorName` and `-SelectorValue`). Other parameters used can be `-PolicyId`, `-SelectorId` and `-ValueId`. In this case, you cannot mix parameters. For example, you cannot use `-PolicyName` with `-SelectorId`

In some cmdlets, dynamic parameters can be used and will require a specific set of parameters depending on the value supplied.

All cmdlets support the following general common parameters and risk-mitigation parameters

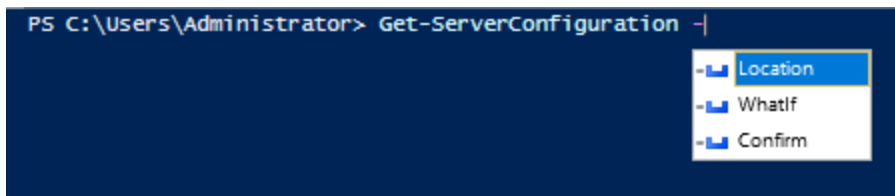
Parameter	Description
<code>-Debug</code>	Specifies whether programmer-level debugging messages are displayed
<code>-ErrorAction</code>	Specifies what action should take place when an error occurs
<code>-ErrorVariable</code>	Specifies the variable in which to place objects when an error occurs

-OutVariable	Specifies the variable in which to place all output objects generated by the cmdlet.
-OutBuffer	Defines the number of objects to store in the output buffer before any objects are passed down the pipeline.
-Verbose	Specifies whether the cmdlet writes explanatory messages that can be displayed at the command line
-WarningAction	Specifies what action should take place when the cmdlet writes a warning message
-WarningVariable	Save warning messages in the specified variable.
-Confirm	Specifies whether the cmdlet displays a prompt that asks if the user is sure that they want to continue.
-WhatIf	Specifies whether the cmdlet writes a message that describes the effects of running the cmdlet without actually performing any action.

## Tab Completion, Parameter Completion, and IntelliSense

Cmdlet name completion occurs as expected with PowerShell. For example, typing the partial cmdlet `get-ser` and then pressing the Tab key will auto-complete the cmdlet name. Pressing the Tab key subsequent times allows scrolling through every cmdlet that begins `get-ser`.

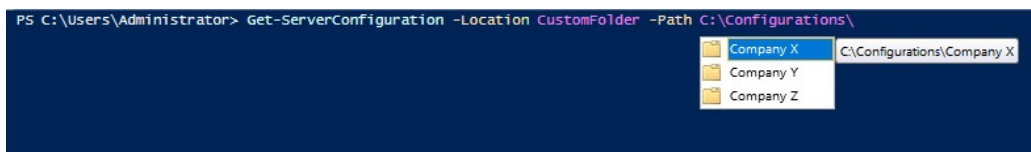
IntelliSense presents available parameters when entering the hyphen character after a cmdlet name. For example,



```
PS C:\Users\Administrator> Get-ServerConfiguration -|
```

The screenshot shows a PowerShell prompt where the command `Get-ServerConfiguration -|` has been entered. A dropdown menu is visible, listing three parameters: `-Location`, `-Whatif`, and `-Confirm`. The `-Location` parameter is currently selected and highlighted in blue.

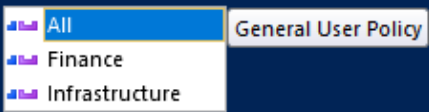
Parameter completion is available for most parameters. For example,



```
PS C:\Users\Administrator> Get-ServerConfiguration -Location CustomFolder -Path C:\Configurations\
```

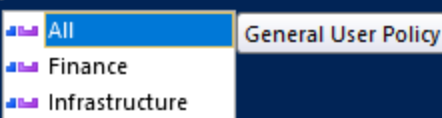
The screenshot shows a PowerShell prompt where the command `Get-ServerConfiguration -Location CustomFolder -Path C:\Configurations\` has been entered. A dropdown menu is visible, listing three folders: `Company X`, `Company Y`, and `Company Z`. The `Company X` folder is currently selected and highlighted in blue. The full path `C:\Configurations\Company X` is visible to the right of the dropdown.

```
PS C:\Users\Administrator> Get-Policy -Name
```



When a unique identifier describes the entity that the action affects, the parameter completion will show friendly names as a choice. When selecting a specific friendly name, the cmdlet completes the parameter value with the unique identifier; for example:

```
PS C:\Users\Administrator> Get-Policy -Id |
```



Selecting the value All provides:

```
PS C:\Users\Administrator> Get-Policy -Id 40428d09-d2c6-4505-af49-f023b1387dfa
```

If a parameter requires an array or list of values, separate the values using a comma, then tab completion can be used to view additional values.

## Pipelining

All cmdlets accept parameters from the pipeline by name. Generally, cmdlets accept the friendly name for the entity from the pipeline by value.

For example, the following cmdlets show bindings for the parameter -Name:

```
PS C:\Users\Administrator> Get-Policy -Name All
Name Description
-----
All General User Policy

PS C:\Users\Administrator> 'All' | Get-Policy
Name Description
-----
All General User Policy

PS C:\Users\Administrator> New-Object PSObject | Add-Member NoteProperty Name 'All' -PassThru | Get-Policy
Name Description
-----
All General User Policy
```

# Selectors, Markings, Policies and Help Cmdlets

You can use the following cmdlets to get selectors, markings, policies, and help.

## Selectors

Cmdlet	Description
Get-Selector	to view all selectors in the selector library
Get-Selector -Name <name>	to view a specific selector; for example Get-Selector - SelectorName Classification to view the 'Classification' selector
Get-SelectorValue - SelectorName <name>	to view all the values for a specific selector; for example Get-SelectorValue - SelectorName Classification to view all values for the 'Classification' selector
Get-SelectorValue - SelectorName <name> - ValueName <value>	to view a specific value for a specific selector; for example Get-SelectorValue - SelectorName Classification -ValueName Non-Business to view the 'Non-Business' value for the 'Classification' selector
Get -Selector -Name <name>   Format-List -Property *	pipe the cmdlet to view all properties on the specific selector; for example, Get -Selector -Name Classification   Format-List - Property * to view all the properties on the 'Classification' selector

## Marking Formats

Cmdlet	Description
Get-MarkingFormat	to view all marking formats in the marking format library
Get-MarkingFormat - Name <name>	to view a specific marking format; for example Get-MarkingFormat - Name FLOT to view the first line of text marking format

# Policies

Cmdlet	Description
Get-Policy	to view all policies for the configuration
Get-Policy -Name <name>   Format-List Property *	to view all the properties on a specific policy; for example Get -Policy -Name MyCompanyClassification   Format-List Property *
Get-PolicySelector -PolicyName <name>	to view all selectors associated with a specific policy; for example Get-PolicySelector -PolicyName MyCompanyClassification
Get-PolicySelectorValue -PolicyName <name>-SelectorName <name>	to view a specific selector associated with a specific policy; for example Get-PolicySelectorValue -PolicyName MyCompanyClassification -SelectorName Classification

# Help

The Classifier Administration PowerShell Module and the associated cmdlets support the Get-Help cmdlet.

Cmdlet	Description
Get-Help <cmdlet>	to view help on a specific cmdlet; for example Get -Help Get -ServerConfiguration to view help on the Get -ServerConfiguration cmdlet
Get-Help <cmdlet> -examples	to view examples on a specific cmdlet; for example Get -Help Get -ServerConfiguration -examples to view examples on the Get -ServerConfiguration cmdlet
Get-Help <cmdlet> -detailed	to view detailed information on a specific cmdlet; for example Get -Help Get -ServerConfiguration -detailed to view detailed information on the Get -ServerConfiguration cmdlet
Get-Help <cmdlet> -full	to view technical information on a specific cmdlet; for example Get -Help Get -ServerConfiguration -full to view technical information on the Get -ServerConfiguration cmdlet