PANOS PS Module Help

# New-PANOSConnection

## Synopsis

Creates an object which contains the necessary information for establishing a connection to a PAN-OS firewall. Most Cmdlets in this module require this object to do their work.

## Detailed Description

The New-PANOSConnection Cmdlet creates an object that represents a connection to a PANOS firewall, and includes properties like: hostname, vsys and access token. Practically all Cmdlets in this module have a mandatory parameter called Connection, which takes the object produced by New-PANOSConnection Cmdlet. Since Connection objects stores API Access Token, care must be taken when storing these objects, see the second example which demonstrates the use of Export-Clixml to securely store the Connection object to a file.

## Parameters

|  |  |
| --- | --- |
| HostName | Firewall’s network name as it appears in the subject field of the SSL certificate installed on the firewall. |
| AccessToken | XML API Access Token. See PAN-OS 6.1 XML API Reference (section 2.1 – Key Generation) for details on how to generate such token. |
| Vsys | Name of the Vsys to which the connection is bound. If you need to execute commands against multiple Vsys then you will need to create a separate Connection object – one per Vsys. |

## Examples

### Example 1

This example shows how to build a connection object and pass it to another Cmdlet.

Note that the API token has to be converted to SecureString, which is then passed to the New-PANOSConnection Cmdlet.

$accessToken = ConvertTo-SecureString "LUFRPT01b…W6K1BqZ0J1WXJxTGU2b3ptV2tuSllBcFdKYz0=" -AsPlainText -Force

$connection = New-PANOSConnection -HostName 'pa.corp.com' -Vsys 'vsys1' -AccessToken $accessToken

Get-PANOSAddress -Name 'dc01' -FromCandidateConfig -Connection $connection

### Example 2

This example shows how to securely store the API token.

The first Cmdlet converts the API token to SecureString and passes the output to Export-Clixml Cmdlet which de-serializes SecureString to disk and encrypts the file using DPAPI. Note, only the user that executed this command will be able to decrypt this file. The advantage of this approach is that once the Connection object is stored in the encrypted form on disk you no longer need to supply the Access Token in clear text.

ConvertTo-SecureString "LUFRPT01b…W6K1BqZ0J1WXJxTGU2b3ptV2tuSllBcFdKYz0=" -AsPlainText -Force | Export-Clixml C:\PSScripts\panosAccessToken

$connection = New-PANOSConnection -HostName 'pa.corp.com' -Vsys 'vsys1' -AccessToken (Import-CliXml c:\PSScripts\panosAccessToken)

Get-PANOSAddress -Name 'dc01' -FromCandidateConfig -Connection $connection

# Get-PANOSTrafficLog

## Synopsis

Gets PANOS traffic log entries.

## Detailed Description

The objective of this Cmdlet is to provide parity with the functionality provided by the PANOS Web Management Interface in viewing, querying and exporting traffic log entries.

## Parameters

|  |  |
| --- | --- |
| Connection | Object that represents a connection to PANOS firewall, see New-PANOSConnection help for details |
| Query | This parameter expects a value that is equivalent to a filter supplied in PANOS Web Interface. For example, the filter below will limit the output of the Cmdlet to only log entries recorded after a specific date/time value and where destination matches the supplied IP address.  ( receive\_time geq '2015/04/18 10:16:44' ) and ( addr.dst in 23.59.190.121 ).  It is recommended to test the value of this parameter in the Web Interface to ensure its correctness. |
| Page | Since the number of entries in PANOS traffic log may easily span millions, PANOS (in order to conserve CPU and memory resources) will only return the first 5000 log entries that satisfy the query condition. In most scenarios, this is a very reasonable behavior, since we should always try to supply the most restrictive Query to limit the output only to the relevant entries. In some scenario (ex. exporting traffic log to a CSV file) you may need to ensure that all log entries that satisfy our Query parameter are returned. To accommodate this scenario supply the Page switch. When the Page switch is supplied this Cmdlet will detect a condition where only the first 5000 records were returned, and will repeatedly re-formulate the Query so that the remaining items are returned until all records that satisfy the original query are returned.  Note: use this switch with caution especially when the supplied Query parameter is broad, since this may lead to a very long execution time of the CmdLet, and consequently high consumption of the management CPU on PANOS. |
| Delay | PANOS uses the asynchronous model when processing traffic log queries. In other words, once a query is received, PANOS immediately returns a job Id to the caller and commences the work involved in collecting the requested data. The caller then makes another request to retrieve the log entries gathered by the job (identified by the ID). The Cmdlet handles this for you behind the scene. However, PANOS requires some time to collect the log entries that match the Query parameter, hence the Cmdlet must allow for a delay between the request and the consumption of the log data.  The value of the Delay is something that you will need to experiment with in your environment, since this value varies depending on the model of the firewall (higher end models have faster CPUs and seem to respond to queries faster). The values that we observe varied from 1 to 4 seconds. Note that the Cmdlet, before consuming the data, will check the status of the job, and if the job status indicates that it is still in progress an exception will be thrown, thus giving you an indication to increase the value for this parameter. |
| ResolveHostName | When this switch is supplied, the Cmdlet will attempt to perform a DNS lookup for the IP addresses found in the Source and Destination fields of log entries.  The outcome of the DNS queries is placed into SourceHostName and DestinationHostName fields of the returned log entries. If DNS query fails, the IP address of the query will be placed into the above mentioned fields.  This switch, when used with queries that return a large number of entries, will significantly increase the execution time of the Cmdlet. |

## Examples

### Example 1

These commands retrieve traffic log entries that match the filter supplied by the $query variable.

The Get-PANOSTraffic Cmldet will insert a four-second delay between the request and the consumption of the traffic log data.

Since the Page switch is not supplied this will return at most 5000 records.

$connection = New-PANOSConnection -HostName 'firewall1.corp.net' -Vsys 'vsys1' -AccessToken (Import-CliXml c:\PSScripts\panosAccessToken.txt)

$query = "( receive\_time leq '2015/04/20 10:16:44' ) and ( receive\_time geq '2015/04/20 09:16:44' ) and ( addr.dst in 23.59.190.121 ) and ( action eq deny )"

Get-PANOSTrafficLog -Query $query -Connection $connection -Delay 4 | ft

### Example 2

These commands illustrate how to use the Get-PANOSTrafficLog Cmdlet for the purposes of extracting large quantities of log entries to an external data store (ex. CSV) file.

Note the use of the Page switch, ensures that all records matching the query will be returned.

$connection = New-PANOSConnection -HostName 'firewall1.corp.net' -Vsys 'vsys1' -AccessToken (Import-CliXml c:\PSScripts\panosAccessToken.txt)

$query = "(Rule eq 'web-farm' )"

Get-PANOSTrafficLog -Query $query -Page -Connection $connection -Delay 4 |

convertto-csv -NoTypeInformation -Delimiter "," | % {$\_ -replace '"',''} |

Out-File -FilePath C:\Data\webfarmRule.csv -Force -Encoding ascii

### Example 3

These commands show how to merge traffic log data from multiple firewalls by supplying an array of Connection object to the Connection parameter. This is particularly useful in a load-balanced scenario.

Note, that the output of the Ge-PANOSTrafficLog Cmdlet – the log entry object - contains the serial number of the Firewall on which the entry was recorded, this will allow you to trace the log entry to the firewall that produced it.

$connectionFw1 = New-PANOSConnection -HostName 'firewall1.it.msft.net' -Vsys 'vsys3' -AccessToken (Import-CliXml c:\PSScripts\panosAccessToken.txt)

$connectionFw2 = New-PANOSConnection -HostName 'firewall2.it.msft.net' -Vsys 'vsys3' -AccessToken (Import-CliXml c:\PSScripts\panosAccessToken.txt)

$query = "( addr.dst in 23.59.190.121 ) and ( action eq deny )"

Get-PANOSTrafficLog -Query $query -Connection $connectionFw1, $connectionFw2 -Delay 4 -Verbose | Select-Object -Property ReceiveTime, Source, Destination, Rule, Action, App | ft

# Get-PANOSAddress

## Synopsis

Gets Address objects from PANOS.

## Parameters

|  |  |
| --- | --- |
| Connection | Object that represents a connection to PANOS firewall, see New-PANOSConnection help for details |
| Name | Name of the address object to get. |
| FirewallObject | An object of type PANOS.AddressObject. When supplied, this Cmdlet will search for an Address object on PANOS, which matches both the name and the IP address of the supplied object. See example 3 for more details. |
| FromCandidateConfig | When this switch is provided the search will be conducted against the Candidate Config, otherwise (default) the Running Config will be searched. |

## Examples

### Example 1

These commands will retrieve all Address objects from the Candidate Config.

$connection = New-PANOSConnection -HostName 'pa.corp.com' -Vsys 'vsys1' -AccessToken (Import-CliXml c:\PSScripts\panosAccessToken)

Get-PANOSAddress -Connection $connection -FromCandidateConfig

### Example 2

These commands will retrieve a single Address objects (from the Running Config), identified by the name supplied in the Name parameter. Null will be return if no Address object with such name exists on the firewall.

$connection = New-PANOSConnection -HostName 'pa.corp.com' -Vsys 'vsys1' -AccessToken (Import-CliXml c:\PSScripts\panosAccessToken)

Get-PANOSAddress -Connection $connection -Name web01

### Example 3

These commands will retrieve a single Address objects (from the Running Config), identical to the object supplied in the FirewallObject parameter. For the match to be considered successful, both the name and the IP address must match.

Null will be return if no Address object exists on the firewall.

$connection = New-PANOSConnection -HostName 'pa.corp.com' -Vsys 'vsys1' -AccessToken (Import-CliXml c:\PSScripts\panosAccessToken)

$addressToSearch = New-Object -TypeName PANOS.AddressObject -ArgumentList "web01", ([System.Net.IPAddress]::Parse("25.1.1.4"))

Get-PANOSAddress -Connection $connection -FirewallObject $addressToSearch

### Example 4

These commands will retrieve all Address objects with a specified IP address.

$connection = New-PANOSConnection -HostName 'pa.corp.com' -Vsys 'vsys1' -AccessToken (Import-CliXml c:\PSScripts\panosAccessToken)

Get-PANOSAddress -Connection $connection | Where { $\_.Address -eq '25.1.1.4' }

# Add-PANOSAddress

## Synopsis

Adds new Address objects to the Candidate Config.

## Parameters

|  |  |
| --- | --- |
| Connection | Object that represents a connection to PANOS firewall, see New-PANOSConnection help for details |
| Name | Name to be assigned to the new address. |
| IpAddress | IP address to be assigned to the new address object. |
| Description | Description of the address object. |
| PanosAddress | An object of type PANOS.AddressObject. Typically, this parameter is used when accepting an Address object from the Pipeline, where the previous command returns PANOS.AddressObject (ex. Get-PANOSAddress). |

## Examples

### Example 1

These commands create a new Address object.

$connection = New-PANOSConnection -HostName 'palab.corp.com' -Vsys 'vsys1' -AccessToken (Import-CliXml c:\PSScripts\panosAccessToken)

Add-PANOSAddress -Connection $connection -Name "web01" -IpAddress '10.10.10.12' -Description "Web Server 01"

### Example 2

These commands create a new Address object and demonstrate the option of passing a PANOS.Address object via Pipeline to the PanosAddress parameter.

$connection = New-PANOSConnection -HostName 'palab.corp.com' -Vsys 'vsys1' -AccessToken (Import-CliXml c:\PSScripts\panosAccessToken)

New-Object -TypeName PANOS.AddressObject -ArgumentList "web02", ([System.Net.IPAddress]::Parse("25.1.1.14")) | Add-PANOSAddress -Connection $connection

### Example 3

These commands create a new Address object and then retrieve the newly created object from the Candidate Config (note the use of the PassThru switch to pass the newly created object to the pipeline.

$connection = New-PANOSConnection -HostName 'palab.corp.com' -Vsys 'vsys1' -AccessToken (Import-CliXml c:\PSScripts\panosAccessToken)

Add-PANOSAddress -Connection $connection -Name "web03" -IpAddress '10.10.10.14' -Description "Web Server 03" -PassThru | Get-PANOSAddress -Connection $connection -FromCandidateConfig

# Remove-PANOSAddress

## Synopsis

Removes Address objects.

## Parameters

|  |  |
| --- | --- |
| Connection | Object that represents a connection to PANOS firewall, see New-PANOSConnection help for details |
| Name | Name of the Address object to remove. |
| PanosAddress | An object of type PANOS.AddressObject. Typically, this parameter is used when accepting an Address object from the Pipeline, where the previous command returns PANOS.AddressObject (ex. Get-PANOSAddress). |

## Examples

### Example 1

These commands remove an Address object identified by name.

$connection = New-PANOSConnection -HostName 'pa.corp.com' -Vsys 'vsys1' -AccessToken (Import-CliXml c:\PSScripts\panosAccessToken)

Remove-PANOSAddress -Connection $connection -Name web12

### Example 2

These commands remove an Address object which was initially retrieved via the Get-PANOSAddress Cmdlet and then passed via the pipeline to the PanosAddress parameter of the Remove-PANOSAddress.

$connection = New-PANOSConnection -HostName 'pa.corp.com' -Vsys 'vsys1' -AccessToken (Import-CliXml c:\PSScripts\panosAccessToken)

Get-PANOSAddress -Connection $connection -Name web12 | Remove-PANOSAddress -Connection $connection

### Example 3

These commands remove all Address objects which contain a specific IP address. The first command in the pipeline retrieves all addresses, then the Where-Object Cmdlet filters-out all addresses that do not contain the supplied IP address. Finally, the matched addresses are passed to the Remove-PANOSAddress for removal.

$connection = New-PANOSConnection -HostName 'pa.corp.com' -Vsys 'vsys1' -AccessToken (Import-CliXml c:\PSScripts\panosAccessToken)

Get-PANOSAddress -Connection $connection -FromCandidateConfig | where { $\_.Address -eq '10.160.49.218' } | Remove-PANOSAddress -Connection $connection