

# PAN-OS-PHP User Guide

Palo Alto Networks  
3000 Tannery Way  
Santa Clara, CA 95054  
[www.paloaltonetworks.com](http://www.paloaltonetworks.com)

Document Author	Martin Smid
Document Owner	TBC
Department/Team	Security
Created	11-August-2021
Status	Draft - (last update 12-Sep 2022)

# Table of Contents

Table of Contents .....	2
Version Control .....	4
Introduction .....	5
Assumptions .....	5
Audience .....	5
List of Abbreviations .....	5
Download & Install .....	6
MAC OS .....	6
Windows OS .....	7
Update .....	8
Basic Use .....	9
Mandatory Arguments .....	9
Optional Arguments .....	11
Autocompletion .....	13
Type .....	14
Rule .....	15
Filters .....	15
Address .....	22
Filters .....	22
Actions .....	22
XML-issue .....	23
Actions .....	25
ExportToExcel .....	25
Additional Features .....	26
Display Configured Admins .....	26
Display Connected Admins .....	26
Disconnect Connected Admins .....	26
Display Panorama Configuration overwritten on Firewall .....	27
Configuration Validation .....	27
JSON Format Output .....	28
Display Shadow Rules .....	29
SET Command Output .....	30
Configuration Import into Panorama .....	31
Bulk Rule configuration .....	32
Use Cases and Examples .....	33

Rule Audit (show rules applicable to traffic X) .....	33
Export security rules with a specific tag .....	33
Export NAT rules .....	34
Export rules with expired schedules and schedules expiring in 20 days .....	34
Config size reduction/cleanup tasks after multiple migrations .....	34
Remove Unused objects .....	35
Merge objects with duplicate values .....	36
Move objects between Device Groups .....	37
Features in Development .....	38
PAN-OS-PHP Docker API .....	38

# Version Control

<b>Document control</b>			
<b>Security level</b>	Confidential (can be shared with customers on as needed basis)		
<b>Company</b>			
<b>Department</b>	Security		
<b>Author</b>	Martin Smid, Palo Alto Networks		
<b>Reviewed by</b>		<b>Date</b>	
<b>Approved by</b>		<b>Date</b>	
<b>Version</b>	<b>Date</b>	<b>Change/Comment</b>	<b>By</b>

# Introduction

The purpose of this document is to provide detailed steps on how to perform some of the most common actions and tasks using the PAN-OS-PHP tool. The aim is not to cover every single possible command or functionality.

## Assumptions

It is assumed that the users of the tool have a good understanding of PAN-OS functionality and basic understanding of filters and syntaxes.

## Audience

This document is created primarily for firewall admins and SMEs.

## List of Abbreviations

3DES	Triple DES
AES	Advanced Encryption Standard
CIDR	Classless Inter-Domain Routing
CLI	Command Line Interface
DES	Data Encryption Standard
DG	Device Group
DH	Diffie Hellman
IKE	Internet Key Exchange
IP	Internet Protocol
IPSec	Internet Protocol Security
ISAKMP	Internet Security Association Key Management Protocol
GUI	Graphical User Interface
NAT	Network Address Translation
OS	Operating System
PA	Palo Alto
PAN	Palo Alto Networks
PBF	Policy Based Forwarding (aka Policy Based Routing)
PHP	Hypertext Preprocessor
RSA	Rivest Shamir Adleman
SHA	Secure Hash
SME	Subject Matter Expert
SSH	Secure Shell
SRE	Symmetric Return Enforcement
URL	Uniform Resource Locator
VPN	Virtual Private Network

## Download & Install

- If the steps below are not clear, or further details are required, please follow the installation flowchart available here <https://github.com/PaloAltoNetworks/pan-os-php/blob/main/READMEdocker.md>
- **Make sure that the path where you save the final installation of PAN-OS-PHP tool doesn't have any special characters**

## MAC OS

- 1) Install Docker Desktop (reboot is needed)
- 2) Start Docker Desktop
- 3) Open MAC Terminal
- 4) Navigate to the folder with your offline configuration using the CD command
- 5) Run the following command in Terminal

```
docker run -v ${PWD}:/share -it swaschkut/pan-os-php:latest
```
- 6) Docker container instance starts
- 7) Check the current version of the tool by running the following command

```
pan-os-php version
```
- 8) Your files should be available in the /share folder inside the Container

## Windows OS

- 1) Install Docker Desktop (reboot is needed)
- 2) Start Docker Desktop
  - a) If Docker complains about WSL not being installed properly, follow the steps below
  - b) If Docker doesn't complain about WSL, please skip to step 5
- 3) Browse to <https://docs.microsoft.com/en-gb/windows/wsl/install-manual>
- 4) Follow steps 2 - 5 on the Microsoft support page to install WSL
- 5) Open PowerShell
- 6) Navigate to the folder with your offline configuration using the CD command
- 7) Run the following command in PowerShell

```
docker run -v ${PWD}:/share -it swaschut/pan-os-php:latest
```
- 8) Docker container instance starts
- 9) Check the current version of the tool by running the following command

```
pan-os-php version
```
- 10) Your files should be available in the /share folder inside the Container

# Update

- In order to keep the Docker image updated, make sure you run the following command.
  - `docker pull swaschkut/pan-os-php:latest`

## Example of an image getting updated

```
PS C:\> docker pull swaschkut/pan-os-php:latest
latest: Pulling from swaschkut/pan-os-php
08c01a0ec47e: Already exists
7f27e7bad0b7: Pull complete
edbc9d21c4b0: Pull complete
6010f3624ecd: Pull complete
8e02b1c58ed1: Pull complete
607be555494c: Pull complete
d2500a199346: Pull complete
5ef37324a6e2: Pull complete
cd1dea69e80: Pull complete
511dfd12c6f4: Pull complete
ff3c92cbe744: Pull complete
5035307eac0e: Pull complete
cf376880b755: Pull complete
02f4c359eda8: Pull complete
ad8d147ea269: Pull complete
1c5f2b1c8fe5: Pull complete
a4fde1679cb5: Pull complete
Digest: sha256:b8bdc7bd03ba207a8fd77a1371c7b8aef6841b053a1fe671d1a113a26bd56242
Status: Downloaded newer image for swaschkut/pan-os-php:latest
docker.io/swaschkut/pan-os-php:latest
```

## Example of an image which is already fully up-to-date

```
PS C:\> docker pull swaschkut/pan-os-php:latest
latest: Pulling from swaschkut/pan-os-php
Digest: sha256:b8bdc7bd03ba207a8fd77a1371c7b8aef6841b053a1fe671d1a113a26bd56242
Status: Image is up to date for swaschkut/pan-os-php:latest
docker.io/swaschkut/pan-os-php:latest
PS C:\>
```

## Basic Use

- The use of single quote ' is recommended for all commands, however, it is not required unless there is a space character inside the arguments (especially in filters, or if names of Device Groups, Vsys or Templates contain a space)

## Mandatory Arguments

```
pan-os-php type=[TYPE] in=[SOURCE_CONFIG] 'actions=[what actions to take]'
```

- Can be provided in any order
- Mandatory arguments will change depending on the type that's being used and what the end goal is
- **IN**
  - provides the input method, it can be a file or the IP/hostname of a firewall
    - example:
      - 'in=D:\path to\my secret\file.xml'
      - 'in=api://panorama.mycompany.com'
- **TYPE**
  - specifies which script inside the PAN-OS-PHP framework will be called
    - example:
      - type=rule
      - type=address

- **ACTIONS**

- A list of actions to perform on the specified configuration
- List of all available actions can be obtained with
  - `pan-os-php type=[TYPE] listactions`
- Several actions can be chained with a forward slash character "/" acting as a separator between the different actions
- When an action requires arguments, they are specified after a colon character ":"
- Certain actions can have additional arguments separated by comma character ",", from the main argument
- Certain arguments can be executed in parallel, and these are separated by pipe character "|"
  - Example:
    - `pan-os-php type=rule 'in=input-file.xml' 'out=output-file.xml' 'location=any' 'actions=exporttoexcel:EXPORT.xls,ResolveAddressSummary|ResolveServiceSummary/tag-add:Low-Hit-Rule' 'filter=(tag has unused-rule)'`
    - The command above will do the following:
- Search for rules which have the tag "unused-rule"
- Add a tag "Low-Hit-Rule" into the specified rules
- Export the resulting rules into Excel and at the same time resolve the address objects and address groups to their actual IP addresses and also resolve the service objects and service groups to their actual port numbers
  - Here is a breakdown of the actual separators
    - Colon character after "exporttoexcel" is followed by the filename to be created by the script
    - Comma character after "EXPORT.xls" is followed by the additional arguments for the exporttoexcel action
    - Pipe character is used to chain together two arguments into one
      - If a comma character is used to separate the 'ResolveAddressSummary' and "ResolveServiceSummary", they will be treated as two separate arguments and the script will end with an error
    - Forward slash character is used to separate the "exporttoexcel" action and "tag-add" action

## Optional Arguments

```
pan-os-php type=[TYPE] in=[SOURCE_CONFIG_FILE_NAME.xml] 'actions=[what actions to take]' 'filter=[which  
specific parts of the configuration are to be changed]' 'location=[which vsys or Device Group to process]'  
'out=[OUTPUT_CONFIG_FILE_NAME.xml]'
```

- **FILTER**

- Only the rules matching a filter will be considered for the Actions
- Remember to add apostrophes ' around this argument
- List of all available filters can be obtained with
  - `pan-os-php type=[TYPE] listfilters`

- **LOCATION**

- Select the VSYS or Device Group(s) against which the script is supposed to run
- Example:
  - `'location=vsys2'`
  - `'location=shared'`
  - `'location=deviceGroup1,deviceGroup2'`
  - `'location=any'`
- Panorama:
  - If the location is not specified, the script will use Shared as the default location and process only the configuration located inside Shared
- Firewall:
  - If the location is not specified, the script will use vsys1 as the default location and process only the configuration located inside vsys1
- If the location is specified as "any", the script will look through the entire configuration file

- **OUT**
  - Use this if you wish to save the amended configuration into an XML file
  - Example
    - `'out=output-file.xml'`

## Autocompletion

- Available only when using Docker PAN-OS-PHP
- Currently works only for the arguments of the pan-os-php command and some of its arguments (such as Type)
  - The functionality will be extended to the other arguments as well
- Examples:
  - pan-os-php {tab}

```
root@e4e159590b06:/share# pan-os-php
actions=          in=          location=          shadow-displaycurlrequest          stats
apitimeout=       listactions      out=              shadow-enablexmlduplicatesdeletion  template=
debugapi          listfilters      shadow-apikeynohidden  shadow-ignoreinvalidaddressobjects  type=
filter=           loadpanoramapushedconfig  shadow-apikeynosave    shadow-json                          version
help              loadplugin=      shadow-disableoutputformatting  shadow-reducexml
```

- pan-os-php type={tab}

```
root@e4e159590b06:/share# pan-os-php type=
address            config-size        license            rule-merger        software-preparation  traffic-log
address-merger     device            maxmind-update    schedule           software-remove       upload
addressgroup-merger  diff             override-finder   securityprofile    spiffy               userid-mgr
appid-enabler      download-predefined  playbook         securityprofilegroup  stats               util_get-action-filter
application         gratuitous-arp     protocol-number-download  service            system-log            virtualwire
bpa-generator      interface         register-ip-mgr   service-merger     tag                  xml-issue
config-commit      ironskillet-update  routing          servicegroup-merger  tag-merger           xml-op-json
config-download-all  key-manager        rule             software-download   threat               zone
```

## Type

- Type is a subset of scripts available inside PAN-OS-PHP and each type has its own set of available actions and filters, meaning not all actions and filters are made available across all types
- Certain filters and actions are available across multiple types
  - These are listed in the Filter and Actions sections in this document
- Certain filters and actions are available only in a specific type
  - These are documented under each respective type
- Certain types do not even require action argument at all

## Rule

- Focuses on policy configuration (security rules, NAT rules etc)

## Filters

### DST and SRC Filters

- **has**
  - Based on object names not values (including ghost objects)
  - Returns partial AND full matches
  - Does **NOT** match "any"
  - Does **NOT** work on nested groups or objects inside groups
  - Example:
    - filter is 8.8.8.8 and the rule destination has objects named 8.8.8.8 and 4.4.4.4, it will match
    - filter is 8.8.8.8 and the rule destination has an object named 8.8.8.8, it will match
    - filter is 8.8.8.8 and the rule destination has an object named obj-8.8.8.8, it will **NOT** match
- **has.only**
  - Based on object names not values (including ghost objects)
  - The filter will return only full matches
  - This filter does **NOT** match "any"
  - Example:
    - filter is 8.8.8.8 and the rule destination has objects named 8.8.8.8 and 4.4.4.4, it will **NOT** match
    - filter is 8.8.8.8 and the rule destination has an object named 8.8.8.8, it will match
    - filter is 8.8.8.8 and the rule destination has an object named obj-8.8.8.8, it will **NOT** match

- **is.fully.included.in.list**

- Based on object values (including ghost objects)
- Searches inside groups, nested groups and object values
- Returns only full matches
- Does **NOT** match "any"
- In case of subnets, all the subnets inside the rule destination or source must match the searched subnets
  - in other words, the searched subnet or subnets must be larger than or equal to all subnet(s) inside the rule
  - following that logic, rules with "any" will always be "larger", which is why rules with "any" do **NOT** match
- Example:
  - filter is 8.8.8.8 and the rule destination is 8.8.8.8 and 4.4.4.4, it will **NOT** match
  - filter is 8.8.8.8 and the rule destination is 8.8.8.8, it will match
  - filter is 10.10.10.0/24 and the rule destination is 10.10.10.10, it will match
  - filter is 10.10.10.0/24 and the rule destination is 10.10.10.0/24, it will match
  - filter is 10.10.10.0/24 and the rule destination is 10.10.10.10 and 10.10.10.0/24 and 10.10.10.0/16, it will **NOT** match
  - filter is 10.0.0.0/8 and the rule destination is 10.11.11.0/24 and 10.11.11.0/16, it will match
  - filter is 10.0.0.0/8 and the rule destination is 10.11.11.0/24 and 10.11.11.0/16 and 1.1.1.1/32, it will **NOT** match
  - filter is 10.0.0.0/8,20.20.0.0/16,4.0.0.0/8 and the rule destination is 10.10.0.0/16 and 4.4.4.4 and 10.20.0.0/16, it will match
  - filter is 10.0.0.0/8,20.20.0.0/16,4.0.0.0/8 and the rule destination is 4.4.4.4, it will match

- **is.partially.included.in.list**

- Based on object values (including ghost objects)
- Searches inside groups, nested groups and object values
- Returns only partial matches
- **DOES** match "any"
- In case of subnets, at least one subnet inside the rule destination or source must match within the searched subnets, but not all subnets in the rule must match the filter (that would be fully.included match)
  - in other words, the searched subnet or subnets must be smaller than the subnet(s) inside the rule
  - following that logic, rules with "any" will always be "larger", which is why rules with "any" will always match
- Example:
  - filter is 8.8.8.8 and the rule destination is 8.8.8.8 and 4.4.4.4, it will match
  - filter is 8.8.8.8 and the rule destination is 8.8.8.8, it will **NOT** match
  - filter is 10.10.10.0/24 and the rule destination is 10.10.10.10, it will **NOT** match
  - filter is 10.10.10.0/24 and the rule destination is 10.10.10.0/24, it will **NOT** match

- filter is 10.10.10.0/24 and the rule destination is 10.10.10.10 and 10.10.10.0/24 and 10.10.10.0/16, it will match
- filter is 10.0.0.0/8 and the rule destination is 10.11.11.0/24 and 10.11.11.0/16, it will **NOT** match
- filter is 10.0.0.0/8 and the rule destination is 10.11.11.0/24 and 10.11.11.0/16 and 1.1.1.1/32, it will match
- filter is 10.0.0.0/8,20.20.0.0/16,4.0.0.0/8 and the rule destination is 10.10.0.0/16 and 4.4.4.4 and 10.20.0.0/16, it will **NOT** match
- filter is 10.0.0.0/8,20.20.0.0/16,4.0.0.0/8 and the rule destination is 4.4.4.4, it will **NOT** match
- filter is 10.0.0.0/8,20.20.0.0/16,4.0.0.0/8 and the rule destination is 10.10.0.0/16 and 4.4.4.4 and 10.20.0.0/16 and 192.168.0.0/16, it will match

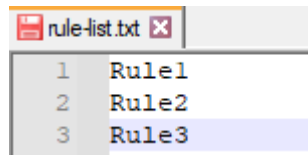
- **is.partially.or.fully.included.in.list**

- This search is based on object values (including ghost objects)
- This will search inside groups, nested groups and object values
- The filter will return partial AND full matches
- This filter **DOES** match "any"
- Example:
  - filter is 8.8.8.8 and the rule destination is 8.8.8.8 and 4.4.4.4, it will match
  - filter is 8.8.8.8 and the rule destination is 8.8.8.8, it will match
  - filter is 10.10.10.0/24 and the rule destination is 10.10.10.10, it will match
  - filter is 10.10.10.0/24 and the rule destination is 10.10.10.0/24, it will match
  - filter is 10.10.10.0/24 and the rule destination is 10.10.10.10 and 10.10.10.0/24 and 10.10.10.0/16, it will match
  - filter is 10.0.0.0/8 and the rule destination is 10.11.11.0/24 and 10.11.11.0/16, it will match
  - filter is 10.0.0.0/8 and the rule destination is 10.11.11.0/24 and 10.11.11.0/16 and 1.1.1.1/32, it will match
  - filter is 10.0.0.0/8,20.20.0.0/16,4.0.0.0/8 and the rule destination is 10.10.0.0/16 and 4.4.4.4 and 10.20.0.0/16, it will match
  - filter is 10.0.0.0/8,20.20.0.0/16,4.0.0.0/8 and the rule destination is 4.4.4.4, it will match
  - filter is 10.0.0.0/8,20.20.0.0/16,4.0.0.0/8 and the rule destination is 10.10.0.0/16 and 4.4.4.4 and 10.20.0.0/16 and 192.168.0.0/16, it will match

## Name Filter

- **name.is.in.file**
  - Provides the ability to read a list of from a text file
  - Useful for situation when a specific list of rules needs to be amended
  - The format of the text file is simply “1 rule name per line”
  - Example

```
'filter=(name is.in.file rule-list.txt) '
```



## Service Filter

- **has.value.recursive**
  - Based on object values
  - Searches inside groups, nested groups and object values
  - Does **NOT** match "any"
  - Does **NOT** match source port values
  - The examples below were tested with individual objects, re-tested with the same objects inside groups, re-tested with the same groups and objects inside another groups (nested groups)
    - Instead of listing every possible variation of service objects, groups and nested groups, the example simply says "the rule has X"
  - Example:
    - filter is 22 and the rule has a port 22, it will match
    - filter is 22 and the rule has a range 22-80, it will match
    - filter is 101 and the rule has a range 100-400, it will match
    - filter is 23-79 and the rule has a range 22-80, it will match
    - filter is 22-1024 and the rule has a range 22-80, it will **NOT** match
    - filter is 21-22 and the rule has two separate ports 21 and 22, it will match
    - filter is 80 and the rule has a port 8080, it will **NOT** match
- **has**
  - Based on object names (not values)
  - The filter will return partial AND full matches
  - This filter does **NOT** match "any"
  - Does **NOT** work on nested groups or objects inside groups
  - Example:
    - filter is FTP and the rule has two service objects named FTP and SSH, it will match
    - filter is FTP and the rule has two service objects named FTP, it will match

- **port.counter = > <**
  - This search is based on the total number of **unique** ports inside a service objects/groups referenced in a rule
    - Port objects/groups with TCP/20-30, TCP/20-25, and UDP/20-40, have a total of 11 TCP ports (30-20+1) and 6 open UDP ports
      - The TCP/20-25 doesn't count because the same ports are already represented by the range TCP/20-30
  - "any" has a value of 131072 and is included in the search
    - 0-65535 for TCP and 0-65535 for UDP = 2 x 65536 = 131072
  - Searches inside groups, nested groups and object values
  - This filter does **NOT** match source port values
  - **Compatibility with App-ID defined ports is currently in development**
  - The examples below were tested with individual objects, re-tested with the same objects inside groups, re-tested with the same groups and objects inside another groups (nested groups)
    - instead of listing every possible variation of service objects, groups and nested groups, the example simply says "the rule has X"
  - Example:
    - filter is port.counter > 11 and the rule has a range 20-30, it will **NOT** match
    - filter is port.counter > 10 and the rule has a range 20-30, it will match
    - filter is port.counter > 11 and the rule has a TCP range 20-30 and TCP range 20-25, it will **NOT** match
    - filter is port.counter > 16 and the rule has a TCP range 20-30 and UDP range 20-25, it will match
    - filter is port.counter > 17 and the rule has a TCP range 20-30 and UDP range 20-25, it will **NOT** match
    - filter is port.counter = 131072 and the rule has "any", it will match
- **port.tcp.counter = > <**
  - Works the same way as the port.counter.greater.than but works only with TCP ports
- **port.udp.counter = > <**
  - Works the same way as the port.counter.greater.than but works only with UDP ports

## Schedule Filter

- is.expired
  - Identifies only expired schedules

## Schedule.expire.in.days Filter

- This is a separate filter from “Schedule” filter
- Identifies all schedules which have already expired or will expire in the set amount of days
- This feature used to be part of the schedule filter, but has been moved out and made into an individual filter
- Typical operands <, >, = are supported; negative and decimal values are also allowed
- The script calculates the time difference between the current time of the time zone referenced in the configuration and the end-time referenced in the schedule
  - In other words, the time zone or the current OS time of the device where the PAN-OS-PHP script is executed is not relevant, the only relevant information is:
    - The time zone inside the configuration file
    - The current time of that time zone
    - The end-time inside the schedule
  - The result is represented as a real number, which allows the use of negative values as well as decimal points
- Example:
  - Time zone inside the FW configuration is Europe/London, schedule end-time is 15:00, the time in London at the point of running the script is 14:00, the local time of the device where the script is being run is 16:00 (Asia/Riyadh time zone)
    - filter is (schedule.expire.in.days > 1), this will match the schedule
      - if the time zone inside the configuration was changed to Asia/Riyadh, it would no longer match

## Address

- Focuses on address objects and address groups configuration

## Filters

- **ip.count**
  - Identifies the number of IP addresses inside an object
    - currently searches only through ip-netmask and ip-range objects
  - Example:
    - `filter=(ip.count > 254)` will match /24 and larger ip-netmask objects and ip-range objects with 255 or bigger ranges
- **object is.ip-range**
  - Identifies objects of ip-range type
    - `filter=(object is.ip-range)`

## Actions

- **move-range2network**
  - Changes the object type from ip-range to ip-netmask
  - Currently only works for range objects which can be replaced with a single CIDR subnet
  - Example:
    - range object 10.0.0.0-10.0.0.10 will **NOT** be changed
    - range object 10.0.0.0-10.0.0.255 will be changed to 10.0.0.0/24 ip-netmask object
    - range object 10.0.0.32-10.0.0.63 will be changed to 10.0.0.32/27 ip-netmask object

## XML-issue

```
pan-os-php type=xml-issue in=config.xml out=fixed-config.xml | tee cleanup-log.txt
```

- Identifies and fixes problems inside configuration
  - Examples of the problems the script looks for are:
    - Duplicate sources and destinations
      - Meaning the same IP being twice as source, or twice as destination in the same rule
    - Duplicate members inside the same address or service groups
      - Meaning the same address/service object being referenced twice inside the same group
    - Two “ “ space characters directly after one another inside a rule name
      - This causes problems with command execution in CLI
- These problems might be currently accepted by PAN-OS and cause no operational impact, but it is still recommended to fix them
- PAN-OS can (and does) receive additional validation in new releases and the configuration acceptable by one PAN-OS version, might not be acceptable by a new one
  - An example of this is PAN-OS 10.0, which no longer allows duplicate members inside the same address or service groups

### Address Groups and Service Groups

On upgrade to PAN-OS 10.0, the Panorama management server checks for duplicate addresses in address groups (**Objects > Address Groups**) and services in service groups (**Objects > Service Groups**) that you created with CLI, and fails to commit any configuration changes if duplicate address objects and services exist.

**Workaround:** Before you upgrade to PAN-OS 10.0, modify your address group and service group configurations and rename any duplicate address objects or services.

- This type does not support the SET command feature, therefore the changes need to be applied through configuration snapshot import into PAN-OS or via API
- Clean-up steps are as follows

1. Run the script
    - a. make sure to specify output file using the “out=” argument
    - b. make sure you save the log file using output redirection “| tee” argument
  2. Upload the cleaned up XML file into PAN-OS device and load it
  3. Search through the log file for “FIX\_MANUALLY” string
    - a. the log file is the the file used for the output redirection
    - b. The number of the required manual fixes is available in the summary provided by the script
  4. Fix the issues tagged as “FIX\_MANUALLY”
    - a. Each log entry with this tag will contain information about the issue and the exact position inside the XML file
    - b. If the problem is not fixable via GUI or CLI, the XML file will need to be fixed manually inside a text editor and then reuploaded to the PAN-OS device
  5. Once all the manual fixes have been done (and if needed, re-uploaded to the device), commit the configuration
- Always make sure the fixed configuration is tested in a lab, especially if the script identifies thousands of problems and also if there are manual fixes required

# Actions

- Actions listed below are available across multiple Types

## ExportToExcel

`exporttoexcel:EXPORT.xls, [ARGUMENT1] | [ARGUMENT2],`

- Most commonly used in Address and Rule Types
- This action exports rules matched by the filters into an HTML formatted file which can be opened and reformatted in Excel
- The action will create Excel files with different columns based on the chosen Type
  - Using ExportToExcel with Address Type will generate a spreadsheet with information about the objects
  - Using ExportToExcel with Rule Type will generate a spreadsheet with information about the rules
- The action also uses different arguments based on the chosen Type
- Rule Type columns
  - these columns are not part of the rule itself in the configuration, but are added automatically to provide additional information
    - Location
      - the Device Group (Panorama) or vsys (Firewall) where the specific rule is located
    - Rulebase
      - pre-rulebase or post-rule base (Panorama), or it remains blank (Firewall)
    - Type
      - defines the rule type such as NAT rule, security rule etc
    - Service\_count
      - number of ports (UDP + TCP) inside the rule
      - if the rule has “service” filter defined as “any”, the number of ports is 131072 (65536 for TCP and 65536 for UDP)
    - Service\_count\_tcp
      - number of TCP ports inside the rule
    - Service\_count\_udp
      - number of UDP ports inside the rule

## Additional Features

### Display Configured Admins

```
pan-os-php type=device in=api://[IP_ADDRESS] actions=system-mgt-config_users
pan-os-php type=device in=[SOURCE_CONFIG] actions=system-mgt-config_users
```

- displays all configured admin accounts, their name, role, authentication profiles and assigned permissions

### Display Connected Admins

```
pan-os-php type=device in=api://[IP_ADDRESS] actions=system-admin-session:display
```

- API-only
- displays the currently logged in admins, their IP, when they connected and their method of access

### Disconnect Connected Admins

```
pan-os-php type=device in=api://[IP_ADDRESS] actions=system-admin-session:delete,[hours]
```

- API-only
- disconnects admin sessions which have been idle for longer than the specified number of hours
  - if the hours are not specified, the script disconnects any idle session older than 8 hour
- [hours] can also be defined as a decimal number
  - for example: 0.1 represents 6 minutes (1/10th of an hour), 0.25 represents 15 minutes

## Display Panorama Configuration overwritten on Firewall

```
pan-os-php type=override-finder in=api://[PANORAMA MGMT-IP] cycleConnectedFirewalls
pan-os-php type=override-finder in=api://[FW-SERIAL#]@[PANORAMA MGMT-IP]
```

- API-only
- script reaching out to all Panorama connected Firewalls
- displays the configuration from Panorama which is locally overwritten by the Firewall configuration

## Configuration Validation

```
pan-os-php type=service in=api://[IP_ADDRESS] 'filter=( ( (object is.tcp) and (name regex /udp/i ) ) or (
(object is.udp ) and (name regex /tcp/i ) ) )' location=any actions=exporttoexcel:mixed_service_protocol.html
```

```
pan-os-php type=service in=[SOURCE_CONFIG] 'filter=( ( (object is.tcp) and (name regex /udp/i ) ) or ( (object
is.udp ) and (name regex /tcp/i ) ) )' location=any actions=exporttoexcel:mixed_service_protocol.html
```

- Compares service objects names with their values
- The specific example above finds all service object which have:
  - TCP in name, but protocol udp configured
  - UDP in name but protocol tcp configured
- This validation and manual cleanup is required in order to avoid inaccurate results when using service-merger utility
- This particular validation is planned to be integrated into:
  - `pan-os-php type=xml-issue in=api://[IP_Address]`

## JSON Format Output

```
pan-os-php type=address in=api://1.1.1.1 location=device-group shadow-json
```

- similar to Excel export, JSON format is available to offer another way how to further work with the output data
- the shadow-json argument will display the relevant objects with all possible values
- JSON format output example:

```
"PanoramaConf": [  
  {  
    "header": "* processing store 'PanoramaConf: \/ DeviceGroup:child \/ AddressStore:address' that  
holds 9 objects",  
    "sub": {  
      "name": "child",  
      "type": "DeviceGroup",  
      "store": "AddressStore",  
      "object": {  
        "wildcard": {  
          "name": "wildcard",  
          "actions": {  
            "display": {  
              "name": "display"  
            }  
          },  
          "type": "Address",  
          "value": "10.17.0.16\/0.0.254.15",  
          "tag": "",  
          "description": ""  
        },  
        "addr2b": {  
          "name": "addr2b",
```

```
        "actions": {
            "display": {
                "name": "display"
            }
        },
        "type": "Address",
        "value": "8.8.4.4",
        "tag": "",
        "description": ""
    },
```

## Display Shadow Rules

```
pan-os-php type=device in=api://FW-MGMT-IP actions=display-shadowrule 'filter=(name eq vsys1)
loadpanoramapushedconfig
pan-os-php type=device in=api://FW-MGMT-IP/merged-config actions=display-shadowrule 'filter=(name eq
vsys2) loadpanoramapushedconfig
```

- This is an API-only feature
- It can be run against a firewall or Panorama
  - “/merged-config” needs to be specified in the API path above only if a template configuration is being pushed from Panorama
- Firewall specific notes
  - panorama pushed template
    - when connecting to a Firewall via API, the script will only retrieve the local configuration for interfaces and routing
    - to display the merged configuration (local + Panorama pushed Template config), please use the following input syntax:
      - in=api://FW-MGMT-IP/merged-config
  - panorama pushed rule/objects
    - when connecting to a Firewall via API, the script will only retrieve the local configuration for rules and objects
    - to display the merged configuration (local + Panorama pushed Device Group config), please use the following argument:
      - loadpanoramapushedconfig

## SET Command Output

```
pan-os-php type=address-merger in=input.xml out=output.xml location=any allowmergingwithupperlevel  
outputformatset=FILE_setcommands.txt
```

- Most PAN-OS-PHP scripts (Types) which manipulates the configuration file, can display the full set commands for the changes
  - Some Types, such as “xml-issue” do not support the SET Command Output feature
- To store the set commands in a file, please use the following argument:
  - outputformatset=FILE.txt

## Configuration Import into Panorama

```
pan-os-php type=upload in=CONFIG-FILE.XML out=api://MGMT-IP 'fromXpath=SOURCE_CONFIG_XPATH'  
'toXpath=TARGET_PANORAMA_XPATH'
```

- There are several ways already in place to import configuration:
  - Load Configuration Snapshot via GUI
  - Load Config Partial via the CLI
- The main benefit of this feature compared to the two mentioned above is that it does NOT require Full Commit
- The other advantage of this feature is that multiple source XPATHs can be defined with a single command using logical OR statements
- This allows import of entire Device Group and Template configuration into Panorama without the need for change freeze
  - This is particularly useful for environments with 24/7 operation and “follow the sun” model
- Always make sure the Device Groups and Templates in question are already present in the target Panorama configuration
- Examples:

### Import all objects at once

- for comparison, LCP would require multiple commands

```
pan-os-php type=upload in=CONFIG-FILE.XML out=api://192.168.1.100  
'fromXpath=/config/devices/entry[@name="localhost.localdomain"]/device-group/entry[@name="NEW-  
DG"]/*[name()="address" or name()="address-group" or name()="service" or name()="service-group" or  
name()="tag"]' 'toXpath=/config/devices/entry[@name="localhost.localdomain"]/device-group/entry[@name="NEW-  
DG"]'
```

### Import Pre-rules:

```
pan-os-php type=upload in=CONFIG-FILE.XML out=api://192.168.1.100  
'fromXpath=/config/devices/entry[@name="localhost.localdomain"]/device-group/entry[@name="NEW-DG"]/="pre-  
rulebase"]' 'toXpath=/config/devices/entry[@name="localhost.localdomain"]/device-group/entry[@name="NEW-DG"]'
```

## Bulk Rule configuration

- Several actions inside PAN-OS-PHP have a FastAPI version along with their “base” version and other versions
  - For example
    - securityProfile-Group-Set
    - securityProfile-Group-Set-FastAPI
    - securityProfile-Group-Set-Force
- The advantage of the FastAPI version of the command is that it pushes all changes at once instead of rule by rule
- If a rule already has the particular attribute configured, it will be overwritten
  - For example, if a rule already has a Security Profile Group assigned, and the above action securityProfile-Group-Set-FastAPI is used, the existing Security Profile Group will be replaced with the one being pushed by the API call
  - It is recommended to pair this action with a filter to ensure only specific rules are amended
  - Alternatively, this action is perfectly suited for situations where a particular rule attribute must be identical across all rules, such as:
    - logging profile
    - log at session end
- Not all actions have this option as not all changes can be pushed “all-in-one”
  - The simplest way to find out which actions have this option is to run the following command
  - `pan-os-php type=rule listactions | grep FastAPI`
- Examples:

### Push the same Security Profile Group to a list of rules

```
pan-os-php type=rule location=DEVICE-GROUP1 in=api://10.1.1.1 actions=securityProfile-Group-Set-FastAPI:DEFAULT-SPG 'filter=(name is.in.file rule-list.txt)'
```

### Push the same Logging Profile to all rules

```
pan-os-php type=rule location=any in=api://10.1.1.1 actions=logSetting-set-FastAPI:FORWARD-LOGS-TO-PANORAMA
```

### Push the same Security Profile Group to rules which do not have one already

```
pan-os-php type=rule location=any in=api://10.1.1.1 actions=securityProfile-Group-Set-FastAPI:DEFAULT-SPG 'filter=(secprof not.set)'
```

## Use Cases and Examples

- These are specific use cases for the above listed features

### Rule Audit (show rules applicable to traffic X)

```
pan-os-php type=rule in=[SOURCE_CONFIG] 'location=AUDITED-DG' 'actions=exporttoexcel:EXCEL-FILE.XLS,ResolveAddressSummary|ResolveServiceSummary' 'filter=( (from has ZONE1) AND ((src is.partially.or.fully.included.in.list 10.11.12.100,10.11.12.200,172.16.17.100,172.16.17.200) OR (src is.any)) AND ((service has.value.recursive 3389) OR (service has.value.recursive 49152-65535) OR OR (service is.any)) )'
```

- finds rules with the ZONE1 as source zone AND one of the IP addresses as source (including inside larger subnets) AND one of the ports as destination service (including inside port ranges)
- exports all identified rules into Excel and resolves objects to their port numbers and IP addresses

### Export security rules with a specific tag

```
pan-os-php type=rule in=[SOURCE_CONFIG] 'location=any' 'filter=(tag has unused-rule)' 'actions=exporttoexcel:EXPORTED-EXCEL-FILE.XLS,ResolveAddressSummary|ResolveServiceSummary'
```

- finds rules with the tag 'unused-rule' and exports them into Excel and resolves objects to their port numbers and IP addresses

## Export NAT rules

```
pan-os-php type=rule in=[SOURCE_CONFIG] location=[SPECIFIC-DG] ruletype=nat 'actions=exporttoexcel:NAT-RULES.xls,ResolveAddressSummary'
```

- exports all NAT rules into Excel and resolves objects to their IP addresses

## Export rules with expired schedules and schedules expiring in 20 days

```
pan-os-php type=rule in=[SOURCE_CONFIG] 'actions=exporttoexcel:EXPORT.xls,ResolveScheduleSummary'  
'filter=(schedule.expire.in.days < 20)'
```

- The "ResolveScheduleSummary" argument in the export action will include the actual configured expiry time and date inside each schedule

## Config size reduction/cleanup tasks after multiple migrations

```
pan-os-php type=config-size in=[SOURCE_CONFIG]
```

- Provides size information about the different configuration parts to show which ones consume the most space
  - These can then be investigated and cleaned up using other pan-os-php scripts
- It also provides an estimate of how much space can be saved by removing typical XML overhead such as spaces and newline characters
  - This is useful only for transferring the file, once the XML is imported into PAN-OS, the removed characters and spaces are added back

## Remove Unused objects

```
pan-os-php type=address in=api://[IP_ADDRESS] location=any actions=delete 'filter=(object is.unused.recursive)'
```

```
pan-os-php type=address in=[SOURCE_CONFIG] location=any actions=delete 'filter=(object is.unused.recursive)'
```

- Deletes unused address objects within a configuration
- It is recommended to use Display or ExportToExcel actions first to get an idea of what will be deleted
- The process can be repeated with service objects and service groups using "type=service"
- 'is.unused.recursive' considers an object or a group unused only if they are **NOT** referenced in any rules, interfaces, static routes **AND** if they are referenced inside a group which itself is not referenced anywhere
  - Example:
    - ObjectA is a member of GroupA
      - ObjectA and GroupA are not referenced anywhere else in the configuration
      - Both ObjectA and GroupA will be considered unused and will be removed
    - ObjectA is a member of GroupA
      - ObjectA is referenced in a security rule, GroupA is not referenced anywhere in the configuration
      - ObjectA will remain, GroupA will be removed
    - ObjectA is a member of GroupA and GroupA is a member of GroupB
      - ObjectA and GroupA and GroupB are not referenced anywhere else in the configuration
      - GroupB will be removed
      - ObjectA and GroupA will remain
        - This is a known limitation of the recursive search, caused by groups being nested inside groups
        - Simple workaround is to run the removal script again after GroupB has been removed
- 'is.unused' considers an object or a group unused only if it is not referenced anywhere at all
  - if an object is referenced inside an object group, the object itself is considered used regardless of whether the group itself is referenced anywhere or not

## Merge objects with duplicate values

```
pan-os-php type=address-merger in=api://[IP_ADDRESS] allowMergingWithUpperLevel MergeCountLimit=10  
exportCSV=FILE.html actions=merge 'pickfilter=(name regex /^corp-/)'
```

- This script merges objects with duplicate values and provides detailed logs on all changes
- The merge count limit is useful to limit the iterations for large configs which may take a long time to complete
- The option for merging with the upper level is useful to move objects into Shared
- The script can be used on services and groups with the following types: service-merger, servicegroup-merger, and addressgroup-merger
- Remember if the "location" argument is not specified, the script will use Shared by default
- The "pickfilter" option is used to prefer a certain object with a certain keywords or prefix to adhere to a naming convention. In this example, objects with **corp-** will be preferred to be kept as the primary object in the merge operations.
- With the additional argument: **exportCSV=FILE.html**, you can get the full list of address objects which are retained in the configuration and which ones are replaced by the remaining objects
  - you can also use **actions=display** instead of **actions=merge** in order to see the output directly in the CLI (without making any changes)
  - if the number of objects is too large, the CSV export is a better option for review

## Move objects between Device Groups

```
pan-os-php type=address in=api://[IP_ADDRESS] location=device-group 'actions=move:shared,removeIfMatch'  
pan-os-php type=address in=[SOURCE_CONFIG] location=device-group 'actions=move:shared,removeIfMatch'
```

- This script moves objects from a device group into Shared
- Caution should be taken if your Panorama manages smaller firewall models which can handle less objects than the larger models
- The process can be repeated with service and group objects
- The optional mode specifier can be used to change the default way conflicts are handled which is "skipIfConflict"
- Please note it is possible to use the same script to move objects the other way round (from Shared to a specific Device Group)
  - `pan-os-php type=address in=api://[IP_ADDRESS] location=shared 'actions=move:[CHILD-DG]'`
- Here are few more examples of the command
  - `pan-os-php type=address in=api://1.1.1.1 'actions=move:[TARGET-DG]' location=shared  
'filter=(reflocation is.only [TARGET-DG]) and (!object is.group)'`

# Features in Development

## PAN-OS-PHP Docker API

- Offers a GUI for PAN-OS-PHP
- It can be started using the following command
  - `docker run -d -p 8082:80 swaschkut/pan-os-php-api:latest`
- Additional information can be found at <https://github.com/PaloAltoNetworks/pan-os-php/wiki/docker>
- Once the docker instance is running, please open Google Chrome and browse to <http://localhost:8082>
  - Other browsers have issues displaying all the content
- The GUI can be used to create and also execute PAN-OS-PHP CLI commands, API calls or playbooks
- **This feature is currently in beta; not everything will be fully functional and the UI design is also in development**

