Palo Alto Networks PAN-OS SDK for Python Documentation

Release 1.6.0

Brian Torres-Gil

Nov 30, 2021

Contents

Contents:

CHAPTER 1

Palo Alto Networks PAN-OS SDK for Python

The PAN-OS SDK for Python (pan-os-python) is a package to help interact with Palo Alto Networks devices (including physical and virtualized Next-generation Firewalls and Panorama). The pan-os-python SDK is object oriented and mimics the traditional interaction with the device via the GUI or CLI/API.

- Documentation: http://pan-os-python.readthedocs.io

1.1 Features

- Object model of Firewall and Panorama configuration
- Multiple connection methods including Panorama as a proxy
- All operations natively vsys-aware
- Support for high availability pairs and retry/recovery during node failure
- Batch User-ID operations
- Device API exception classification

1.2 Status

Palo Alto Networks PAN-OS SDK for Python is considered stable. It is fully tested and used in many production environments. Semantic versioning is applied to indicate bug fixes, new features, and breaking changes in each version.

1.3 Install

Install using pip:

```
pip install pan-os-python
```

Upgrade to the latest version:

```
pip install --upgrade pan-os-python
```

If you have poetry installed, you can also add pan-os-python to your project:

```
poetry add pan-os-python
```

1.4 How to import

To use pan-os-python in a project:

```
import panos
```

You can also be more specific about which modules you want to import:

```
from panos import firewall
from panos import network
```

1.5 A few examples

For configuration tasks, create a tree structure using the classes in each module. Nodes hierarchy must follow the model in the Configuration Tree.

The following examples assume the modules were imported as such:

```
from panos import firewall
from panos import network
```

Create an interface and commit:

```
fw = firewall.Firewall("10.0.0.1", api_username="admin", api_password="admin")
eth1 = network.EthernetInterface("ethernet1/1", mode="layer3")
fw.add(eth1)
eth1.create()
fw.commit()
```

Operational commands leverage the 'op' method of the device:

```
fw = firewall.Firewall("10.0.0.1", api_username="admin", api_password="admin")
print fw.op("show system info")
```

Some operational commands have methods to refresh the variables in an object:

```
# populates the version, serial, and model variables from the live device
fw.refresh_system_info()
```

See more examples in the Usage Guide.

1.6 Upgrade from pandevice

This `pan-os-python` package is the evolution of the older `pandevice` package. To upgrade from `pandevice` to `pan-os-python`, follow these steps.

Step 1. Ensure you are using python3

> Python2 is end-of-life and not supported by `pan-os-python`.

Step 2. Uninstall pandevice:

```
pip uninstall pandevice
 # or
poetry remove pandevice
```

Step 3. Install pan-os-python:

```
pip3 install pan-os-python
 # or
poetry add pan-os-python
```

Step 4. Change the import statements in your code from `pandevice` to `panos`. For example:

```
import pandevice
from pandevice.firewall import Firewall

 # would change to

import panos
from panos.firewall import Firewall
```

Step 5. Test your script or application

> There are no known breaking changes between `pandevice v0.14.0` and `pan-os-python v1.0.0`, but it is a major upgrade so please verify everything works as expected.

1.7 Contributors

- Brian Torres-Gil - btorresgil
- Garfield Freeman - shinmog
- John Anderson - lampwins
- Aditya Sripal - AdityaSripal

Thank you to Kevin Steves, creator of the pan-python library

CHAPTER 2

Getting Started

2.1 Install

Install using pip:

```
pip install pan-os-python
```

Upgrade to the latest version:

```
pip install --upgrade pan-os-python
```

If you have poetry installed, you can also add pan-os-python to your project:

```
poetry add pan-os-python
```

2.2 Import the classes

To use pan-os-python in a project:

```
import panos
```

You can also be more specific about which modules you want to import:

```
from panos import base
from panos import firewall
from panos import panorama
from panos import policies
from panos import objects
from panos import network
from panos import device
```

Or, even *more* specific by importing a specific class:

```
from panos.firewall import Firewall
```

2.3 Connect to a Firewall or Panorama

A PanDevice is a Firewall or a Panorama. It's called a PanDevice because that is the class that Firewall and Panorama inherit from. Everything connects back to a PanDevice, so creating one is often the first step:

```
from panos.firewall import Firewall
from panos.panorama import Panorama
fw = Firewall('10.0.0.1', 'admin', 'mypassword')  # Create a firewall object
pano = Panorama('10.0.0.5', 'admin', 'mypassword')  # Create a panorama object
```

You can also create a Firewall or Panorama object from a live device. In this example, 10.0.0.1 is a firewall and 10.0.0.5 is a Panorama. The device type is determined by checking the live device.:

```
>>> from panos.base import PanDevice

>>> device1 = PanDevice.create_from_device('10.0.0.1', 'admin', 'mypassword')
>>> type(device1)
<class 'panos.firewall.Firewall'>

>>> device2 = PanDevice.create_from_device('10.0.0.5', 'admin', 'mypassword')
>>> type(device2)
<class 'panos.panorama.Panorama'>
```

2.4 Operational commands

Operational commands are used to get or clear the current operational state of the device or make operational requests such as content upgrades. Most any command that is not a config mode or debug command is an operational command. These include many 'show' commands such as `show system info` and `show interface ethernet1/1` and 'request' commands. You cannot use operational commands to change the running configuration of the firewall or Panorama. See *Configure your device* below to configure your firewall by changing the running configuration.

Perform operational commands using the `op` method on a PanDevice instance. By default, the return value is an xml.etree.ElementTree object which can be easily parsed:

```
from panos import firewall
fw = firewall.Firewall('10.0.0.1', 'admin', 'mypassword')
element_response = fw.op('show system info')
```

Use the `xml` argument to return a string of xml. This is harder to parse, but sometimes a string is needed such as when saving to a file.:

```
xml_str_response = fw.op('show system info', xml=True)
```

Important: When passing the cmd as a command string (not XML) you must include any non-keyword strings in the command inside double quotes (`"`). Here's some examples:

```
fw.op('clear session all filter application "facebook-base"')
# The string "facebook-base" must be in quotes because it is not a keyword
```

(continues on next page)

(continued from previous page)

```
fw.op('show interface "ethernet1/1"')
# The string "ethernet1/1" must be in quotes because it is not a keyword
```

This works by converting all unquoted arguments in cmd to XML elements and double quoted arguments as text after removing the quotes. For example:

- `show system info` -> `<show><system><info></info></system></show>`
- `show interface "ethernet1/1"` -> `<show><interface>ethernet1/1</interface></show>`

The command's XML is then sent to the firewall.

Parse the result

You can parse an ElementTree using the python ElementTree library.

Assuming the first `op()` call returns a response with this XML (output simplified for example purposes):

```
<response status="success">
    <result>
        <ifnet>
            <counters>
                <ifnet>
                    <entry>
                        <name>ethernet1/1</name>
                        <ipackets>329744</ipackets>
                        <opackets>508805</opackets>
                        <ierrors>0</ierrors>
                    </entry>
                </ifnet>
            </counters>
            <name>ethernet1/1</name>
            <zone>DMZ</zone>
        </ifnet>
        <hw>
            <name>ethernet1/1</name>
            <mac>08:30:6b:1e:55:42</mac>
            <state>up</state>
        </hw>
    </result>
</response>
```

Then this example collects the zone, mac address, and packet output for ethernet1/1:

```
response = fw.op('show interface "ethernet1/1"')

name = response.find(".//zone").text
# name = "DMZ"

mac_address = response.find("./result/hw/mac").text
# mac_address = "08:30:6b:1e:55:42"

counter_entries = response.findall(".//counters/ifnet/entry")
packets_out = [(counters.find("./name").text, int(counters.find("./opackets").text))␣
↪for counters in counter_entries]
# packets_out = [("ethernet1/1", 508805)]
```

In the example above, we use a deep search to find the `<zone>` element, an absolute path to get the `<mac>` element,

and a findall with both deep search and relative path to get packets out for every subinterface. In this example there are no subinterfaces, so it returns one list item.

2.5 Configure your device

You can configure your firewall or Panorama with a configuration tree using PanObjects. Everything in pan-os-python is a PanObject. They are like building blocks to build out a configuration. There are many methods available to build up the configuration tree and interact with the live device:

Common configuration methods of PanObject

Build the configuration tree: `add()`, `remove()`, `find()`, and `findall()`

Push changed configuration to the live device: `apply()`, `create()`, and `delete()`

Pull configuration from the live device: `refresh()`, `refreshall()`

There are other useful methods besides these. See *Useful Methods* for a table of all the methods and what they do. All methods are also documented in the *panos.base.PanObject* API reference.

Configuration examples

In each of these examples, assume a Firewall and Panorama object have been instantiated:

```
from panos.firewall import Firewall
from panos.panorama import Panorama
from panos.objects import AddressObject

fw = Firewall("10.0.0.1", "admin", "mypassword")
pano = Panorama("10.0.0.5", "admin", "mypassword")
```

Create an address object on a firewall:

```
webserver = AddressObject("Apache-webserver", "5.5.5.5", description="Company web␣
↪server")
fw.add(webserver)
webserver.create()
```

In this example, add() makes the AddressObject a child of the Firewall. This does not make any change to the live device. The create() method pushes the new AddressObject to the live device represented by 'fw'.

If you lose the handle to the AddressObject, you can always retreive it from a parent node with one of the *find* methods. For example:

```
webserver = fw.find("Apache-webserver", AddressObject)
```

Remove the description of that same address object:

```
webserver.description = None
webserver.apply()
```

The apply() method is used instead of create() because it is destructive. The create() method will never remove a variable or object, only add or change it.

Delete the entire address object:

```
webserver.delete()
```

The delete() method removes the object from the live device *and* the configuration tree. In this example, after delete() is called, 'webserver' is no longer a child of 'fw'.

Retrieve configuration

The previous section describes how to build a configuration tree yourself. But many cases require you to pull configuration from the firewall to populate a PanDevice configuration tree. This technique allows many advantages including tracking current state of the device, and checking if the configuration change is already on the firewall to prevent an unnecessary commit.

In this example, the live device has 3 address objects. Pull the address objects from the live device and add them into the configuration tree:

```
>>> fw.children
[]
>>> AddressObject.refreshall(fw, add=True)
>>> fw.children
[<panos.objects.AddressObject object at 0x108080e90>,
 <panos.objects.AddressObject object at 0x108080f50>,
 <panos.objects.AddressObject object at 0x108080ed0>]
```

It's also possible to refresh the variables of an existing object:

```
>>> adserver = AddressObject("ADServer")
>>> fw.add(adserver)
>>> adserver.value
None
>>> adserver.refresh()
>>> adserver.value
"4.4.4.4"
```

CHAPTER 3

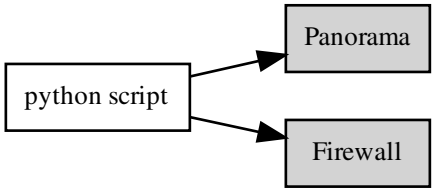
How-to Guides

3.1 Connect via Panorama

Making changes to Panorama is always done the same way, with a connection to Panorama. But, there are a different options to make local changes to a Firewall.

Option 1: Connect to the Firewall and Panorama directly

When making changes to Panorama, connect to Panorama. When making changes to the Firewall, connect directly to the Firewall.



This method is best in the following cases:

- Firewall managment IP is accessible to the script
- The credentials for both devices are known
- The permissions/role for the user are set on both devices
- The serial of the firewall is unknown, but the management IP is known

To use this method:

1. Create a *panos.firewall.Firewall* instance and a *panos.panorama.Panorama* instance.
2. In both instances, set the 'hostname' attribute and either the 'api_key' or the 'api_username' and 'api_password' attributes.

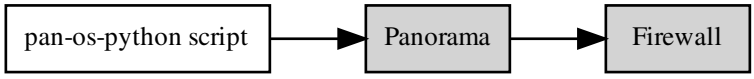
Example:

```
# Instantiate a Firewall with hostname and credentials
fw = firewall.Firewall("10.0.0.1", "admin", "mypassword")
# Instantiate a Panorama with hostname and credentials
pano = panorama.Panorama("10.0.0.5", "admin", "mypassword")
# Change to Firewall
fw.add(objects.AddressObject("Server", "2.2.2.2")).create()
# Change to Panorama
pano.add(panorama.DeviceGroup("CustomerA")).create()
```

In this example, the address object is added to the Firewall directly, without any connection to Panorama. Then a device-group is created on Panorama directly, without any connection to the Firewall.

Option 2: Connect to Firewall via Panorama

When making changes to the Firewall, connect to Panorama which will proxy the connection to the Firewall. Meaning all connections are to Panorama.



This method is best in the following cases:

- The Firewall management IP is unknown or not reachable from the script
- You only store one set of credentials (Panorama)
- The serial of the firewall is known or can be determined from Panorama

To use this method:

1. Create a *panos.firewall.Firewall* instance and a *panos.panorama.Panorama* instance.
2. In the Panorama instance, set the 'hostname' attribute and either the 'api_key' or the 'api_username' and 'api_password' attributes.
3. In the Firewall instance, set the 'serial' attribute.
4. Add the Firewall as a child of Panorama, or as a child of a DeviceGroup under Panorama.

Example:

```
# Instantiate a Firewall with serial
fw = firewall.Firewall(serial="0002487YR3880")
# Instantiate a Panorama with hostname and credentials
pano = panorama.Panorama("10.0.0.5", "admin", "mypassword")
# Add the Firewall as a child of Panorama
pano.add(fw)
# Change to Firewall via Panorama
fw.add(objects.AddressObject("Server", "2.2.2.2")).create()
# Change to Panorama directly
pano.add(panorama.DeviceGroup("CustomerA")).create()
```

In this example, both changes are made with connections to Panorama. First, the address object is added to the Firewall by connecting to Panorama which proxies the API call to the Firewall. Then a device-group is created on Panorama directly.

3.2 Work with Virtual Systems (VSYS)

There's a great blog post by the Developer Relations team on how to work with vsys in python. You can read it here:

https://medium.com/palo-alto-networks-developer-blog/handling-pan-os-vsys-in-pandevice-212fe892d303

A Firewall PanDevice can represent a firewall or a virtual system (vsys). By default, a Firewall instance represents a single context firewall, or 'vsys1' on a multi-vsys firewall.

When working with a firewall with multi-vsys mode enabled, there are two methods to work with vsys:

Method 1: A different Firewall instance for each vsys

Each Firewall object has a 'vsys' attribute which is assigned the vsys id. For example:

```
fw_vsys2 = firewall.Firewall("10.0.0.1", "admin", "mypassword", vsys="vsys2")
fw_vsys3 = firewall.Firewall("10.0.0.1", "admin", "mypassword", vsys="vsys3")
```

When using this method, non-vsys-specific configuration should be modified using a 'shared' PanDevice:

```
fw = firewall.Firewall("10.0.0.1", "admin", "mypassword", vsys="shared")
```

To create or delete an entire vsys, use the create_vsys() and delete_vsys() methods:

```
fw_vsys2.create_vsys()
fw_vsys3.delete_vsys()
```

Method 2: A single Firewall instance with Vsys child instances

Create Vsys instances and add them to a 'shared' PanDevice:

```
fw = firewall.Firewall("10.0.0.1", "admin", "mypassword", vsys="shared")
vsys2 = device.Vsys("vsys2")
vsys3 = device.Vsys("vsys3")
fw.add(vsys2)
fw.add(vsys3)
```

Configuration objects are added to the Vsys instances instead of the Firewall instance:

```
ao = vsys2.add(objects.AddressObject("MyIP", "2.2.2.2"))
ao.create()
```

The vsys itself can be created and deleted using the standard configuration tree methods:

```
vsys2.create()
vsys3.delete()
```

3.3 High Availability Pairs

This library tries to handle High Availability (HA) pairs of devices as elegantly as possible. Having two devices can pose challenges because some configuration needs to be applied to both firewalls, while other configuration should be applied only to the active firewall. Also, two devices implies two pan-os-python configuration trees. But, pan-os-python offers a few features to make working with HA pairs easier:

- Only one configuration tree to manage for an HA pair
- Automatically knows which firewall to talk to
- Detects when a firewall is not reachable and automatically switches to the other firewall
- Knows which configuration should be applied to the active firewall and which should be made on both firewalls, and handles this for you under the hood

There's just a couple extra steps to ensure your HA experience is smooth. While not strictly necessary, it's a good idea to verify the state of the HA before making configuration changes, so you know configuration will sync properly to the standby device.

Here's an example of configuration with an HA pair of firewalls:

```
from panos.firewall import Firewall
from panos.objects import AddressObject

# Don't assume either firewall is primary or active.
# Just start by telling pan-os-python they are an HA pair
# and how to connect to them.
fw = Firewall('10.0.0.1', 'admin', 'password')
fw.set_ha_peers(Firewall('10.0.0.2', 'admin', 'password'))

# Notice I didn't save the second firewall to a variable, because I don't need it.
# The point is to treat the HA pair as one firewall, so we only need one variable.
# This way, we have only one pan-os-python configuration tree to manage,
# NOT one tree for each fw in the pair.

# At this point, it's a good idea to collect the active/passive state from
# the live devices. This stores which firewall is active to an internal
# state machine in the Firewall object.
fw.refresh_ha_active()

# Now, verify the config is synced between the devices.
# If it's not synced, force config synchronization from active to standby
if not fw.config_synced():
    fw.synchronize_config()  # blocks until synced or error

# Now, it's completely safe to use all the configuration methods as usual
# on the one fw variable.
obj = AddressObject('test', '10.0.1.1')
fw.add(obj)
obj.create()
```

In the above code, we added the AddressObject to the `fw` variable. Even though we created this above with the IP of 10.0.0.1, it represents both firewalls in the pair. So when we create the AddressObject on the live device, pan-os-python will reach out to the active firewall in the pair. It will automatically detect if the active failed and switch to standby.

Note: We didn't save the second firewall to a variable, because our `fw` variable represents both firewalls, but if you need to access the second firewall as a variable, it's available to you at `fw.ha_peer`.

3.4 Optimize with Bulk Operations

Each API call takes time and consumes management plane resources on the firewall or Panorama. While this won't affect traffic, it does limit the number of changes that can take place in a time period.

Example: if you're adding policy for all your branch offices and need to add 200 address groups with 20 address objects each, creating them individually would be 200 x 20 + 200 = 4200 API calls. If your device can process an API call in 1 second, then this operation would take *over an hour* to complete. Even if you applied concurrency up to 5 API calls simultaneously, it's still over 10 minutes of waiting.

We can do this faster with **bulk operations**.

The methods used to push these objects to a live device individually are `create()`, `apply()`, and `delete()`. Each of these has a bulk counterpart: `create_similar()`, `apply_similar()`, and `delete_similar()`.

The bulk version of the method is called exactly the same way as the individual version, but the behavior is different. Instead of sending this single object to the device, all objects in the configuration tree with this type and location in the tree are pushed to the live device in a single API call.

Here's code for the above example using individual API calls and using bulk operations:

```
from panos.firewall import Firewall
from panos.objects import AddressObject, AddressGroup

# Build out the configuration tree with a Firewall object at the root and an
# array of AddressObjects and AddressGroups as children of the Firewall
fw1 = Firewall('10.0.0.1', 'admin', 'password')
# Create 200 AddressGroups with 20 AddressObjects each
for i in range(0, 200):
    addr_objects = [AddressObject('object{}'.format(i*20+j), '192.168.{0}.{1}'.
↪format(i, j)) for j in range(0, 20)]
    fw.extend(addr_objects)
    grp = AddressGroup('group{}'.format(i), addr_objects)
    fw.add(grp)

# The config tree is built, now we need to push it to the live device.

# Option 1: Push each address object and group one at a time
#           (takes over 1 hour)
for obj in fw.findall(AddressObject):
    obj.create()
for grp in fw.findall(AddressGroup):
    grp.create()

# Option 2: Push all the address objects at once, then all the address groups at once
#           (takes 2-3 seconds)
fw.find('object1').create_similar()
fw.find('group1').create_similar()
```

Bulk operations for the win!

One thing to keep in mind when using bulk operations is that the methods will push any objects that share the same type and **location**. This means if you call a bulk operation method on an AddressObject in vsys2, pan-os-python will NOT push the AddressObjects in vsys3, or Device Group 7, or the shared scope. Under the hood, it verifies that the objects share the same XPath and type before they are pushed to the live device.

3.5 Connect to PAN-OS 8.0 and higher

Starting in PAN-OS 8.0, the default TLS version has changed from 1.0 to 1.1 to enhance the security of the management connection. This can cause connection problems for systems with older OpenSSL versions that don't support TLS 1.1, such as MacOSX Sierra. TLS 1.1 is supported in OpenSSL 1.0.1 and higher.

Suggestions for connecting to PAN-OS 8.0

Options 1:

If using OSX, install homebrew, then use homebrew to install python. Python from homebrew will come with an updated OpenSSL version, and it is best practice to install it anyway to prevent pollution of your system python.

After installing homebrew using the instructions on their website, type the following in an OSX termainal to install python:

```
brew install python
```

Option 2:

Upgrade OpenSSL using your OS package manager. For example, in Ubuntu you would type *apt-get install openssl*. If a newer OpenSSL is not available, upgrade the OS distribution to a newer version. The procedure will differ depending on your OS distro. Please refer to the instructions for upgrading your OS.

Option 3:

Set the firewall minimum TLS version back to TLS 1.0. To do this, in the Device tab, create a self-signed CA certificate on the firewall and assign it to a new SSL/TLS Service Profile with the Minimum TLS version set to TLS 1.0. Then, assign the SSL/TLS Server Profile to the management interface at Device tab -> Setup -> Management -> General Settings.

CHAPTER 4

Examples

4.1 Example scripts

There are several example scripts written as CLI programs in the [examples directory](https://github.com/PaloAltoNetworks/pan-os-python/tree/develop/examples).

4.2 Cookbook examples

4.2.1 Get the version of a firewall

```
from panos.firewall import Firewall

fw = Firewall("10.0.0.1", "admin", "mypassword")
version = fw.refresh_system_info().version
print version
```

Example output:

```
10.0.3
```

We use `refresh_system_info()` here instead of an op commands because this method saves the version information to the Firewall object which tells all future API calls what format to use to be compatible with this version.

4.2.2 Print a firewall rule

```
from panos.firewall import Firewall
from panos.policies import Rulebase, SecurityRule

# Create a config tree for the rule
```

(continues on next page)

(continued from previous page)

```
fw = Firewall("10.0.0.1", "admin", "mypassword", vsys="vsys1")
rulebase = fw.add(Rulebase())
rule = rulebase.add(SecurityRule("my-rule"))

# Refresh the rule from the live device and print it
rule.refresh()
print(rule.about())
```

4.2.3 List of firewall rules by name

```
from panos.firewall import Firewall
from panos.policies import Rulebase, SecurityRule

# Create config tree and refresh rules from live device
fw = Firewall("10.0.0.1", "admin", "mypassword", vsys="vsys1")
rulebase = fw.add(Rulebase())
rules = SecurityRule.refreshall(rulebase)

for rule in rules:
    print(rule.name)
```

4.2.4 List of pre-rules on Panorama

```
from panos.panorama import Panorama
from panos.policies import PreRulebase, SecurityRule

# Create config tree and refresh rules from live device
pano = Panorama("10.0.0.1", "admin", "mypassword")
pre_rulebase = pano.add(PreRulebase())
rules = SecurityRule.refreshall(pre_rulebase)

for rule in rules:
    print(rule.name)
```

4.2.5 List firewall devices in Panorama

Print the serial, hostname, and management IP of all firewalls that Panorama knows about.

```
from panos.panorama import Panorama
from panos.device import SystemSettings

# Create config tree root
pano = Panorama("10.0.0.1", "admin", "mypassword")

# Refresh firewalls from live Panorama
devices = pano.refresh_devices(expand_vsys=False, include_device_groups=False)

# Print each firewall's serial and management IP
for device in devices:
    system_settings = device.find("", SystemSettings)
    print(f"{device.serial} {system_settings.hostname} {system_settings.ip_address}")
```

Example output:

```
310353000003333 PA-VM-1 10.1.1.1
310353000003334 PA-VM-2 10.1.1.2
```

4.2.6 Upgrade a firewall

```
from panos.firewall import Firewall

fw = Firewall("10.0.0.1", "admin", "mypassword")
fw.software.upgrade_to_version("10.1.5")
```

This simple example will upgrade from any previous version to the target version and handle all intermediate upgrades and reboots.

CHAPTER 5

API Reference

5.1 Useful Methods

These methods are the most commonly used and can be called on any object in a configuration tree.

5.1.1 Configuration Methods

Modify the configuration tree or the live device with these methods.

- **C:** Changes the pan-os-python configuration tree
- **L:** Connects to a live device (firewall or Panorama) via the API
- **M:** Modifies the live device by making a change to the device's configuration
- **B:** Bulk operation modifies more than one object in a single API call

Method	C	L	M	B	Description
add()					Add an object as a child of this object
extend()					Add a list of objects as children
insert()					Insert an object as a child at an index
pop()					Remove a child object at an index
remove()					Remove a child object from this object
remove_by_name()					Remove a child object by its name
removeall()					Remove all children of this object
refresh()					Set params of object from live device
refreshall()					Pull all children from the live device
refresh_variable()					Set a single param from the live device
create()					Push object to the live device (nd)
apply()					Push object to the live device (d)
update()					Push single object param to live device
delete()					Delete from live device and config tree
rename()					Rename on live device and config tree
move()					Reorder on live device and config tree
create_similar()					Push objects of this type to live device (nd)
apply_similar()					Push objects of this type to live device (d)
delete_similar()					Delete objects of this type from live device

- (d): Destructive - Method *overwrites* an object on the live device with the same name
- (nd): Non-destructive - Method *combines* object with one on live device with the same name

5.1.2 Navigation Methods

These methods help you locate objects and information in an existing configuration tree. These are commonly used when you have used `refreshall` to pull a lot of nested objects and you're either looking for a specific object or aggregate stats on the objects.

Method	Description
find()	Return object by name and type
findall()	Return all objects of a type
find_index()	Return the index of a child object
find_or_create()	Return object by name and type, creates object if not in config tree
findall_or_create()	Return all objects of type, creates an object if none exist
nearest_pandevice()	Return the nearest parent Firewall or Panorama object in tree
panorama()	Return the nearest parent Panorama object
devicegroup()	Return the nearest parent DeviceGroup object
vsys	Return the vsys that contains this object

5.1.3 Informational Methods

These methods provide information about an object in the configuration tree.

Method	Description
about()	Return all the params set on this object and their values
equal()	Test if two objects are equal and return a boolean
xpath()	Return the XPath of this object
element()	Return the XML of this object as an ElementTree
element_str()	Return the XML of this object as a string

5.1.4 Device Methods

These methods can be called on a PanDevice object (a Firewall or Panorama), but not on any other PanObject.

Method	Description
refresh_system_info()	Return and retain important information about the device
commit()	Trigger a commit on a Firewall or Panorama
commit_all()	Trigger a configuration push from Panorama to the Firewalls
syncjob()	Wait for a job on the device to finish
refresh_devices()	Pull all the devices attached to Panorama as Firewall objects
op()	Execute an operational command
watch_op()	Same as 'op', then watch for a specific result

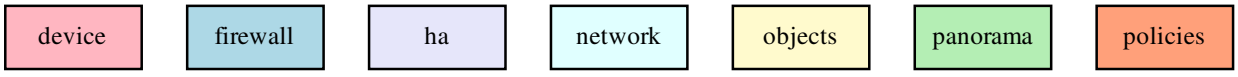
There are many other convenience methods available. They're all documented in the *PanDevice* class.

5.2 Configuration tree diagrams

These diagrams illustrates the possible tree structures for a Firewall/Panorama configuration.

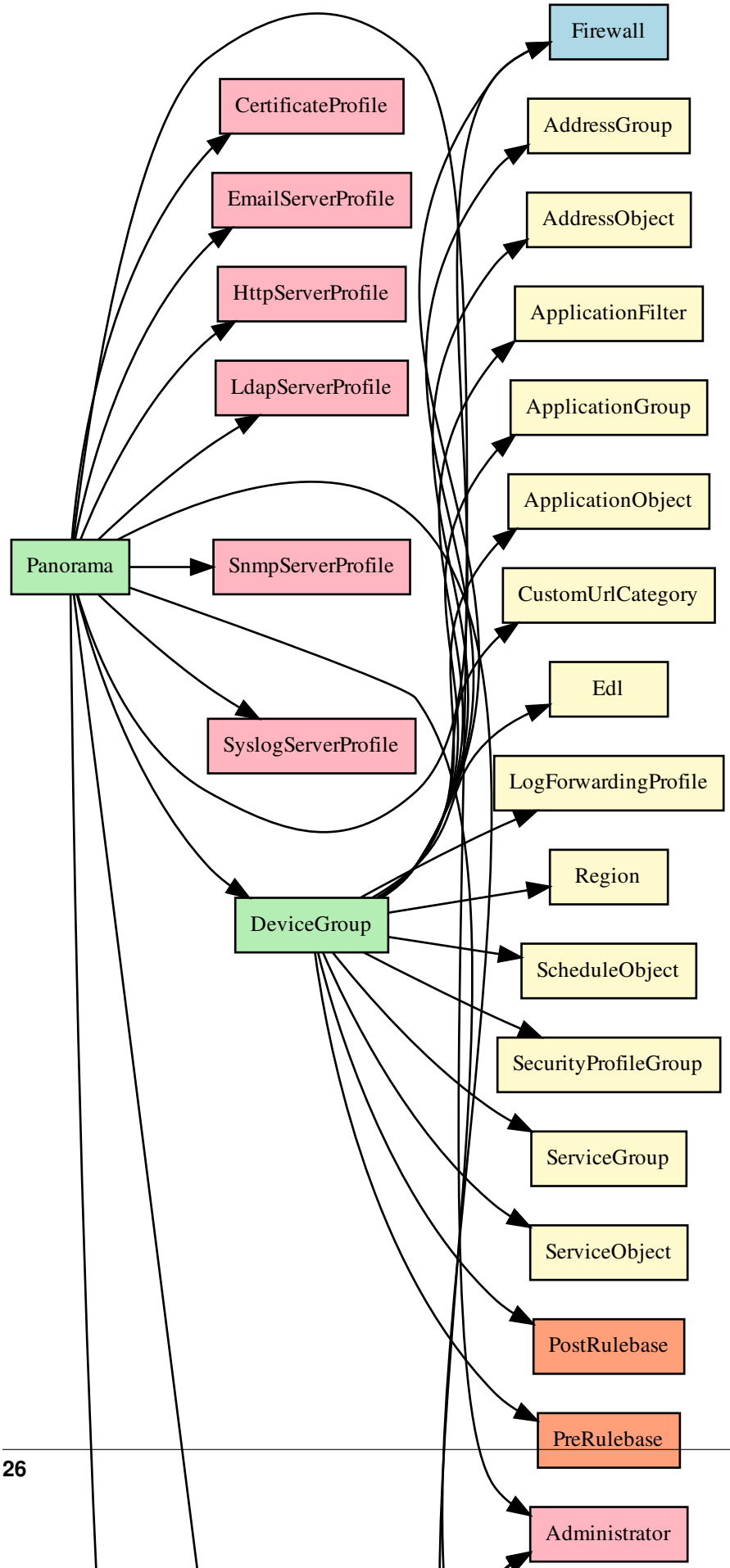
The tree diagrams are broken out into partial diagrams by module or function for better readability. The nodes are color coded by the module they are in according to the legend.

5.2.1 Module Legend

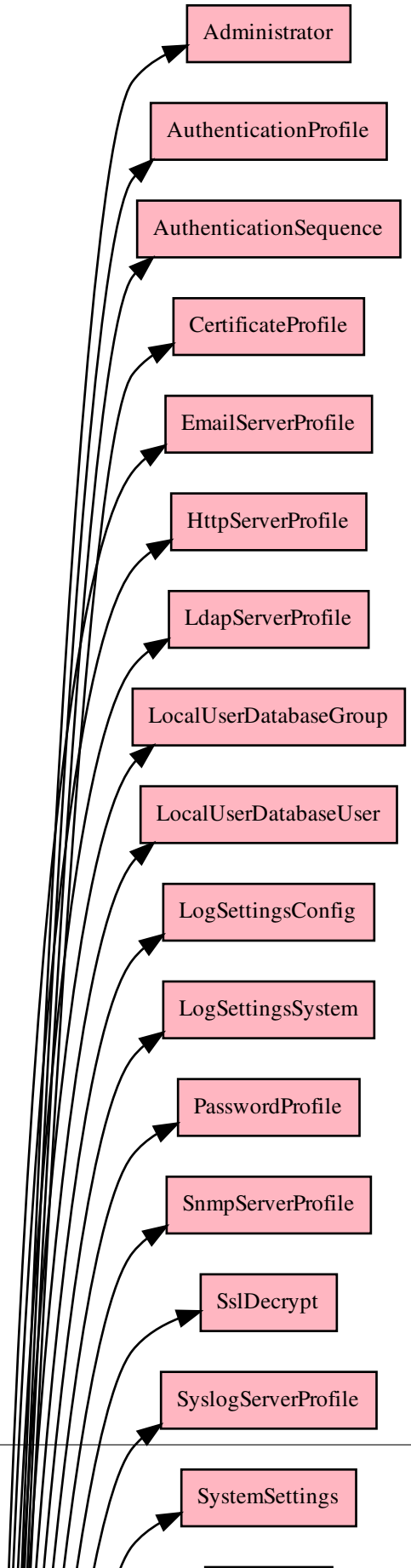


5.2.2 Panorama

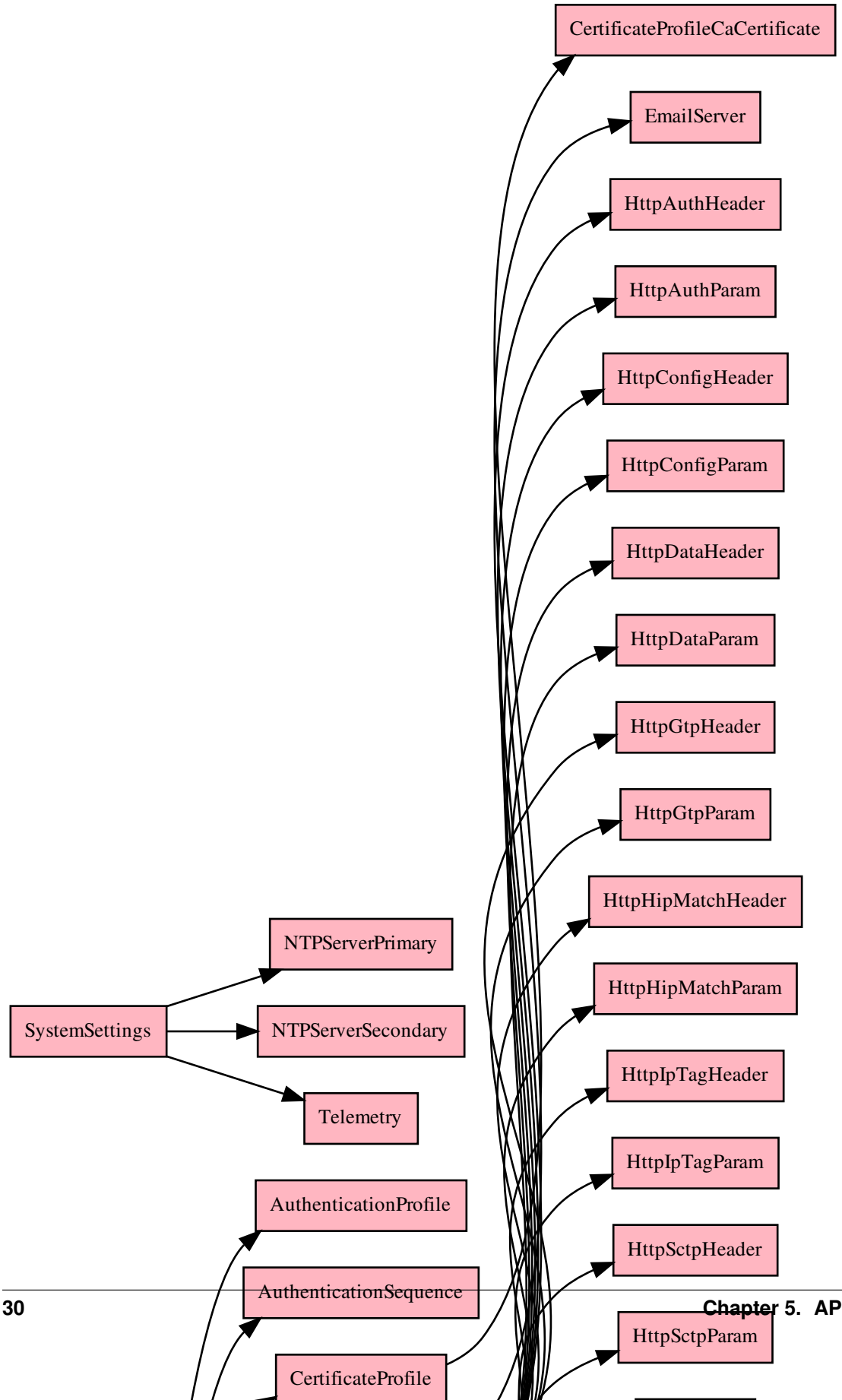
A Panorama object can contain a DeviceGroup or Firewall, each of which can contain configuration objects. (see *Firewall* below for objects that can be added to the Firewall object)


Firewall
CertificateProfile
AddressGroup
EmailServerProfile
AddressObject
HttpServerProfile
ApplicationFilter
LdapServerProfile
ApplicationGroup
ApplicationObject
Panorama
SnmpServerProfile
CustomUrlCategory
Edl
SyslogServerProfile
LogForwardingProfile
Region
DeviceGroup
ScheduleObject
SecurityProfileGroup
ServiceGroup
ServiceObject
PostRulebase
PreRulebase
Administrator

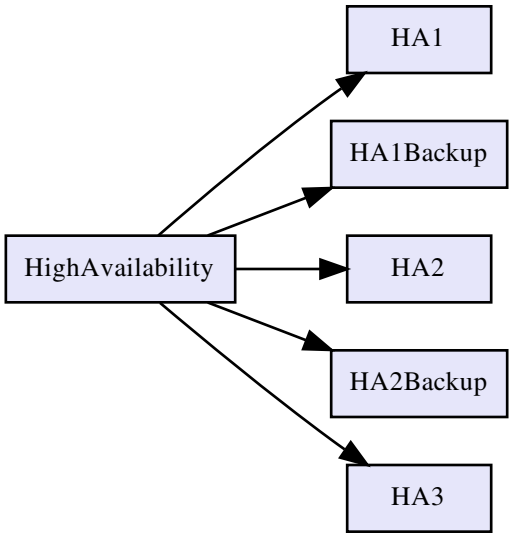
5.2.3 Firewall



5.2.4 Device


CertificateProfileCaCertificate
EmailServer
HttpAuthHeader
HttpAuthParam
HttpConfigHeader
HttpConfigParam
HttpDataHeader
HttpDataParam
HttpGtpHeader
HttpGtpParam
HttpHipMatchHeader
HttpHipMatchParam
HttpIpTagHeader
HttpIpTagParam
HttpSctpHeader
HttpSctpParam
NTPServerPrimary
SystemSettings
NTPServerSecondary
Telemetry
AuthenticationProfile
AuthenticationSequence
CertificateProfile

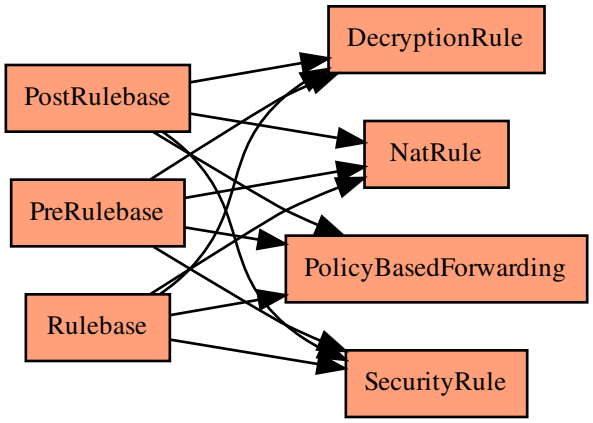
5.2.5 HA



5.2.6 Network

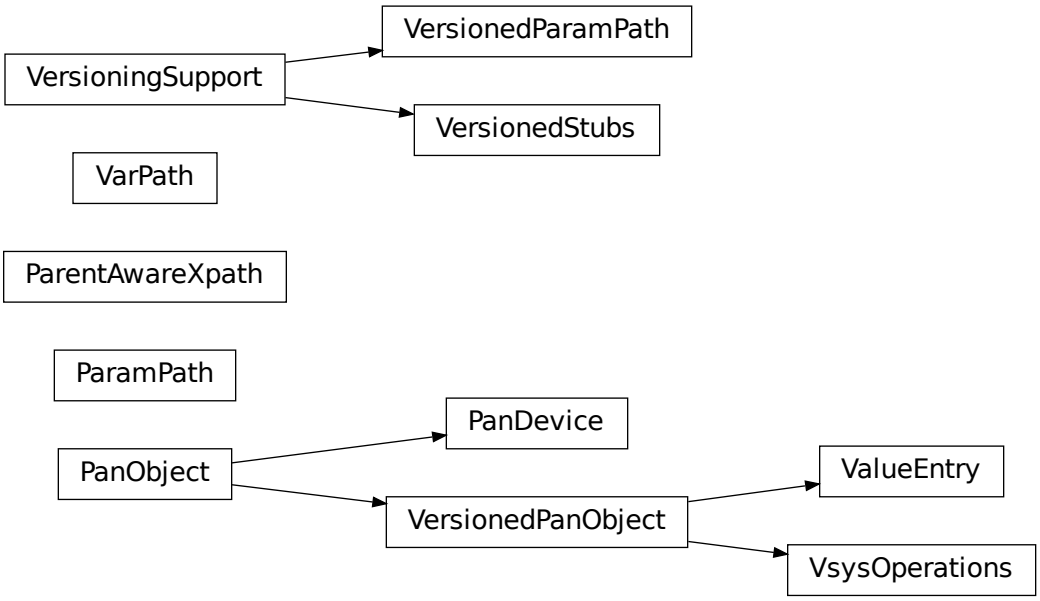

Vlan
StaticMac
IpsecTunnel
IpsecTunnelIpv4ProxyId
IpsecTunnelIpv6ProxyId
Dhcp
DhcpRelay
DhcpRelayIpv6Address
BgpAuthProfile
BgpDampeningProfile
BgpPeer
BgpPeerGroup
BgpPolicyAdvertiseFilter
BgpPolicyAggregationAddress
BgpPolicySuppressFilter
BgpPolicyConditionalAdvertisement
BgpPolicyNonExistFilter
BgpPolicyAddressPrefix
Bgp
BgpPolicyExportRule
BgpPolicyImportRule
BgpRedistributionRule
BgpOutboundRouteFilter
BgpRoutingOptions
OspfAreaInterface
OspfNeighbor
OspfArea
OspfNssaExternalRange
Ospf
OspfAuthProfile
OspfRange
OspfExportRules
OspfAuthProfileMd5
RedistributionProfile
VirtualRouter
RedistributionProfileIPv6
RipAuthProfile
RipAuthProfileMd5
Rip
RipExportRule
RipInterface
StaticRoute
StaticRouteV6
PathMonitorDestination
Arp
VlanInterface
AggregateInterface
Layer2Subinterface
EthernetInterface
Layer3Subinterface
IPv6Address

5.2.7 Policies



5.3 Module: base

5.3.1 Inheritance diagram



5.3.2 Class Reference

Base object classes for inheritence by other classes

class panos.base.**PanDevice**(*hostname*, *api_username=None*, *api_password=None*, *api_key=None*, *port=443*, *is_virtual=None*, *timeout=1200*, *interval=0.5*, **args*, ***kwargs*)

A Palo Alto Networks device

The device can be of any type (currently supported devices are firewall, or panorama). The class handles common device functions that apply to all device types.

Usually this class is not instantiated directly. It is the base class for a firewall.Firewall object or a panorama.Panorama object.

Parameters

- **hostname** – Hostname or IP of device for API connections
- **api_username** – Username of administrator to access API
- **api_password** – Password of administrator to access API
- **api_key** – The API Key for connecting to the device's API
- **port** – Port of device for API connections
- **is_virtual** (*bool*) – Physical or Virtual firewall
- **timeout** – The timeout for asynchronous jobs
- **interval** – The interval to check asynchronous jobs

ha_peer

The HA peer device of this PanDevice

Type *PanDevice*

activate()

Make this PanDevice active and the other passive

activate_feature_using_authorization_code(*code*)

Updates a license using the given auth code.

Modifies the live device

Parameters **code** (*str*) – The authorization code.

Raises PanActivateFeatureAuthCodeError

active()

Return the active device in the HA Pair

clock()

Gets the current time on PAN-OS.

Returns datetime.datetime

commit(*sync=False*, *exception=False*, *cmd=None*, *admins=None*, *sync_all=False*)

Trigger a commit

Parameters

- **sync** (*bool*) – Block until the commit is finished (Default: False)
- **exception** (*bool*) – Create an exception on commit errors (Default: False)
- **cmd** (*str*) – Commit options in XML format
- **admins** (*str/list*) – name or list of admins whose changes need to be committed

- **sync_all** (*bool*) – If this is a Panorama commit, wait for firewalls jobs to finish (Default: False)

Returns Commit results

Return type dict

config_sync_state()
Get the current configuration synchronization state from the live device

Returns Current configuration sync state, or None if HA is not enabled

Return type str

config_synced()
Check if configuration is synchronized between HA peers

Returns True if synchronized, False if not

Return type bool

classmethod create_from_device(*hostname*, *api_username=None*, *api_password=None*, *api_key=None*, *port=443*)
Factory method to create a *panos.firewall.Firewall* or *panos.panorama.Panorama* object from a live device

Connects to the device and detects its type and current state in order to create a PanDevice subclass.

Parameters

- **hostname** – Hostname or IP of device for API connections
- **api_username** – Username of administrator to access API
- **api_password** – Password of administrator to access API
- **api_key** – The API Key for connecting to the device's API
- **port** – Port of device for API connections

Returns New subclass instance (Firewall or Panorama instance)

Return type *PanDevice*

fetch_licenses_from_license_server()
Fetches licenses from the license server.

Modifies the live device

Note: For namedtuple objects, you can access the variables via its index like a normal tuple or via name like a class.

Returns

A list of namedtuples of the licenses with the following attributes:

- feature (str): the feature name
- description (str): description
- serial (str): the license's serial number
- issued (datetime.date/None): issue date
- expires (datetime.date/None): expiration date, or None if the license does not expire
- expired (bool): True if the license is currently expired
- authcode (str/None): license's authcode

Return type list

get_device_version()
Gets the current version on the PanDevice.

ha_pair()
List containing this firewall and its HA peer

Returns

self and self.ha_peer in a list. If there is not ha_peer, then a single item list containing only self is returned.

Return type list

is_active()
Return True if this device is active

map_ha(*method_name*, **args*, ***kwargs*)
Apply to both devices in HA Pair

Invoke a method of this class on both this instance and its HA peer

Parameters

- **method_name** – The name of the method in this class (or subclass) to invoke
- ***args** – Arguments to pass to the method
- ****kwargs** – Keyword arguments to pass to the method

Returns A tuple of the return values of invoking the method on each device. The first item in the tuple is always from invoking the method on self, and the second item is from invoking the method on the ha_peer. The second item is None if there is no HA Peer.

nearest_pandevice()
The nearest *panos.base.PanDevice* object.

This method is used to determine the device to apply this object to.

Returns

The PanDevice object closest to this object in the configuration tree.

Return type *PanDevice*

Raises PanDeviceNotSet – There is no PanDevice object in the tree.

op(*cmd=None*, *vsys=None*, *xml=False*, *cmd_xml=True*, *extra_qs=None*, *retry_on_peer=False*)
Perform operational command on this device

Operational commands are most any command that is not a debug or config command. These include many 'show' commands such as `show system info`.

When passing the cmd as a command string (not XML) you must include any non-keyword strings in the command inside double quotes ("). Here's some examples:

```
# The string "facebook-base" must be in quotes because it is not a keyword
fw.op('clear session all filter application "facebook-base"')

# The string "ethernet1/1" must be in quotes because it is not a keyword
fw.op('show interface "ethernet1/1"')
```

Parameters

- **cmd** (*str*) – The operational command to execute

- **vsys** (*str*) – Vsys id.
- **xml** (*bool*) – Return value should be a string (Default: False)
- **cmd_xml** (*bool*) – True: cmd is not XML, False: cmd is XML (Default: True)
- **extra_qs** – Extra parameters for API call
- **retry_on_peer** (*bool*) – Try on active Firewall first, then try on passive Firewall

Returns The result of the operational command. May also return a string of XML if xml=True

Return type xml.etree.ElementTree

passive()
Return the passive device in the HA Pair

pending_changes(*retry_on_peer=True*)
Check if there are pending changes on the live device

Parameters **retry_on_peer** (*bool*) – Try on active Firewall first, if connection error try on passive Firewall

Returns True if pending changes, False if not

Return type bool

plugins()
Returns plugin information.

Each dict in the list returned has the following keys:

- name
- version
- release_date
- release_note_url
- package_file
- size
- platform
- installed
- downloaded

Returns list of dicts

predefined = None
Predefined object subsystem

See Also: *panos.predefined*

refresh_ha_active()
Refresh which device is active using the live device

Returns Current HA state of this device

Return type str

refresh_system_info()
Refresh system information variables.

Variables refreshed:

- PAN-OS version
- platform
- serial
- content version (if this is a *panos.firewall.Firewall*)
- multi_vsys (if this is a *panos.firewall.Firewall*)

Returns version, platform, serial

Return type namedtuple

refresh_version()
Refresh version of PAN-OS

Version is stored in self.version and returned

Returns version of PAN-OS

Return type str

request_license_info()
Returns the licenses currently installed on this device.

Touches the live device

Note: For namedtuple objects, you can access the variables via its index like a normal tuple or via name like a class.

Returns

A list of namedtuples of the licenses with the following attributes:

- feature (str): the feature name
- description (str): description
- serial (str): the license's serial number
- issued (datetime.date/None): issue date
- expires (datetime.date/None): expiration date, or None if the license does not expire
- expired (bool): True if the license is currently expired
- authcode (str/None): license's authcode

Return type list

request_password_hash(*value*)
Request a password hash from the live device.

This function does not modify the live device, but it does interact with the live device to generate the password hash.

Parameters **value** (*str*) – The password

Returns A hashed version of the password provided.

Return type str

Raises `ValueError` – If the password hash is not found.

set_config_changed(*scope=None*)
Set flag that configuration of this device has changed

This is useful for checking if a commit is necessary by knowing if the configuration was actually changed. This method is already used by every pan-os-python package method that makes a configuration change. But this method could also by run directly to force a 'dirty' configuration state in a PanDevice object.

Parameters **scope** – vsys in which configuration was changed, or 'shared'

set_dns_servers(*primary*, *secondary=None*)
Set the device DNS Servers

Convenience method to set the firewall or Panorama dns servers

Parameters

- **primary** (*str*) – IP address of primary DNS server
- **secondary** (*str*) – IP address of secondary DNS server

set_failed()
Set this PanDevice as a failed HA Peer

API calls will no longer be attempted to this device until one of the following conditions:

1. self.ha_failed is set to False
2. self.ha_failed is set to True on the peer device

Returns The HA Peer device

Return type *PanDevice*

set_ha_peers(*device*)
Establish an HA peer relationship between two PanDevice objects

Parameters **device** – The HA peer device

set_hostname(*hostname*)
Set the device hostname

Convenience method to set the firewall or Panorama hostname

Parameters **hostname** (*str*) – hostname to set (should never be None)

set_ntp_servers(*primary*, *secondary=None*)
Set the device NTP Servers

Convenience method to set the firewall or Panorama NTP servers

Parameters

- **primary** (*str*) – IP address of primary DNS server
- **secondary** (*str*) – IP address of secondary DNS server

show_system_info()
Returns the data from "show system info".

Returns dict

synchronize_config()
Force configuration synchronization from this device to its HA peer

syncjob(*job_id*, *sync_all=False*, *interval=0.5*)
Block until job completes and return result

Parameters

- **job_id** (*int*) – job ID, or response XML from job creation
- **sync_all** (*bool*) – Wait for all devices to complete if commit all operation
- **interval** (*float*) – Interval in seconds to check if job is complete

Returns Job result

Return type dict

syncreboot(*interval=5.0*, *timeout=600*)
Block until reboot completes and return version of device

test_security_policy_match(*source*, *destination*, *protocol*, *application=None*, *category=None*, *port=None*, *user=None*, *from_zone=None*, *to_zone=None*, *show_all=False*)
Test security policy match using the given criteria.

This function will always return a list for its results. If *show_all* is set to False, then the list will only have one entry in it. The keys in each dict are as follows:

- name (str): rule's name
- index (int): the index of the security rule
- action (str): the security rule's action

Parameters

- **source** (*str*) – Source IP address.
- **destination** (*str*) – Destination IP address.
- **protocol** (*int*) – IP protocol value (1-255).
- **application** (*str*) – Application name.
- **category** (*str*) – Category name.
- **port** (*int*) – Destination port.
- **user** (*str*) – Source user.
- **from_zone** (*str*) – Source zone name.
- **to_zone** (*str*) – Destination zone name.
- **show_all** (*bool*) – Show all potential match rules until first allow.

Returns List of dicts

toggle_ha_active()
Switch the active device in this HA Pair

update_connection_method()
Regenerate the xapi object used to connect to the device

This is only necessary if the API key, password, hostname, or other connectivity information in this object has changed. In this case, the xapi object used to communicate with the firewall must be regenerated to use the new connectivity information.

The new xapi is stored in the PanDevice object and returned.

Returns The xapi object which is also stored in self.xapi.

Return type XapiWrapper

userid = None
User-ID subsystem

See Also: *panos.userid*

watch_op(*cmd*, *path*, *value*, *vsys=None*, *cmd_xml=True*, *interval=1.0*)
Watch an operational command for an expected value

Blocks script execution until the value exists or timeout expires

Parameters

- **cmd** (*str*) – Operational command to run
- **path** (*str*) – XPath to the value to watch
- **value** (*str*) – The value expected before method completes
- **vsys** (*str*) – Vsys id for the operational command
- **cmd_xml** (*bool*) – True: cmd is not XML, False: cmd is XML (Default: True)
- **interval** (*float*) – Interval in seconds to check if the value exists

whoami()
Returns which user you're currently authenticated as.

NOTE: PAN-OS 10.0+

Returns string

class panos.base.**PanObject**(**args*, ***kwargs*)
Base class for all package objects

This class defines an object that can be placed in a tree to generate configuration.

Parameters **name** (*str*) – The name of this object

uid
The unique identifier for this object if it has one. If it doesn't have one, then this returns the class name.

Type str

vsys
The vsys id for this object (eg. 'vsys2') or 'shared' if no vsys

Type str

about(*parameter=None*)
Return information about this object or the given parameter.

If no parameter is specified, then invoking this function is similar to doing *vars(obj)*: it will return a dict of key/value pairs, with the difference being that the keys are all specifically parameters attached to this *VersionedPanObject*, and the values being what the current settings are for those keys.

If a parameter is specified and this object is connected to a parent PanDevice, then version specific information on the parameter is returned.

If a parameter is specified but this object is not connected to a PanDevice instance, then all versioning information for the given parameter is returned.

Parameters **parameter** (*str*) – The parameter to get info for.

Returns An informational dict about either the object as a whole or the specified parameter.

Return type dict

Raises `AttributeError` – If a parameter is specified that does not exist on this object.

add(*child*)
Add a child node to this node

Parameters **child** (`PanObject`) – Node to add as a child

Returns Child node

Return type *PanObject*

apply()
Apply this object to the device, replacing any existing object of the same name

Modifies the live device

apply_similar()
Bulk apply all objects similar to this one.

Modifies the live device

This is similar to apply(), except instead of calling apply only on this object, it calls apply for all objects that share the same xpath as this object, recursively searching the entire object tree from the nearest firewall or panorama instance.

As an example, if you called apply_similar on an object representing ethernet1/5.42, all of the subinterfaces for ethernet1/5 would be included in the resulting XML document, regardless of which vsys those subinterfaces existed in.

Since apply does a replace of the config at the given xpath, please be careful when using this function that all objects, whether they be updated or not, exist in your pan-os-python object tree.

create()
Create this object on the device

Modifies the live device

This method is nondestructive. If the object exists, the variables are added to the device without changing existing variables on the device. If a variables already exists on the device and this object has a different value, the value on the firewall is changed to the value in this object.

create_similar()
Bulk create all objects similar to this one.

Modifies the live device

This is similar to create(), except instead of calling create only on this object, it calls create for all objects that share the same xpath as this object, recursively searching the entire object tree from the nearest firewall or panorama instance.

As an example, if you called create_similar on an object representing ethernet1/5.42, all of the subinterfaces for ethernet1/5 would be included in the resulting XML document, regardless of which vsys those subinterfaces existed in.

delete()
Delete this object from the firewall

Modifies the live device

delete_similar()
Bulk delete all objects similar to this one.

Modifies the live device

This is similar to delete(), except instead of calling delete only on this object, it calls delete for all objects that share the same xpath as this object, recursively searching the entire object tree from the nearest firewall or panorama instance.

As an example, if you called delete_similar on an object representing ethernet1/5.42, all of the subinterfaces in your pan-os-python object tree for ethernet1/5 would be removed.

devicegroup()
The nearest *panos.panorama.DeviceGroup* object.

This method is used to determine the device to apply this object to.

Returns The DeviceGroup object closest to this object in the configuration tree, or None if there is no DeviceGroup in the path to this node.

Return type *DeviceGroup*

element(*with_children=True*, *comparable=False*)
Construct an ElementTree for this PanObject and all its children

Parameters

- **with_children** (*bool*) – Include children in element.
- **comparable** (*bool*) – Element will be used in a comparison with another.

Returns

An ElementTree instance representing the xml form of this object and its children

Return type xml.etree.ElementTree

element_str(*pretty_print=False*)
The XML representation of this PanObject and all its children.

Parameters **pretty_print** (*bool*) – Return the resulting string pretty_printed with indentation.

Returns XML form of this object and its children

Return type str

equal(*panobject*, *force=False*, *compare_children=True*)
Compare this object to another object

Equality of the objects is determined by the XML they generate, not by the values of their variables.

Parameters

- **panobject** (PanObject) – The object to compare with this object
- **force** (*bool*) – Do not raise a PanObjectError if the objects are different classes
- **compare_children** (*bool*) – Not supported in this object, use True

Raises PanObjectError – Raised if the objects are different types that would not normally be comparable

Returns True if the XML of the objects is equal, False if not

Return type bool

extend(*children*)
Add a list of child nodes to this node

Parameters **children** (*list*) – List of PanObject instances

find(*name*, *class_type=None*, *recursive=False*)
Find an object in the configuration tree by name

Parameters

- **name** (*str*) – Name of the object to find
- **class_type** – Class to look for
- **recursive** (*bool*) – Find recursively (Default: False)

Returns The object in the tree that fits the criteria, or None if no object is found

Return type *PanObject*

find_index(*name=None*, *class_type=None*)
Finds the first index of the given name and class type.

If name is None, just find the first instance of class_type.

If class_type is unspecified, it defaults to the current class type.

Parameters

- **name** (*str*) – Name of the child node
- **class_type** (*class*) – Restrict the find to children of this type

Returns the index of the first matching child

Return type int

find_or_create(*name*, *class_type*, **args*, ***kwargs*)
Find an object in the configuration tree by name, and create it if it doesn't exist

If the object does not exist, it is created and added to the current object.

Parameters

- **name** (*str*) – Name of the object to find
- **class_type** – Class to look for or create
- ***args** – Arguments to pass to the __init__ method of class_type
- ***kwargs** – Keyworkd arguments to pass to the __init__ method of class_type

Returns The object in the tree that fits the criteria, or None if no object is found

Return type *PanObject*

findall(*class_type*, *recursive=False*)
Find all objects of a class in configuration tree

Parameters

- **class_type** – Class to look for
- **recursive** (*bool*) – Find recursively (Default: False)

Returns List of 'class_type' objects

Return type list

findall_or_create(*class_type*, **args*, ***kwargs*)
Find all object in the configuration tree by class, and create a new object if none exist

If no objects of this type exist, one is created and added to the current object.

Parameters

- **class_type** – Class to look for or create
- ***args** – Arguments to pass to the __init__ method of class_type
- ****kwargs** – Keyworkd arguments to pass to the __init__ method of class_type

Returns List of 'class_type' objects

Return type list

fulltree()
Display a graph of the entire configuration tree

This method is only for use in Jupyder Notebooks

insert(*index*, *child*)
Insert a child node at a specific index

This is useful for ordering or reordering security policy rules

Parameters

- **index** (*int*) – The index where the child obj should be inserted
- **child** (PanObject) – Node to add as a child

Returns Child node

Return type *PanObject*

move(*location*, *ref=None*, *update=True*)
Moves the current object.

Modifies the live device

This is useful for stuff like moving one security policy above another.

If this object's parent is a rulebase object, then this object is also moved to the appropriate position in the local pan-os-python object tree.

Parameters

- **location** (*str*) – Any of the following: before, after, top, or bottom
- **ref** (*PanObject/str*) – If location is "before" or "after", move this object before/after the ref object. If this is a string, then the string should just be the name of the object.
- **update** (*bool*) – If this is set to False, then only move this object in the pan-os-python object tree, do not actually perform the MOVE operation on the live device. Note that in order for this object to be moved in the pan-os-python object tree, the parent object must be a rulebase object.

Raises ValueError

nearest_pandevice()
The nearest *panos.base.PanDevice* object to.

This method is used to determine the device to apply this object.

Returns

The PanDevice object closest to this object in the configuration tree.

Return type *PanDevice*

Raises PanDeviceNotSet – There is no PanDevice object in the tree.

panorama()
The nearest *panos.panorama.Panorama* object.

This method is used to determine the device to apply this object to.

Returns

The Panorama object closest to this object in the configuration tree

Return type *Panorama*

Raises PanDeviceNotSet – There is no Panorama object in the tree.

pop(*index*)
Remove and return the object at an index

Parameters **index** (*int*) – Index of the object to remove and return

Returns The object removed from the children of this node

Return type *PanObject*

refresh(*running_config=False*, *refresh_children=True*, *exceptions=True*, *xml=None*)
Refresh all variables and child objects from the device.

Parameters

- **running_config** (*bool*) – Set to True to refresh from the running configuration (Default: False)
- **xml** (*xml.etree.ElementTree*) – XML from a configuration to use instead of refreshing from a live device
- **refresh_children** (*bool*) – Set to False to prevent refresh of child objects (Default: True)
- **exceptions** (*bool*) – Set to False to prevent exceptions on failure (Default: True)

refresh_variable(*variable*, *running_config=False*, *exceptions=False*)
Refresh a single variable of an object.

Don't use for variables with replacements or selections in path.

Parameters

- **variable** (*str*) – Variable name to refresh.
- **running_config** (*bool*) – Set to True to refresh from the running configuration (Default: False)
- **exceptions** (*bool*) – Set to False to prevent exceptions on failure (Default: True)

Returns New value of the refreshed variable.

Raises PanObjectMissing – When the object this variable is connected to does not exist.

classmethod refreshall(*parent*, *running_config=False*, *add=True*, *exceptions=False*, *name_only=False*)
Factory method to instantiate class from live device.

This method is a factory for the class. It takes an firewall or Panorama and gets the xml config from the live device. It generates instances of this class for each item this class represents in the xml config. For example, if the class is AddressObject and there are 5 address objects on the firewall, then this method will generate 5 instances of the class AddressObject.

Parameters

- **parent** (PanObject) – A PanDevice, or a PanObject subclass with a PanDevice as its parental root.
- **running_config** (*bool*) – False for candidate config, True for running config.
- **add** (*bool*) – Update the objects of this type in pan-os-python with the refreshed values.
- **exceptions** (*bool*) – If False, exceptions are ignored if the xpath can't be found.
- **name_only** (*bool*) – If True, refresh only the name of the object, but not its variables. This results in a smaller response to the API call when only the object name is needed.

Returns created instances of class

Return type list

refreshall_from_xml(*xml*, *refresh_children=True*, *variables=None*)

Factory method to instantiate class from firewall config.

This method is a factory for the class. It takes an xml config from a firewall and generates instances of this class for each item this class represents in the xml config. For example, if the class is AddressObject and there are 5 address objects on the firewall, then this method will generate 5 instances of the class AddressObject.

Parameters

- **xml** (*xml.etree.ElementTree*) – A section of XML configuration from a firewall or Panorama. It should not contain the response or result tags.
- **refresh_children** (*bool*) – Refresh children objects or not.
- **variables** (*iterable*) – A list or tuple of the variables to parse from the XML. Note that this is only used when invoked against classes not derived from VersionedPanObject.

Returns created instances of class

Return type list

remove(*child*)

Remove the child from this node

Parameters **child** (PanObject) – Child to remove

remove_by_name(*name*, *cls=None*)

Remove a child node by name

If the class is not specified, then it defaults to type(self).

Parameters **name** (*str*) – Name of the child node

Keyword Arguments **cls** (*class*) – Restrict removal to instances of this class

Returns The removed node, otherwise None

Return type *PanObject*

removeall(*cls=None*)

Remove all children of a type

Not recursive.

Parameters **cls** (*class*) – The class of objects to remove

Returns List of PanObjects that were removed

Return type list

rename(*new_name*)
Change the name of this object.

Modifies the live device

NOTE: This does not change any references that may exist in your pan-os-python object hierarchy, but it does update the name of the object itself.

Parameters **new_name** (*str*) – The new UID for this object.

retrieve_panos_version()
Gets the panos_version of the closest PanDevice.

If this object is not attached to a PanDevice, then a very large number is returned to ensure that the newest version of the object and xpath is presented to the user.

Returns The version as (x, y, z)

Return type tuple

tree()
Display a graph of the configuration tree

The tree includes this object and its children, recursively.

This method is only for use in Jupyder Notebooks

uid
Returns the unique identifier of this object as a string.

update(*variable*)
Change the value of a variable

Modifies the live device

Do not attempt this on an element variable (|) or variable with replacement {{}} If the variable's value is None, then a delete API call is attempted.

Parameters **variable** (*str*) – The name of an instance variable to update on the device

classmethod variables()
Defines the variables that exist in this object. Override in each subclass.

vsys
Return the vsys for this object

Traverses the tree to determine the vsys from a *panos.firewall.Firewall* or *panos.device.Vsys* instance somewhere before this node in the tree.

Returns The vsys id (eg. vsys2)

Return type str

xml_merge(*root*, *elements*)
Merges other elements into the root element.

This differs from xml_combine in a few important ways:

1) The base tag of root is valid
2) The root element must be a valid ElementTree object
3) Individual Nones in the elements iterable are ignored

Parameters

- **root** (*xml.etree.ElementTree*) – The root element.

- **elements** (*iterable*) – Other xml.etree.ElementTree instances (or None) that should be merged into `root` as well.

Returns The final merged root element.

Return type xml.etree.ElementTree

xpath(*root=None*)
Return the full xpath for this object

Xpath in the form: parent's xpath + this object's xpath + entry or member if applicable.

Parameters **root** – The root to use for this object (default: this object's root)

Returns The full xpath to this object

Return type str

xpath_nosuffix()
Return the xpath without the suffix

This is used by refreshall().

Returns The xpath without entry or member on the end

Return type str

xpath_short(*root=None*)
Return an xpath for this object without the final segment

Xpath in the form: parent's xpath + this object's xpath. Used for set API calls.

Parameters **root** – The root to use for this object (default: this object's root)

Returns The xpath without the final segment

Return type str

class panos.base.**ParamPath**(*param*, *path=None*, *vartype=None*, *condition=None*, *values=None*, *exclude=False*)
Configuration parameter within the object.

Parameters

- **param** (*str*) – The name of the instance parameter in the class
- **path** – The relative xpath to the variable.
- **vartype** – The type of variable (None, 'member', 'entry', 'yesno', 'int', 'exist').
- **condition** (*dict*) – Other settings that must be true for this param to appear in the XML. The keys of the condition should be other parameter names, with the value being what the necessary value of that parameter should be.
- **values** (*list*) – Valid values this param can be set to. This is not enforced in any way from the user's perspective when setting parameters, but these values are referenced when parsing any XML returned from a live device.
- **exclude** (*bool*) – Exclude this param from the resultant XML.

about(*version_header=None*)
Returns information about this ParamPath as a dict.

element(*elm*, *settings*, *comparable=False*)
Create the xml.etree.ElementTree for this parameter.

Parameters

- **elm** (*xml.etree.ElementTree*) – the root node for which to append onto this param's XML.
- **settings** (*dict*) – All parameter settings for the `VersionedPanObject`.
- **comparable** (*bool*) – Make necessary adjustments to the XML for comparison's sake.

Returns The `elm` passed in, modified to contain this parameter in the XML. If this param should not be contained in the full `VersionedPanObject`'s XML, then None is returned.

Return type xml.etree.ElementTree

parse_value_from_xml_last_tag(*elm*, *settings*)

Actually do the parsing for this parameter.

The value parsed is saved into the `settings` dict.

Parameters

- **elm** (*xml.etree.ElementTree*) – The final (deepest) tag in the XML document passed in to `parse_xml()` that contains the actual value to parse out for this parameter.
- **settings** (*dict*) – The dict where the parsed value will be saved.

Raises `ValueError` – If a param is in an incorrect format.

parse_xml(*xml*, *settings*, *possibilities*)

Parse the XML to find this parameter's value.

Both this parameter, and any other parameters that may be discovered during the parsing of this parameter, will be saved in the `settings` dict passed in to this function.

Parameters

- **xml** (*xml.etree.ElementTree*) – The XML to parse.
- **settings** (*dict*) – Current known values for this object's parameters.
- **possibilities** (*dict*) – A dict where the key is a parameter's name, and the value is a list of strings that that param could be in the XML.

class panos.base.**ParentAwareXpath**

Class to handle xpaths of objects.

Some objects have a different xpath based on where in the tree they are located. This class allows you configure various xpaths that can vary both on version and what the parent class is.

If no explicit parent is specified, then the global parent of 'None' is assumed.

add_profile(*version=None*, *value=None*, *parents=None*, *parent_param=None*, *parent_param_values=None*)

Adds support for the given versions, specific to the parents.

If no parents are specified, then a parent of `None` is assumed, which is the global parent type.

Version support per parent must be in ascending order.

Parameters

- **version** (*str*) – The version number (default: '0.0.0').
- **value** (*str*) – The xpath setting.
- **parents** (*list/tuple*) – The parent classes this version/value is valid for.
- **parent_param** (*str*) – Parent param to key off of.
- **parent_param_values** (*list*) – Values of the parent param to key off of.

class panos.base.**ValueEntry**(**args*, ***kwargs*)
Base class for objects that only have a value element.

class panos.base.**VarPath**(*path*, *variable=None*, *vartype=None*, *default=None*, *xmldefault=None*, *condition=None*, *order=100*)
Configuration variable within the object

Parameters

- **path** (*str*) – The relative xpath to the variable
- **variable** (*str*) – The name of the instance variable in the class
- **vartype** (*str*) – The type of variable (None, 'member', 'entry', 'bool', 'int', 'exist', 'none')
- **default** – The default value if no value is specified during __init__ of the object
- **xmldefault** (*bool*) – The default value if no value exists in the xml from a device
- **condition** (*str*) – In the format othervariable:value where this variable is only considered if othervariable equals value
- **order** (*int*) – The order of this variable relative to other variables in this constructor of the class that contains this variables. Defaults to 100, set variable order to less than or greater than 100 to alter the order of the variables.

about()
Returns information about this VarPath as a dict.

class panos.base.**VersionedPanObject**(**args*, ***kwargs*)
Base class for all versioned package objects.

This class is an extention of *panos.base.PanObject* that supports versioning.

Parameters

- **name** (*str*) – The name of this object.
- ***args** – Variable length list of values to initialize this object.
- ****kwargs** – Keyword args to initialize this object.

uid
The unique identifier for this object if it has one. If it doesn't have one, then this returns the class name.

Type str

vsys
The vsys id for this object (e.g. 'vsys2') or 'shared' if no vsys.

Type str

XPATH
The xpath for this object, based on where in the tree it currently resides, as well as the versioning.

Type str

element(*with_children=True*, *comparable=False*)
Return an xml.etree.ElementTree for this object and its children.

Parameters

- **with_children** (*bool*) – Include the children objects.
- **comparable** (*bool*) – Element will be used in a comparison with another.

Returns xml.etree.ElementTree for this object.

equal(*panobject*, *force=False*, *compare_children=True*)
Compare this object to another object

Equality of the objects is determined by the XML they generate, not by the values of their variables.

Parameters

- **panobject** (VersionedPanObject) – The object to compare with this object
- **force** (*bool*) – Do not raise a PanObjectError if the objects are different classes
- **compare_children** (*bool*) – Include children of the PanObject in the comparison

Raises PanObjectError – Raised if the objects are different types that would not normally be comparable

Returns True if the XML of the objects is equal, False if not

Return type bool

parse_xml(*xml*)
Parse the given XML into this object's parameters.

Parameters **xml** (*xml.etree.ElementTree*) – The XML to parse values from.

class panos.base.**VersionedParamPath**(*name*, *default=None*, *version=None*, ***kwargs*)
A wrapper class for ParamPath objects.

Specifying any kwargs will be interpreted as args for the first profile to add for this parameter. If there are no kwargs specified, then any version that may or may not have been passed in is ignored.

The values stored in each profile added are the kwargs used to initialize the ParamPath object. The name should not be specified, as that will be passed in positionally for you.

Parameters

- **name** (*str*) – The parameter name. Any hyphens in the name are replaced with underscores, as hyphens are not a valid variable character.
- **default** – The default value this parameter should take when the user is creating a VersionedPanObject, but doesn't specify a value.
- **version** (*str*) – A version string like '1.2.3' or None. If the version is None, then the version is set to '0.0.0'.
- ****kwargs** – Various ParamPath parameters for the given version.

add_profile(*version=None*, ***kwargs*)
Add support for version version.

Parameters

- **version** (*str*) – The version to add support for. If this is unspecified, then the version defaults to '0.0.0'.
- ****kwargs** – The various ParamPath arguments to use for the given version. Note that if your kwargs do not contain a path, then this means that the variable will only be present in the resulting XML if another VersionedParamPath references this parameter in it's path.

class panos.base.**VersionedStubs**

add_profile(*version=None*, **paths*)
Adds the following stubs for the specified version.

Parameters

- **version** (*str*) – The version to add support for.
- ***paths** (*str*) – Variable length arg list of paths for this version.

class panos.base.**VersioningSupport**

A class that supports getting version specific values of something.

Versions of the value are added in ascending order using `add_profile()`, then can be retrieved by using `_get_versioned_value()`. You can specify how the retrieved value is cast by overriding `_cast_version_value()`.

add_profile(*version=None*, *value=None*)

Add support for version `version` that returns `value`.

Version support must be added in ascending order.

Parameters

- **version** (*str*) – The version to add support for. If this is unspecified, then the version defaults to '0.0.0'.
- **value** – The value to be retrieved for this version.

Raises `ValueError` – If the given version is lower than the most recent version.

class panos.base.**VsysOperations**(**args*, ***kwargs*)

Modify PanObject methods to set vsys import configuration.

XPATH_IMPORT

Returns the version specific xpath import for this object.

apply()

Apply this object to the device, replacing any existing object of the same name

Modifies the live device

create()

Create this object on the device

Modifies the live device

This method is nondestructive. If the object exists, the variables are added to the device without changing existing variables on the device. If a variables already exists on the device and this object has a different value, the value on the firewall is changed to the value in this object.

create_import(*vsys=None*)

Create a vsys import for the object

Parameters **vsys** (*str*) – Override the vsys

delete()

Delete this object from the firewall

Modifies the live device

delete_import(*vsys=None*)

Delete a vsys import for the object

Parameters **vsys** (*str*) – Override the vsys

classmethod refreshall(*parent*, *running_config=False*, *add=True*, *exceptions=False*, *name_only=False*, *matching_vsys=True*)

Factory method to instantiate class from live device.

This method is a factory for the class. It takes an firewall or Panorama and gets the xml config from the live device. It generates instances of this class for each item this class represents in the xml config. For example, if the class is AddressObject and there are 5 address objects on the firewall, then this method will generate 5 instances of the class AddressObject.

Parameters

- **parent** (PanObject) – A PanDevice, or a PanObject subclass with a PanDevice as its parental root.
- **running_config** (*bool*) – False for candidate config, True for running config.
- **add** (*bool*) – Update the objects of this type in pan-os-python with the refreshed values.
- **exceptions** (*bool*) – If False, exceptions are ignored if the xpath can't be found.
- **name_only** (*bool*) – If True, refresh only the name of the object, but not its variables. This results in a smaller response to the API call when only the object name is needed.

Returns created instances of class

Return type list

set_vsys(*vsys_id*, *refresh=False*, *update=False*, *running_config=False*, *return_type='object'*)

Set the vsys for this interface.

Creates a reference to this interface in the specified vsys and removes references to this interface from all other vsys. The vsys will be created if it doesn't exist.

Parameters

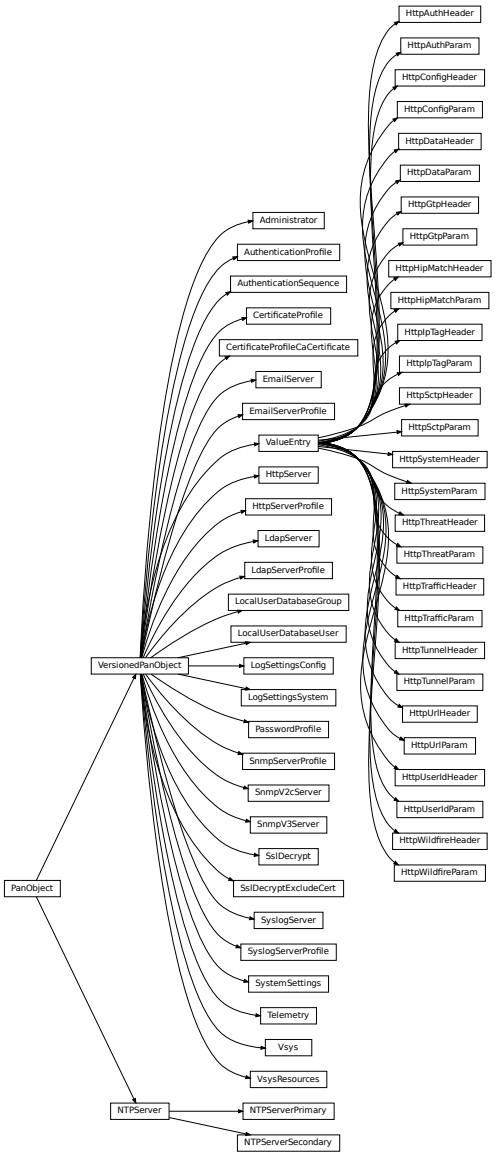
- **vsys_id** (*str*) – The vsys id to set for this object (eg. vsys2)
- **refresh** (*bool*) – Refresh the relevant current state of the device before taking action (Default: False)
- **update** (*bool*) – Apply the changes to the device (Default: False)
- **running_config** (*bool*) – If refresh is True, refresh from the running configuration (Default: False)
- **return_type** (*str*) – Specify what this function returns, can be either 'object' (the default) or 'bool'. If this is 'object', then the return value is the device.Vsys in question. If this is 'bool', then the return value is a boolean that tells you about if the live device needs updates (update=False) or was updated (update=True).

Returns The vsys for this interface after the operation completes

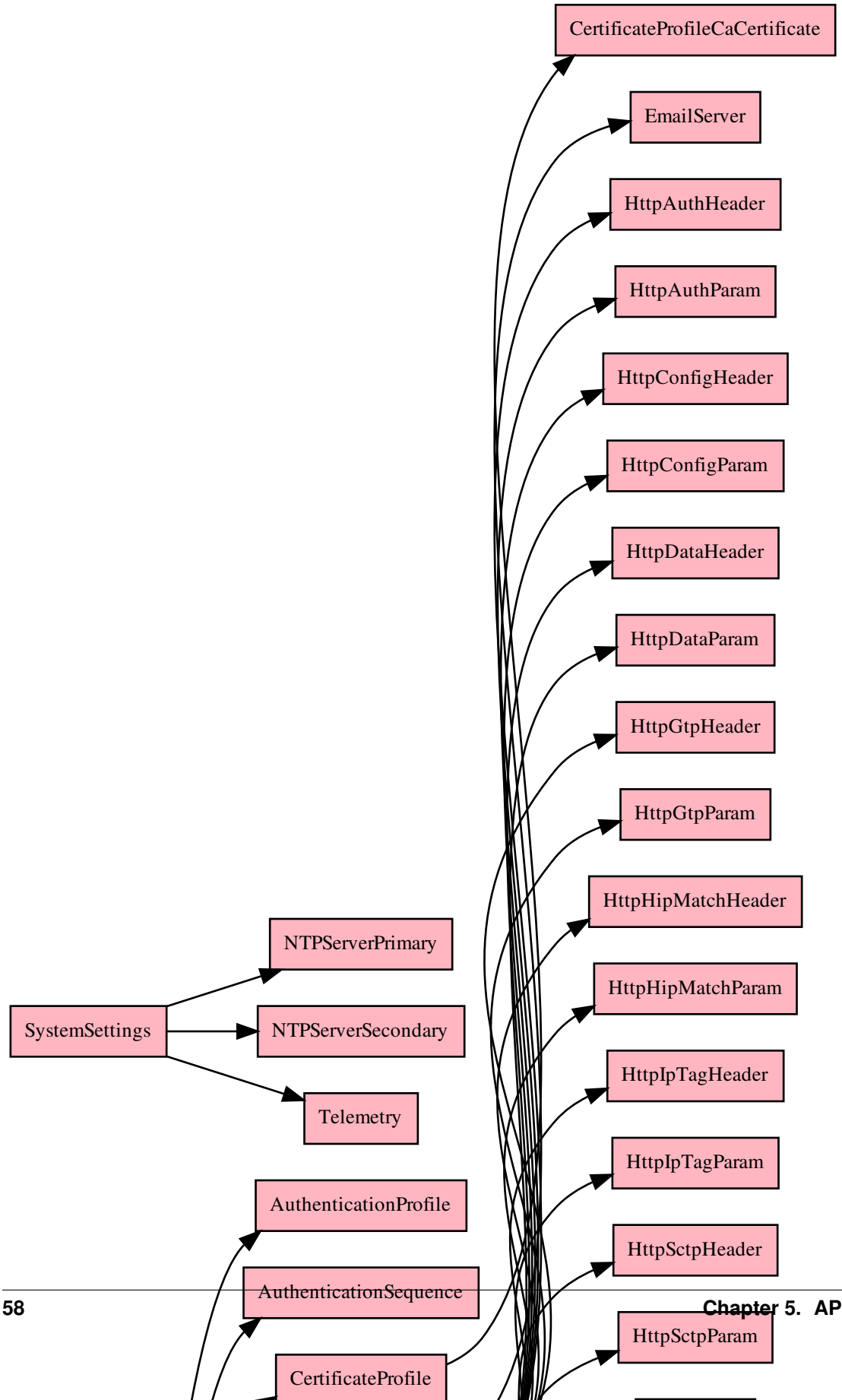
Return type *Vsys*

5.4 Module: device

5.4.1 Inheritance diagram



5.4.2 Configuration tree diagram



5.4.3 Class Reference

Device module contains objects that exist in the 'Device' tab in the firewall GUI

class panos.device.**Administrator**(**args*, ***kwargs*)

Administrator object

Parameters

- **name** (*str*) – Admin name
- **authentication_profile** (*str*) – The authentication profile
- **web_client_cert_only** (*bool*) – Use only client certificate authentication (Web)
- **superuser** (*bool*) – Admin type - superuser
- **superuser_read_only** (*bool*) – Admin type - superuser, read only
- **panorama_admin** (*bool*) – Panonrama - a panorama admin only
- **device_admin** (*bool*) – Admin type - device admin
- **device_admin_read_only** (*bool*) – Admin type - device admin, read only
- **vsys** (*list/str*) – Physical firewalls: the vsys this admin should manage
- **vsys_read_only** (*list/str*) – Physical firewalls: the vsys this read only admin should manage
- **ssh_public_key** (*str*) – Use Public Key Authentication (SSH)
- **role_profile** (*str*) – The role based profile
- **password_hash** (*encrypted str*) – The encrypted password
- **password_profile** (*str*) – The password profile for this user
- **vsys_device** (*list*) – The vsys list (excluded)
- **vsys_read_only_device** (*list*) – The read-only device list (excluded)

change_password(*new_password*)

Update the password.

Modifies the live device

Parameters **new_password** (*str*) – The new password for this user.

class panos.device.**AuthenticationProfile**(**args*, ***kwargs*)

Authentication profile object.

Note: This is valid for PAN-OS 8.0+.

Parameters

- **name** (*string*) – The name
- **profile_type** – Authentication profile type. Valid values are "none" (default), "kerberos", "ldap", "local-database", "radius", "saml-idp", or "tacplus".
- **server_profile** (*string*) – Login method server profile
- **retrieve_user_group** (*bool*) – Retrieve user group from RADIUS or TACACS+
- **ldap_login_attribute** (*string*) – LDAP login attribute
- **ldap_password_expiry_warning** (*string*) – LDAP number of days prior to warning a user about password expiry

- **kerberos_realm** (*string*) – Kerberos realm name to be used for authentication
- **saml_request_signing_certificate** (*string*) – SAML-IDP request signing certificate
- **saml_enable_single_logout** (*bool*) – SAML enable single_logout
- **saml_certificate_profile** (*string*) – SAML certificate profile
- **saml_username_attribute** (*string*) – SAML attribute name usrname
- **saml_user_group_attribute** (*string*) – SAML attribute name user group
- **saml_admin_role_attribute** (*string*) – SAML attribute name admin role
- **saml_access_domain_attribute** (*string*) – SAML attribute name access domain
- **user_domain** (*string*) – User domain
- **username_modifier** (*string*) – Username modifier
- **sso_realm** (*string*) – Single-sign-on Kerberos realm
- **sso_service_principal** (*string*) – Single-sign-on Kerberos service principal
- **sso_keytab** (*string*) – Single-sign-on Kerberos keytab
- **mfa_enable** (*bool*) – Multi factor auth enable
- **mfa_factors** (*list*) – Multi factor auth factors
- **allow_list** (*list*) – Allow users
- **failed_attempts** (*int*) – number of permitted failed attempts
- **lockout_time** (*int*) – amount of time use will be locked

class panos.device.**AuthenticationSequence**(**args*, ***kwargs*)

Authentication Sequence object.

Note: This is valid for PAN-OS 7.0+.

Parameters

- **name** (*string*) – The name
- **authentication_profiles** (*list*) – The authentication profiles
- **use_domain_find_profile** (*bool*) – Use domain find profile

class panos.device.**CertificateProfile**(**args*, ***kwargs*)

Certificate profile object.

Parameters

- **name** (*str*) – The name
- **username_field** (*str*) – The username field. Valid values are “subject”, “subject-alt”, or “none”.
- **username_field_value** (*str*) – The value for the given *username_field*.
- **domain** (*str*) – The domain.
- **use_crl** (*bool*) – Use CRL.
- **use_ocsp** (*bool*) – Use OCSP.
- **crl_receive_timeout** (*int*) – CRL receive timeout (sec).

- **ocsp_receive_timeout** (*int*) – OCSP receive timeout (sec).
- **certificate_status_timeout** (*int*) – Certificate status timeout (sec).
- **block_unknown_certificate** (*bool*) – Block session if certificate status is unknown.
- **block_certificate_timeout** (*bool*) – Block if a session certificate status can't be retrieved within timeout.
- **block_unauthenticated_certificate** (*bool*) – (PAN-OS 7.1) Block session if the certificate was not issued to the authenticating device.
- **block_expired_certificate** (*bool*) – (PAN-OS 8.1) Block session if the certificate is expired.
- **ocsp_exclude_nonce** (*bool*) – (PAN-OS 9.0) Whether to exclude nonce extension for OCSP requests.

class panos.device.**CertificateProfileCaCertificate**(**args*, ***kwargs*)

CA certificate for a certificate profile.

Parameters

- **name** (*str*) – The name.
- **default_ocsp_url** (*str*) – Default URL for OCSP verification.
- **ocsp_verify_certificate** (*str*) – Certificate to verify signature in OCSP response.
- **template_name** (*str*) – (PAN-OS 9.0+) Template name / OID for the certificate.

class panos.device.**EmailServer**(**args*, ***kwargs*)

An email server in a email server profile.

Parameters

- **name** (*str*) – The name
- **display_name** (*str*) – Display name
- **from** (*str*) – From email address
- **to** (*str*) – To email address
- **also_to** (*str*) – Additional destination email address
- **email_gateway** (*str*) – IP address or FQDN of email gateway to use
- **protocol** (*str*) – (PAN-OS 10.0+) SMTP for clear-text or TLS for encrypted
- **port** (*int*) – (PAN-OS 10.0+) Port number
- **tls_version** (*str*) – (PAN-OS 10.0+) TLS handshake protocol version.
- **auth** (*str*) – (PAN-OS 10.0+) Authentication type.
- **certificate_profile** (*str*) – (PAN-OS 10.0+) Certificate profile for validating server certificate.
- **username** (*str*) – (PAN-OS 10.0+) Authentication username.
- **password** (*str*) – (PAN-OS 10.0+) Authentication password.

class panos.device.**EmailServerProfile**(**args*, ***kwargs*)

An email server profile.

Parameters

- **name** (*str*) – The name
- **config** (*str*) – Custom config log format
- **system** (*str*) – Custom system log format
- **threat** (*str*) – Custom threat log format
- **traffic** (*str*) – Custom traffic log format
- **hip_match** (*str*) – Custom HIP match log format
- **url** (*str*) – (PAN-OS 8.0+) Custom URL log format
- **data** (*str*) – (PAN-OS 8.0+) Custom data log format
- **wildfire** (*str*) – (PAN-OS 8.0+) Custom WildFire log format
- **tunnel** (*str*) – (PAN-OS 8.0+) Custom tunnel log format
- **user_id** (*str*) – (PAN-OS 8.0+) Custom user-ID log format
- **gtp** (*str*) – (PAN-OS 8.0+) Custom GTP log format
- **auth** (*str*) – (PAN-OS 8.0+) Custom authentication log format
- **sctp** (*str*) – (PAN-OS 8.1+) Custom SCTP log format
- **iptag** (*str*) – (PAN-OS 9.0+) Custom Iptag log format
- **escaped_characters** (*str*) – Characters to be escaped
- **escape_character** (*str*) – Escape character

class panos.device.**HttpAuthHeader**(**args*, ***kwargs*)

HTTP header for auth.

Note: This is valid for PAN-OS 8.0+

Parameters

- **name** (*str*) – The header name
- **value** (*str*) – The header value

class panos.device.**HttpAuthParam**(**args*, ***kwargs*)

HTTP param for auth.

Note: This is valid for PAN-OS 8.0+

Parameters

- **name** (*str*) – The param name
- **value** (*str*) – The param value

class panos.device.**HttpConfigHeader**(**args*, ***kwargs*)

HTTP header for config.

Note: This is valid for PAN-OS 8.0+

Parameters

- **name** (*str*) – The header name
- **value** (*str*) – The header value

class panos.device.**HttpConfigParam**(**args*, ***kwargs*)
HTTP param for config.

Note: This is valid for PAN-OS 8.0+

Parameters

- **name** (*str*) – The param name
- **value** (*str*) – The param value

class panos.device.**HttpDataHeader**(**args*, ***kwargs*)
HTTP header for data.

Note: This is valid for PAN-OS 8.0+

Parameters

- **name** (*str*) – The header name
- **value** (*str*) – The header value

class panos.device.**HttpDataParam**(**args*, ***kwargs*)
HTTP param for data.

Note: This is valid for PAN-OS 8.0+

Parameters

- **name** (*str*) – The param name
- **value** (*str*) – The param value

class panos.device.**HttpGtpHeader**(**args*, ***kwargs*)
HTTP header for GTP.

Note: This is valid for PAN-OS 8.0+

Parameters

- **name** (*str*) – The header name
- **value** (*str*) – The header value

class panos.device.**HttpGtpParam**(**args*, ***kwargs*)
HTTP param for GTP.

Note: This is valid for PAN-OS 8.0+

Parameters

- **name** (*str*) – The param name
- **value** (*str*) – The param value

class panos.device.**HttpHipMatchHeader**(**args*, ***kwargs*)
HTTP header for HIP match.

Note: This is valid for PAN-OS 8.0+

Parameters

- **name** (*str*) – The header name
- **value** (*str*) – The header value

class panos.device.**HttpHipMatchParam**(**args*, ***kwargs*)
HTTP param for HIP match.

Note: This is valid for PAN-OS 8.0+

Parameters

- **name** (*str*) – The param name
- **value** (*str*) – The param value

class panos.device.**HttpIpTagHeader**(**args*, ***kwargs*)
HTTP header for IP tag.

Note: This is valid for PAN-OS 8.0+

Parameters

- **name** (*str*) – The header name
- **value** (*str*) – The header value

class panos.device.**HttpIpTagParam**(**args*, ***kwargs*)
HTTP param for IP tag.

Note: This is valid for PAN-OS 8.0+

Parameters

- **name** (*str*) – The param name
- **value** (*str*) – The param value

class panos.device.**HttpSctpHeader**(**args*, ***kwargs*)
HTTP header for SCTP.

Note: This is valid for PAN-OS 8.0+

Parameters

- **name** (*str*) – The header name
- **value** (*str*) – The header value

class panos.device.**HttpSctpParam**(**args*, ***kwargs*)
HTTP param for SCTP.

Note: This is valid for PAN-OS 8.0+

Parameters

- **name** (*str*) – The param name
- **value** (*str*) – The param value

class panos.device.**HttpServer**(**args*, ***kwargs*)
A single HTTP server in a HTTP server profile.

Parameters

- **name** (*str*) – The name
- **address** (*str*) – IP address or FQDN of HTTP server to use
- **protocol** (*str*) – HTTPS (default) or HTTP
- **port** (*int*) – Port number (default: 443).

- **tls_version** (*str*) – (PAN-OS 9.0+) TLS handshake protocol version. Valid values are 1.0, 1.1, or 1.2.
- **certificate_profile** (*str*) – (PAN-OS 9.0+) Certificate profile for validating server certificate
- **http_method** (*str*) – HTTP method to use (default: POST).
- **username** (*str*) – Username for basic HTTP auth
- **password** (*str*) – Password for basic HTTP auth

class panos.device.**HttpServerProfile**(**args*, ***kwargs*)

A HTTP server profile.

Note: This is valid for PAN-OS 8.0+.

Parameters

- **name** (*str*) – The name
- **tag_registration** (*bool*) – The server should have User-ID agent running in order for tag registration to work
- **config_name** (*str*) – Name for custom config format
- **config_uri_format** (*str*) – URI format for custom config format
- **config_payload** (*str*) – Payload for custom config format
- **system_name** (*str*) – Name for custom system format
- **system_uri_format** (*str*) – URI format for custom system format
- **system_payload** (*str*) – Payload for custom system format
- **threat_name** (*str*) – Name for custom threat format
- **threat_uri_format** (*str*) – URI format for custom threat format
- **threat_payload** (*str*) – Payload for custom threat format
- **traffic_name** (*str*) – Name for custom traffic format
- **traffic_uri_format** (*str*) – URI format for custom traffic format
- **traffic_payload** (*str*) – Payload for custom traffic format
- **hip_match_name** (*str*) – Name for custom HIP match format
- **hip_match_uri_format** (*str*) – URI format for custom HIP match format
- **hip_match_payload** (*str*) – Payload for custom HIP match format
- **url_name** (*str*) – Name for custom url format
- **url_uri_format** (*str*) – URI format for custom url format
- **url_payload** (*str*) – Payload for custom url format
- **data_name** (*str*) – Name for custom data format
- **data_uri_format** (*str*) – URI format for custom data format
- **data_payload** (*str*) – Payload for custom data format
- **wildfire_name** (*str*) – Name for custom wildfire format
- **wildfire_uri_format** (*str*) – URI format for custom wildfire format

- **wildfire_payload** (*str*) – Payload for custom wildfire format
- **tunnel_name** (*str*) – Name for custom tunnel format
- **tunnel_uri_format** (*str*) – URI format for custom tunnel format
- **tunnel_payload** (*str*) – Payload for custom tunnel format
- **user_id_name** (*str*) – Name for custom User-ID format
- **user_id_uri_format** (*str*) – URI format for custom User-ID format
- **user_id_payload** (*str*) – Payload for custom User-ID format
- **gtp_name** (*str*) – Name for custom GTP format
- **gtp_uri_format** (*str*) – URI format for custom GTP format
- **gtp_payload** (*str*) – Payload for custom GTP format
- **auth_name** (*str*) – Name for custom auth format
- **auth_uri_format** (*str*) – URI format for custom auth format
- **auth_payload** (*str*) – Payload for custom auth format
- **sctp_name** (*str*) – (PAN-OS 8.1+) Name for custom SCTP format
- **sctp_uri_format** (*str*) – (PAN-OS 8.1+) URI format for custom SCTP format
- **sctp_payload** (*str*) – (PAN-OS 8.1+) Payload for custom SCTP format
- **iptag_name** (*str*) – (PAN-OS 9.0+) Name for custom IP tag format
- **iptag_uri_format** (*str*) – (PAN-OS 9.0+) URI format for custom IP tag format
- **iptag_payload** (*str*) – (PAN-OS 9.0+) Payload for custom IP tag format

class panos.device.**HttpSystemHeader**(**args*, ***kwargs*)

HTTP header for system.

Note: This is valid for PAN-OS 8.0+

Parameters

- **name** (*str*) – The header name
- **value** (*str*) – The header value

class panos.device.**HttpSystemParam**(**args*, ***kwargs*)

HTTP param for system.

Note: This is valid for PAN-OS 8.0+

Parameters

- **name** (*str*) – The param name
- **value** (*str*) – The param value

class panos.device.**HttpThreatHeader**(**args*, ***kwargs*)

HTTP header for threat.

Note: This is valid for PAN-OS 8.0+

Parameters

- **name** (*str*) – The header name
- **value** (*str*) – The header value

class panos.device.**HttpThreatParam**(**args*, ***kwargs*)
HTTP param for threat.

Note: This is valid for PAN-OS 8.0+

Parameters

- **name** (*str*) – The param name
- **value** (*str*) – The param value

class panos.device.**HttpTrafficHeader**(**args*, ***kwargs*)
HTTP header for traffic.

Note: This is valid for PAN-OS 8.0+

Parameters

- **name** (*str*) – The header name
- **value** (*str*) – The header value

class panos.device.**HttpTrafficParam**(**args*, ***kwargs*)
HTTP param for traffic.

Note: This is valid for PAN-OS 8.0+

Parameters

- **name** (*str*) – The param name
- **value** (*str*) – The param value

class panos.device.**HttpTunnelHeader**(**args*, ***kwargs*)
HTTP header for tunnel.

Note: This is valid for PAN-OS 8.0+

Parameters

- **name** (*str*) – The header name
- **value** (*str*) – The header value

class panos.device.**HttpTunnelParam**(**args*, ***kwargs*)
HTTP param for tunnel.

Note: This is valid for PAN-OS 8.0+

Parameters

- **name** (*str*) – The param name
- **value** (*str*) – The param value

class panos.device.**HttpUrlHeader**(**args*, ***kwargs*)
HTTP header for URL.

Note: This is valid for PAN-OS 8.0+

Parameters

- **name** (*str*) – The header name
- **value** (*str*) – The header value

class panos.device.**HttpUrlParam**(**args*, ***kwargs*)
HTTP param for URL.

Note: This is valid for PAN-OS 8.0+

Parameters

- **name** (*str*) – The param name
- **value** (*str*) – The param value

class panos.device.**HttpUserIdHeader**(**args*, ***kwargs*)
HTTP header for user-ID.

Note: This is valid for PAN-OS 8.0+

Parameters

- **name** (*str*) – The header name
- **value** (*str*) – The header value

class panos.device.**HttpUserIdParam**(**args*, ***kwargs*)
HTTP param for user-ID.

Note: This is valid for PAN-OS 8.0+

Parameters

- **name** (*str*) – The param name
- **value** (*str*) – The param value

class panos.device.**HttpWildfireHeader**(**args*, ***kwargs*)
HTTP header for WildFire.

Note: This is valid for PAN-OS 8.0+

Parameters

- **name** (*str*) – The header name
- **value** (*str*) – The header value

class panos.device.**HttpWildfireParam**(**args*, ***kwargs*)
HTTP param for WildFire.

Note: This is valid for PAN-OS 8.0+

Parameters

- **name** (*str*) – The param name
- **value** (*str*) – The param value

class panos.device.**LdapServer**(**args*, ***kwargs*)
An ldap server in a ldap server profile

Parameters

- **name** (*str*) – The name
- **address** (*str*) – IP address or FQDN of ldap server to use
- **port** (*str*) – port number

class panos.device.**LdapServerProfile**(**args*, ***kwargs*)
An ldap server profile.

Note: Valid for PAN-OS 7.0+.

Parameters

- **name** (*str*) – The name
- **ldap_type** (*str*) – Ldap profile type. Valid values are "other" (default), "active-directory", "e-directory", or "sun".
- **base** (*str*) – Base DN
- **bind_dn** (*str*) – Bind DN
- **bind_password** (*str*) – Bind password
- **bind_timelimit** (*int*) – Bind timeout
- **timelimit** (*int*) – Search timeout
- **retry_interval** (*int*) – Retry interval
- **ssl** (*bool*) – Require ssl/ttls secured connection
- **verify_server_certificate** (*bool*) – Verify server certificate for ssl sessions
- **disabled** (*bool*) – Disabled or not

class panos.device.**LocalUserDatabaseGroup**(**args*, ***kwargs*)
A Local User Database group.

Parameters

- **name** (*str*) – Name.
- **users** (*list*) – The local users in this group.

class panos.device.**LocalUserDatabaseUser**(**args*, ***kwargs*)
A Local User Database User.

Parameters

- **name** (*str*) – Name.
- **password_hash** (*str*) – The password hash.
- **disabled** (*bool*) – Set to True if the user is disabled.

change_password(*new_password*)
Update the password.

Modifies the live device

Parameters **new_password** (*str*) – The new password for this user.

class panos.device.**LogSettingsConfig**(**args*, ***kwargs*)
Firewall or Panorama device log settings configuration

Note: This is valid for PANS-OS 8.0+.

Parameters

- **name** (*string*) – The name
- **filter** (*string*) – Valid values are "All logs" (default) or create your own filter
- **description** (*string*) – Description

- **send_to_panorama** (*bool*) – Send to panorama
- **send_email** (*list*) – Send email profile
- **send_snmp** (*list*) – Send snmp profile
- **send_syslog** (*list*) – Send syslog profile
- **send_http** (*list*) – Send http profile

class panos.device.**LogSettingsSystem**(**args*, ***kwargs*)
Firewall or Panorama device log settings system

Note: This is valid for PANS-OS 8.0+.

Parameters

- **name** (*string*) – The name
- **filter** (*string*) – Valid values are "All logs" (default) or create your own filter
- **description** (*string*) – Description
- **send_to_panorama** (*bool*) – Send to panorama
- **send_email** (*list*) – Send email profile
- **send_snmp** (*list*) – Send snmp profile
- **send_syslog** (*list*) – Send syslog profile
- **send_http** (*list*) – Send http profile

class panos.device.**NTPServer**(**args*, ***kwargs*)
A primary or secondary NTP server

This is an abstract base class, do not instantiate it.

Parameters **address** (*str*) – The IP address of the NTP server

classmethod variables()
Defines the variables that exist in this object. Override in each subclass.

class panos.device.**NTPServerPrimary**(**args*, ***kwargs*)
A primary NTP server

Add to a *panos.device.SystemSettings* object

Parameters **address** (*str*) – IP address or hostname of NTP server

class panos.device.**NTPServerSecondary**(**args*, ***kwargs*)
A secondary NTP server

Add to a *panos.device.SystemSettings* object

Parameters **address** (*str*) – IP address or hostname of NTP server

class panos.device.**PasswordProfile**(**args*, ***kwargs*)
Password profile object

Parameters

- **name** (*str*) – Password profile name
- **expiration** (*int*) – Number of days until the password expires
- **warning** (*int*) – Number of days warning before password expires
- **login_count** (*int*) – Post expiration admin login count

- **grace_period** (*int*) – Post expiration grace period

class panos.device.**SnmpServerProfile**(**args*, ***kwargs*)
SNMP server profile.

Parameters

- **name** (*str*) – The name
- **version** (*str*) – SNMP version. Valid values are v2c (default) or v3.

class panos.device.**SnmpV2cServer**(**args*, ***kwargs*)
SNMP V2C server in a server.

Parameters

- **name** (*str*) – The name
- **manager** (*str*) – IP address or FQDN of SNMP manager to use
- **community** (*str*) – SNMP community

class panos.device.**SnmpV3Server**(**args*, ***kwargs*)
SNMP V3 server.

Parameters

- **name** (*str*) – The name
- **manager** (*str*) – IP address or FQDN of SNMP manager to use
- **user** (*str*) – User
- **engine_id** (*str*) – A hex number
- **auth_password** (*str*) – Authentication protocol password
- **priv_password** (*str*) – Privacy protocol password

class panos.device.**SslDecrypt**(**args*, ***kwargs*)
SSL decrypt configuration for certificates.

Note: PAN-OS 8.0+

Parameters

- **forward_trust_certificate_rsa** (*str*) – RSA CA certificate for trusted sites.
- **forward_trust_certificate_ecdsa** (*str*) – ECDSA CA certificate for trusted sites.
- **forward_untrust_certificate_rsa** (*str*) – RSA CA certificate for untrusted sites.
- **forward_untrust_certificate_ecdsa** (*str*) – ECDSA CA certificate for untrusted sites.
- **root_ca_excludes** (*list*) – List of predefined root CAs to not trust.
- **trusted_root_cas** (*list*) – List of trusted root CAs.
- **disabled_predefined_exclude_certificates** (*list*) – Disabled predefined SSL exclude certificates.

class panos.device.**SslDecryptExcludeCert**(**args*, ***kwargs*)
SSL decryption exclusion object.

Note: PAN-OS 8.0+

Parameters

- **name** (*str*) – The name.
- **description** (*str*) – Description.
- **exclude** (*bool*) – Exclude boolean.

class panos.device.**SyslogServer**(**args*, ***kwargs*)

A single syslog server in a syslog server profile.

Parameters

- **name** (*str*) – The name
- **server** (*str*) – IP address or FQDN of the syslog server
- **transport** (*str*) – Syslog transport. Valid values are UDP (default), TCP, or SSL.
- **port** (*int*) – Syslog port number.
- **format** (*str*) – Format of the syslog message. Valid values are BSD (default) or IETF.
- **facility** (*str*) – Syslog facility. Valid values are LOG_USER (default), or LOG_LOCAL0 through LOG_LOCAL7.

class panos.device.**SyslogServerProfile**(**args*, ***kwargs*)

A syslog server profile.

Parameters

- **name** (*str*) – The name
- **config** (*str*) – Custom config log format
- **system** (*str*) – Custom system log format
- **threat** (*str*) – Custom threat log format
- **traffic** (*str*) – Custom traffic log format
- **hip_match** (*str*) – Custom HIP match log format
- **url** (*str*) – (PAN-OS 8.0+) Custom URL log format
- **data** (*str*) – (PAN-OS 8.0+) Custom data log format
- **wildfire** (*str*) – (PAN-OS 8.0+) Custom WildFire log format
- **tunnel** (*str*) – (PAN-OS 8.0+) Custom tunnel log format
- **user_id** (*str*) – (PAN-OS 8.0+) Custom user-ID log format
- **gtp** (*str*) – (PAN-OS 8.0+) Custom GTP log format
- **auth** (*str*) – (PAN-OS 8.0+) Custom authentication log format
- **sctp** (*str*) – (PAN-OS 8.1+) Custom SCTP log format
- **iptag** (*str*) – (PAN-OS 9.0+) Custom Iptag log format
- **escaped_characters** (*str*) – Characters to be escaped
- **escape_character** (*str*) – Escape character

class panos.device.**SystemSettings**(**args*, ***kwargs*)

Firewall or Panorama device system settings

Add only one of these to a parent object.

If you want to configure DHCP on the management interface, you should specify settings for *dhcp_send_hostname* and *dhcp_send_client_id*.

Parameters

- **hostname** (*str*) – The hostname of the device
- **domain** (*str*) – The domain of the device
- **ip_address** (*str*) – Management interface IP address
- **netmask** (*str*) – Management interface netmask
- **default_gateway** (*str*) – Management interface default gateway
- **ipv6_address** (*str*) – Management interface IPv6 address
- **ipv6_default_gateway** (*str*) – Management interface IPv6 default gateway
- **dns_primary** (*str*) – Primary DNS server IP address
- **dns_secondary** (*str*) – Secondary DNS server IP address
- **timezone** (*str*) – Device timezone
- **panorama** (*str*) – IP address of primary Panorama
- **panorama2** (*str*) – IP address of secondary Panorama
- **login_banner** (*str*) – Login banner text
- **update_server** (*str*) – IP or hostname of the update server
- **verify_update_server** (*bool*) – Verify the update server identity
- **dhcp_send_hostname** (*bool*) – (DHCP Mngt) Send Hostname
- **dhcp_send_client_id** (*bool*) – (DHCP Mngt) Send Client ID
- **accept_dhcp_hostname** (*bool*) – (DHCP Mngt) Accept DHCP hostname
- **accept_dhcp_domain** (*bool*) – (DHCP Mngt) Accept DHCP domain name

class panos.device.**Telemetry**(**args*, ***kwargs*)

Share telemetry data with Palo Alto Networks.

Join other Palo Alto Networks customers in a global sharing community, helping to raise the bar against the latest attack techniques. Your participation allows us to deliver new threat prevention controls across the attack lifecycle. Choose the type of data you share across applications, threat intelligence, and device health information to improve the fidelity of the protections we deliver. This is an opt-in feature controlled with granular policy, and we encourage you to join the community.

Add only one of these to a firewall.

Parameters

- **app_reports** (*bool*) – Application reports
- **threat_reports** (*bool*) – Threat preventioin reports
- **url_reports** (*bool*) – URL reports
- **file_type_reports** (*bool*) – File type identification reports
- **threat_data** (*bool*) – Threat prevention data
- **threat_pcaps** (*bool*) – Enable sending packet captures with threat prevention information. This requires that "threat_data" also be enabled.

- **product_usage_stats** (*bool*) – Health and performance reports
- **passive_dns_monitoring** (*bool*) – Passive DNS monitoring

class panos.device.**Vsys**(**args*, ***kwargs*)
Virtual System (VSYS)

You can interact with virtual systems in two different ways:

Method 1. Use a *panos.firewall.Firewall* object with the 'vsys' variable set to a vsys identifier (eg. 'vsys2'). In this case, you don't need to use this Vsys class. Add other PanObject instances (like *panos.objects.AddressObject*) to the Firewall instance

Method 2. Add an instance of this Vsys class to a *panos.firewall.Firewall* object. It is best practice to set the Firewall instance's 'shared' variable to True when using this method. Add other PanObject instances (like *panos.objects.AddressObject*) to the Vsys instance.

Parameters

- **name** (*str*) – Vsys identifier (eg. 'vsys1', 'vsys5', etc)
- **display_name** (*str*) – Friendly name of the vsys
- **interface** (*list*) – A list of strings with names of interfaces or a list of *panos.network.Interface* objects
- **vlans** (*list*) – A list of strings of VLANs
- **virtual_wires** (*list*) – A list of strings of virtual wires
- **virtual_routers** (*list*) – A list of strings of virtual routers
- **visible_vsys** (*list*) – A list of strings of the vsys visible
- **dns_proxy** (*str*) – DNS Proxy server
- **decrypt_forwarding** (*bool*) – Allow forwarding of decrypted content

vsys
Return the vsys for this object

Traverses the tree to determine the vsys from a *panos.firewall.Firewall* or *panos.device.Vsys* instance somewhere before this node in the tree.

Returns The vsys id (eg. vsys2)

Return type str

class panos.device.**VsysResources**(**args*, ***kwargs*)
Resource constraints for a Vsys

Parameters

- **max_security_rules** (*int*) – Maximum security rules
- **max_nat_rules** (*int*) – Maximum nat rules
- **max_ssl_decryption_rules** (*int*) – Maximum ssl decryption rules
- **max_qos_rules** (*int*) – Maximum QOS rules
- **max_application_override_rules** (*int*) – Maximum application override rules
- **max_pbf_rules** (*int*) – Maximum policy based forwarding rules
- **max_cp_rules** (*int*) – Maximum captive portal rules
- **max_dos_rules** (*int*) – Maximum DOS rules

- **max_site_to_site_vpn_tunnels** (*int*) – Maximum site-to-site VPN tunnels
- **max_concurrent_ssl_vpn_tunnels** (*int*) – Maximum ssl VPN tunnels
- **max_sessions** (*int*) – Maximum sessions

5.5 Module: errors

5.5.1 Inheritance diagram

5.5.2 Class Reference

Exception classes used by pan-os-python package

exception panos.errors.**PanActivateFeatureAuthCodeError**(**args*, ***kwargs*)

exception panos.errors.**PanApiKeyNotSet**(**args*, ***kwargs*)

exception panos.errors.**PanCommitFailed**(**args*, ***kwargs*)

exception panos.errors.**PanCommitInProgress**(**args*, ***kwargs*)

exception panos.errors.**PanCommitNotNeeded**(**args*, ***kwargs*)

exception panos.errors.**PanConnectionTimeout**(**args*, ***kwargs*)

exception panos.errors.**PanDeviceError**(**args*, ***kwargs*)
Exception for errors in the PanDevice class

The PanDevice class may raise errors when problems occur such as response parsing problems. This exception class is raised on those errors. This class is not for errors connecting to the API, as pan.xapi.PanXapiError is responsible for those.

message
The error message for the exception

pan_device
A reference to the PanDevice that generated the exception

exception panos.errors.**PanDeviceNotSet**(**args*, ***kwargs*)

exception panos.errors.**PanDeviceXapiError**(**args*, ***kwargs*)
General error returned by an API call

exception panos.errors.**PanHAConfigSyncFailed**(**args*, ***kwargs*)

exception panos.errors.**PanHASyncInProgress**(**args*, ***kwargs*)

exception panos.errors.**PanInstallInProgress**(**args*, ***kwargs*)

exception panos.errors.**PanInvalidCredentials**(**args*, ***kwargs*)

exception panos.errors.**PanJobTimeout**(**args*, ***kwargs*)

exception panos.errors.**PanLockError**(**args*, ***kwargs*)

exception panos.errors.**PanNoSuchNode**(**args*, ***kwargs*)

exception panos.errors.**PanNotAttachedOnPanorama**(**args*, ***kwargs*)

exception panos.errors.**PanNotConnectedOnPanorama**(**args*, ***kwargs*)

exception panos.errors.**PanObjectError**(**args*, ***kwargs*)

exception panos.errors.**PanObjectMissing**(**args*, ***kwargs*)

exception panos.errors.**PanOutdatedSslError**(**args*, ***kwargs*)

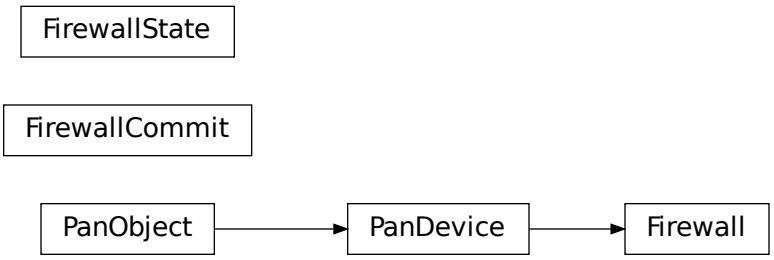
exception panos.errors.**PanPendingChanges**(**args*, ***kwargs*)

exception panos.errors.**PanSessionTimedOut**(**args*, ***kwargs*)

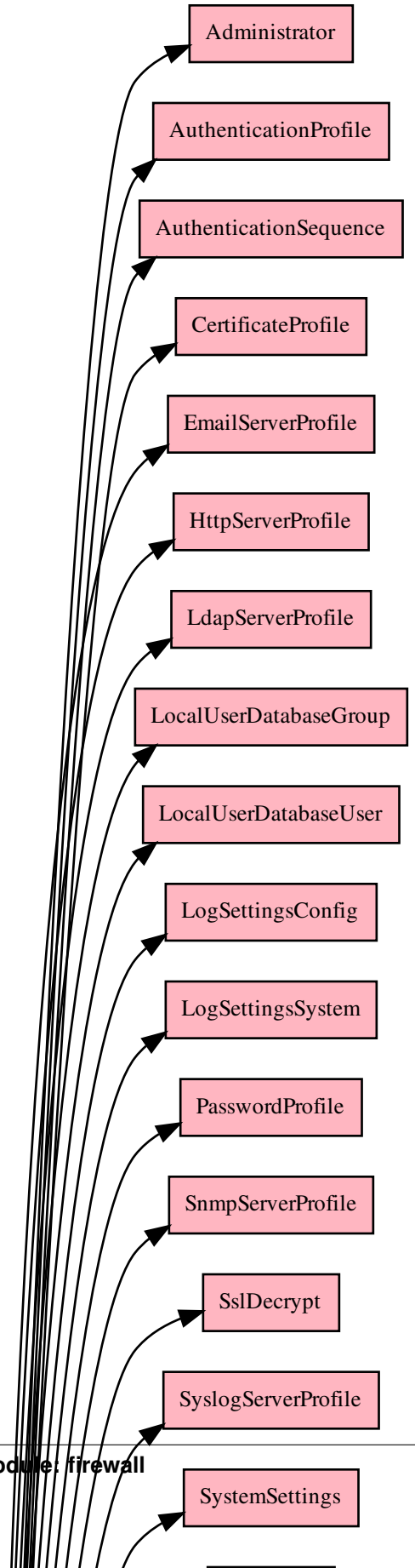
exception panos.errors.**PanURLError**(**args*, ***kwargs*)

5.6 Module: firewall

5.6.1 Inheritance diagram



5.6.2 Configuration tree diagram



5.6.3 Class Reference

Palo Alto Networks Firewall object

class panos.firewall.**Firewall**(*hostname=None*, *api_username=None*, *api_password=None*, *api_key=None*, *serial=None*, *port=443*, *vsys=None*, *is_virtual=None*, *multi_vsys=None*, **args*, ***kwargs*)

A Palo Alto Networks Firewall

This object can represent a firewall physical chassis, virtual firewall, or individual vsys.

Parameters

- **hostname** – Hostname or IP of device for API connections
- **api_username** – Username of administrator to access API
- **api_password** – Password of administrator to access API
- **api_key** – The API Key for connecting to the device's API
- **serial** – The serial number of this firewall
- **port** – Port of device for API connections
- **vsys** – The vsys of this firewall (eg. "vsys1", "vsys2", etc.)
- **is_virtual** (*bool*) – Physical or Virtual firewall
- **timeout** – The timeout for asynchronous jobs
- **interval** – The interval to check asynchronous jobs

apply()

Apply this object to the device, replacing any existing object of the same name

Modifies the live device

create()

Create this object on the device

Modifies the live device

This method is nondestructive. If the object exists, the variables are added to the device without changing existing variables on the device. If a variables already exists on the device and this object has a different value, the value on the firewall is changed to the value in this object.

create_vsys()

Create the vsys on the live device that this Firewall object represents

delete()

Delete this object from the firewall

Modifies the live device

delete_vsys()

Delete the vsys on the live device that this Firewall object represents

element()

Construct an ElementTree for this PanObject and all its children

Parameters

- **with_children** (*bool*) – Include children in element.
- **comparable** (*bool*) – Element will be used in a comparison with another.

Returns

An ElementTree instance representing the xml form of this object and its children

Return type xml.etree.ElementTree

op(*cmd=None*, *vsys=None*, *xml=False*, *cmd_xml=True*, *extra_qs=None*, *retry_on_peer=False*)
Perform operational command on this Firewall

Parameters

- **cmd** (*str*) – The operational command to execute
- **vsys** (*str*) – Vsys id. Defaults to the vsys of the firewall or the Vsys object in the parent tree.
- **xml** (*bool*) – Return value should be a string (Default: False)
- **cmd_xml** (*bool*) – True: cmd is not XML, False: cmd is XML (Default: True)
- **extra_qs** – Extra parameters for API call
- **retry_on_peer** (*bool*) – Try on active Firewall first, then try on passive Firewall

Returns The result of the operational command. May also return a string of XML if xml=True

Return type xml.etree.ElementTree

organize_into_vsys(*create_vsys_objects=True*, *refresh_vsys=True*)
Organizes all imported objects under the appropriate Vsys object.

Parameters

- **create_vsys_objects** (*bool*) – Create the vsys objects (True) or use the ones already connected to this firewall (False).
- **refresh_vsys** (*bool*) – Refresh all vsys objects' parameters before doing the reorganization or not. This is assumed True if create_vsys_objects is True.

refreshall_from_xml(*xml*, *refresh_children=False*, *variables=None*)
Factory method to instantiate class from firewall config.

This method is a factory for the class. It takes an xml config from a firewall and generates instances of this class for each item this class represents in the xml config. For example, if the class is AddressObject and there are 5 address objects on the firewall, then this method will generate 5 instances of the class AddressObject.

Parameters

- **xml** (*xml.etree.ElementTree*) – A section of XML configuration from a firewall or Panorama. It should not contain the response or result tags.
- **refresh_children** (*bool*) – Refresh children objects or not.
- **variables** (*iterable*) – A list or tuple of the variables to parse from the XML. Note that this is only used when invoked against classes not derived from `VersionedPanObject`.

Returns created instances of class

Return type list

shared = None
Set to True to act on the shared part of this firewall

state = None
Panorama state variables refreshed by Panorama

vsys
Return the vsys for this object

Traverses the tree to determine the vsys from a *panos.firewall.Firewall* or *panos.device.Vsys* instance somewhere before this node in the tree.

Returns The vsys id (eg. vsys2)

Return type str

class panos.firewall.**FirewallCommit**(*description=None*, *admins=None*, *exclude_device_and_network=False*, *exclude_shared_objects=False*, *exclude_policy_and_objects=False*, *force=False*)

Normalization of a firewall commit.

Instances of this class can be passed in to `Firewall.commit()` (inherited from *panos.base.PanDevice.commit()*) as the `cmd` parameter.

Parameters

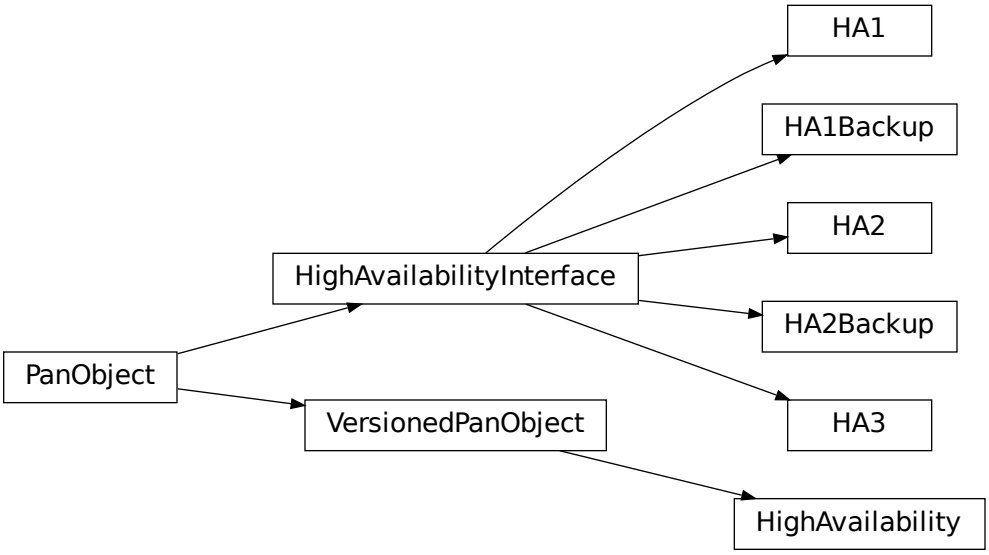
- **description** (*str*) – The commit message.
- **admins** (*list*) – (PAN-OS 8.0+) List of admins whose changes are to be committed.
- **exclude_device_and_network** (*bool*) – Set to True to exclude device and network changes.
- **exclude_shared_objects** (*bool*) – Set to True to exclude shared objects changes.
- **exclude_policy_and_objects** (*bool*) – Set to True to exclude policy and objects changes.
- **force** (*bool*) – Set to True to force a commit even if one is not needed.

element()
Returns an xml representation of the commit requested.

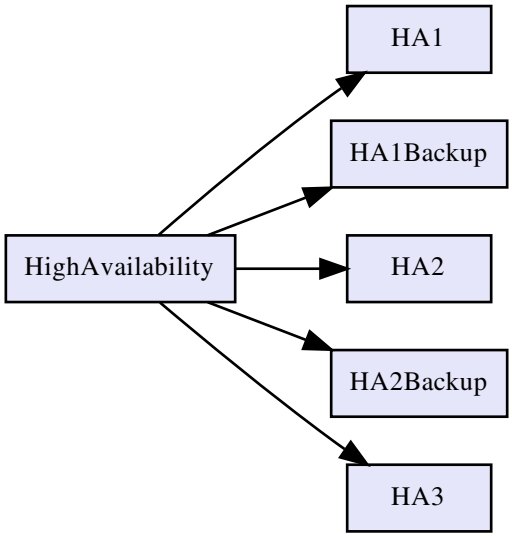
Returns xml.etree.ElementTree

5.7 Module: ha

5.7.1 Inheritance diagram



5.7.2 Configuration tree diagram



5.7.3 Class Reference

High availability objects to configure HA for a firewall or Panorama

class panos.ha.**HA1**(**args*, ***kwargs*)
HA1 interface

Parameters

- **ip_address** (*str*) – IP of the interface
- **netmask** (*str*) – Netmask of the interface
- **port** (*str*) – Interface to use for this HA interface (eg. ethernet1/5)
- **gateway** (*str*) – Default gateway of the interface
- **link_speed** (*str*) – Link speed
- **link_duplex** (*str*) – Link duplex
- **monitor_hold_time** (*int*) – Monitor hold time

classmethod variables()
Defines the variables that exist in this object. Override in each subclass.

class panos.ha.**HA1Backup**(**args*, ***kwargs*)
HA1 Backup interface

Parameters

- **ip_address** (*str*) – IP of the interface
- **netmask** (*str*) – Netmask of the interface
- **port** (*str*) – Interface to use for this HA interface (eg. ethernet1/5)
- **gateway** (*str*) – Default gateway of the interface
- **link_speed** (*str*) – Link speed
- **link_duplex** (*str*) – Link duplex

class panos.ha.**HA2**(**args*, ***kwargs*)
HA2 interface

Parameters

- **ip_address** (*str*) – IP of the interface
- **netmask** (*str*) – Netmask of the interface
- **port** (*str*) – Interface to use for this HA interface (eg. ethernet1/5)
- **gateway** (*str*) – Default gateway of the interface
- **link_speed** (*str*) – Link speed
- **link_duplex** (*str*) – Link duplex

class panos.ha.**HA2Backup**(**args*, ***kwargs*)
HA2 Backup interface

Parameters

- **ip_address** (*str*) – IP of the interface
- **netmask** (*str*) – Netmask of the interface

- **port** (*str*) – Interface to use for this HA interface (eg. ethernet1/5)
- **gateway** (*str*) – Default gateway of the interface
- **link_speed** (*str*) – Link speed
- **link_duplex** (*str*) – Link duplex

class panos.ha.**HA3**(**args*, ***kwargs*)
HA3 interface

Parameters

- **port** (*str*) – Interface to use for this HA interface (eg. ethernet1/5)
- **link_speed** (*str*) – Link speed
- **link_duplex** (*str*) – Link duplex

classmethod variables()
Defines the variables that exist in this object. Override in each subclass.

class panos.ha.**HighAvailability**(**args*, ***kwargs*)
High availability configuration base object

All high availability configuration is in this object or is a child of this object

Parameters

- **name** – (unused, and may be omitted)
- **enabled** (*bool*) – Enable HA (Default: True)
- **group_id** (*int*) – The group identifier
- **description** (*str*) – Description for HA pairing
- **config_sync** (*bool*) – Enabled configuration synchronization (Default: True)
- **peer_ip** (*str*) – HA Peer's HA1 IP address
- **mode** (*str*) – Mode of HA: 'active-passive' or 'active-active' (Default: 'active-passive')
- **passive_link_state** (*str*) – Passive link state
- **state_sync** (*bool*) – Enabled state synchronization (Default: False)
- **ha2_keepalive** (*bool*) – Enable HA2 keepalives
- **ha2_keepalive_action** (*str*) – HA2 keepalive action
- **ha2_keepalive_threshold** (*int*) – HA2 keepalive threshold
- **peer_ip_backup** (*str*) – HA Peer's HA1 backup IP address
- **device_id** (*int*) – HA3 device id (0 or 1)
- **session_owner_selection** (*str*) – active-active session owner mode
- **session_setup** (*str*) – active-active session setup mode
- **tentative_hold_time** (*int*) – active-active tentative hold timer
- **sync_qos** (*bool*) – active-active network sync qos
- **sync_virtual_router** (*bool*) – active-active network sync virtual router
- **ip_hash_key** (*str*) – active-active hash key used by ip-hash algorithm

class panos.ha.**HighAvailabilityInterface**(**args*, ***kwargs*)
Base class for high availability interface classes

Do not instantiate this class. Use its subclasses.

delete_interface(*interface=None*, *pan_device=None*)
Delete the data interface used by this HA interface

Parameters

- **interface** (HighAvailabilityInterface) – The HA interface (HA1, HA2, etc)
- **pan_device** (PanDevice) – The PanDevice object to apply the change

delete_old_interface()
Delete the data interface previously used by this HA interface

Use this if the 'port' of an HA interface was changed and the old interface needs to be cleaned up.

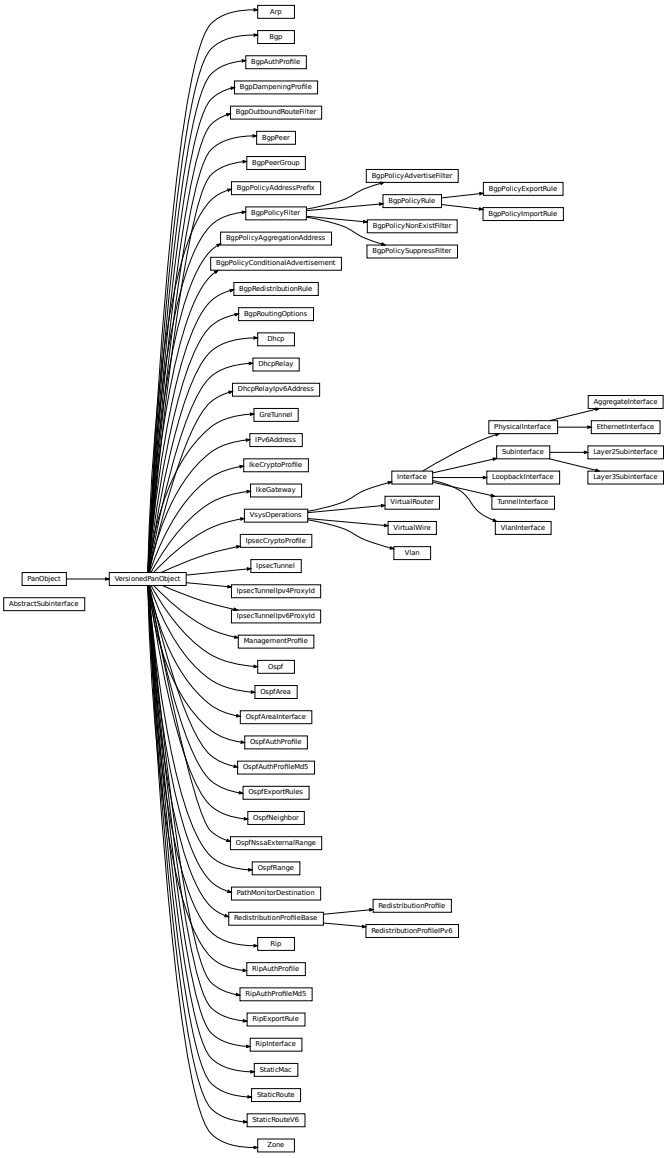
setup_interface()
Setup the data interface as an HA interface

Use this method to automatically convert the data interface to 'ha' mode. This must be done *before* this HA interface is created on the firewall.

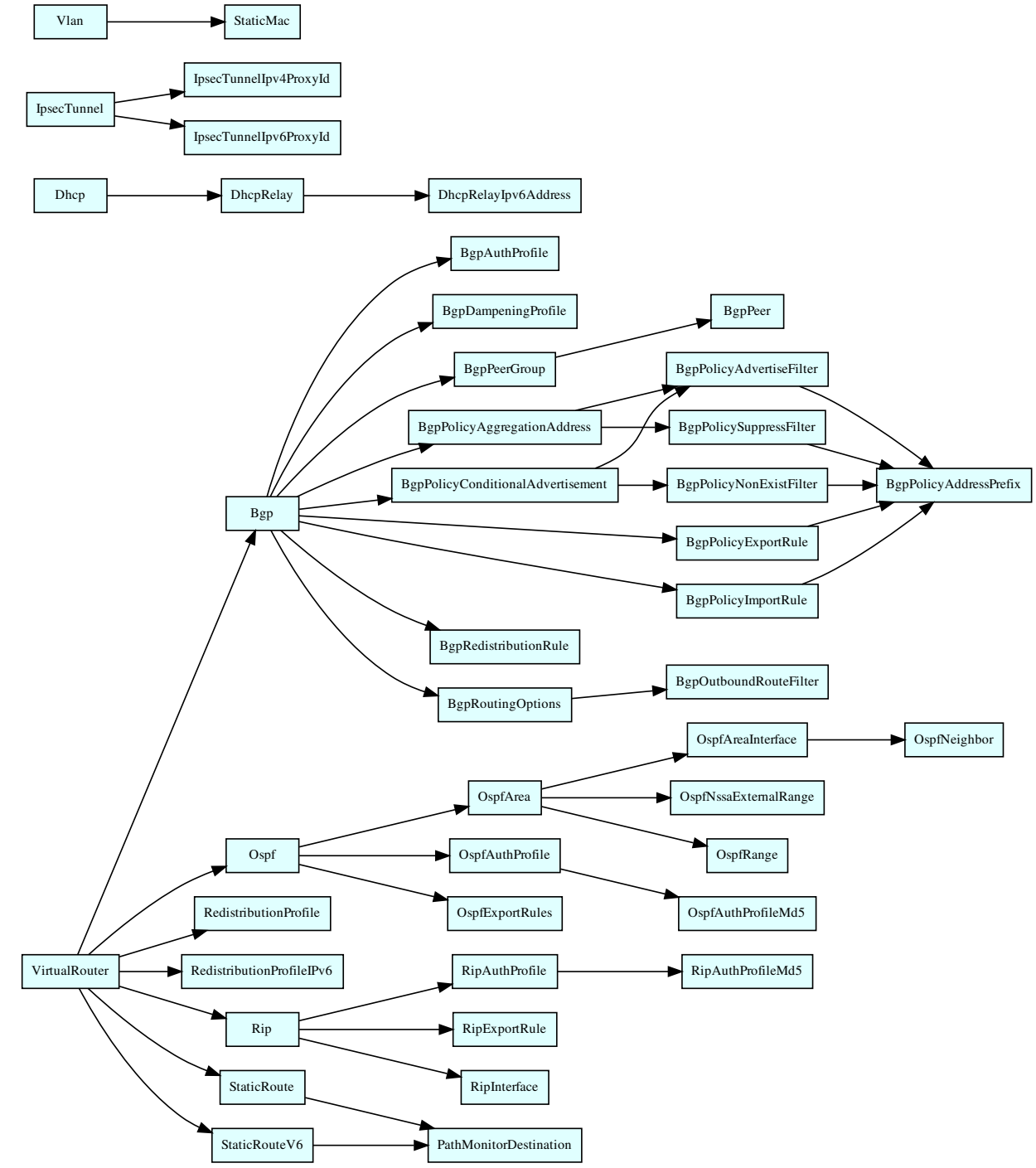
classmethod variables()
Defines the variables that exist in this object. Override in each subclass.

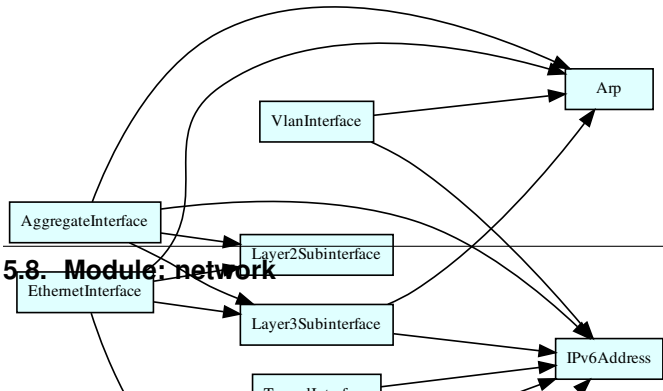
5.8 Module: network

5.8.1 Inheritance diagram



5.8.2 Configuration tree diagram





5.8.3 Class Reference

Network module contains objects that exist in the 'Network' tab in the firewall GUI

class panos.network.**AbstractSubinterface**(*name*, *tag*, *parent=None*)
When a subinterface is needed, but the layer is unknown

Kindof like a placeholder or reference for a Layer2Subinterface or Layer3Subinterface. This class gets a parent which is the ethernet or aggregate interface, but it should not be added to the parent interface with add().

Parameters

- **name** (*str*) – Name of the interface (eg. ethernet1/1.5)
- **tag** (*int*) – Tag for the interface, aka vlan id
- **parent** (Interface) – The base interface for this subinterface

delete()
Deletes both Layer3 and Layer2 subinterfaces by name

This is necessary because an AbstractSubinterface's mode is unknown.

get_layered_subinterface(*mode*, *add=True*)
Instantiate a regular subinterface type from this AbstractSubinterface

Converts an abstract subinterface to a real subinterface by offering it a mode.

Parameters

- **mode** (*str*) – Mode of the subinterface ('layer3' or 'layer2')
- **add** (*bool*) – Add the newly instantiated subinterface to the base interface object

Returns A *panos.network.Layer3Subinterface* or *panos.network.Layer2Subinterface* instance, depending on the mode argument

Return type *Subinterface*

nearest_pandevice()
The PanDevice parent for this instance

Returns Parent PanDevice instance (Firewall or Panorama)

Return type *PanDevice*

set_name()
Create a name appropriate for a subinterface if it isn't already created

Example

If self.name is 'ethernet1/1' and self.tag is 5, this method will change the name to 'ethernet1/1.5'.

set_virtual_router(*virtual_router_name*, *refresh=False*, *update=False*, *running_config=False*)
Set the virtual router for this interface

Creates a reference to this interface in the specified virtual router and removes references to this interface from all other virtual routers. The virtual router will be created if it doesn't exist.

Parameters

- **virtual_router_name** (*str*) – The name of the VirtualRouter or a *panos.network.VirtualRouter* instance

- **refresh** (*bool*) – Refresh the relevant current state of the device before taking action (Default: False)
- **update** (*bool*) – Apply the changes to the device (Default: False)
- **running_config** – If refresh is True, refresh from the running configuration (Default: False)

Returns The zone for this interface after the operation completes

Return type *Zone*

class panos.network.**AggregateInterface**(**args*, ***kwargs*)

Aggregate interface (eg. 'ae1')

Parameters

- **name** (*str*) – Name of interface (eg. 'ae1')
- **mode** (*str*) –

  Mode of the interface:

  - layer3
  - layer2
  - virtual-wire
  - ha

  Not all modes apply to all interface types (Default: layer3)
- **ip** (*tuple*) – Layer3: Interface IPv4 addresses
- **ipv6_enabled** (*bool*) – Layer3: IPv6 Enabled (requires IPv6Address child object)
- **management_profile** (ManagementProfile) – Layer3: Interface Management Profile
- **mtu** (*int*) – Layer3: MTU for interface
- **adjust_tcp_mss** (*bool*) – Layer3: Adjust TCP MSS
- **netflow_profile** (*str*) – Netflow profile
- **lldp_enabled** (*bool*) – Enable LLDP
- **lldp_profile** (*str*) – Reference to an lldp profile
- **comment** (*str*) – The interface's comment
- **ipv4_mss_adjust** (*int*) – Layer3: TCP MSS adjustment for ipv4
- **ipv6_mss_adjust** (*int*) – Layer3: TCP MSS adjustment for ipv6
- **enable_dhcp** (*bool*) – Enable DHCP on this interface
- **create_dhcp_default_route** (*bool*) – Layer3: Create default route pointing to default gateway provided by server
- **dhcp_default_route_metric** (*int*) – Layer3: Metric for the DHCP default route
- **lacp_enable** (*bool*) – Enables LACP
- **lacp_passive_pre_negotiation** (*bool*) – Enable LACP passive pre-negotiation, off by default
- **lacp_mode** (*str*) – Set LACP mode to 'active' or 'passive'

- **lacp_rate** (*str*) – Set LACP transmission-rate to 'fast' or 'slow'

class panos.network.**Arp**(**args*, ***kwargs*)
Static ARP Mapping

Can be added to various interfaces.

Parameters

- **ip** (*str*) – The IP address
- **hw_address** (*str*) – The MAC address for the static ARP
- **interface** (*str*) – The interface (when attached to VlanInterface only)

class panos.network.**Bgp**(**args*, ***kwargs*)
BGP Process

Parameters

- **enable** (*bool*) – Enable BGP (Default: True)
- **router_id** (*str*) – Router ID in IP format (eg. 1.1.1.1)
- **reject_default_route** (*bool*) – Reject default route
- **allow_redist_default_route** (*bool*) – Allow redistribution in default route
- **install_route** (*bool*) – Populate BGP learned route to global route table
- **ecmp_multi_as** (*bool*) – Support multiple AS in ECMP
- **enforce_first_as** (*bool*) – Enforce First AS for EBGP
- **local_as** (*int*) – local AS number
- **global_bfd_profile** (*str*) – BFD Profile

class panos.network.**BgpAuthProfile**(**args*, ***kwargs*)
BGP Authentication Profile

Parameters

- **name** (*str*) – Name of Auth Profile
- **secret** (*str*) – shared secret for the TCP MD5 authentication.

class panos.network.**BgpDampeningProfile**(**args*, ***kwargs*)
BGP Dampening Profile

Parameters

- **name** (*str*) – Name of Dampening Profile
- **enable** (*bool*) – Enable profile (Default: True)
- **cutoff** (*float*) – Cutoff threshold value
- **reuse** (*float*) – Reuse threshold value
- **max_hold_time** (*int*) – Maximum of hold-down time (in seconds)
- **decay_half_life_reachable** (*int*) – Decay half-life while reachable (in seconds)
- **decay_half_life_unreachable** (*int*) – Decay half-life while unreachable (in seconds)

class panos.network.**BgpOutboundRouteFilter**(**args*, ***kwargs*)
BGP Outbound Route Filtering

NOTE: This functionality is not enabled yet in PanOS

Parameters

- **enable** (*bool*) – enable prefix-based outbound route filtering.
- **max_received_entries** (*int*) – maximum of ORF prefixes to receive.
- **cisco_prefix_mode** (*bool*) – ORF vendor-compatible mode

class panos.network.**BgpPeer**(**args*, ***kwargs*)
BGP Peer

Parameters

- **name** (*str*) – Name of BGP Peer
- **enable** (*bool*) – Enable Peer (Default: True)
- **peer_as** (*str*) – peer AS number
- **enable_mp_bgp** (*bool*) – enable MP-BGP extentions
- **address_family_identifier** (*str*) – peer address family type * ipv4 * ipv6
- **subsequent_address_unicast** (*bool*) – select SAFI for this peer
- **subsequent_address_multicast** (*bool*) – select SAFI for this peer
- **local_interface** (*str*) – interface to accept BGP session
- **local_interface_ip** (*str*) – specify exact IP address if interface has multiple addresses
- **peer_address_ip** (*str*) – IP address of peer
- **connection_authentication** (*str*) – BGP auth profile name
- **connection_keep_alive_interval** (*int*) – keep-alive interval (in seconds)
- **connection_min_route_adv_interval** (*int*) – Minimum Route Advertisement Interval (in seconds)
- **connection_multihop** (*int*) – IP TTL value used for sending BGP packet. set to 0 means eBGP use 2, iBGP use 255
- **connection_open_delay_time** (*int*) – open delay time (in seconds)
- **connection_hold_time** (*int*) – hold time (in seconds)
- **connection_idle_hold_time** (*int*) – idle hold time (in seconds)
- **connection_incoming_allow** (*bool*) – allow incoming connections
- **connection_outgoing_allow** (*bool*) – allow outgoing connections
- **connection_incoming_remote_port** (*int*) – restrict remote port for incoming BGP connections
- **connection_outgoing_local_port** (*int*) – use specific local port for outgoing BGP connections
- **enable_sender_side_loop_detection** (*bool*) –
- **reflector_client** (*str*) –

  - non-client
  - client
  - meshed-client
- **peering_type** (*str*) –
  - unspecified
  - bilateral
- **max_prefixes** (*int*) – maximum of prefixes to receive from peer
- **bfd_profile** (*str*) – BFD configuration * Inherit-vr-global-setting * None * Pre-existing BFD profile name * None

class panos.network.**BgpPeerGroup**(**args*, ***kwargs*)
BGP Peer Group

Parameters

- **name** (*str*) – Name of BGP Peer Group
- **enable** (*bool*) – Enable Peer Group (Default: True)
- **aggregated_confed_as_path** (*bool*) – the peers understand aggregated confederation AS path
- **soft_reset_with_stored_info** (*bool*) – soft reset with stored info
- **type** (*str*) – peer group type I('ebgp')/I('ibgp')/I('ebgp-confed')/I('ibgp-confed')
- **export_nexthop** (*str*) – export locally resolved nexthop I('resolve')/I('use-self')
- **import_nexthop** (*str*) – override nexthop with peer address I('original')/I('use-peer'), only with 'ebgp'
- **remove_private_as** (*bool*) – remove private AS when exporting route, only with 'ebgp'

class panos.network.**BgpPolicyAddressPrefix**(**args*, ***kwargs*)
BGP Policy Address Prefix with Exact

Parameters

- **name** (*str*) – address prefix
- **exact** (*str*) – match exact prefix length

class panos.network.**BgpPolicyAdvertiseFilter**(**args*, ***kwargs*)
BGP Policy Advertise Filter

Parameters

- **name** (*str*) – Name of filter
- **enable** (*bool*) – Enable rule.
- **match_afi** (*str*) – Address Family Identifier * ip * ipv6
- **match_safi** (*str*) – Subsequent Address Family Identifier * ip * ipv6
- **match_route_table** (*str*) – Route table to match rule * unicast * multicast * both
- **match_nexthop** (*list*) – Next-hop attributes
- **match_from_peer** (*list*) – Filter by peer that sent this route

- **match_med** (*int*) – Multi-Exit Discriminator
- **match_as_path_regex** (*str*) – AS-path regular expression
- **match_community_regex** (*str*) – Community AS-path regular expression
- **match_extended_community_regex** (*str*) – Extended Community AS-path regular expression

class panos.network.**BgpPolicyAggregationAddress**(**args*, ***kwargs*)

BGP Policy Aggregation Address

Parameters

- **name** (*str*) – Sddress prefix
- **enable** (*bool*) – Enable aggregation for this prefix
- **prefix** (*str*) – Aggregating address prefix
- **summary** (*bool*) – Summarize route
- **as_set** (*bool*) – Generate AS-set attribute
- **attr_local_preference** (*int*) – New local preference value
- **attr_med** (*int*) – New MED value
- **attr_weight** (*int*) – New weight value
- **attr_nexthop** (*str*) – Nexthop address
- **attr_origin** (*str*) – New route origin * igp * egp * incomplete
- **attr_as_path_limit** (*int*) – Add AS path limit attribute if it does not exist
- **attr_as_path_type** (*str*) – AS path update options * none (string, not to be confused with the Python type None) * remove * prepend * remove-and-prepend
- **attr_as_path_prepend_times** (*int*) – Prepend local AS for specified number of times * only valid when attr_as_path_type is 'prepend' or 'remove-and-prepend'
- **attr_community_type** (*str*) – Community update options * none (string, not to be confused with the Python type None) * remove-all * remove-regex * append * overwrite
- **attr_community_argument** (*str*) – Argument to the attr community value if needed * None * local-as * no-advertise * no-export * nopeer * regex * 32-bit value * AS:VAL
- **attr_extended_community_type** (*str*) – Extended community update options * none (string, not to be confused with the Python type None) * remove-all * remove-regex * append * overwrite
- **attr_extended_community_argument** (*str*) – Argument to the attr extended community value if needed

class panos.network.**BgpPolicyConditionalAdvertisement**(**args*, ***kwargs*)

BGP Conditional Advertisement Policy

Parameters

- **name** (*str*) – Name of Conditional Advertisement Policy
- **enable** (*bool*) – enable prefix-based outbound route filtering.
- **used_by** (*list*) – peer-groups that use this rule.

class panos.network.**BgpPolicyExportRule**(**args*, ***kwargs*)

BGP Policy Export Rule

Parameters

- **name** (*str*) – The name
- **enable** (*bool*) – Enable rule.
- **match_afi** (*str*) – Address Family Identifier * ip * ipv6
- **match_safi** (*str*) – Subsequent Address Family Identifier * ip * ipv6
- **match_route_table** (*str*) – Route table to match rule * unicast * multicast * both
- **match_nexthop** (*list*) – Next-hop attributes
- **match_from_peer** (*list*) – Filter by peer that sent this route
- **match_med** (*int*) – Multi-Exit Discriminator
- **match_as_path_regex** (*str*) – AS-path regular expression
- **match_community_regex** (*str*) – AS-path regular expression
- **match_extended_community_regex** (*str*) – AS-path regular expression
- **used_by** (*list*) – Peer-groups that use this rule.
- **action** (*str*) – The action
- **action_local_preference** (*int*) – New local preference value
- **action_med** (*int*) – New MED value
- **action_nexthop** (*str*) – Nexthop address
- **action_origin** (*str*) – New route origin * igp * egp * incomplete
- **action_as_path_limit** (*int*) – Add AS path limit attribute if it does not exist
- **action_as_path_type** (*str*) – AS path update options * none (string, not to be confused with the Python type None) * remove * prepend * remove-and-prepend
- **action_as_path_prepend_times** (*int*) – Prepend local AS for specified number of times * only valid when action_as_path_type is 'prepend' or 'remove-and-prepend'
- **action_community_type** (*str*) – Community update options * none (string, not to be confused with the Python type None) * remove-all * remove-regex * append * overwrite
- **action_community_argument** (*str*) – Argument to the action community value if needed * None * local-as * no-advertise * no-export * nopeer * regex * 32-bit value * AS:VAL
- **action_extended_community_type** (*str*) – Extended community update options * none (string, not to be confused with the Python type None) * remove-all * remove-regex * append * overwrite
- **action_extended_community_argument** (*str*) – Argument to the action extended community value if needed

class panos.network.**BgpPolicyFilter**(**args*, ***kwargs*)

Base class for BGP Policy Match Filters

Do not instantiate this class, use one of:

- BgpPolicyImportRule
- BgpPolicyExportRule

Parameters

- **name** (*str*) – Name of filter
- **enable** (*bool*) – Enable rule.
- **match_afi** (*str*) – Address Family Identifier * ip * ipv6
- **match_safi** (*str*) – Subsequent Address Family Identifier * ip * ipv6
- **match_route_table** (*str*) – Route table to match rule * unicast * multicast * both
- **match_nexthop** (*list*) – Next-hop attributes
- **match_from_peer** (*list*) – Filter by peer that sent this route
- **match_med** (*int*) – Multi-Exit Discriminator
- **match_as_path_regex** (*str*) – AS-path regular expression
- **match_community_regex** (*str*) – Community AS-path regular expression
- **match_extended_community_regex** (*str*) – Extended Community AS-path regular expression

class panos.network.**BgpPolicyImportRule**(**args*, ***kwargs*)

BGP Policy Import Rule

Parameters

- **name** (*str*) – The name
- **enable** (*bool*) – Enable rule.
- **match_afi** (*str*) – Address Family Identifier * ip * ipv6
- **match_safi** (*str*) – Subsequent Address Family Identifier * ip * ipv6
- **match_route_table** (*str*) – Route table to match rule * unicast * multicast * both
- **match_nexthop** (*list*) – Next-hop attributes
- **match_from_peer** (*list*) – Filter by peer that sent this route
- **match_med** (*int*) – Multi-Exit Discriminator
- **match_as_path_regex** (*str*) – AS-path regular expression
- **match_community_regex** (*str*) – AS-path regular expression
- **match_extended_community_regex** (*str*) – AS-path regular expression
- **used_by** (*list*) – Peer-groups that use this rule.
- **action** (*str*) – The action
- **action_local_preference** (*int*) – New local preference value
- **action_med** (*int*) – New MED value
- **action_nexthop** (*str*) – Nexthop address
- **action_origin** (*str*) – New route origin * igp * egp * incomplete
- **action_as_path_limit** (*int*) – Add AS path limit attribute if it does not exist
- **action_as_path_type** (*str*) – AS path update options * none (string, not to be confused with the Python type None) * remove * prepend * remove-and-prepend
- **action_as_path_prepend_times** (*int*) – Prepend local AS for specified number of times * only valid when action_as_path_type is 'prepend' or 'remove-and-prepend'

- **action_community_type** (*str*) – Community update options * none (string, not to be confused with the Python type None) * remove-all * remove-regex * append * overwrite
- **action_community_argument** (*str*) – Argument to the action community value if needed * None * local-as * no-advertise * no-export * nopeer * regex * 32-bit value * AS:VAL
- **action_extended_community_type** (*str*) – Extended community update options * none (string, not to be confused with the Python type None) * remove-all * remove-regex * append * overwrite
- **action_extended_community_argument** (*str*) – Argument to the action extended community value if needed
- **action_dampening** (*str*) – Route flap dampening profile
- **action_weight** (*int*) – New weight value

class panos.network.**BgpPolicyNonExistFilter**(**args*, ***kwargs*)

BGP Policy Non-Exist Filter

Parameters

- **name** (*str*) – Name of filter
- **enable** (*bool*) – Enable rule.
- **match_afi** (*str*) – Address Family Identifier * ip * ipv6
- **match_safi** (*str*) – Subsequent Address Family Identifier * ip * ipv6
- **match_route_table** (*str*) – Route table to match rule * unicast * multicast * both
- **match_nexthop** (*list*) – Next-hop attributes
- **match_from_peer** (*list*) – Filter by peer that sent this route
- **match_med** (*int*) – Multi-Exit Discriminator
- **match_as_path_regex** (*str*) – AS-path regular expression
- **match_community_regex** (*str*) – Community AS-path regular expression
- **match_extended_community_regex** (*str*) – Extended Community AS-path regular expression

class panos.network.**BgpPolicyRule**(**args*, ***kwargs*)

Base class for BGP Policy Import/Export Rules

Do not instantiate this class, use one of:

- BgpPolicyImportRule
- BgpPolicyExportRule

Parameters

- **name** (*str*) – The name
- **enable** (*bool*) – Enable rule.
- **match_afi** (*str*) – Address Family Identifier * ip * ipv6
- **match_safi** (*str*) – Subsequent Address Family Identifier * ip * ipv6
- **match_route_table** (*str*) – Route table to match rule * unicast * multicast * both
- **match_nexthop** (*list*) – Next-hop attributes

- **match_from_peer** (*list*) – Filter by peer that sent this route
- **match_med** (*int*) – Multi-Exit Discriminator
- **match_as_path_regex** (*str*) – AS-path regular expression
- **match_community_regex** (*str*) – AS-path regular expression
- **match_extended_community_regex** (*str*) – AS-path regular expression
- **used_by** (*list*) – Peer-groups that use this rule.
- **action** (*str*) – The action
- **action_local_preference** (*int*) – New local preference value
- **action_med** (*int*) – New MED value
- **action_nexthop** (*str*) – Nexthop address
- **action_origin** (*str*) – New route origin * igp * egp * incomplete
- **action_as_path_limit** (*int*) – Add AS path limit attribute if it does not exist
- **action_as_path_type** (*str*) – AS path update options * none (string, not to be confused with the Python type None) * remove * prepend * remove-and-prepend
- **action_as_path_prepend_times** (*int*) – Prepend local AS for specified number of times * only valid when action_as_path_type is 'prepend' or 'remove-and-prepend'
- **action_community** (*str*) – Community update options * none (string, not to be confused with the Python type None) * remove-all * remove-regex * append * overwrite
- **action_community_argument** (*str*) – Argument to the action community value if needed * None * local-as * no-advertise * no-export * nopeer * regex * 32-bit value * AS:VAL
- **action_extended_community_type** (*str*) – Extended community update options * none (string, not to be confused with the Python type None) * remove-all * remove-regex * append * overwrite
- **action_extended_community_argument** (*str*) – Argument to the action extended community value if needed

class panos.network.**BgpPolicySuppressFilter**(**args*, ***kwargs*)

BGP Policy Suppress Filter

Parameters

- **name** (*str*) – Name of filter
- **enable** (*bool*) – Enable rule.
- **match_afi** (*str*) – Address Family Identifier * ip * ipv6
- **match_safi** (*str*) – Subsequent Address Family Identifier * ip * ipv6
- **match_route_table** (*str*) – Route table to match rule * unicast * multicast * both
- **match_nexthop** (*list*) – Next-hop attributes
- **match_from_peer** (*list*) – Filter by peer that sent this route
- **match_med** (*int*) – Multi-Exit Discriminator
- **match_as_path_regex** (*str*) – AS-path regular expression
- **match_community_regex** (*str*) – Community AS-path regular expression

- **match_extended_community_regex** (*str*) – Extended Community AS-path regular expression

class panos.network.**BgpRedistributionRule**(**args*, ***kwargs*)
BGP Policy Address Prefix with Exact

Parameters

- **name** (*str*) – Redistribution profile name
- **enable** (*bool*) – Enable redistribution rule.
- **address_family_identifier** (*str*) – Select redistribution profile type * ipv4 * ipv6
- **route_table** (*str*) – Select destination SAFI for redistribution * unicast * multicast * both
- **set_origin** (*str*) – Add the ORIGIN path attribute * igp * egp * incomplete
- **set_med** (*int*) – Add the MULTI_EXIT_DISC path attribute
- **set_local_preference** (*int*) – Add the LOCAL_PREF path attribute
- **set_as_path_limit** (*int*) – Add the AS_PATHLIMIT path attribute
- **set_community** (*list*) – Add the COMMUNITY path attribute
- **set_extended_community** (*list*) – Add the EXTENDED COMMUNITY path attribute
- **metric** (*int*) – Metric value

class panos.network.**BgpRoutingOptions**(**args*, ***kwargs*)
BGP Routing Options

Parameters

- **as_format** (*str*) – AS format ('2-byte'/'4-byte')
- **always_compare_med** (*bool*) – always compare MEDs
- **deterministic_med_comparison** (*bool*) – deterministic MEDs comparison
- **default_local_preference** (*int*) – default local preference
- **graceful_restart_enable** (*bool*) – enable graceful restart
- **gr_stale_route_time** (*int*) – time to remove stale routes after peer restart (in seconds)
- **gr_local_restart_time** (*int*) – local restart time to advertise to peer (in seconds)
- **gr_max_peer_restart_time** (*int*) – maximum of peer restart time accepted (in seconds)
- **reflector_cluster_id** (*str*) – route reflector cluster ID
- **confederation_member_as** (*str*) – 32-bit value in decimal or dot decimal AS.AS format
- **aggregate_med** (*bool*) – aggregate route only if they have same MED attributes

class panos.network.**Dhcp**(**args*, ***kwargs*)
DHCP config.

Parameters **name** (*str*) – Interface name.

class panos.network.**DhcpRelay**(**args*, ***kwargs*)
DHCP relay config.

Parameters

- **name** (*str*) – The (interface) name
- **enabled** (*bool*) – Enabled.
- **servers** (*list*) – Relay server IP addresses.
- **ipv6_enabled** (*bool*) – Enable DHCPv6 relay.

class panos.network.**DhcpRelayIpv6Address**(**args*, ***kwargs*)
DHCP relay IPv6 address.

Parameters

- **name** (*str*) – DHCP server IPv6 address.
- **interface** (*str*) – Outgoing interface when using an IPv6 multicast address for the DHCPv6 server.

class panos.network.**EthernetInterface**(**args*, ***kwargs*)
Ethernet interface (eg. 'ethernet1/1')

Parameters

- **name** (*str*) – Name of interface (eg. 'ethernet1/1')
- **mode** (*str*) –

  Mode of the interface:

  - layer3
  - layer2
  - virtual-wire
  - tap
  - ha
  - decrypt-mirror
  - aggregate-group

  Not all modes apply to all interface types (Default: layer3)
- **ip** (*tuple*) – Layer3: Interface IPv4 addresses
- **ipv6_enabled** (*bool*) – Layer3: IPv6 Enabled (requires IPv6Address child object)
- **management_profile** (ManagementProfile) – Layer3: Interface Management Profile
- **mtu** (*int*) – Layer3: MTU for interface
- **adjust_tcp_mss** (*bool*) – Layer3: Adjust TCP MSS
- **netflow_profile** (*str*) – Netflow profile
- **lldp_enabled** (*bool*) – Layer2: Enable LLDP
- **lldp_profile** (*str*) – Layer2: Reference to an lldp profile
- **netflow_profile_l2** (*str*) – Netflow profile
- **link_speed** (*str*) – Link speed: eg. auto, 10, 100, 1000

- **link_duplex** (*str*) – Link duplex: eg. auto, full, half
- **link_state** (*str*) – Link state: eg. auto, up, down
- **aggregate_group** (*str*) – Aggregate interface (eg. ae1)
- **comment** (*str*) – The interface's comment
- **ipv4_mss_adjust** (*int*) – (PAN-OS 7.1+) TCP MSS adjustment for ipv4
- **ipv6_mss_adjust** (*int*) – (PAN-OS 7.1+) TCP MSS adjustment for ipv6
- **enable_dhcp** (*bool*) – Enable DHCP on this interface
- **create_dhcp_default_route** (*bool*) – Create default route pointing to default gateway provided by server
- **dhcp_default_route_metric** (*int*) – Metric for the DHCP default route
- **enable_untagged_subinterface** (*bool*) – (PAN-OS 7.1+) Enable untagged subinterface
- **decrypt_forward** (*bool*) – (PAN-OS 8.1+) Decrypt forward.
- **rx_policing_rate** (*int*) – (PAN-OS 8.1+) Receive policing rate
- **tx_policing_rate** (*int*) – (PAN-OS 8.1+) Transmit policing rate
- **dhcp_send_hostname_enable** (*bool*) – Enable send firewall or custom hostname to DHCP server
- **dhcp_send_hostname_value** (*string*) – Set interface hostname

class panos.network.**GreTunnel**(**args*, ***kwargs*)

GRE Tunnel configuration.

Note: This is valid for PAN-OS 9.0+

Parameters

- **name** – GRE tunnel name.
- **interface** – Interface to terminate tunnel.
- **local_address_type** – Type of local address. Can be "ip" (default) or "floating-ip".
- **local_address_value** – IP address value.
- **peer_address** – Peer IP address.
- **tunnel_interface** – To apply GRE tunnels to tunnel interface.
- **ttl** (*int*) – TTL.
- **copy_tos** (*bool*) – Copy IP TOS bits from inner packet to GRE packet.
- **enable_keep_alive** (*bool*) – Enable tunnel monitoring.
- **keep_alive_interval** (*int*) – Interval.
- **keep_alive_retry** (*int*) – Retry.
- **keep_alive_hold_timer** (*int*) – Hold timer.
- **disabled** (*bool*) – Disable the GRE tunnel.

class panos.network.**IPv6Address**(**args*, ***kwargs*)

IPv6 Address

Can be added to any *panos.network.Interface* subclass that supports IPv6.

Parameters

- **address** (*str*) – The IPv6 address
- **enable_on_interface** (*bool*) – Enabled IPv6 on the interface this object was added to
- **prefix** (*bool*) – Use interface ID as host portion
- **anycast** (*bool*) – Enable anycast
- **advertise_enabled** (*bool*) – Enabled router advertisements
- **valid_lifetime** (*int*) – Valid lifetime
- **preferred_lifetime** (*int*) – Preferred lifetime
- **onlink_flag** (*bool*) –
- **auto_config_flag** (*bool*) –

class panos.network.**IkeCryptoProfile**(**args*, ***kwargs*)

IKE SA proposal.

Parameters

- **name** – IKE crypto profile name
- **dh_group** (*string/list*) – phase-1 DH group: group1, group2, group5, group14, group19 (7.0+), or group20 (7.0+).
- **authentication** (*string/list*) – hashing algorithm: md5, sha1, sha256, sha384, or sha512.
- **encryption** (*string/list*) – encryption algorithm: des (7.1+), 3des, aes128 / aes-128-cbc, aes192 / aes-192-cbc, or aes256 / aes-256-cbc. If you need to be able to work with older than 7.0 firewalls, then use set_encryption().
- **lifetime_seconds** (*int*) – IKE SA lifetime in seconds
- **lifetime_minutes** (*int*) – IKE SA lifetime in minutes
- **lifetime_hours** (*int*) – IKE SA lifetime in hours
- **lifetime_days** (*int*) – IKE SA lifetime in days
- **authentication_multiple** (*int*) – (7.0+) IKEv2 SA reauthentication interval equals authentication_multiple * lifetime; 0 means reauthentication is disabled.

set_encryption(*value*)

Version agnostic set for encryption.

This object should be connected to a panos.Firewall before invocation.

Valid values include the following:

- des (7.1+)
- 3des
- aes128
- aes-128-cbc
- aes192
- aes-192-cbc
- aes256

- aes-256-cbc

Raises

- `PanDeviceNotSet` – if there is no Firewall in the object tree
- `ValueError` – if value is not one of the above, or you attempt to configure 3des with this object connected to a PANOS 7.0 or earlier firewall.

class panos.network.**IkeGateway**(**args*, ***kwargs*)

IKE Gateway.

Parameters

- **name** – IKE gateway name
- **version** – (7.0+) ikev1, ikev2, or ikev2-prefered (default: ikev1)
- **enable_ipv6** (*bool*) – (7.0+) enable IPv6
- **disabled** (*bool*) – (7.0+) disable this object
- **peer_ip_type** – ip, dynamic, or fqdn (8.1+) (default: ip)
- **peer_ip_value** – the IP for peer_ip_type of 'ip' or 'fqdn'
- **interface** – local gateway end-point
- **local_ip_address_type** – ip or floating-ip
- **local_ip_address** – IP address if interface has multiple addresses
- **auth_type** – pre-shared-key or certificate (default: pre-shared-key)
- **pre_shared_key** – The string used as pre-shared key
- **local_id_type** – ipaddr, fqdn, ufqdn, keyid, or dn
- **local_id_value** – The value for local_id_type
- **peer_id_type** – ipaddr, fqdn, ufqdn, keyid, or dn
- **peer_id_value** – The value for peer_id_type
- **peer_id_check** – exact or wildcard (default: exact)
- **local_cert** – Local certificate name
- **cert_enable_hash_and_url** (*bool*) – (7.0+) Use hash-and-url for local certificate.
- **cert_base_url** – (7.0+) The host and directory part of URL for local certificates (http only).
- **cert_use_management_as_source** (*bool*) – (7.0+) Use management interface IP as source to retrieve http certificates
- **cert_permit_payload_mismatch** (*bool*) – Permit peer identification and certificate payload identification mismatch.
- **cert_profile** – Local certificate name
- **cert_enable_strict_validation** (*bool*) – Enable strict validation of peer's extended key use
- **enable_passive_mode** (*bool*) – Enable passive mode (responder only)
- **enable_nat_traversal** (*bool*) – Enable NAT traversal

- **nat_traversal_keep_alive** (*int*) – sending interval for NAT keep-alive packets (in seconds)
- **nat_traversal_enable_udp_checksum** (*bool*) – enable UDP checksum
- **enable_fragmentation** (*bool*) – Enable IKE fragmentation
- **ikev1_exchange_mode** – auto, main, or aggressive
- **ikev1_crypto_profile** – IKE SA crypto oprofile name
- **enable_dead_peer_detection** (*bool*) – enable Dead-Peer-Detection
- **dead_peer_detection_interval** (*int*) – sending interval for probing packets (in seconds)
- **dead_peer_detection_retry** (*int*) – number of retries before disconnection
- **ikev1_send_commit_bit** (*bool*) – Send commit bit
- **ikev1_initial_contact** (*bool*) – send initial contact
- **ikev2_crypto_profile** – (7.0+) IKE SE crypto profile name
- **ikev2_cookie_validation** (*bool*) – (7.0+) require cookie
- **ikev2_send_peer_id** (*bool*) – (7.0+) send peer ID
- **enable_liveness_check** (*bool*) – (7.0+) enable sending empty information liveness check message
- **liveness_check_interval** (*int*) – (7.0+) delay interval before sending probing packets (in seconds)

class panos.network.**Interface**(**args*, ***kwargs*)

Base class for all interfaces

Do not instantiate this object. Use a subclass. Methods in this class are available to all interface subclasses.

Parameters

- **name** (*str*) – Name of the interface
- **state** (*str*) – Link state, 'up' or 'down'

full_delete(*refresh=False*, *delete_referencing_objects=False*, *include_vsys=False*)

Delete the interface and all references to the interface

Often when deleting an interface there is an API error because there are still references to the interface from zones, virtual-router, vsys, etc. This method deletes all references to the interface before deleting the interface itself.

Parameters

- **refresh** (*bool*) – Refresh the current state of the device before taking action
- **delete_referencing_objects** (*bool*) – Delete the entire object that references this interface

get_counters()

Pull the counters for an interface

Returns

counter name as key, counter as value, None if interface is not configured

Return type dict

refresh_state()
Pull the state of the interface from the firewall

The attribute 'state' is populated with the current state from the firewall.

Returns The current state from the firewall

Return type str

set_virtual_router(*virtual_router_name*, *refresh=False*, *update=False*, *running_config=False*, *return_type='object'*)
Set the virtual router for this interface

Creates a reference to this interface in the specified virtual router and removes references to this interface from all other virtual routers. The virtual router will be created if it doesn't exist.

Parameters

- **virtual_router_name** (*str*) – The name of the VirtualRouter or a *panos.network.VirtualRouter* instance
- **refresh** (*bool*) – Refresh the relevant current state of the device before taking action (Default: False)
- **update** (*bool*) – Apply the changes to the device (Default: False)
- **running_config** – If refresh is True, refresh from the running configuration (Default: False)
- **return_type** (*str*) – Specify what this function returns, can be either 'object' (the default) or 'bool'. If this is 'object', then the return value is the VirtualRouter in question. If this is 'bool', then the return value is a boolean that tells you about if the live device needs updates (update=False) or was updated (update=True).

Returns The zone for this interface after the operation completes

Return type *Zone*

set_vlan(*vlan_name*, *refresh=False*, *update=False*, *running_config=False*, *return_type='object'*)
Set the vlan for this interface

Creates a reference to this interface in the specified vlan and removes references to this interface from all other interfaces. The vlan will be created if it doesn't exist.

Parameters

- **vlan_name** (*str*) – The name of the vlan or a *panos.network.Vlan* instance
- **refresh** (*bool*) – Refresh the relevant current state of the device before taking action (Default: False)
- **update** (*bool*) – Apply the changes to the device (Default: False)
- **running_config** – If refresh is True, refresh from the running configuration (Default: False)
- **return_type** (*str*) – Specify what this function returns, can be either 'object' (the default) or 'bool'. If this is 'object', then the return value is the Vlan in question. If this is 'bool', then the return value is a boolean that tells you about if the live device needs updates (update=False) or was updated (update=True).

Raises AttributeError – if this class is not allowed to use this function.

Returns The VLAN for this interface after the operation completes

Return type *Vlan*

set_zone(*zone_name*, *mode=None*, *refresh=False*, *update=False*, *running_config=False*, *return_type='object'*)

Set the zone for this interface

Creates a reference to this interface in the specified zone and removes references to this interface from all other zones. The zone will be created if it doesn't exist.

Parameters

- **zone_name** (*str*) – The name of the Zone or a *panos.network.Zone* instance
- **mode** (*str*) – The mode of the zone. See *panos.network.Zone* for possible values
- **refresh** (*bool*) – Refresh the relevant current state of the device before taking action (Default: False)
- **update** (*bool*) – Apply the changes to the device (Default: False)
- **running_config** – If refresh is True, refresh from the running configuration (Default: False)
- **return_type** (*str*) – Specify what this function returns, can be either 'object' (the default) or 'bool'. If this is 'object', then the return value is the Zone in question. If this is 'bool', then the return value is a boolean that tells you about if the live device needs updates (update=False) or was updated (update=True).

Returns The zone for this interface after the operation completes

Return type *Zone*

up()

Link state of interface

Returns

True if state is 'up', False if state is 'down', 'unconfigured' or other

Return type bool

class panos.network.**IpsecCryptoProfile**(**args*, ***kwargs*)

IPSec SA proposals.

Parameters

- **name** – IPSec crypto profile name
- **esp_encryption** (*string/list*) – des, 3des, null, aes128 / aes-128-cbc, aes192 / aes-192-cbc, aes256 / aes-256-cbc, aes-128-gcm (7.0+), or aes-256-gcm (7.0+). If you need to write a script that works older than 7.0 firewalls, then please use set_esp_encryption().
- **esp_authentication** (*string/list*) – none, md5, sha1, sha256, sha384, or sha512
- **ah_authentication** (*string/list*) – md5, sha1, sha256, sha384, or sha512
- **dh_group** – no-pfs, group1, group2, group5, group14, group19, or group20
- **lifetime_seconds** (*int*) – IPSec SA lifetime in seconds
- **lifetime_minutes** (*int*) – IPSec SA lifetime in minutes
- **lifetime_hours** (*int*) – IPSec SA lifetime in hours
- **lifetime_days** (*int*) – IPSec SA lifetime in days
- **lifesize_kb** (*int*) – IPSec SA lifesize in kilobytes (KB)
- **lifesize_mb** (*int*) – IPSec SA lifesize in megabytes (MB)

- **lifesize_gb** (*int*) – IPSec SA lifesize in gigabytes (GB)
- **lifesize_tb** (*int*) – IPSec SA lifesize in terabytes (TB)

set_esp_encryption(*value*)
Version agnostic set for esp_encryption.

This object should be connected to a panos.Firewall before invocation.

Valid values include the following:

- des
- 3des
- aes128
- aes-128-cbc
- aes192
- aes-192-cbc
- aes256
- aes-256-cbc
- aes-128-gcm (7.0+)
- aes-256-gcm (7.0+)
- null

Parameters **value** (*string/list*) – values to put in esp_encryption

Raises

- PanDeviceNotSet – if there is no Firewall in the object tree
- ValueError – if value is not one of the above, or you attempt to configure aes-128-gcm or aes-256-gcm with this object connected to a PANOS 6.1 firewall.

class panos.network.**IpsecTunnel**(**args*, ***kwargs*)
IPSec Tunnel

A large number of params have prefixes:

- ak: Auto Key
- mk: Manual Key
- gps: GlobalProtect Satellite

Only attach IpsecTunnelIpv4ProxyId or IpsecTunnelIpv4ProxyId objects to this one if you are using type='auto-key'.

Parameters

- **name** – IPSec tunnel name
- **tunnel_interface** – apply IPSec VPN tunnels to tunnel interface
- **ipv6** (*bool*) – (7.0+) use IPv6 for the IPSec tunnel
- **type** – auto-key (default), manual-key, or global-protect-satellite
- **ak_ike_gateway** (*string/list*) – IKE gateway name
- **ak_ipsec_crypto_profile** – IPSec crypto profile name

- **mk_local_spi** – outbound SPI in hex
- **mk_interface** – interface to terminate tunnel
- **mk_remote_spi** – inbound SPI in hex
- **mk_remote_address** – tunnel peer IP address
- **mk_local_address_ip** – exact IP address if interface has multiple IP addresses
- **mk_local_address_floating_ip** – floating IP address in HA Active-Active configuration
- **mk_protocol** – esp or ah
- **mk_auth_type** – md5, sha1, sha256, sha384, or sha512
- **mk_auth_key** – the key for the given mk_auth_type
- **mk_esp_encryption** – des, 3des, aes128 / aes-128-cbc, aes192 / aes-192-cbc, aes256 / aes-256-cbc, or null. The various "aes" options changed in version 7.0 onward. If you need to make a script that is compatible with 6.1 PANOS, then use "set_mk_esp_encryption()". Passing it either "aes128" or "aes-128-cbc" will have it set the appropriate string for the given version.
- **mk_esp_encryption_key** – The ESP encryption key for mk_esp_encryption type
- **gps_portal_address** – GlobalProtect portal address
- **gps_prefer_ipv6** (*bool*) – (8.0+) perfer to register portal in IPv6
- **gps_interface** – interface to communicate with portal
- **gps_interface_ipv4_ip** – exact IPv4 IP address if interface has multiple IP addresses
- **gps_interface_ipv6_ip** – (8.0+) exact IPv6 IP address if interface has multiple IP addresses
- **gps_interface_ipv4_floating_ip** – (7.0+) floating IPv4 IP address in HA Active-Active configuration
- **gps_interface_ipv6_floating_ip** – (8.0+) floating IPv6 IP address in HA Active-Active configuration
- **gps_publish_connected_routes** (*bool*) – enable publishing of connected and static routes
- **gps_publish_routes** (*str/list*) – specify list of routes to publish to GlobalProtect gateway
- **gps_local_certificate** – GlobalProtect satellite certificate file name
- **gps_certificate_profile** – profile for authenticating GlobalProtect gateway certificates
- **anti_replay** (*bool*) – enable anti-replay check on this tunnel
- **copy_tos** (*bool*) – copy IP TOS bits from inner packet to IPSec packet (not recommended)
- **copy_flow_label** (*bool*) – (7.0+) copy IPv6 flow label for 6in6 tunnel from inner packet to IPSec packet (not recommended)
- **enable_tunnel_monitor** (*bool*) – enable tunnel monitoring on this tunnel
- **tunnel_monitor_dest_ip** – destination IP to send ICMP probe

- **tunnel_monitor_proxy_id** – (7.0+) which proxy-id (or proxy-id-v6) the monitoring traffic will use
- **tunnel_monitor_profile** – monitoring action
- **disabled** (*bool*) – (7.0+) disable the IPSec tunnel

set_mk_esp_encryption(*value*)
Version agnostic set for mk_esp_encryption.

This object should be connected to a panos.Firewall before invocation.

Valid values include the following:

- des
- 3des
- aes128
- aes-128-cbc
- aes192
- aes-192-cbc
- aes256
- aes-256-cbc
- null

Raises

- PanDeviceNotSet – if there is no Firewall in the object tree
- ValueError – if value is not one of the above

class panos.network.**IpsecTunnelIpv4ProxyId**(**args*, ***kwargs*)
IKEv1 proxy-id for auto-key IPSec tunnels.

Parameters

- **name** – The proxy ID
- **local** – IP subnet or IP address represents local network
- **remote** – IP subnet or IP address represents remote network
- **any_protocol** (*bool*) – Any protocol
- **number_protocol** (*int*) – Numbered Protocol: protocol number (1-254)
- **tcp_local_port** (*int*) – Protocol TCP: local port
- **tcp_remote_port** (*int*) – Protocol TCP: remote port
- **udp_local_port** (*int*) – Protocol UDP: local port
- **udp_remote_port** (*int*) – Protocol UDP: remote port

class panos.network.**IpsecTunnelIpv6ProxyId**(**args*, ***kwargs*)
IKEv1 IPv6 proxy-id for auto-key IPSec tunnels.

NOTE: Only supported in 7.0 and forward.

Parameters

- **name** – The proxy ID

- **local** – IP subnet or IP address represents local network
- **remote** – IP subnet or IP address represents remote network
- **any_protocol** (*bool*) – Any protocol
- **number_protocol** (*int*) – Numbered Protocol: protocol number (1-254)
- **tcp_local_port** (*int*) – Protocol TCP: local port
- **tcp_remote_port** (*int*) – Protocol TCP: remote port
- **udp_local_port** (*int*) – Protocol UDP: local port
- **udp_remote_port** (*int*) – Protocol UDP: remote port

class panos.network.**Layer2Subinterface**(**args*, ***kwargs*)
Ethernet or Aggregate Subinterface in Layer 2 mode.

Parameters

- **name** (*str*) – The name
- **tag** (*int*) – Tag for the interface, aka vlan id
- **lldp_enabled** (*bool*) – Enable LLDP
- **lldp_profile** (*str*) – Reference to an lldp profile
- **netflow_profile_l2** (*str*) – Netflow profile
- **comment** (*str*) – The interface's comment

class panos.network.**Layer3Subinterface**(**args*, ***kwargs*)
Ethernet or Aggregate Subinterface in Layer 3 mode.

Parameters

- **name** (*str*) – The name
- **tag** (*int*) – Tag for the interface, aka vlan id
- **ip** (*tuple*) – Interface IPv4 addresses
- **ipv6_enabled** (*bool*) – IPv6 Enabled (requires IPv6Address child object)
- **management_profile** (ManagementProfile) – Interface Management Profile
- **mtu** (*int*) – MTU for interface
- **adjust_tcp_mss** (*bool*) – Adjust TCP MSS
- **netflow_profile** (*str*) – Netflow profile
- **comment** (*str*) – The interface's comment
- **ipv4_mss_adjust** (*int*) – TCP MSS adjustment for ipv4
- **ipv6_mss_adjust** (*int*) – TCP MSS adjustment for ipv6
- **enable_dhcp** (*bool*) – Enable DHCP on this interface
- **create_dhcp_default_route** (*bool*) – Create default route pointing to default gateway provided by server
- **dhcp_default_route_metric** (*int*) – Metric for the DHCP default route
- **decrypt_forward** (*bool*) – (PAN-OS 8.1+) Decrypt forward.

class panos.network.**LoopbackInterface**(**args*, ***kwargs*)
Loopback interface

Parameters

- **name** (*str*) – The name
- **ip** (*tuple*) – Interface IPv4 addresses
- **ipv6_enabled** (*bool*) – IPv6 Enabled (requires IPv6Address child object)
- **management_profile** (ManagementProfile) – Interface Management Profile
- **mtu** (*int*) – MTU for interface
- **adjust_tcp_mss** (*bool*) – Adjust TCP MSS
- **netflow_profile** (*str*) – Netflow profile
- **comment** (*str*) – The interface's comment
- **ipv4_mss_adjust** (*int*) – TCP MSS adjustment for ipv4
- **ipv6_mss_adjust** (*int*) – TCP MSS adjustment for ipv6

class panos.network.**ManagementProfile**(**args*, ***kwargs*)
Interface management provile.

Add to any of the following interfaces:

- Layer3Subinterface
- EthernetInterface
- AggregateInterface
- VlanInterface
- LoopbackInterface
- TunnelInterface

Parameters

- **name** (*str*) – The name
- **ping** (*bool*) – Enable ping
- **telnet** (*bool*) – Enable telnet
- **ssh** (*bool*) – Enable ssh
- **http** (*bool*) – Enable http
- **http_ocsp** (*bool*) – Enable http-ocsp
- **https** (*bool*) – Enable https
- **snmp** (*bool*) – Enable snmp
- **response_pages** (*bool*) – Enable response pages
- **userid_service** (*bool*) – Enable userid service
- **userid_syslog_listener_ssl** (*bool*) – Enable userid syslog listener ssl
- **userid_syslog_listener_udp** (*bool*) – Enable userid syslog listener udp
- **permitted_ip** (*list*) – The list of permitted IP addresses

class panos.network.**Ospf**(**args*, ***kwargs*)
OSPF Process

Parameters

- **enable** (*bool*) – Enable OSPF (Default: True)
- **router_id** (*str*) – Router ID in IP format (eg. 1.1.1.1)
- **reject_default_route** (*bool*) – Reject default route
- **allow_redist_default_route** (*bool*) – Allow redistribution in default route
- **rfc1583** (*bool*) – rfc1583
- **spf_calculation_delay** (*int*) – SPF calculation delay
- **lsa_interval** (*int*) – LSA interval
- **graceful_restart_enable** (*bool*) – Enable OSPF graceful restart
- **gr_grace_period** (*int*) – Graceful restart period
- **gr_helper_enable** (*bool*) – Graceful restart helper enable
- **gr_strict_lsa_checking** (*bool*) – Graceful restart strict lsa checking
- **gr_max_neighbor_restart_time** (*int*) – Graceful restart neighbor restart time

class panos.network.**OspfArea**(**args*, ***kwargs*)
OSPF Area

Parameters

- **name** (*str*) – Area in IP format
- **type** (*str*) – Type of area, 'normal', 'stub', or 'nssa' (Default: normal)
- **accept_summary** (*bool*) – Accept summary route - stub and nssa only
- **default_route_advertise** (*str*) – 'disable' or 'advertise' (Default: disable) - stub and nssa only
- **default_route_advertise_metric** (*int*) – Default route metric - stub and nssa only
- **default_route_advertise_type** (*str*) – 'ext-1' or 'ext2' (Default: ext-2 - nssa only

class panos.network.**OspfAreaInterface**(**args*, ***kwargs*)
OSPF Area Interface

Parameters

- **name** (*str*) – Name of the interface (interface must exist)
- **enable** (*bool*) – OSPF enabled on this interface
- **passive** (*bool*) – Passive mode
- **link_type** (*str*) – Link type, 'broadcast', 'p2p', or 'p2mp' (Default: broadcast)
- **metric** (*int*) – Metric
- **priority** (*int*) – Priority id
- **hello_interval** (*int*) – Hello interval
- **dead_counts** (*int*) – Dead counts

- **retransmit_interval** (*int*) – Retransmit interval
- **transit_delay** (*int*) – Transit delay
- **gr_delay** (*int*) – Graceful restart delay
- **authentication** (*str*) – Reference to a *panos.network.OspfAuthProfile*

class panos.network.**OspfAuthProfile**(**args*, ***kwargs*)
OSPF Authentication Profile

Parameters

- **name** (*str*) – Name of Auth Profile
- **type** (*str*) – 'password' or 'md5'
- **password** (*str*) – The password if type is set to 'password'. If type is set to 'md5', add a *panos.network.OspfAuthProfileMd5*

class panos.network.**OspfAuthProfileMd5**(**args*, ***kwargs*)
OSPF Authentication Profile

Parameters

- **keyid** (*int*) – Identifier for key
- **key** (*str*) – The authentication key
- **preferred** (*bool*) – This key is preferred

class panos.network.**OspfExportRules**(**args*, ***kwargs*)
OSPF Export Rules

Parameters

- **name** (*str*) – IP subnet or *panos.network.RedistributionProfile*
- **new_path_type** (*str*) – New path type, 'ext-1' or 'ext-2' (Default: ext-2)
- **new_tag** (*str*) – New tag (int or IP format)
- **metric** (*int*) – Metric

class panos.network.**OspfNeighbor**(**args*, ***kwargs*)
OSPF Neighbor

Parameters

- **name** (*str*) – IP of neighbor
- **metric** (*int*) – Metric

class panos.network.**OspfNssaExternalRange**(**args*, ***kwargs*)
OSPF NSSA External Range

Parameters

- **name** (*str*) – IP network with prefix
- **mode** (*str*) – 'advertise' or 'suppress' (Default: advertise)

class panos.network.**OspfRange**(**args*, ***kwargs*)
OSPF Range

Parameters

- **name** (*str*) – IP network with prefix

- **mode** (*str*) – 'advertise' or 'suppress' (Default: advertise)

class panos.network.**PathMonitorDestination**(**args*, ***kwargs*)
PathMonitorDestination Static Route

Parameters

- **name** (*str*) – Name of Path Monitor Destination
- **enable** (*bool*) – Enable Path Monitor Destination
- **source** (*str*) – Source ip of interface
- **destination** (*str*) – Destination ip
- **interval** (*int*) – Ping Interval (sec) (Default: 3)
- **count** (*int*) – Ping count (Default: 5)

class panos.network.**PhysicalInterface**(**args*, ***kwargs*)
Absract base class for Ethernet and Aggregate Interfaces

Do not instantiate this object. Use a subclass.

set_zone(*zone_name*, *mode=None*, *refresh=False*, *update=False*, *running_config=False*, *return_type='object'*)
Set the zone for this interface

Creates a reference to this interface in the specified zone and removes references to this interface from all other zones. The zone will be created if it doesn't exist.

Parameters

- **zone_name** (*str*) – The name of the Zone or a *panos.network.Zone* instance
- **mode** (*str*) – The mode of the zone. See *panos.network.Zone* for possible values
- **refresh** (*bool*) – Refresh the relevant current state of the device before taking action (Default: False)
- **update** (*bool*) – Apply the changes to the device (Default: False)
- **running_config** – If refresh is True, refresh from the running configuration (Default: False)
- **return_type** (*str*) – Specify what this function returns, can be either 'object' (the default) or 'bool'. If this is 'object', then the return value is the Zone in question. If this is 'bool', then the return value is a boolean that tells you about if the live device needs updates (update=False) or was updated (update=True).

Returns The zone for this interface after the operation completes

Return type *Zone*

class panos.network.**RedistributionProfile**(**args*, ***kwargs*)
Redistribution Profile

Parameters

- **name** (*str*) – Name of profile
- **priority** (*int*) – Priority id
- **action** (*str*) – 'no-redist' or 'redist'
- **filter_type** (*tuple*) – Any of 'static', 'connect', 'rip', 'ospf', or 'bgp'
- **filter_interface** (*tuple*) – Filter interface

- **filter_destination** (*tuple*) – Filter destination
- **filter_nexthop** (*tuple*) – Filter nexthop
- **ospf_filter_pathtype** (*tuple*) – Any of 'intra-area', 'inter-area', 'ext-1', or 'ext-2
- **ospf_filter_area** (*tuple*) – OSPF filter on area
- **ospf_filter_tag** (*tuple*) – OSPF filter on tag
- **bgp_filter_community** (*tuple*) – BGP filter on community
- **bgp_filter_extended_community** (*tuple*) – BGP filter on extended community

class panos.network.**RedistributionProfileBase**(**args*, ***kwargs*)
Redistribution Profile

Parameters

- **name** (*str*) – Name of profile
- **priority** (*int*) – Priority id
- **action** (*str*) – 'no-redist' or 'redist'
- **filter_type** (*tuple*) – Any of 'static', 'connect', 'rip', 'ospf', or 'bgp'
- **filter_interface** (*tuple*) – Filter interface
- **filter_destination** (*tuple*) – Filter destination
- **filter_nexthop** (*tuple*) – Filter nexthop
- **ospf_filter_pathtype** (*tuple*) – Any of 'intra-area', 'inter-area', 'ext-1', or 'ext-2
- **ospf_filter_area** (*tuple*) – OSPF filter on area
- **ospf_filter_tag** (*tuple*) – OSPF filter on tag
- **bgp_filter_community** (*tuple*) – BGP filter on community
- **bgp_filter_extended_community** (*tuple*) – BGP filter on extended community

class panos.network.**RedistributionProfileIPv6**(**args*, ***kwargs*)
Redistribution Profile

Parameters

- **name** (*str*) – Name of profile
- **priority** (*int*) – Priority id
- **action** (*str*) – 'no-redist' or 'redist'
- **filter_type** (*tuple*) – Any of 'static', 'connect', 'rip', 'ospf', or 'bgp'
- **filter_interface** (*tuple*) – Filter interface
- **filter_destination** (*tuple*) – Filter destination
- **filter_nexthop** (*tuple*) – Filter nexthop
- **ospf_filter_pathtype** (*tuple*) – Any of 'intra-area', 'inter-area', 'ext-1', or 'ext-2
- **ospf_filter_area** (*tuple*) – OSPF filter on area
- **ospf_filter_tag** (*tuple*) – OSPF filter on tag
- **bgp_filter_community** (*tuple*) – BGP filter on community
- **bgp_filter_extended_community** (*tuple*) – BGP filter on extended community

class panos.network.**Rip**(**args*, ***kwargs*)
Add to a *panos.network.VirtualRouter* instance.

Parameters

- **enable** (*bool*) – Enable RIP
- **reject_default_route** (*bool*) – Reject default route
- **allow_redist_default_route** (*bool*) – Allow Redistribute Default Route
- **delete_intervals** (*int*) – Delete Intervals
- **expire_intervals** (*int*) – Expire Intervals
- **interval_seconds** (*int*) – Interval Seconds (sec)
- **update_intervals** (*int*) – Update Intervals
- **global_bfd_profile** (*str*) – Global BFD profile

class panos.network.**RipAuthProfile**(**args*, ***kwargs*)
Rip Authentication Profile

Parameters

- **name** (*str*) – Name of Auth Profile
- **auth_type** (*str*) – 'password' or 'md5'
- **password** (*str*) – The password if auth_type is set to 'password'. If auth_type is set to 'md5', add a *panos.network.RipAuthProfileMd5*

class panos.network.**RipAuthProfileMd5**(**args*, ***kwargs*)
Rip Authentication Profile

Parameters

- **keyid** (*int*) – Identifier for key
- **key** (*str*) – The authentication key
- **preferred** (*bool*) – This key is preferred

class panos.network.**RipExportRule**(**args*, ***kwargs*)
Rip Export Rules

Parameters

- **name** (*str*) – IP subnet or *panos.network.RedistributionProfile*
- **metric** (*int*) – Metric

class panos.network.**RipInterface**(**args*, ***kwargs*)
Rip Interface

Add to a *panos.network.Rip* instance.

Parameters

- **name** (*str*) – Interface name
- **enable** (*bool*) – Enable
- **advertise_default_route** – Advertise default route * advertise * disable
- **metric** (*int*) – Default route metric. Requires {advertise_default_route: "advertise"}
- **auth_profile** (*str*) – Auth profile name

- **mode** (*str*) – Mode of RipInterface * normal (default) * passive * send-only

class panos.network.**StaticMac**(**args*, ***kwargs*)
Static MAC address for a Vlan

Can be added to a *panos.network.Vlan* object

Parameters

- **mac** (*str*) – The MAC address
- **interface** (*str*) – Name of an interface

class panos.network.**StaticRoute**(**args*, ***kwargs*)
Static Route

Add to a *panos.network.VirtualRouter* instance.

Parameters

- **name** (*str*) – The name
- **destination** (*str*) – Destination network
- **nexthop_type** (*str*) – ip-address, discard, or next-vr
- **nexthop** (*str*) – Next hop IP address or Next VR Name
- **interface** (*str*) – Next hop interface
- **admin_dist** (*str*) – Administrative distance
- **metric** (*int*) – Metric (Default: 10)
- **enable_path_monitor** (*bool*) – Enable Path Monitor
- **failure_condition** (*str*) – Path Monitor failure condition set 'any' or 'all'
- **preemptive_hold_time** (*int*) – Path Monitor Preemptive Hold Time in minutes

class panos.network.**StaticRouteV6**(**args*, ***kwargs*)
IPV6 Static Route

Add to a *panos.network.VirtualRouter* instance.

Parameters

- **name** (*str*) – The name
- **destination** (*str*) – Destination network
- **nexthop_type** (*str*) – ip-address or discard
- **nexthop** (*str*) – Next hop IP address
- **interface** (*str*) – Next hop interface
- **admin_dist** (*str*) – Administrative distance
- **metric** (*int*) – Metric (Default: 10)
- **enable_path_monitor** (*bool*) – Enable Path Monitor
- **failure_condition** (*str*) – Path Monitor failure condition set 'any' or 'all'
- **preemptive_hold_time** (*int*) – Path Monitor Preemptive Hold Time in minutes

class panos.network.**Subinterface**(**args*, ***kwargs*)
Subinterface class

Do not instantiate this object. Use a subclass.

set_name()
Create a name appropriate for a subinterface if it isn't already

class panos.network.**TunnelInterface**(**args*, ***kwargs*)
Tunnel interface

Parameters

- **name** (*str*) – The name
- **ip** (*tuple*) – Interface IPv4 addresses
- **ipv6_enabled** (*bool*) – IPv6 Enabled (requires IPv6Address child object)
- **management_profile** (ManagementProfile) – Interface Management Profile
- **mtu** (*int*) – MTU for interface
- **netflow_profile** (*str*) – Netflow profile
- **comment** (*str*) – The interface's comment

class panos.network.**VirtualRouter**(**args*, ***kwargs*)
Virtual router

Parameters

- **name** (*str*) – Name of virtual router (Default: "default")
- **interface** (*list*) – List of interface names
- **ad_static** (*int*) – Administrative distance for this protocol
- **ad_static_ipv6** (*int*) – Administrative distance for this protocol
- **ad_ospf_int** (*int*) – Administrative distance for this protocol
- **ad_ospf_ext** (*int*) – Administrative distance for this protocol
- **ad_ospfv3_int** (*int*) – Administrative distance for this protocol
- **ad_ospfv3_ext** (*int*) – Administrative distance for this protocol
- **ad_ibgp** (*int*) – Administrative distance for this protocol
- **ad_ebgp** (*int*) – Administrative distance for this protocol
- **ad_rip** (*int*) – Administrative distance for this protocol

class panos.network.**VirtualWire**(**args*, ***kwargs*)
Virtual wires (vwire)

Parameters

- **name** (*str*) – The vwire name
- **tag** (*int*) – Tag for the interface, aka vlan id
- **interface1** (*str*) – The first interface to use
- **interface2** (*str*) – The second interface to use
- **multicast** (*bool*) – Enable multicast firewalling or not
- **pass_through** (*bool*) – Enable link state pass through or not

class panos.network.**Vlan**(**args*, ***kwargs*)

Parameters

- **name** (*str*) – The name
- **interface** (*list*) – List of interface names
- **virtual_interface** (VlanInterface) – The layer3 vlan interface for this vlan

class panos.network.**VlanInterface**(**args*, ***kwargs*)

Vlan interface

Parameters

- **name** (*str*) – Interface name
- **ip** (*tuple*) – Interface IPv4 addresses
- **ipv6_enabled** (*bool*) – IPv6 Enabled (requires IPv6Address child object)
- **management_profile** (ManagementProfile) – Interface Management Profile
- **mtu** (*int*) – MTU for interface
- **adjust_tcp_mss** (*bool*) – Adjust TCP MSS
- **netflow_profile** (*str*) – Netflow profile
- **comment** (*str*) – The interface's comment
- **ipv4_mss_adjust** (*int*) – TCP MSS adjustment for ipv4
- **ipv6_mss_adjust** (*int*) – TCP MSS adjustment for ipv6
- **enable_dhcp** (*bool*) – Enable DHCP on this interface
- **create_dhcp_default_route** (*bool*) – Create default route pointing to default gateway provided by server
- **dhcp_default_route_metric** (*int*) – Metric for the DHCP default route

set_vlan_interface(*vlan_name*, *refresh=False*, *update=False*, *running_config=False*, *return_type='object'*)

Sets the VLAN's VLAN interface to this VLAN interface

Creates a reference to this interface in the specified vlan and removes references to this interface from all other VLANs. The vlan will be created if it doesn't exist.

Parameters

- **vlan_name** (*str*) – The name of the vlan or a *panos.network.Vlan* instance
- **refresh** (*bool*) – Refresh the relevant current state of the device before taking action (Default: False)
- **update** (*bool*) – Apply the changes to the device (Default: False)
- **running_config** – If refresh is True, refresh from the running configuration (Default: False)
- **return_type** (*str*) – Specify what this function returns, can be either 'object' (the default) or 'bool'. If this is 'object', then the return value is the Vlan in question. If this is 'bool', then the return value is a boolean that tells you about if the live device needs updates (update=False) or was updated (update=True).

Returns The VLAN for this interface after the operation completes

Return type *Vlan*

class panos.network.**Zone**(**args*, ***kwargs*)
Security zone

Parameters

- **name** (*str*) – Name of the zone
- **mode** (*str*) – The mode of the security zone. Must match the mode of the interface. Possible values: tap, virtual-wire, layer2, layer3, external
- **interface** (*list*) – List of interface names or instantiated subclasses of *panos.network.Interface*.
- **zone_profile** (*str*) – Zone protection profile
- **log_setting** (*str*) – Log forwarding setting
- **enable_user_identification** (*bool*) – If user identification is enabled
- **include_acl** (*list/str*) – User identification ACL include list
- **exclude_acl** (*list/str*) – User identification ACL exclude list
- **enable_packet_buffer_protection** (*bool*) – (PAN-OS 8.0+) Enable packet buffer protection
- **enable_device_identification** (*bool*) – (PAN-OS 10.0+) Enable device identification
- **device_include_acl** (*list*) – (PAN-OS 10.0+) Device include ACLs list
- **device_exclude_acl** (*list*) – (PAN-OS 10.0+) Device exclude ACLs list

panos.network.**interface**(*name*, **args*, ***kwargs*)
Interface object factory

Creates an interface object of type determined by the name of the interface.

Parameters

- **name** (*str*) – Name of the interface to create (eg. ethernet1/1.5)
- **mode** (*str*) – Mode of the interface. Possible values: layer3, layer2, virtual-wire, tap, ha, aggregate-group. Default: None

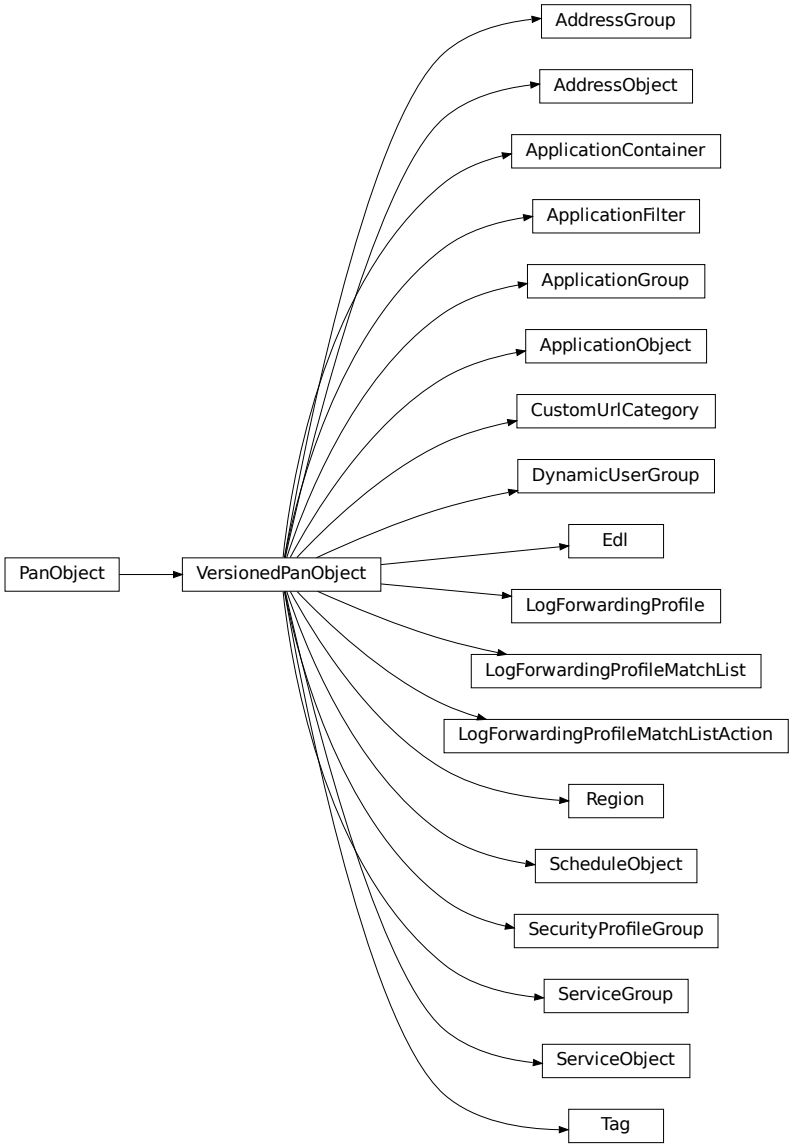
Keyword Arguments **tag** (*int*) – Tag for the interface, aka vlan id

Returns An instantiated subclass of *panos.network.Interface*

Return type *Interface*

5.9 Module: objects

5.9.1 Inheritance diagram



5.9.2 Class Reference

Objects module contains objects that exist in the 'Objects' tab in the firewall GUI

class panos.objects.**AddressGroup**(**args*, ***kwargs*)

Address Group

Parameters

- **name** (*str*) – Name of the address group
- **static_value** (*list*) – Values for a static address group
- **dynamic_value** (*str*) – Registered-ip tags for a dynamic address group
- **description** (*str*) – Description of this object
- **tag** (*list*) – Administrative tags (not to be confused with registered-ip tags)

class panos.objects.**AddressObject**(**args*, ***kwargs*)

Address Object

Parameters

- **name** (*str*) – Name of the object
- **value** (*str*) – IP address or other value of the object
- **type** (*str*) – Type of address: * ip-netmask (default) * ip-range * ip-wildcard (added in PAN-OS 9.0) * fqdn
- **description** (*str*) – Description of this object
- **tag** (*list*) – Administrative tags

class panos.objects.**ApplicationContainer**(**args*, ***kwargs*)

ApplicationContainer object

This is a special class that is used in the predefined module. It acts much like an ApplicationGroup object but exists only in the predefined context. It is more or less a way that Palo Alto groups predefined applications together.

Parameters

- **name** (*str*) – The name
- **applications** (*list*) – List of memeber applications

class panos.objects.**ApplicationFilter**(**args*, ***kwargs*)

ApplicationFilter Object

Parameters

- **name** (*str*) – Name of the object
- **category** (*list*) – Application category
- **subcategory** (*list*) – Application subcategory
- **technology** (*list*) – Application technology
- **risk** (*list*) – Application risk
- **evasive** (*bool*) –
- **excessive_bandwidth_use** (*bool*) –
- **prone_to_misuse** (*bool*) –
- **is_saas** (*bool*) –
- **transfers_files** (*bool*) –
- **tunnels_other_apps** (*bool*) –

- **used_by_malware** (*bool*) –
- **has_known_vulnerabilities** (*bool*) –
- **pervasive** (*bool*) –
- **tag** (*list*) – Administrative tags

class panos.objects.**ApplicationGroup**(**args*, ***kwargs*)
ApplicationGroup Object

Parameters

- **name** (*str*) – Name of the object
- **value** (*list*) – List of application values
- **tag** (*list*) – Administrative tags

class panos.objects.**ApplicationObject**(**args*, ***kwargs*)
Application Object

Parameters

- **name** (*str*) – Name of the object
- **category** (*str*) – Application category
- **subcategory** (*str*) – Application subcategory
- **technology** (*str*) – Application technology
- **risk** (*int*) – Risk (1-5) of the application
- **default_type** (*str*) – Default identification type of the application
- **default_port** (*list*) – Default ports
- **default_ip_protocol** (*str*) – Default IP protocol
- **default_icmp_type** (*int*) – Default ICMP type
- **default_icmp_code** (*int*) – Default ICMP code
- **parent_app** (*str*) – Parent Application for which this app falls under
- **timeout** (*int*) – Default timeout
- **tcp_timeout** (*int*) – TCP timeout
- **udp_timeout** (*int*) – UDP timeout
- **tcp_half_closed_timeout** (*int*) – TCP half closed timeout
- **tcp_time_wait_timeout** (*int*) – TCP wait time timeout
- **evasive_behavior** (*bool*) – Applicaiton is actively evasive
- **consume_big_bandwidth** (*bool*) – Application uses large bandwidth
- **used_by_malware** (*bool*) – Application is used by malware
- **able_to_transfer_file** (*bool*) – Application can do file transfers
- **has_known_vulnerability** (*bool*) – Application has known vulnerabilities
- **tunnel_other_application** (*bool*) –
- **tunnel_applications** (*list*) – List of tunneled applications
- **prone_to_misuse** (*bool*) –

- **pervasive_use** (*bool*) –
- **file_type_ident** (*bool*) –
- **virus_ident** (*bool*) –
- **data_ident** (*bool*) –
- **description** (*str*) – Description of this object
- **tag** (*list*) – Administrative tags

Please refer to https://applipedia.paloaltonetworks.com/ for more info on these params

class panos.objects.**CustomUrlCategory**(**args*, ***kwargs*)
Custom url category group

Parameters

- **name** (*str*) – The name
- **url_value** (*list*) – Values to include in custom URL category object
- **description** (*str*) – Description of this object
- **type** (*str*) – (PAN-OS 9.0+) The type

class panos.objects.**DynamicUserGroup**(**args*, ***kwargs*)
Dynamic user group.

Note: PAN-OS 9.1+

Parameters

- **name** – Name of the dynamic user group
- **description** (*str*) – Description of this object
- **filter** – Tag-based filter.
- **tag** (*list*) – Administrative tags

class panos.objects.**Edl**(**args*, ***kwargs*)
External Dynamic List.

Parameters

- **name** (*str*) – The name.
- **edl_type** (*str*) – The EDL type.
- **description** (*str*) – Description.
- **source** (*str*) – Source.
- **exceptions** (*list*) – (PAN-OS 8.0+) Exceptions.
- **certificate_profile** (*str*) – (PAN-OS 8.0+) Profile for authenticating client certificates.
- **username** (*str*) – (PAN-OS 8.0+) Username auth.
- **password** (*str*) – (PAN-OS 8.0+) Password auth.
- **expand_domain** (*bool*) – (PAN-OS 9.0+) Enable/disable expand domain (requires *edl_type=domain*).
- **repeat** (*str*) – Retrieval interval. Valid values are “five-minute”, “hourly”, “daily”, “weekly”, or “monthly”.

- **repeat_at** (*str*) – The time specification for the given repeat value.
- **repeat_day_of_week** (*str*) – For *repeat=daily*, the day of the week.
- **repeat_day_of_month** (*int*) – For *repeat=monthly*, the day of the month.

class panos.objects.**LogForwardingProfile**(**args*, ***kwargs*)
A log forwarding profile.

Note: This is valid for PAN-OS 8.0+

Parameters

- **name** (*str*) – The name
- **description** (*str*) – The description
- **enhanced_logging** (*bool*) – (PAN-OS 8.1+) Enabling enhanced application logging

class panos.objects.**LogForwardingProfileMatchList**(**args*, ***kwargs*)
A log forwarding profile match list entry.

Note: This is valid for PAN-OS 8.0+

Parameters

- **name** (*str*) – The name
- **description** (*str*) – Description
- **log_type** (*str*) – Log type. Valid values are traffic, threat, wildfire, url, data, gtp, tunnel, auth, or sctp (PAN-OS 8.1+).
- **filter** (*str*) – The filter.
- **send_to_panorama** (*bool*) – Send to panorama or not
- **snmp_profiles** (*str/list*) – List of SnmpServerProfiles.
- **email_profiles** (*str/list*) – List of EmailServerProfiles.
- **syslog_profiles** (*str/list*) – List of SyslogServerProfiles.
- **http_profiles** (*str/list*) – List of HttpServerProfiles.

class panos.objects.**LogForwardingProfileMatchListAction**(**args*, ***kwargs*)
Action for a log forwarding profile match list entry.

Note: This is valid for PAN-OS 8.0+

Parameters

- **name** (*str*) – The name
- **action_type** (*str*) – Action type. Valid values are tagging (default) or (PAN-OS 8.1+) integration.
- **action** (*str*) – The action. Valid values are add-tag, remove-tag, or (PAN-OS 8.1+) Azure-Security-Center-Integration.
- **target** (*str*) – The target. Valid values are source-address or destination-address.
- **registration** (*str*) – Registration. Valid values are localhost, panorama, or remote.
- **http_profile** (*str*) – The HTTP profile for registration of "remote".
- **tags** (*str/list*) – List of administrative tags.
- **timeout** (*int*) – (PAN-OS 9.0+) Timeout in minutes

class panos.objects.**Region**(**args*, ***kwargs*)
Region.

Parameters

- **name** (*str*) – Name of the region
- **address** (*list*) – List of IP networks
- **latitude** (*float*) – Latitude of the region
- **longitude** (*float*) – Longitude of the region

class panos.objects.**ScheduleObject**(**args*, ***kwargs*)
Schedule Object

"Date and Time Range" Example: 2019/11/01@00:15-2019/11/28@00:30 "Time Range" Example: 17:00-19:00

Parameters

- **name** (*str*) – Name of the object
- **disable_override** (*bool*) – "True" to set disable-override
- **type** (*str*) – Type of Schedule: "recurring" or "non-recurring"
- **non_recurring_date_time** (*list/str*) – "Date and Time Range" string for a non-recurring schedule
- **recurrence** (*str*) – "daily" or "weekly" recurrence
- **daily_time** (*list/str*) – "Time Range" for a daily recurring schedule
- **weekly_sunday_time** (*list/str*) – "Time Range" for a weekly recurring schedule (Sunday)
- **weekly_monday_time** (*list/str*) – "Time Range" for a weekly recurring schedule (Monday)
- **weekly_tuesday_time** (*list/str*) – "Time Range" for a weekly recurring schedule (Tuesday)
- **weekly_wednesday_time** (*list/str*) – "Time Range" for a weekly recurring schedule (Wednesday)
- **weekly_thursday_time** (*list/str*) – "Time Range" for a weekly recurring schedule (Thursday)
- **weekly_friday_time** (*list/str*) – "Time Range" for a weekly recurring schedule (Friday)
- **weekly_saturday_time** (*list/str*) – "Time Range" for a weekly recurring schedule (Saturday)

class panos.objects.**SecurityProfileGroup**(**args*, ***kwargs*)
Security Profile Group object

Parameters

- **name** (*str*) – The group name
- **virus** (*str*) – Antivirus profile
- **spyware** (*str*) – Anti-spyware profile
- **vulnerability** (*str*) – Vulnerability protection profile

- **url_filtering** (*str*) – URL filtering profile
- **file_blocking** (*str*) – File blocking profile
- **data_filtering** (*str*) – Data filtering profile
- **wildfire_analysis** (*str*) – WildFire analysis profile

class panos.objects.**ServiceGroup**(**args*, ***kwargs*)
ServiceGroup Object

Parameters

- **name** (*str*) – Name of the object
- **value** (*list*) – List of service values
- **tag** (*list*) – Administrative tags

class panos.objects.**ServiceObject**(**args*, ***kwargs*)
Service Object

Parameters

- **name** (*str*) – Name of the object
- **protocol** (*str*) – Protocol of the service, either tcp or udp
- **source_port** (*str*) – Source port of the protocol, if any
- **destination_port** (*str*) – Destination port of the service
- **description** (*str*) – Description of this object
- **tag** (*list*) – Administrative tags

class panos.objects.**Tag**(**args*, ***kwargs*)
Administrative tag

Parameters

- **name** (*str*) – Name of the tag
- **color** (*str*) – Color ID (eg. 'color1', 'color4', etc). You can use *color_code()* to generate the ID.
- **comments** (*str*) – Comments

static color_code(*color_name*)
Return the color code for a color

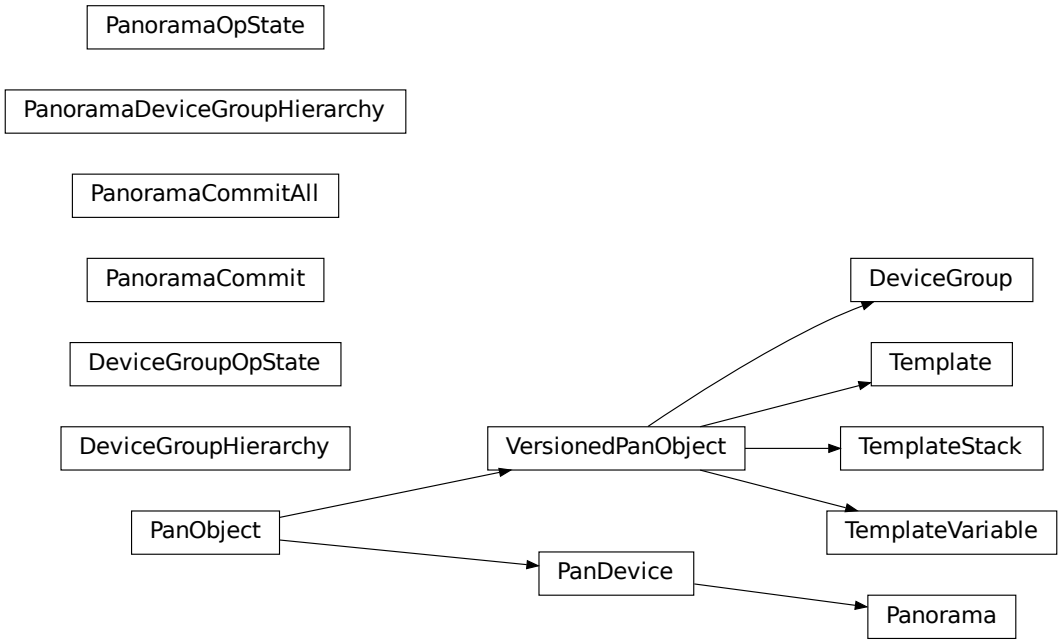
Parameters **color_name** (*str*) – One of the following colors:

- red
- green
- blue
- yellow
- copper
- orange
- purple
- gray
- light green

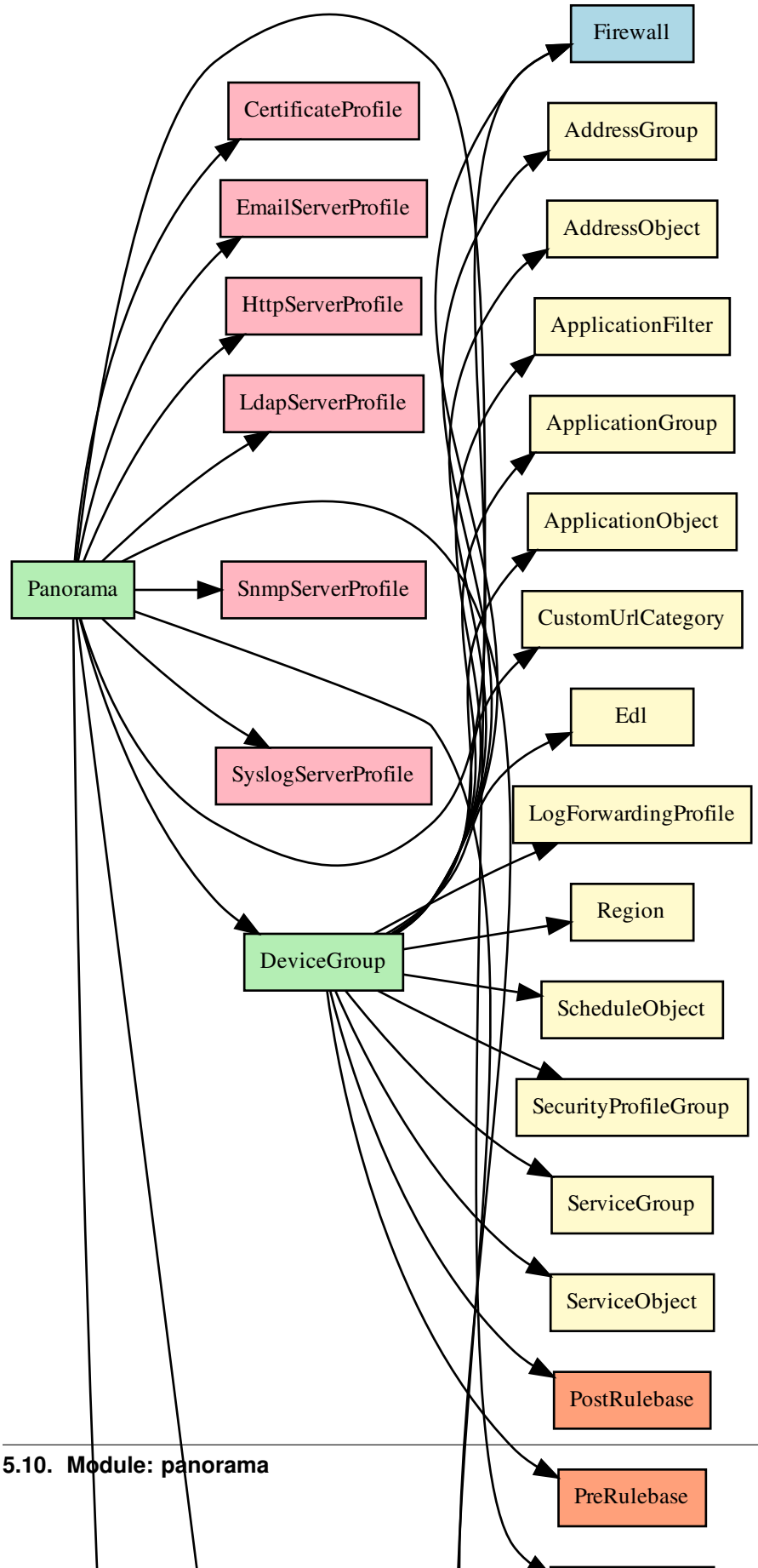
- cyan
- light gray
- blue gray
- lime
- black
- gold
- brown

5.10 Module: panorama

5.10.1 Inheritance diagram



5.10.2 Configuration tree diagram



5.10.3 Class Reference

Panorama and all Panorama related objects

class panos.panorama.**DeviceGroup**(**args*, ***kwargs*)
Panorama Device-group

This class and the *panos.panorama.Panorama* classes are the only objects that can have a *panos.firewall.Firewall* child object. In addition to a Firewall, a DeviceGroup can have the same children objects as a *panos.firewall.Firewall* or *panos.device.Vsys*.

See also *Configuration tree diagrams*

Parameters

- **name** (*str*) – Name of the device-group
- **tag** (*list*) – Tags as strings

devicegroup()
The nearest *panos.panorama.DeviceGroup* object.

This method is used to determine the device to apply this object to.

Returns The DeviceGroup object closest to this object in the configuration tree, or None if there is no DeviceGroup in the path to this node.

Return type *DeviceGroup*

vsys
Return the vsys for this object

Traverses the tree to determine the vsys from a *panos.firewall.Firewall* or *panos.device.Vsys* instance somewhere before this node in the tree.

Returns The vsys id (eg. vsys2)

Return type str

class panos.panorama.**DeviceGroupHierarchy**(*obj*)
Operational state handling for device group hierarchy.

Parameters **parent** (*str*) – This device group's parent.

refresh()
Refresh the `parent` from the state.

update()
Change this device group's hierarchical parent.

Modifies the live device

This operation results in a job being submitted to the backend, which this function will block until the move is completed. The return value of this function is what is returned from *panos.base.PanDevice.syncjob()*.

Returns Job result

Return type dict

class panos.panorama.**DeviceGroupOpState**(*obj*)
Operational state handling for device group classes.

class panos.panorama.**Panorama**(*hostname*, *api_username=None*, *api_password=None*, *api_key=None*, *port=443*, **args*, ***kwargs*)
Panorama device

This is the only object in the configuration tree that cannot have a parent. If it is in the configuration tree, then it is the root of the tree.

Parameters

- **hostname** – Hostname or IP of device for API connections
- **api_username** – Username of administrator to access API
- **api_password** – Password of administrator to access API
- **api_key** – The API Key for connecting to the device's API
- **port** – Port of device for API connections
- **timeout** – The timeout for asynchronous jobs
- **interval** – The interval to check asynchronous jobs

FIREWALL_CLASS
alias of *panos.firewall.Firewall*

commit_all(*sync=False*, *sync_all=True*, *exception=False*, *devicegroup=None*, *serials=()*, *cmd=None*, *description=None*, *include_template=None*)
Trigger a commit-all (commit to devices) on Panorama

NOTE: Use the new panorama.PanoramaCommitAll with commit() instead.

Parameters

- **sync** (*bool*) – Block until the Panorama commit is finished (Default: False)
- **sync_all** (*bool*) – Block until every Firewall commit is finished, requires sync=True (Default: False)
- **exception** (*bool*) – Create an exception on commit errors (Default: False)
- **devicegroup** (*str*) – Limit commit-all to a single device-group
- **serials** (*list*) – Limit commit-all to these serial numbers
- **cmd** (*str*) – Commit options in XML format
- **description** – Commit description
- **include_template** (*bool*) – Include template changes in this push

Returns Commit results

Return type dict

generate_vm_auth_key(*lifetime*)
Generates a VM auth key to be placed in a VM's init-cfg.txt.

Parameters **lifetime** (*int*) – The lifetime (in hours).

Raises PanDeviceError

Returns has "authkey" and "expires" keys.

Return type dict

get_vm_auth_keys()
Returns the current VM auth keys.

Raises PanDeviceError

Returns list of dicts. Each dict has "authkey" and "expires" keys.

Return type list

op(*cmd=None*, *vsys=None*, *xml=False*, *cmd_xml=True*, *extra_qs=None*, *retry_on_peer=False*)
Perform operational command on this Panorama

Parameters

- **cmd** (*str*) – The operational command to execute
- **vsys** (*str*) – Ignored for Panorama
- **xml** (*bool*) – Return value should be a string (Default: False)
- **cmd_xml** (*bool*) – True: cmd is not XML, False: cmd is XML (Default: True)
- **extra_qs** – Extra parameters for API call
- **retry_on_peer** (*bool*) – Try on active Firewall first, then try on passive Firewall

Returns The result of the operational command. May also return a string of XML if xml=True

Return type xml.etree.ElementTree

panorama()
The nearest *panos.panorama.Panorama* object.

This method is used to determine the device to apply this object to.

Returns

The Panorama object closest to this object in the configuration tree

Return type *Panorama*

Raises PanDeviceNotSet – There is no Panorama object in the tree.

refresh_devices(*devices=()*, *only_connected=False*, *expand_vsys=True*, *include_device_groups=True*, *add=False*, *running_config=False*)
Refresh device groups and devices using config and operational commands

Uses operational command in addition to configuration to gather as much information as possible about Panorama connected devices. The operational commands used are 'show devices all/connected' and 'show devicegroups'.

Information gathered about each device includes:

- management IP address (can be different from hostname)
- serial
- version
- high availability peer releationships
- panorama connection status
- device-group sync status

Parameters

- **devices** (*list*) – Limit refresh to these serial numbers
- **only_connected** (*bool*) – Ignore devices that are not 'connected' to Panorama (Default: False)
- **expand_vsys** (*bool*) – Instantiate a Firewall object for every Vsys (Default: True)
- **include_device_groups** (*bool*) – Instantiate *panos.panorama.DeviceGroup* objects with Firewall objects added to them.

- **add** (*bool*) – Add the new tree of instantiated DeviceGroup and Firewall objects to the Panorama config tree. Warning: This removes all current DeviceGroup and Firewall objects from the configuration tree, and all their children, so it is typically done before building a configuration tree. (Default: False)
- **running_config** (*bool*) – Refresh devices from the running configuration (Default: False)

Returns If 'include_device_groups' is True, returns a list containing new DeviceGroup instances which contain new Firewall instances. Any Firewall that is not in a device-group is in the list with the DeviceGroup instances. If 'include_device_groups' is False, returns a list containing new Firewall instances.

Return type list

class panos.panorama.**PanoramaCommit**(*description=None*, *admins=None*, *device_groups=None*, *templates=None*, *template_stacks=None*, *wildfire_appliances=None*, *wildfire_clusters=None*, *log_collectors=None*, *log_collector_groups=None*, *exclude_device_and_network=False*, *exclude_shared_objects=False*, *force=False*)

Normalization of a Panorama commit.

This performs a commit to Panorama. Changes must first be committed to Panorama before they can be pushed out elsewhere, such as to device groups or log collectors.

Instances of this class can be passed in to `Panorama.commit()` (inherited from *panos.base.PanDevice.commit()*) as the `cmd` parameter.

Parameters

- **description** (*str*) – The commit message.
- **admins** (*list*) – (PAN-OS 8.0+) List of admins whose changes are to be committed.
- **device_groups** (*list*) – List of device groups to save changes for.
- **templates** (*list*) – List of templates to save changes for.
- **template_stacks** (*list*) – List of template stacks to save changes for.
- **wildfire_appliances** (*list*) – List of Wildfire appliances to save changes for.
- **wildfire_clusters** (*list*) – List of Wildfire clusters to save changes for.
- **log_collectors** (*list*) – List of log collectors to save changes for.
- **log_collector_groups** (*list*) – List of log collector groups to save changes for.
- **exclude_device_and_network** (*bool*) – Set to True to exclude device and network changes.
- **exclude_shared_objects** (*bool*) – Set to True to exclude shared objects changes.
- **force** (*bool*) – Set to True to force a commit even if one is not needed.

element()

Returns an xml representation of the commit requested.

Returns xml.etree.ElementTree

class panos.panorama.**PanoramaCommitAll**(*style*, *name*, *description=None*, *include_template=None*, *force_template_values=None*, *devices=None*)

Normalization of a Panorama commit all.

This performs a commit-all in Panorama, pushing config out to the specified location.

Instances of this class can be passed in to `Panorama.commit()` (inherited from *panos.base.PanDevice.commit()*) as the `cmd` parameter.

Parameters

- **style** (*str*) – The type of commit-all to perform: * device group * template * template stack * log collector group * wildfire appliance * wildfire cluster
- **name** (*str*) – The name of the location to push the config to (e.g. - name of the device group, name of the template, etc).
- **description** (*str*) – The commit message.
- **include_template** (*bool*) – (For *device group* style commits) Set to True to include template changes.
- **force_template_values** (*bool*) – (For *device group*, *template*, or *template stack* style commits) Set to True to force template values.
- **devices** (*list*) – (For *device group*, *template*, or *template stack* style commits) Specific devices to commit to.

element()

Returns an xml representation of the commit all.

Returns xml.etree.ElementTree

class panos.panorama.**PanoramaDeviceGroupHierarchy**(*obj*)

Operational state handling for device group hierarchy.

fetch()

Returns a dict of device groups and their parents.

Keys in the dict are the device group's name, while the value is the name of that device group's parent. Top level device groups will have a parent of `None`.

Returns dict

class panos.panorama.**PanoramaOpState**(*obj*)

Panorama OP state handling.

class panos.panorama.**Template**(**args*, ***kwargs*)

A panorama template.

Parameters

- **name** – Template name
- **description** – Description
- **devices** (*str/list*) – The list of serial numbers in this template
- **default_vsys** – The default vsys in case of a single vsys firewall
- **multi_vsys** (*bool*) – (6.1 and lower) Multi virtual systems boolean
- **mode** – (6.1 and lower) Can be fips, cc, or normal (default: normal)
- **vpn_disable_mode** (*bool*) – (6.1 and lower) VPN disable mode

apply_similar()

Bulk apply all objects similar to this one.

Modifies the live device

This is similar to apply(), except instead of calling apply only on this object, it calls apply for all objects that share the same xpath as this object, recursively searching the entire object tree from the nearest firewall or panorama instance.

As an example, if you called apply_similar on an object representing ethernet1/5.42, all of the subinterfaces for ethernet1/5 would be included in the resulting XML document, regardless of which vsys those subinterfaces existed in.

Since apply does a replace of the config at the given xpath, please be careful when using this function that all objects, whether they be updated or not, exist in your pan-os-python object tree.

create_similar()
Bulk create all objects similar to this one.

Modifies the live device

This is similar to create(), except instead of calling create only on this object, it calls create for all objects that share the same xpath as this object, recursively searching the entire object tree from the nearest firewall or panorama instance.

As an example, if you called create_similar on an object representing ethernet1/5.42, all of the subinterfaces for ethernet1/5 would be included in the resulting XML document, regardless of which vsys those subinterfaces existed in.

delete_similar()
Bulk delete all objects similar to this one.

Modifies the live device

This is similar to delete(), except instead of calling delete only on this object, it calls delete for all objects that share the same xpath as this object, recursively searching the entire object tree from the nearest firewall or panorama instance.

As an example, if you called delete_similar on an object representing ethernet1/5.42, all of the subinterfaces in your pan-os-python object tree for ethernet1/5 would be removed.

class panos.panorama.**TemplateStack**(**args*, ***kwargs*)
Template stack.

NOTE: Template stacks were introduced in PAN-OS 7.0. Attempting to use this class on PAN-OS 6.1 or earlier will result in an error.

Parameters

- **name** – Stack name
- **description** – The description
- **templates** (*str/list*) – The list of templates in this stack
- **devices** (*str/list*) – The list of serial numbers in this template

apply_similar()
Bulk apply all objects similar to this one.

Modifies the live device

This is similar to apply(), except instead of calling apply only on this object, it calls apply for all objects that share the same xpath as this object, recursively searching the entire object tree from the nearest firewall or panorama instance.

As an example, if you called apply_similar on an object representing ethernet1/5.42, all of the subinterfaces for ethernet1/5 would be included in the resulting XML document, regardless of which vsys those subinterfaces existed in.

Since apply does a replace of the config at the given xpath, please be careful when using this function that all objects, whether they be updated or not, exist in your pan-os-python object tree.

create_similar()
Bulk create all objects similar to this one.

Modifies the live device

This is similar to create(), except instead of calling create only on this object, it calls create for all objects that share the same xpath as this object, recursively searching the entire object tree from the nearest firewall or panorama instance.

As an example, if you called create_similar on an object representing ethernet1/5.42, all of the subinterfaces for ethernet1/5 would be included in the resulting XML document, regardless of which vsys those subinterfaces existed in.

delete_similar()
Bulk delete all objects similar to this one.

Modifies the live device

This is similar to delete(), except instead of calling delete only on this object, it calls delete for all objects that share the same xpath as this object, recursively searching the entire object tree from the nearest firewall or panorama instance.

As an example, if you called delete_similar on an object representing ethernet1/5.42, all of the subinterfaces in your pan-os-python object tree for ethernet1/5 would be removed.

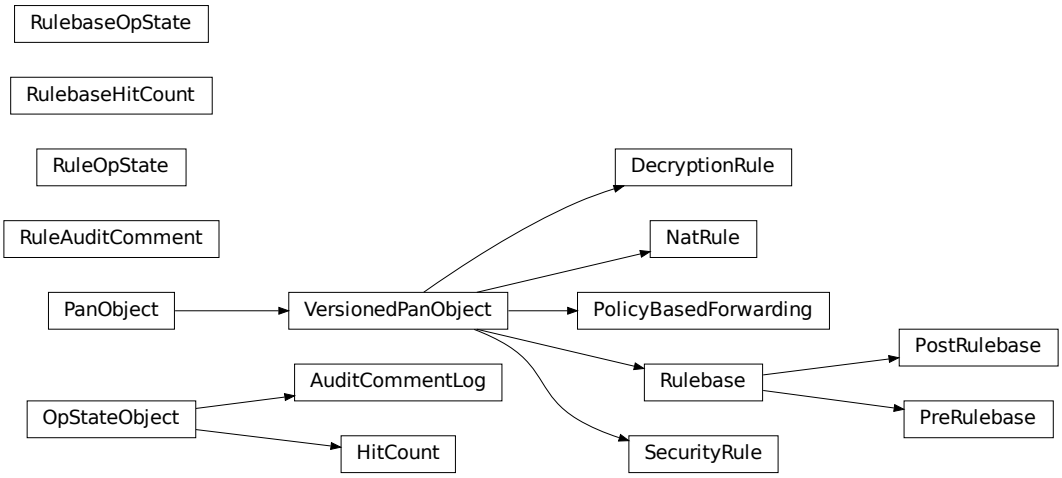
class panos.panorama.**TemplateVariable**(**args*, ***kwargs*)
Template or template stack variable.

Parameters

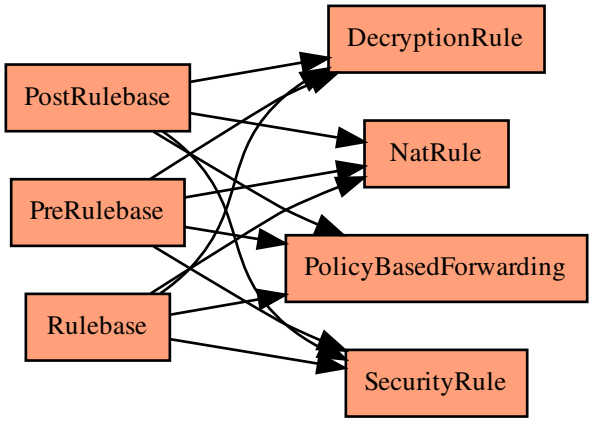
- **name** – The name.
- **value** – The variable value.
- **variable_type** – The variable type: * ip-netmask (default) * ip-range * fqdn * group-id * interface * device-priority (PAN-OS 9.0+) * device-id (PAN-OS 9.0+)

5.11 Module: policies

5.11.1 Inheritance diagram



5.11.2 Configuration tree diagram



5.11.3 Class Reference

Policies module contains policies and rules that exist in the 'Policies' tab in the firewall GUI

class panos.policies.**AuditCommentLog**(*elm*)
A single audit comment log entry.

class panos.policies.**DecryptionRule**(**args*, ***kwargs*)
Decryption rule.

PAN-OS 7.0+

Parameters

- **name** (*str*) – The name
- **description** (*str*) – The descripton
- **uuid** (*str*) – (PAN-OS 9.0+) The UUID for this rule.
- **source_zones** (*list*) – The source zones.
- **source_addresses** (*list*) – The source addresses.
- **negate_source** (*bool*) – Negate the source addresses.
- **source_users** (*list*) – The source users.
- **source_hip** (*list*) – (PAN-OS 10.0+) The source HIP info.
- **destination_zones** (*list*) – The destination zones.
- **destination_addresses** (*list*) – The destination addresses.
- **negate_destination** (*bool*) – Negate the destination addresses.
- **destination_hip** (*list*) – The destination HIP info.
- **tags** (*list*) – The administrative tags.
- **disabled** (*bool*) – If the rule is disabled or not.
- **services** (*list*) – Services.
- **url_categories** (*list*) – URL categories.
- **action** (*str*) – The action. Valid values are "no-decrypt" (default), "decrypt", or "decrypt-and-forward" (PAN-OS 8.1+).
- **decryption_type** (*str*) – The decryption type. Valid values are "ssl-forward-proxy", "ssh-proxy", or "ssl-inbound-inspection".
- **ssl_certificate** (*str*) – The SSL cert.
- **decryption_profile** (*str*) – The decryption profile.
- **forwarding_profile** (*str*) – (PAN-OS 8.1+) The forwarding profile.
- **group_tag** (*str*) – (PAN-OS 9.0+) The group tag.
- **log_successful_tls_handshakes** (*bool*) – (PAN-OS 10.0+) Log successful TLS handshakes.
- **log_failed_tls_handshakes** (*bool*) – (PAN-OS 10.0+) Log failed TLS handshakes.
- **log_setting** (*str*) – (PAN-OS 10.0+) Log setting.

class panos.policies.**HitCount**(*obj=None*, *name=None*, *elm=None*)

Hit count operational data.

class panos.policies.**NatRule**(**args*, ***kwargs*)

NAT Rule

Both the naming convention and the order of the parameters tries to closly match what is presented in the GUI.

There are groupings of parameters that give hints to the sections that they contribute towards:

- source_translation_<etc>

- source_translation_fallback_<etc>
- source_translation_static_<etc>
- destination_translation_<etc>

Parameters

- **name** (*str*) – Name of the rule
- **description** (*str*) – The description
- **nat_type** (*str*) – Type of NAT
- **fromzone** (*list*) – From zones
- **tozone** (*list*) – To zones
- **to_interface** (*str*) – Egress interface from route lookup
- **service** (*str*) – The service
- **source** (*list*) – Source addresses
- **destination** (*list*) – Destination addresses
- **source_translation_type** (*str*) – Type of source address translation
- **source_translation_address_type** (*str*) – Address type for Dynamic IP And Port or Dynamic IP source translation types
- **source_translation_interface** (*str*) – Interface of the source address translation for Dynamic IP and Port source translation types
- **source_translation_ip_address** (*str*) – IP address of the source address translation for Dynamic IP and Port source translation types
- **source_translation_translated_addresses** (*list*) – Translated addresses of the source address translation for Dynamic IP And Port or Dynamic IP source translation types
- **source_translation_fallback_type** (*str*) – Type of fallback for Dynamic IP source translation types
- **source_translation_fallback_translated_addresses** (*list*) – Addresses for translated address types of fallback source translation
- **source_translation_fallback_interface** (*str*) – The interface for the fallback source translation
- **source_translation_fallback_ip_type** (*str*) – The type of the IP address for the fallback source translation IP address
- **source_translation_fallback_ip_address** (*str*) – The IP address of the fallback source translation
- **source_translation_static_translated_address** (*str*) – The IP address for the static source translation
- **source_translation_static_bi_directional** (*bool*) – Allow reverse translation from translated address to original address
- **destination_translated_address** (*str*) – Translated destination IP address
- **destination_translated_port** (*int*) – Translated destination port number
- **ha_binding** (*str*) – Device binding configuration in HA Active-Active mode

- **disabled** (*bool*) – Disable this rule
- **negate_target** (*bool*) – Target all but the listed target firewalls (applies to panorama/device groups only)
- **target** (*list*) – Apply this policy to the listed firewalls only (applies to panorama/device groups only)
- **tag** (*list*) – Administrative tags
- **destination_dynamic_translated_address** (*str*) – (PAN-OS 8.1+) Dynamic destination translated address.
- **destination_dynamic_translated_port** (*int*) – (PAN-OS 8.1+) Dynamic destination translated port.
- **destination_dynamic_translated_distribution** (*str*) – (PAN-OS 8.1+) Dynamic destination translated distribution.
- **uuid** (*str*) – (PAN-OS 9.0+) The UUID for this rule.
- **group_tag** (*str*) – (PAN-OS 9.0+) The group tag.

class panos.policies.**OpStateObject**

A container object for opstate data.

class panos.policies.**PolicyBasedForwarding**(**args*, ***kwargs*)

PBF rule.

Parameters

- **name** (*str*) – The name
- **description** (*str*) – The descripton
- **tags** (*str/list*) – List of tags
- **from_type** (*str*) – Source from type. Valid values are 'zone' (default) or 'interface'.
- **from_value** (*str/list*) – The source values for the given type.
- **source_addresses** (*str/list*) – List of source IP addresses.
- **source_users** (*str/list*) – List of source users.
- **negate_source** (*bool*) – Set to negate the source.
- **destination_addresses** (*str/list*) – List of destination addresses.
- **negate_destination** (*bool*) – Set to negate the destination.
- **applications** (*str/list*) – List of applications.
- **services** (*str/list*) – List of services.
- **schedule** (*str*) – The schedule.
- **disabled** (*bool*) – Set to disable this rule.
- **action** (*str*) – The action to take. Valid values are 'forward' (default), 'forward-to-vsys', 'discard', or 'no-pbf'.
- **forward_vsys** (*str*) – The vsys to forward to if action is set to forward to a vsys.
- **forward_egress_interface** (*str*) – The egress interface.
- **forward_next_hop_type** (*str*) – The next hop type. Valid values are 'ip-address', 'fqdn', or None (default).

- **forward_next_hop_value** (*str*) – The next hop value if the forward next hop type is not None.
- **forward_monitor_profile** (*str*) – The monitor profile to use.
- **forward_monitor_ip_address** (*str*) – The monitor IP address.
- **forward_monitor_disable_if_unreachable** (*bool*) – Set to disable this rule if nexthop / monitor IP is unreachable.
- **enable_enforce_symmetric_return** (*bool*) – Set to enforce symmetric return.
- **symmetric_return_addresses** (*str/list*) – List of symmetric return addresses.
- **active_active_device_binding** (*str*) – Active/Active device binding.
- **target** (*list*) – Apply this policy to the listed firewalls only (applies to panorama/device groups only)
- **negate_target** (*bool*) – Target all but the listed target firewalls (applies to panorama/device groups only)
- **uuid** (*str*) – (PAN-OS 9.0+) The UUID for this rule.
- **group_tag** (*str*) – (PAN-OS 9.0+) The group tag.

class panos.policies.**PostRulebase**(**args*, ***kwargs*)
Post-rulebase for a Panorama

Panorama only. For Firewall, use *panos.policies.Rulebase*.

class panos.policies.**PreRulebase**(**args*, ***kwargs*)
Pre-rulebase for a Panorama

Panorama only. For Firewall, use *panos.policies.Rulebase*.

class panos.policies.**RuleAuditComment**(*obj*)
Operational state handling for a rule's audit comments.

Note: Audit comments are present in PAN-OS 9.0+.

current()
Returns the current audit comment.

Returns string

history(*count=100*, *direction='backward'*, *skip=None*)
Returns a chunk of historical audit comment logs.

Parameters

- **count** (*int*) – Number of audit comments to return, maximum 5000.
- **direction** (*str*) – Specify whether logs are shown oldest first (`forward`) or newest first (`backward`).
- **skip** (*int*) – Specify the number of logs to skip when doing log retrieval. This is useful when retrieving logs in batches where you can skip the previously retrieved logs.

Returns list of *panos.policies.AuditCommentLog*

update(*comment*)
Sets an audit comment for the given rule.

Parameters **comment** (*str*) – The audit comment.

class panos.policies.**RuleOpState**(*obj*)
Operational state handling for a rule in the rulebase.

class panos.policies.**Rulebase**(**args*, ***kwargs*)
Rulebase for a Firewall

Firewall only. For Panorama, use *panos.policies.PreRulebase* or *panos.policies.PostRulebase*.

class panos.policies.**RulebaseHitCount**(*obj*)
Operational state handling for rulebase hit counts.

refresh(*style*, *rules=None*, *all_rules=False*)
Retrieves hit count information for the specified rules.

PAN-OS 8.1+

Parameters

- **style** (*str*) – The rule style to use. The style can be "application-override", "authentication", "decryption", "dos", "nat", "pbf", "qos", "sdwan", "security", or "tunnel-inspect".
- **rules** (*list*) – A list of rules. This can be a mix of *panos.policies* instances or basic strings. If no rules are given, then the hit count for all rules is retrieved.
- **all_rules** (*bool*) – If this is False, only retrieve hit count information for the rules attached to the rulebase of the specified style. If this is True, then get all rules. Either way, any rule whose hit count is retrieved and is in the object hierarchy has the hit count data saved to its *opstate*.

Returns A dict where the key is the rule name and the value is the hit count information.

Return type dict

class panos.policies.**RulebaseOpState**(*obj*)
Operational state handling for rulebase classes.

class panos.policies.**SecurityRule**(**args*, ***kwargs*)
Security Rule

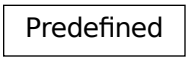
Parameters

- **name** (*str*) – Name of the rule
- **fromzone** (*list*) – From zones
- **tozone** (*list*) – To zones
- **source** (*list*) – Source addresses
- **source_user** (*list*) – Source users and groups
- **hip_profiles** (*list*) – GlobalProtect host integrity profiles
- **destination** (*list*) – Destination addresses
- **application** (*list*) – Applications
- **service** (*list*) – Destination services (ports) (Default: application-default)
- **category** (*list*) – Destination URL Categories
- **action** (*str*) – Action to take (deny, allow, drop, reset-client, reset-server, reset-both) Note: Not all options are available on all PAN-OS versions.
- **log_setting** (*str*) – Log forwarding profile

- **log_start** (*bool*) – Log at session start
- **log_end** (*bool*) – Log at session end
- **description** (*str*) – Description of this rule
- **type** (*str*) – 'universal', 'intrazone', or 'intrazone' (Default: universal)
- **tag** (*list*) – Administrative tags
- **negate_source** (*bool*) – Match on the reverse of the 'source' attribute
- **negate_destination** (*bool*) – Match on the reverse of the 'destination' attribute
- **disabled** (*bool*) – Disable this rule
- **schedule** (*str*) – Schedule Profile
- **icmp_unreachable** (*bool*) – Send ICMP Unreachable
- **disable_server_response_inspection** (*bool*) – Disable server response inspection
- **group** (*str*) – Security Profile Group
- **negate_target** (*bool*) – Target all but the listed target firewalls (applies to panorama/device groups only)
- **target** (*list*) – Apply this policy to the listed firewalls only (applies to panorama/device groups only)
- **virus** (*str*) – Antivirus Security Profile
- **spyware** (*str*) – Anti-Spyware Security Profile
- **vulnerability** (*str*) – Vulnerability Protection Security Profile
- **url_filtering** (*str*) – URL Filtering Security Profile
- **file_blocking** (*str*) – File Blocking Security Profile
- **wildfire_analysis** (*str*) – Wildfire Analysis Security Profile
- **data_filtering** (*str*) – Data Filtering Security Profile
- **uuid** (*str*) – (PAN-OS 9.0+) The UUID for this rule.
- **source_devices** (*list*) – (PAN-OS 10.0+) Host devices subject to the policy.
- **destination_devices** (*list*) – (PAN-OS 10.0+) Destination devices subject to the policy.
- **group_tag** (*str*) – (PAN-OS 9.0+) The group tag.

5.12 Module: predefined

5.12.1 Inheritance diagram



5.12.2 Class Reference

Retrieving and parsing predefined objects from the firewall

class panos.predefined.**Predefined**(*device=None*, **args*, ***kwargs*)
Predefined Objects Subsystem of Firewall

A member of a base.PanDevice object that has special methods for interacting with the predefined objects of the firewall

This class is typically not instantiated by anything but the base.PanDevice class itself. There is an instance of this Predefined class inside every instantiated base.PanDevice class.

Parameters **device** (base.PanDevice) – The firewall or Panorama this Predefined subsystem leverages

application(*name*, *refresh_if_none=True*, *include_containers=True*)
Get a Predefined Application

Return the instance of the application from the given name.

Parameters

- **name** (*str*) – Name of the application
- **refresh_if_none** (*bool*) – Refresh the application if it is not found
- **include_containers** (*bool*) – also search application containers if no match found

Returns Either an ApplicationObject, ApplicationContainerObject, or None

applications(*names*, *refresh_if_none=True*, *include_containers=True*)
Get a list of Predefined Applications

Return a list of the instances of the applications from the given names.

Parameters

- **names** (*list*) – Names of the applications
- **refresh_if_none** (*bool*) – Refresh the application(s) if it is not found
- **include_containers** (*bool*) – also search application containers if no match found

Returns A list of all found ApplicationObjects or ApplicationContainerObjects

object(*name*, *classtype*, *refresh_if_none=True*)
Get object by classtype

For example, if you pass in panos.objects.ApplicationObject as the classtype, an application will be returned

Parameters

- **name** (*str*) – Name of the object
- **classtype** – The class of the object (eg. panos.objects.ApplicationObject
- **refresh_if_none** (*bool*) – Refresh the object if it is not found

objects(*names*, *classtype*, *refresh_if_none=True*)
Get a list of objects by classtype

For example, if you pass in panos.objects.ApplicationObject as the classtype, a list of application will be returned

Parameters

- **names** (*list*) – List of names of the objects
- **classtype** – The class of the object (eg. panos.objects.ApplicationObject
- **refresh_if_none** (*bool*) – Refresh the object if it is not found

refresh_application(*name*)
Refresh a Single Predefined Application

This method refreshes single predefined application or application container (predefined only object).

Parameters **name** (*str*) – Name of the application to refresh

refresh_service(*name*)
Refresh a Single Predefined Service

This method refreshes single predefined service (predefined only object).

Parameters **name** (*str*) – Name of the service to refresh

refresh_tag(*name*)
Refresh a Single Predefined Tag

This method refreshes single predefined tag (predefined only object).

Parameters **name** (*str*) – Name of the tag to refresh

refreshall()
Refresh all Predefined Objects

This method refreshes all predefined objects. This includes applications, application containers, services, and tags.

CAUTION: This method requires a lot of overhead on the device api to respond. Response time will vary by platform, but know that it will generally take longer than a normal api request.

refreshall_applications()
Refresh all Predefined Applications

This method refreshes all predefined applications and application containers.

CAUTION: This method requires a lot of overhead on the device api to respond. Response time will vary by platform, but know that it will generally take longer than a normal api request.

refreshall_services()
Refresh all Predefined Services

This method refreshes all predefined services.

refreshall_tags()
Refresh all Predefined Tags

This method refreshes all predefined tag objects

service(*name*, *refresh_if_none=True*)
Get a Predefined Service

Return the instance of the service from the given name.

Parameters

- **name** (*str*) – Name of the service
- **refresh_if_none** (*bool*) – Refresh the service if it is not found

Returns Either a ServiceObject or None

services(*names*, *refresh_if_none=True*)
Get a list of Predefined Services

Return a list of the instances of the services from the given names.

Parameters

- **names** (*list*) – Names of the services
- **refresh_if_none** (*bool*) – Refresh the service(s) if it is not found

Returns A list of all found ServiceObjects

tag(*name*, *refresh_if_none=True*)
Get a Predefined Tag

Return the instance of the tag from the given name.

Parameters

- **name** (*str*) – Name of the tag
- **refresh_if_none** (*bool*) – Refresh the tag if it is not found

Returns Either a Tag or None

tags(*names*, *refresh_if_none=True*)
Get a list of Predefined Tags

Return a list of the instances of the tags from the given names.

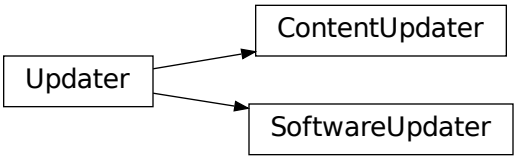
Parameters

- **names** (*list*) – Names of the tags
- **refresh_if_none** (*bool*) – Refresh the tag(s) if it is not found

Returns A list of all found Tags

5.13 Module: updater

5.13.1 Inheritance diagram



5.13.2 Class Reference

Device updater handles software versions and updates for devices

class panos.updater.**ContentUpdater**(*pandevice*)

check()
Trigger PAN-OS to get versions, then synchronize this object instance.

First, PAN-OS will reach out to the upgrade servers to get the list of all version that can be upgraded to. Then synchronizes this current updater state with the live device.

downgrade(*sync=False*)
Return to the previous content version.

Parameters **sync** (*bool, optional*) – Run jobs synchronously and return the result. Defaults to False.

Returns If sync, returns result of install job

download(*sync_to_peer=None*, *sync=False*)
Download the latest content version.

Parameters

- **sync_to_peer** (*bool, optional*) – Send a copy to HA peer. Defaults to None.
- **sync** (*bool, optional*) – Run jobs synchronously and return the result. Defaults to False.

Raises err.PanDeviceError – on unsuccessful download

Returns If sync, returns result of download job

download_install(*version='latest'*, *sync_to_peer=False*, *skip_commit=False*, *sync=False*)
Download and install the requested content version.

Like a combinations of the check(), download(), and install() methods.

Parameters

- **version** (*string*) – Content version (eg. "8357-6464"). Defaults to "latest".

- **sync_to_peer** (*bool, optional*) – Send a copy to HA peer. Defaults to False.
- **skip_commit** (*bool, optional*) – Do not perform a commit after install. Defaults to False.
- **sync** (*bool, optional*) – Run jobs synchronously and return the result. Defaults to False.

info()
Fetch version list from live device.

Synchronizes this current updater state with the live device.

install(*version='latest'*, *sync_to_peer=True*, *skip_commit=False*, *sync=False*)
Install the requested content version.

Parameters

- **version** (*string*) – Content version (eg. "8357-6464"). Defaults to "latest".
- **sync_to_peer** (*bool, optional*) – Send a copy to HA peer. Defaults to True.
- **skip_commit** (*bool, optional*) – Do not perform a commit after install. Defaults to False.
- **sync** (*bool, optional*) – Run jobs synchronously and return the result. Defaults to False.

Raises err.PanDeviceError – on unsuccessful install

Returns If sync, returns result of install job

class panos.updater.**SoftwareUpdater**(*pandevice*)

check()
Trigger PAN-OS to get versions, then synchronize this object instance.

First, PAN-OS will reach out to the upgrade servers to get the list of all version that can be upgraded to. Then synchronizes this current updater state with the live device.

download(*version*, *sync_to_peer=True*, *sync=False*)
PAN-OS downloads the requested version.

Parameters

- **version** (*string*) – PAN-OS version (eg. "10.0.2")
- **sync_to_peer** (*bool, optional*) – Send a copy to HA peer. Defaults to True.
- **sync** (*bool, optional*) – Run job synchronously and return the result. Defaults to False.

Raises err.PanDeviceError – on unsuccessful download

Returns If sync, returns result of PAN-OS download job

download_install(*version*, *load_config=None*, *sync=False*)
Download and install the requested PAN-OS version.

Like a combinations of the check(), download(), and install() methods, but with some additional checks. For example, it will not act if the requested version is already running, and it will skip to the install if it is already downloaded.

Does not perform the required reboot after the install.

Parameters

- **version** (*string*) – PAN-OS version (eg. "10.0.2")
- **load_config** (*string, optional*) – Configuration to use for booting new software. Defaults to None.
- **sync** (*bool, optional*) – Run jobs synchronously and return the result. Defaults to False.

Raises err.PanDeviceError – problem found in pre-download checks

Returns If sync, returns result of PAN-OS install job

download_install_reboot(*version*, *load_config=None*, *sync=False*)
Download and install the requested PAN-OS version, then reboot.

Like a combinations of the check(), download(), and install() methods with a reboot at the end. It has additional checks. For example, it will not act if the requested version is already running, and it will skip to the install if it is already downloaded.

Parameters

- **version** (*string*) – PAN-OS version (eg. "10.0.2")
- **load_config** (*string, optional*) – Configuration to use for booting new software. Defaults to None.
- **sync** (*bool, optional*) – Run jobs synchronously and return the result. Defaults to False.

Raises err.PanDeviceError – problem found in pre-download checks or after reboot

info()
Fetch version list from live device.

Synchronizes this current updater state with the live device.

install(*version*, *load_config=None*, *sync=False*)
Install the requested PAN-OS version.

Does not download the software or perform the reboot required after installation.

Parameters

- **version** (*string*) – PAN-OS version (eg. "10.0.2")
- **load_config** (*string, optional*) – Configuration to use for booting new software. Defaults to None.
- **sync** (*bool, optional*) – Run job synchronously and return the result. Defaults to False.

Raises err.PanDeviceError – on unsuccessful install

Returns If sync, returns result of PAN-OS install job

upgrade_to_version(*target_version*, *dryrun=False*)
Upgrade to the target version, completing all intermediate upgrades.

For example, if firewall is running version 9.0.5 and target version is 10.0.2, then this method will proceed through the following steps:

- Upgrade to 9.1.0 and reboot
- Upgrade to 10.0.0 and reboot
- Upgrade to 10.0.2 and reboot

Does not account for HA pairs.

Example

This shows how to upgrade a firewall to version 10.0.2. This will work regardless of which version the firewall is currently running:

```
from panos.firewall import Firewall

fw = Firewall("10.0.0.5", "admin", "password")
fw.software.upgrade_to_version("10.0.2")
```

Parameters

- **target_version** (*string*) – PAN-OS version (eg. "10.0.2") or "latest"
- **dryrun** (*bool, optional*) – Log what steps would be taken, but don't make any changes to the live device. Defaults to False.

Raises err.PanDeviceError – any problem during the upgrade process

class panos.updater.**Updater**(*pandevice*)
This class is instantiated by the PanDevice class as a software update subsystem

5.14 Module: userid

5.14.1 Inheritance diagram

5.14.2 Class Reference

User-ID and Dynamic Address Group updates using the User-ID API

class panos.userid.**UserId**(*device*, *prefix=''*, *ignore_dup_errors=True*)
User-ID Subsystem of Firewall

A member of a firewall.Firewall object that has special methods for interacting with the User-ID API. This includes login/logout of a user, user/group mappings, and dynamic address group tags.

This class is typically not instantiated by anything but the base.PanDevice class itself. There is an instance of this UserId class inside every instantiated base.PanDevice class.

Support: UserId API is supported on Panorama starting with Panorama 8.0 UserId API is supported on all firewall PAN-OS versions but with varying features as noted in the documentation for each method.

Parameters

- **device** (base.PanDevice) – The firewall or Panorama this user-id subsystem leverages
- **prefix** (*str*) – Prefix to use in all IP tag operations for Dynamic Address Groups
- **ignore_dup_errors** (*bool*) – Devices produce errors when a tag is registered that already exists. Set to true to ignore these errors. (Default: True)

audit_registered_ip(*ip_tags_pairs*, *timeout=None*)

Synchronize the current registered-ip tag list to this exact set of ip-tags

Sets the registered-ip tag list on the device. Regardless of the current state of the registered-ip tag list when this method is called, at the end of the method the list will contain only the ip-tags passed in the argument. The current state of the list is retrieved to reduce the number of operations needed. If the list is currently in the requested state, no API call is made after retrieving the list.

Support: PAN-OS 6.0 and higher

> **Warning:** This will clear any batch without it being sent, and can't be used as part of a batch.

Parameters

- **ip_tags_pairs** (*dict*) – dictionary where keys are ip addresses and values or tuples of tags
- **timeout** (*string*) – The optional timeout value in seconds.

audit_registered_ip_for_tag(*tag*, *ip_addresses*, *timeout=None*)

Synchronize the current registered-ip tag to tag only the specificied IP addresses.

Sets the registered-ip list for a single tag on the device. Regardless of the current state of the registered-ip tag list when this method is called, at the end of the method the list for the specified tag will contain only the ip addresses passed in the argument. The current state of the list is retrieved to reduce the number of operations needed. If the list for this tag is currently in the requested state, no API call is made after retrieving the list.

Support: PAN-OS 6.0 and higher

Warning: This will clear any batch without it being sent, and can't be used as part of a batch.

Parameters

- **tag** (*string*) – Tag to audit
- **ip_addresses** (*list*) – List of IP addresses that should have the tag
- **timeout** (*string*) – The optional timeout value in seconds.

batch_end()

End a batched API call and send it to the firewall

This method usually follows a batch_start() and several other operations.

The API call will not be sent to the firewall until batch_end() is called. This allows multiple operations to be added to a single API call.

batch_start()
Start creating an API call

The API call will not be sent to the firewall until batch_end() is called. This allows multiple operations to be added to a single API call.

clear_registered_ip(*ip=None*, *tags=None*, *prefix=None*)
Unregister registered/tagged addresses

Removes registered addresses used by dynamic address groups. When called without arguments, removes all registered addresses

Note: Passing a single ip and/or single tag to this method results in a response from the firewall that contains only the relevant entries. ie. the filtering is done on the firewall before it responds. Passing a list of multiple ip addresses or tags will result in retreival of the entire tag database from the firewall which is then filtered and returned with only the relevant entries. Therefor, using a single ip or tag is more efficient.

Support: PAN-OS 6.0 and higher

> **Warning:** This will clear any batch without it being sent, and can't be used as part of a batch.

Parameters

- **ip** (`list` or `str`) – IP address(es) to remove tags for
- **tags** (`list` or `str`) – Tag(s) to remove
- **prefix** (*str*) – Override class tag prefix

get_group_members(*group*)
Returns a list of users in the given group.

Parameters **group** – The name of the group.

Returns list

get_groups(*style=None*)
Get a list of groups.

Parameters **style** – The type of groups to retrieve. If unspecified, returns a list of all groups. Can be "custom-group", "dynamic", or "xmlapi".

Returns list

get_registered_ip(*ip=None*, *tags=None*, *prefix=None*)
Return registered/tagged addresses

When called without arguments, retrieves all registered addresses.

Note: Passing a single ip and/or single tag to this method results in a response from the firewall that contains only the relevant entries. ie. the filtering is done on the firewall before it responds. Passing a list of multiple ip addresses or tags will result in retreival of the entire tag database from the firewall which is then filtered and returned with only the relevant entries. Therefor, using a single ip or tag is more efficient.

Support: PAN-OS 6.0 and higher

Parameters

- **ip** (`list` or `str`) – IP address(es) to get tags for
- **tags** (`list` or `str`) – Tag(s) to get
- **prefix** (*str*) – Override class tag prefix

Returns ip addresses as keys with tags as values

Return type dict

Raises *PanDeviceError if running PAN-OS < 8.0 and a logfile is returned* – instead of IP/tag mapings.

get_user_tags(*user=None*, *prefix=None*)
Get the dynamic user tags.

Note: PAN-OS 9.1+

Parameters

- **user** – Get only this user's tags, not all users and all tags.
- **prefix** – Override class tag prefix.

Returns Dict where the user is the key and the value is a list of tags.

Return type dict

login(*user*, *ip*, *timeout=None*)
Login a single user

Maps a user to an IP address

This method can be batched with batch_start() and batch_end().

Parameters

- **user** (*str*) – a username
- **ip** (*str*) – an ip address
- **timeout** (*int*) – timeout in minutes to remove this mapping

logins(*users*)
Login multiple users in the same API call

This method can be batched with batch_start() and batch_end().

Parameters **users** – a list of sets of user/ip mappings with optional timeout in minutes eg. [('user1', '10.0.1.1'), ('user2', '10.0.1.2', 60)]

logout(*user*, *ip*)
Logout a single user

Removes a mapping of a user to an IP address

This method can be batched with batch_start() and batch_end().

Parameters

- **user** (*str*) – a username
- **ip** (*str*) – an ip address

logouts(*users*)
Logout multiple users in the same API call

This method can be batched with batch_start() and batch_end().

Parameters **users** – a list of sets of user/ip mappings eg. [(user1, 10.0.1.1), (user2, 10.0.1.2)]

register(*ip*, *tags*, *timeout=None*)
Register an ip tag for a Dynamic Address Group.

This method can be batched with batch_start() and batch_end().

Parameters

- **ip** (list or str) – IP address(es) to tag
- **tags** (list or str) – The tag(s) for the IP address
- **timeout** (*string*) – The optional timeout value in seconds. (Max is 2,592,000 sec (30 days))

send(*uidmessage*)
Send a uidmessage to the User-ID API of a firewall

Used for adhoc User-ID API calls that are not supported by other methods in this class. This method cannot be batched.

Parameters **uidmessage** (*str*) – The UID Message in XML to send to the firewall

set_group(*group*, *users*)
Set a group's membership to the specified users.

This method can be batched with batch_start() and batch_end().

Parameters

- **group** – The group name.
- **users** (*list*) – The users to be in this group.

tag_user(*user*, *tags*, *timeout=None*, *prefix=None*)
Tags the user with the specified tags.

This method can be batched with batch_start() and batch_end().

Note: PAN-OS 9.1+

Parameters

- **user** – The user.
- **tags** (*list*) – The list of tags to apply.
- **timeout** (*int*) – (Optional) The timeout for the given tags.
- **prefix** – Override class tag prefix.

unregister(*ip*, *tags*)
Unregister an ip tag for a Dynamic Address Group

This method can be batched with batch_start() and batch_end().

Parameters

- **ip** (list or str) – IP address(es) with the tag to remove
- **tags** (list or str) – The tag(s) to remove from the IP address

untag_user(*user*, *tags=None*, *prefix=None*)
Removes tags associated with a user.

This method can be batched with batch_start() and batch_end().

Note: PAN-OS 9.1+

Parameters

- **user** – The user.
- **tags** (*list*) – (Optional) Remove only these tags instead of all tags.

- **prefix** – Override class tag prefix.

CHAPTER 6

Release Notes

Release notes (changelogs) are available on GitHub: https://github.com/PaloAltoNetworks/pan-os-python/releases

CHAPTER 7

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

7.1 Types of Contributions

7.1.1 Report Bugs

Report bugs at https://github.com/PaloAltoNetworks/pan-os-python/issues.

7.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to fix it.

7.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" is open to whoever wants to implement it.

7.1.4 Write Documentation

The PAN-OS SDK for Python could always use more documentation, whether as part of the official pan-os-python docs, in docstrings, or even on the web in blog posts, articles, and such.

The main documentation is in the *docs* directory and the API reference is generated from docstrings in the code.

After you set up your development environment, type `poetry run make docs` to generate the documentation locally.

7.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/PaloAltoNetworks/pan-os-python/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

7.2 Get Started!

Ready to contribute some code? Here's how to set up *pan-os-python* for local development.

1. Install python 3.6 or higher

Development must be done using python 3.6 or higher. Development on python 2.7 is no longer supported.

2. Fork the *pan-os-python* repo on GitHub.
3. Clone your fork locally:

```
$ git clone https://github.com/your-username/pan-os-python.git
```

4. Install Poetry

   Poetry is a dependency manager and build tool for python If you don't have poetry installed, use the instructions here to install it:

   https://python-poetry.org/docs/#installation

5. Create a virtual environment with dependencies:

```
$ poetry install
```

6. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

7. Now you can make your changes locally
8. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ poetry run make lint
$ poetry run make bandit
$ poetry run make test
$ poetry run make test-all
$ poetry run make sync-deps
```

9. Commit your changes and push your branch to GitHub:

```
$ git add -A
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

10. Submit a pull request through the GitHub website.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W

X

Z