PAN-OS® and Panorama XML API Usage Guide

Version 7.0

Contents

Palo Alto Networks, Inc.
www.paloaltonetworks.com

Revision Date: October 10, 2015 2:09 PM

1 Using the XML API

In addition to the web interface and a Command Line Interface, PAN-OS provides an XML API to manage both the firewall and Panorama. The API allows access to several types of data on the device for integration with and use in other systems. The API is a web service that is implemented using HTTP requests and responses.

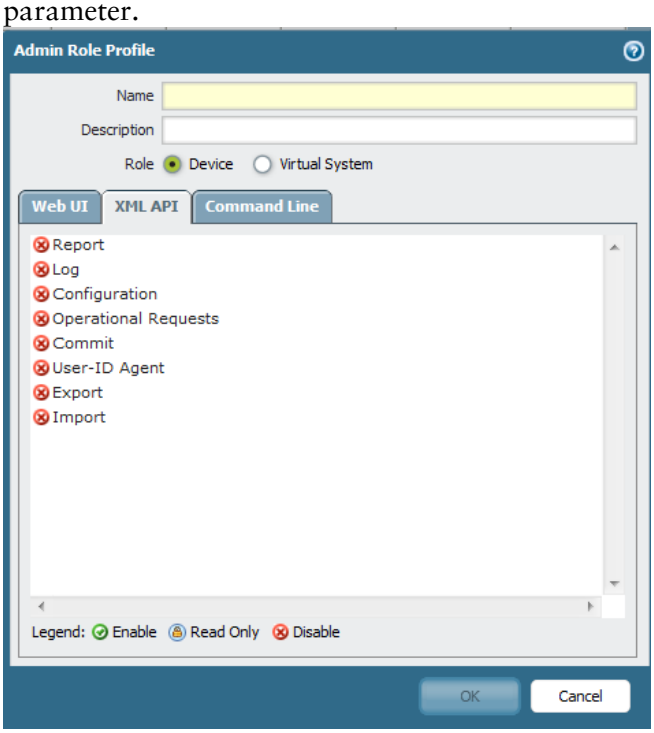
The structure of the URI for the API requests is: http(s)://*hostname*/api/?*request-parameters-values*

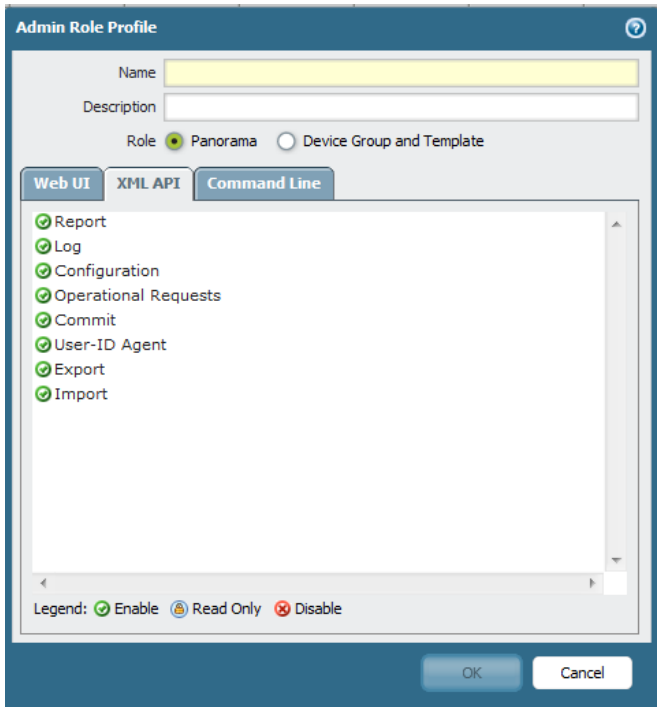
The *hostname* is the device's IP address or Domain name. The *request-parameters-values* is a series of multiple 'parameter=value' pairs separated by the ampersand character (&). The keywords for all the *parameters* are described in this document. The *values* can either be keywords or data-values in standard or XML format. The response data is always in XML format. When using the API with a command line tool such as cURL or wget, you can use both HTTP GET and POST methods.

The API supports the following types of Administrators and Admin roles:

- *Dynamic roles*: Superuser, Superuser (readonly), Device admin, Device admin (readonly), Vsys admin, Vsys admin (readonly)
- *Role-based Admins*: Device, Vsys, Panorama.

For Admin role profiles, you can enable or disable permissions on the web interface or the CLI based on the *type* parameter.





2 API Request Types

There are currently nine different API requests that you can access with the *type* parameter:

- Key Generation: type=*keygen*
- Device Configuration: type=*config*
- Operational Commands: type=*op*
- Commit Configuration: type=*commit*

- Reporting: type=*report*
- Exporting files: type=*export*
- Importing files: type=*import*
- Retrieving logs: type=*log*
- Set or Get User-ID mapping: type=*user-id*

To learn how to use the API browser to craft the syntax for the API query, see API Browser.

2.1 Key Generation

To use the API, you must generate an API key that is required for authenticating API calls. To generate the key, you must construct a URL request using the administrative credentials as follows.
http(s)://*hostname*/api/?type=keygen&user=*username*&password=*password*

Make sure that special characters in the password are URL/percent-encoded.

The result with be an XML block that contains the key. It should look like the following:

```
<response status="success">
    <result>
        <key>gJlQWE56987nBxIqyfa62sZeRtYuIo2BgzEA9UOnlZBhU</key>
    </result>
</response>
```

The key must be URL encoded when used in HTTP requests.
The key generation operation uses the master key for generating keys. If you have not changed the master key from the default, all firewalls with the same username/password will return the same key. You must change the master key on the device if you want different keys returned for the same username/password combination on two different devices.

To revoke or change the key, change the password with the associated admin account. As a best practice, set up a separate admin account for XML API access.

2.2 Device Configuration

The API allows you to configure or retrieve either all or part of the running or candidate device configuration. The API supports nine options that are accessed via the *action* parameter.

- Retrieve running configuration: action=*show*
- Get candidate configuration: action=*get*
- Set candidate configuration: action=*set*
- Edit candidate configuration: action=*edit*
- Delete candidate configuration: action=*delete*
- Rename a configuration object: action=*rename*
- Clone a configuration object: action=*clone*
- Move a configuration object: action=*move*
- Override a template setting: action=*override*
- Move multiple objects in a device group or virtual system: action=*multi-move*
- Clone multiple objects in a device group or virtual system: action=*multi-clone*

2.2.1 Retrieve

Using *action=show* with no additional parameters, will return the entire running configuration. Using the *xpath* parameter, you target a specific portion of the configuration. For example, to retrieve just the security rulebase, use: xpath=/config/devices/entry/vsys/entry/rulebase/security. **NOTE**: There is no trailing backslash character at the end of the *xpath*.

The URL for the API request will be:
http(s)://*hostname*/api/?type=config&action=show&key=*keyvalue*&xpath=/config/devices/entry/vsys/entry/rulebase/security

The XML response for the query should look like the following (truncated):

```
▼<response status="success">
  ▼<result>
    ▼<devices>
      ▼<entry name="localhost.localdomain">
        ▶<network>...</network>
        ▶<deviceconfig>...</deviceconfig>
        ▼<vsys>
          ▼<entry name="vsys1">
              <ssl-decrypt/>
            ▶<application>...</application>
            ▶<application-group>...</application-group>
            ▶<zone>...</zone>
              <service/>
              <service-group/>
              <schedule/>
            ▼<rulebase>
              ▼<security>
                ▼<rules>
                  ▼<entry name="p2p-games-proxies-tunnels">
```

To retrieve ARP information, you can use the following:
http(s)://firewall//api/?type=op&command =<show><arp><entry name='all'/></arp></show>

The XML response for the query should look like the following (truncated):

```
- <response status="success">
  - <result>
      <max>2500</max>
      <total>0</total>
      <timeout>1800</timeout>
      <dp>dp0</dp>
      <entries/>
    </result>
  </response>
```

2.2.2 Get

Beginning with PAN-OS 4.1.0, you can get the candidate configuration from the firewall or Panorama device using the Config Get API request. Use the *xpath* parameter to specify the portion of the configuration to get.
http(s)://hostname/api/?type=config&action=get&xpath=*path-to-config-node*

For instance to get the address objects in a VSYS, you can use the following:
http(s)://hostname//api/?type=config&action=get&xpath=/config/devices/entry/vsys/entry[@name='vsys1']/address

```
<response status="success" code="19">
  <result total-count="1" count="1">
    <address>
      <entry name="sra">
        <ip-netmask>172.16.1.2/32</ip-netmask>
      </entry>
      <entry name="sra-loaner">
        <ip-netmask>172.16.1.3/32</ip-netmask>
      </entry>
    </address>
  </result>
</response>
```

To get the pre-rules pushed from Panorama, you can use the following:
http(s)://firewall//api/?type=config&action=get&xpath=/config/panorama/vsys/entry[@name='vsys']/pre-rulebase/security

You can use this query is to get detail information on Applications and Threats from the firewall.
http(s)://hostname/api/?type=config&action=get&xpath=/config/predefined/threats/vulnerability/entry[@name='30003']

```
<response status="success" code="19">
  <result total-count="1" count="1">
    <entry name="30003" p="yes">
      <threatname>
        Microsoft Windows DCOM RPC Interface Buffer Overrun Vulnerability
      </threatname>
      <cve xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <member>CVE-2003-0352</member>
      </cve>
      <vendor xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <member>MS03-026</member>
      </vendor>
      <category>code-execution</category>
      <severity>critical</severity>
      <affected-host>
        <server>yes</server>
      </affected-host>
      <default-action>reset-server</default-action>
    </entry>
  </result>
</response>
```

http(s)://hostname/api/?type=config&action=get&xpath=/config/predefined/application, provides details on the full list of all applications.
http(s)://hostname/api/?type=config&action=get&xpath=/config/predefined/application/entry[@name='hotmail'], provides details on the specific application.

Refer to the API browser and follow the 'Configuration Commands' link to see all the available config xpaths.

2.2.3 Set

Using *action=set*, you can add or create a new object at a specified location in the configuration hierarchy. Use the *xpath* parameter to specify the location of the object in the configuration, without the name of the node being updated. For example, if you are adding a new rule to the security rulebase, the xpath-value would be */config/devices/entry[@name='localhost.localdomain']/vsys/entry[@name='vsys1']/rulebase/security*. Use

the *element* parameter to specify a value for the object you are adding or creating using its XML representation (as seen in the output of action=show).

For example, to create a new rule called *rule1* in the security policy, use the following Config Set API request:
http(s)://hostname/api/?type=config&action=set&key=keyvalue&xpath=*xpath-value*&element=*element-value*, where

xpath-value is:
/config/devices/entry/vsys/entry/rulebase/security/rules/entry[@name='rule1']

element-value is:
<source><member>src</member></source><destination><member>dst</member></destination><service><member>service</member></service><application><member>application</member></application><action>action</action><source-user><member>src-user</member></source-user><option><disable-server-response-inspection>yes-or-no</disable-server-response-inspection></option><negate-source>yes-or-no</negate-source><negate-destination>yes-or-no</negate-destination><disabled>yes-or-no</disabled><log-start>yes-or-no</log-start><log-end>yes-or-no</log-end><description>description</description><from><member>src-zone</member></from><to><member>dst-zone</member></to>

Use the response from the config show API request to create the xml body for the element.
http(s)://hostname/api/?type=config&action=show

To add an additional member to a group/list, include the 'list' node in the xpath using the *member[text()='name']* syntax and include the members in the element parameter. For example, to add an additional static address object named *abc* to an address group named *test*, use:
http(s)://hostname/api/?type=config&action=set&xpath=/config/devices/entry/vsys/entry[@name='vsys1']/address-group/entry[@name='test']&element=<static><member>abc</member></static>

2.2.4 Edit

Using action=*edit*, you can replace an existing object hierarchy at a specified location in the configuration with a new value. Use the *xpath* parameter to specify the location of the object, including the node to be replaced. Use the *element* parameter to specify a new value for the object using its XML object hierarchy (as seen in the output of action=show). For instance, to replace the application(s) currently used in a rule *rule1* with a new application, use:
http(s)://hostname/api/?type=config&action=edit&key=keyvalue&xpath=*xpath-value*&element=*element-value*, where

xpath=/config/devices/entry/vsys/entry/rulebase/security/rules/entry[@name='rule1']/application
element=<application><member>app-name</member></application>

Use the response from the config show API request to create the xml body for the element.
http(s)://hostname/api/?type=config&action=show

To replace all members in a node with a new set of members, use the entry tag in both the xpath and element parameters. For example, to replace all the address objects in the address group named test with two new static members named *abc* and *xyz*, use:
http(s)://hostname/api/?type=config&action=edit&xpath=/config/devices/entry/vsys/entry[@name='vsys1']/address-group/entry[@name='test']&element=<static><entry name='test'><member>abc</member><member>xyz</member></static></entry>

2.2.5 Delete

Using action=*delete*, you can delete an object at a specified location in the configuration. Use xpath parameter to specify the location of the object to be deleted. For instance, to delete a rule named *rule1* in the security policy, use the below API query:
http(s)://hostname/api/?type=config&action=delete&xpath=/config/devices/entry/vsys/entry/rulebase/security/rules/entry[@name='rule1']
To delete a single member object in a group, use the object name in the xpath as member[text()='name']. For example, to delete a static address object named *abc* in an address group named *test*, use the below xpath:
http(s)://hostname/api/?type=config&action=delete&xpath=/config/devices/entry/vsys/entry[@name='vsys1']/address-group/entry[@name='test']/static/member[text()='abc']

2.2.6 Rename

Using action=*rename*, you can rename an object at a specified location in the configuration. Use the *xpath* parameter to specify the location of the object to be renamed. Use the *newname* parameter to provide a new name for the object.
For instance, to rename an address object called old_address to new_address, use the below API query:
http(s)://hostname/api/?type=config&action=*rename*&xpath=/config/devices/entry/vsys/entry[@name='vsys1']/address/entry[@name='old_address']&*newname*=new_address

2.2.7 Clone

Using action=*clone*, you can clone an existing configuration object. Use the *xpath* parameter to specify the location of the object to be cloned. Use the '*from*' parameter to specify the source object, and the *newname* parameter to provide a name for the cloned object. For instance, to clone a security policy called rule1 into rule2, use the below API query:
http(s)://hostname/api/?type=config&action=*clone*&xpath=/config/devices/entry/vsys/entry[@name='vsys1']/rulebase/security/rules&*from*=/config/devices/entry/vsys/entry[@name='vsys1']/rulebase/security/rules/entry[@name='rule1']&*newname*=rule2

This query returns the following response:
<response status="success" name="rule2"/>

And a corresponding success log is recorded in the config log:
1,2014/03/19 19:07:45,0009C100708,CONFIG,0,0,2014/03/19 19:07:45,10.66.18.1,,clone,admin,Web,Succeeded, config devices entry vsys vsys1 rulebase security rules,384,0x8000000000000000

2.2.8 Move

Using action=*move*, you can move the location of an existing configuration object. Use the *xpath* parameter to specify the location of the object to be moved, the *where* parameter to specify type of move, and *dst* parameter to specify the destination xpath.

- where=*after*&dst=xpath
- where=*before*&dst=xpath
- where=top
- where=bottom

For instance, to move a security policy called rule1 after rule2, use the below API query:
http(s)://hostname/api/?type=config&action=*move*&xpath=/config/devices/entry/vsys/entry[@name='vsys1']/rulebase/security/rules/entry[@name='rule1']&where=after&dst=rule2

2.2.9 Override

Using action=*override*, you can override a setting that was pushed to the device from a template. Use the *xpath* parameter to specify the location of the object to override. For example, the following command shows how to override the SNMP trap profile configuration settings that were pushed to the device using a template:

https://hostname/api/?type=config&action=override&xpath=/config/shared/log-settings/snmptrap&element=<entry name="snmp" src="tpl"><version src="tpl"><v2c src="tpl"><server src="tpl"><entry name="test" src="tpl"><manager src="tpl">2.2.2.2</manager><community src="tpl">test</community></entry></server></v2c></version></entry>

2.2.10 Multi-Move/Multi-Clone

The action=*multi-move* and action=*multi-clone* allow you to move and clone addresses across device groups and virtual systems. Templates do not support the multi-move and multi-clone capability.
The syntax for multi-move and multi-clone specifies the xpath for the destination where the addresses will be moved to, the xpath for the source and the list of objects within the specified source. It also includes a flag for displaying the errors when the firewall performs a referential integrity check on the multi-move or multi-clone action.

To move/clone addresses addr1, addr2, to dg norcal from dg socal the request would be as follows:

https://firewall/api/?type=config&action=multi-move&xpath=/config/devices/entry[@name='localhost.localdomain']/device-group/entry[@name='norcal']/address&element=<selected-list><source xpath="/config/devices/entry[@name='localhost.localdomain']/device-group/entry[@name='socal']/address"><member>addr1</member><member>addr2</member></source></selected-list><all-errors>no</all-errors>

https://firewall/api/?type=config&action=multi-clone&xpath=/config/devices/entry[@name='localhost.localdomain']/device-group/entry[@name='norcal']/address&element=<selected-list><source xpath="/config/devices/entry[@name='localhost.localdomain']/device-group/entry[@name='socal']/address"><member>addr1</member><member>addr2</member></source></selected-list><all-errors>no</all-errors>

2.3 Commit

You can commit candidate configuration to a firewall or Panorama device using the commit API request.

To commit a candidate configuration, use the following:
http(s)://hostname/api/?type=commit&cmd=<commit></commit>

To do a force commit of the candidate configuration, use the following:
http(s)://hostname/api/?type=commit&cmd=<commit><force>*body*</force></commit>

To do a granular or partial commit of the candidate configuration, use the following:
https://{hostname/api/?type=commit&action=partial&cmd=<commit><partial><vsys><member>{vsys_number/name}</member></vsys></partial></commit>&target={device_S/N}&key={api_key}

Refer to the API browser for the different options available for use with force and partial commits. The *body* element in the *cmd* parameter should be replaced by the XML element for the corresponding commit operation.

When there are no pending changes to commit, API request returns:

```
<response status="success" code="19">
        <msg>There are no changes to commit.</msg>
</response>
```

When there are pending changes, the API returns a Job ID for the commit request as below.

```
<response status="success" code="19">
        <result>
                <msg><line>Commit job enqueued with jobid 4</line></msg>
                <job>4</job>
        </result>
</response>
```

You can query the status of the job using the below Operational API request and the corresponding response:

```
http(s)://hostname/api/?type=op&cmd=<show><jobs><id>4</id></jobs></show>
<response status="success">
        <result>
                <job>
                        <tenq>2011/10/20 20:41:44</tenq>
                        <id>4</id>
                        <type>Commit</type>
                        <status>FIN</status>
                        <stoppable>no</stoppable>
                        <result>OK</result>
                        <tfin>20:42:22</tfin>
                        <progress>20:42:22</progress>
                        <details><line>Configuration committed successfully</line></details>
                        <warnings/>
                </job>
        </result>
</response>
```

2.3.1 Commit-all (Panorama)

To centrally manage firewalls from Panorama, you can use the commit-all API request type to push and validate shared policy to the firewalls using device groups and configuration to the firewalls using templates or template stacks.

To validate the policy before pushing it to a device group

```
http(s)://panorama/api/?type=commit&action=all&cmd= <commit-all><shared-policy><validate-only></validate-only></shared-policy></commit-all>
```

To push configuration to a device group (say *west-dg*), use the following:

```
http(s)://panorama/api/?type=commit&action=all&cmd=<commit-all><shared-policy><device-group><entry name="west-dg"/></device-group></shared-policy></commit-all>
```

To push configuration to a VSYS (say *mktg-vsys that is in west-dg*), use the following:
http(s)://panorama/api/?type=commit&action=all&cmd=<commit-all><shared-policy><device-group><name>west-dg</name><devices><entryname="serial_number"><vsys><member>***mktg-vsys***</member></vsys></entry></devices></device-group></shared-policy></commit-all>

To push configuration to a specific device by serial number, use the following:
http(s)://panorama/api/?type=commit&action=all&cmd=<commit-all><shared-policy><device-group><name>west-dg</name><devices><entryname="serial_number"></entry></devices></device-group></shared-policy></commit-all>

Refer to the API browser for other options available for granular commit operations on Panorama. In the *cmd* parameter, you must replace XML element for the corresponding commit-all operation.

2.4 Operational Commands

Use any of the operational commands available on the command line interface using the *Op* API request below:
http(s)://hostname/api/?type=op&cmd=*xml-body*

Refer to the API browser and follow the link for operational commands to see a complete listing of all the different options available for the *xml-body* and their corresponding operation.

Examples of operational API requests include setting, showing, or clearing runtime parameters, saving and loading configurations to disk, retrieving interface or system information, etc.

To request a system restart, use:
http(s)://hostname/api/?type=op&cmd=<request><restart><system></system></restart></request>

To install system software version 4.1.0, use:
http(s)://hostname/api/?type=op&cmd=<request><system><software><install><version>4.1.0</version></install></software></system></request>

To set the system setting to turn on multi-vsys mode, use:
http(s)://hostname/api/?type=op&cmd=<set><system><setting><multi-vsys></multi-vsys></setting></system></set>

To schedule a User Activity Report, use:
http(s)://hostname/api/?type=op&cmd=<schedule><uar-report><user>*username*</user><title>*titlename*</title></uar-report></schedule>

To validate the full configuration on a firewall or Panorama, use
http(s)://hostname/api/?type=op&cmd=<validate><full></full></validate>

Note that you can validate the partial configuration on a firewall, for example by excluding the policy and objects configuration. For example:
http(s)://hostname/api/?type=op&cmd=<validate><partial><device-and-network>excluded</device-and-network></partial></validate>

To save or load config to/from a file, use:
http(s)://hostname/api/?type=op&cmd=<save><config><to>*filename*</to></config></save>, and
http(s)://hostname/api/?type=op&cmd=<load><config><from>*filename*</from></config></load>

2.5 Reporting

The XML API provides a way to quickly pull the results of any report defined in the system using the type=*report* parameter. There are three report stores that can be pulled from:

- Dynamic Reports (ACC reports): reporttype=*dynamic*
- Predefined Reports: reporttype=*predefined*
- Custom Reports: reporttype=*custom*

To retrieve a specific report by name, use the *reportname* parameter:
http(s)://hostname/api/?type=report&reporttype=*dynamic*|*predefined*|*custom*&reportname=*name*

Note: When generating an on-demand report on Panorama, when you use the query, the on-screen output will display a job-id instead of the requested report. To retrieve the report, you must poll the job status using the job id (shown in 2.6.3) until the job completes. On completion, the job status reports as FIN (finished), and the reports displays.

2.5.1 Dynamic reports

The names for the currently supported *dynamic* reports follows:

- acc-summary
- custom-dynamic-report
- top-app-summary
- top-application-categories-summary
- top-application-risk-summary
- top-application-subcategories-summary
- top-application-tech-summary
- top-applications-summary
- top-applications-trsum
- top-attacker-countries-summary
- top-attackers-summary
- top-attacks-acc
- top-blocked-url-categories-summary
- top-blocked-url-summary
- top-blocked-url-user-behavior-summary
- top-data-dst-countries-summary
- top-data-dst-summary
- top-data-egress-zones-summary
- top-data-filename-summary
- top-data-filetype-summary
- top-data-ingress-zones-summary
- top-data-src-countries-summary
- top-data-src-summary
- top-data-type-summary
- top-dst-countries-summary
- top-dst-summary
- top-egress-zones-summary
- top-hip-objects-details
- top-hip-objects-summary
- top-hip-profiles-details
- top-hip-profiles-summary
- top-hip-report-links
- top-hr-applications-summary
- top-ingress-zones-summary
- top-rule-summary
- top-spyware-phonehome-summary
- top-spyware-threats-summary
- top-src-countries-summary
- top-src-summary
- top-threat-egress-zones-summary
- top-threat-ingress-zones-summary
- top-threats-type-summary
- top-url-categories-summary
- top-url-summary
- top-url-user-behavior-summary
- top-victim-countries-summary
- top-victims-summary
- top-viruses-summary
- top-vulnerabilities-summary

You can get the above list of dynamic report names using the below API request, or by following the links on the API browser. http(s)://hostname/api/?type=*report*&reporttype=*dynamic*

For dynamic reports, you can provide the timeframe for the report using the *period* or *starttime* and *endtime* options (use a + instead of a space between the date and timestamp). The number of rows is set via *topn*. The possible values for *period* are:

- last-60-seconds
- last-15-minutes
- last-hour
- last-12-hrs
- last-24-hrs
- last-calendar-day
- last-7-days
- last-7-calendar-days
- last-calendar-week
- last-30-days

2.5.2 Predefined reports

The names for the currently supported *predefined* reports are shown below. Predefined reports always return data for the last 24 hour period. You can also get this list by following the link for predefined reports on the API browser or running this API query: http(s)://hostname/api/?type=*report*&reporttype=*predefined*

- SaaS Application Usage
- bandwidth-trend
- botnet
- hruser-top-applications
- hruser-top-threats
- hruser-top-url-categories
- risk-trend
- risky-users
- spyware-infected-hosts
- threat-trend
- top-application-categories
- top-applications
- top-attackers
- top-attackers-by-countries
- top-attacks
- top-blocked-url-categories
- top-blocked-url-user-behavior
- top-blocked-url-users
- top-blocked-websites
- top-connections
- top-denied-applications
- top-denied-destinations
- top-denied-sources
- top-destination-countries
- top-destinations
- top-egress-interfaces
- top-egress-zones
- top-http-applications
- top-ingress-interfaces
- top-ingress-zones
- top-rules
- top-source-countries
- top-sources
- top-spyware-threats
- top-technology-categories
- top-url-categories
- top-url-user-behavior
- top-url-users
- top-users
- top-victims
- top-victims-by-countries
- top-viruses
- top-vulnerabilities
- top-websites
- unknown-tcp-connections
- unknown-udp-connections
- wildfire-file-digests

2.5.3 Custom reports

For custom reports, the different selection criteria (time frame, group-by, sort-by, etc.) are part of the report definition itself. The API returns any shared custom reports. Note that quotes are not required around the reportname and any spaces in the report name must be URL encoded to %20.

For custom reports created in a specific VSYS, you can retrieve them directly by specifying the vsys parameters.

Note: Panorama reports are retrieved asynchronously and return a job ID. This is unlike pulling reports on firewall device which returns the report data either synchronously or asynchronously.

Step one, retrieve the report definition from the configuration using a Config Get API request. For example, a report named *report-abc*:
http(s)://firewall/api/?type=config&action=get&xpath=/config/devices/entry/vsys/entry[@name='vsys1']/reports/entry[@name='*report-abc*']

```
<response status="success" code="19">
  <result total-count="1" count="1">
    <entry name="srareport">
      <type>
        <appstat>
          <aggregate-by>
            <member>category-of-name</member>
            <member>technology-of-name</member>
          </aggregate-by>
        </appstat>
      </type>
      <period>last-24-hrs</period>
      <topn>10</topn>
      <topm>10</topm>
      <query>(name neq '')</query>
    </entry>
  </result>
</response>
```

Step Two, retrieve a dynamic report data using reporttype=*dynamic*, reportname=*custom-dynamic-report*, and cmd=*report-definition* where report definition is the XML definition retrieved in the previous query.
http(s)://hostname/api/?type=report&reporttype=dynamic&reportname=custom-dynamic-report&cmd=<type><appstat><aggregate-by><member>category-of-name</member><member>technology-of-name</member></aggregate-by></appstat></type><period>last-24-hrs</period><topn>10</topn><topm>10</topm><query>(name neq '') AND (vsys eq 'vsys1')</query>

2.6 Exporting files

You can export certain types of files from the firewall using the type=*export* parameter in the API request. Use the *category* parameter to specify the type of file that you want to export.

- Configuration: category=*configuration*
- Certificates/Keys: category=<*certificate* | *high-availability-key* | *key-pair*>
- Response pages: category=< *application-block-page* | *captive-portal-text* | *file-block-continue-page* | *file-block-page* | *global-protect-portal-custom-help-page* | *global-protect-portal-custom-login-page* | *global-protect-portal-custom-welcome-page* | *ssl-cert-status-page* | *ssl-optout-text* | *url-block-page* | *url-coach-text* | *virus-block-page*>
- Technical support data: category=*tech-support*
- Device State: category=*device-state*

Use wget or cURL tools to export the file from the firewall and save locally with a local file name, as below. Refer to their respective man pages for additional usage information.

>wget --output-document=*filename* "http(s)://firewall/api/?*query-parameters*"

>curl -o *filename* "http(s)://firewall/api/?*query-parameters*"

When using the API query from a web-browser, you can specify to=filename as an optional parameter if you would like to provide a different name when saving the file locally.

2.6.1 Packet Captures

You can export packet captures from the firewall device using the Export API request. The type of PCAP to be exported using the API must be specified using the *category* parameter.

- Application Packet Captures: category=*application-pcap*
- Threat Packet Captures: category=*threat-pcap*
- Debug Filter Packet Captures: category=*filter-pcap* or *filters-pcap*. This PCAP is based on packets that match the filters you have defined on the firewall.
- Data filtering Packet Captures: category=*dlp-pcap*. Data filtering pcaps require a dlp-password parameter.

2.6.1.1 Application PCAPs

Application PCAPs are organized by a *Directory/Filename* structure where the directory is a date in yyyymmdd format. Filename for application pcaps uses a SourceIP-SourcePort-DestinationIP-DestinationPort-SessionID.pcap format.
To get a list of directories for the application PCAPs, you can use the following:
http(s)://firewall/api/?type=export&category=application-pcap, and

To get a listing of the files under a directory, you can use the *from* parameter as follows:
http(s)://firewall/api/?type=export&category=application-pcap&from=*yyyymmdd*, and

To retrieve a specific application PCAP file by its name, you can use the *from* parameter as follows:
http(s)://firewall/api/?type=export&category=application-pcap&from=*yyyymmdd/filename*, and
The file will be retrieved and saved locally using the name *yyyymmdd-filename*.

To retrieve a specific application PCAP file by its name, and save it locally with a custom name, use the *to* parameter as follows:
http(s)://firewall/api/?type=export&category=application-pcap&from=*yyyymmdd/filename*&to=*localfile*, and

2.6.1.2 Threat PCAPs

To export threat PCAPs, you need to provide the device serial number, PCAP ID from the threat log, and the *search time*, which is the time that the PCAP was received on the firewall. Threat PCAP filenames use an pcapID.pcap format.
To retrieve a specific threat PCAP, you must use the following parameters:
http(s)://firewall/api/?type=export&category=threat-pcap &serialno=*number*&pcap-id=*id*&search-time=*yyyymmdd hr:min:sec*

2.6.1.3 Filter PCAPs

To get a list of available filter PCAPs, you can use:
http(s)://firewall/api/?type=export&category=filters-pcap

To retrieve a specific filter PCAP file, you can use:
http(s)://firewall/api/?type=export&category=filters-pcap&from=*filename*

2.6.1.4 Data filtering PCAPs

To get a list of data filtering PCAP file names, you can use:
http(s)://firewall/api/?type=export&category=dlp-pcap&dlp-password=<password>

To retrieve a specific data filtering PCAP file, you can use:

http(s)://firewall/api/?type=export&category=dlp-pcap&dlp-password=<password>&from=*filename*&to=<localfile>

2.6.2 Certificates/Keys

There are additional query parameters to be specified when exporting Certificates/Keys from the firewall.
http(s)://firewall/api/?type=export&category=*certificate*&certificate-name=<certificate_name>&format=<*pkcs12* | *pem*>&include-key=<*yes* | *no*>&vsys=<vsys | omit this parameter to import it into shared location>

- certificate-name: name of the certificate object on the firewall
- format: cerficate format, pkcs12 or pem
- include-key: yes or no parameter to include or exclude the key
- passphrase: required when including the certificate key
- vsys: Virtual System where the certificate object is used. Ignore this parameter if the certificate is a shared object.

2.6.3 Technical Support Data (debug logs etc.)

Since debug log data sizes are large, the API uses an asynchronous job scheduling approach to retrieve technical support data. The initial query creates a Job id with a hash that is used in the follow on queries with the action parameter. The values for the action parameter are:

- When action parameter is not specified, the system creates a new job to retrieve tech support data.
- action=*status*, to check status of the job. Returns either PEND or FIN.
- action=*get*, to retrieve the data when the status shows FIN.
- action=*finish*, to manually delete the job.

Create a job to retrieve technical support data using http(s)://firewall/api/?type=export&category=tech-support, which returns a job id.

```
▼<response status="success" code="19">
  ▼<result>
    ▼<msg>
       <line>Exec job enqueued with jobid 2</line>
     </msg>
     <job>BJ1Lmjc7Reqh9eZTJuzHQJ1STmSo0qMuUrlDTQFH9zA=</job>
   </result>
 </response>
```

Check the status of the job using: http(s)://firewall/api/?type=*export*&category=*tech-support*&action=*get*&job-id=*id*. Use the job id returned in the previous response as the job-id parameter. A status value of 'FIN' indicates the data is ready to be retrieved.

```
<response status="success">
  <result>
    <job>
      <tenq>2012/06/14 10:11:09</tenq>
      <id>2</id>
      <user/>
      <type>Exec</type>
      <status>FIN</status>
      <stoppable>no</stoppable>
      <result>OK</result>
      <tfin>10:12:39</tfin>
      <progress>10:12:39</progress>
      <details/>
      <warnings/>
      <resultfile>//tmp/techsupport.tgz</resultfile>
    </job>
  </result>
</response>
```

Retrieve the data using: http(s)://firewall/api/?type=*export*&category=*tech-support*&action=*get*&job-id=*id*. When using cURL or wget, you can specify the output file name as an option to cURL (-o) or wget (--output-document). After a successful retrieval of the job data, the job is automatically deleted by the system.

To manually delete the job use: http(s)://firewall/api/?type=*export*&category=*tech-support*&action=*finish*&job-id=*id*

```
<response status="success">
  <msg>Job 2 removed.</msg>
</response>
```

2.7 Importing files

Beginning with PAN-OS 5.0.0, you can import certain types of files into the firewall using the type=*import* parameter in the API request. The type of file to be imported must be specified using the *category* parameter.

- Software: category=*software*
- Content: category=<*anti-virus* | *content* | *url-database* | *signed-url-database*>
- Licenses: category=*license*
- Configuration, category=*configuration*
- Certificates/Keys, category=<*certificate* | *high-availability-key* | *key-pair*>
- Response pages, category=< *application-block-page* | *captive-portal-text* | *file-block-continue-page* | *file-block-page* | *global-protect-portal-custom-help-page* | *global-protect-portal-custom-login-page* | *global-protect-portal-custom-welcome-page* | *ssl-cert-status-page* | *ssl-optout-text* | *url-block-page* | *url-coach-text* | *virus-block-page*>
- Clients, category=*global-protect-client*
- Custom logo, category=*custom-logo*

Use wget or cURL tools to import the file to the firewall, as below. Refer to their respective man pages for additional usage information.

>wget --post-file *filename* "http(s)://firewall/api/?*query-parameters*&client=*wget* &file-name=*filename*"

>curl --form file=@*filename* "http(s)://firewall/api/?*query-parameters*"

Note: The API does not support importing files to a device via Panorama. You can achieve this in two steps:

import the file to Panorama first, and then run a request batch upload-install op command to Panorama like so: http://panorama/api/?type=op&cmd=<request><batch><anti-virus><upload-install><uploaded-file>your-file-name-here</uploaded-file><devices>SN12121212</devices></upload-install></anti-virus></batch></request>

2.7.1 Certificates/Keys

There are additional query parameters to be specified when importing Certificates/Keys to the firewall. The type of the certificate or key file is specified using the category parameter

- category=*certificate*
- category=*keypair*
- category=*high-availability-key*

The certificate file import (category=*certificate*) and keypair import (category=*keypair*) take the below additional parameters.

- certificate-name: name of the certificate object on the firewall
- format: cerficate format, pkcs12 or pem
- passphrase: required when including the certificate key
- vsys: Virtual System where the certificate object is used. Ignore this parameter if the certificate is a shared object.

For example, http(s)://firewall/api/?type=*import*&category=*certificate*&certificate-name=<certificate_name>&format=<*pkcs12* | *pem*>&passphrase=<text>&vsys=<vsys | omit this parameter to import it into shared location>

2.7.2 Response pages

Only the GlobalProtect related response pages require an additional parameter for the *profile* where the page should be imported to.

- profile=*profilename*

2.7.3 Custom logo

Custom logos can be imported to different locations based on the where parameter.

- where=<*login-screen* | *main-ui* | *pdf-report-footer* | *pdf-report-header*>

2.8 Retrieving Logs

Beginning with PAN-OS 5.0.0, you can retrieve logs from the firewall using the API with the type=*log* parameter. The type of logs to retrieve must be specified using the log-type parameter.

- log-type=*traffic*, for traffic logs
- log-type=*threat*, for threat logs,
- log-type=*config*, for config logs,
- log-type=*system*, for system logs,
- log-type=*hipmatch*, for HIP logs,
- log-type=*wildfire*, for WildFire logs,
- log-type=*url*, for URL Filtering logs,
- log-type=*data*, for Data Filtering logs.

The other optional parameters to this request are:

- *query* parameter to specify match criteria for the logs. This is similar to the query provided in the WebUI under the Monitor tab when viewing the logs. The query must be URL encoded.
- *nlogs* parameter to specify the number of logs to be retrieved. The default is 20 when the parameter is not specified. The maximum is 5000.
- *skip* parameter to specify the number of logs to skip when doing a log retrieval. The default is 0. This is useful when retrieving logs in batches where you can skip the previously retrieved logs.

Since log data sizes can be large, the API uses an asynchronous job scheduling approach to retrieve log data. The initial query returns a Job id with a Hash that is used in the follow on queries with the action parameter. The values for the action parameter are:

- Unspecified: when the action parameter is not specified, the system creates a new job to retrieve log data.
- action=*get*, to check status and retrieve the log data when the status is FIN. (This is a slight difference from the asynchronous approach to retrieve tech support data where a separation status action was available)
- action=*finish*, to manually delete the job.

To create a job to retrieve all traffic logs that occurred after a certain time, you can use below query.

NOTE: A web-browser will automatically URL encode the parameters, but when using wget/curl tools, the query parameter must be URL encoded.

http(s)://firewall/api/?type=log&log-type=traffic&query=(receive_time geq '2012/06/22 08:00:00')

```
▼<response status="success" code="19">
  ▼<result>
    ▼<msg>
       <line>query job enqueued with jobid 18</line>
     </msg>
     <job>4CkbDkn0186ys2XtWn2fYSd0IcUeF9EpUuixgKQwuwQ=</job>
   </result>
  </response>
```

Retrieve the data using: http(s)://firewall/api/?type=log&action=get&job-id=*id*, where id is the value returned in the previous response.

```
▼<response status="success">
  ▼<result>
    ▶<job>...</job>
    ▼<log>
      ▼<logs count="20" progress="100">
        ▼<entry logid="5753304543500710425">
           <domain>1</domain>
           <receive_time>2012/06/13 15:43:17</receive_time>
           <serial>001606000117</serial>
           <seqno>6784588</seqno>
           <actionflags>0x0</actionflags>
           <type>TRAFFIC</type>
           <subtype>start</subtype>
           <config_ver>1</config_ver>
           <time_generated>2012/06/13 15:43:17</time_generated>
           <src>172.16.1.2</src>
           <dst>10.0.0.246</dst>
           <natsrc>10.16.0.96</natsrc>
           <natdst>10.0.0.246</natdst>
           <rule>default allow</rule>
```

When the job status is FIN (finished), the response automatically includes all the logs in the xml data response. The <log> node in the xml data is not present when the job status is still pending. After successful log data retrieval, the system automatically deletes the job.

To manually delete a log retrieval job, you must run the below query.
http(s)://firewall/api/?type=log&action=finish&job-id=*id*, which on successful completion returns:

```
▼<response status="success">
   <msg>Job 18 removed.</msg>
 </response>
```

2.9 User-ID mapping

Beginning with PAN-OS 5.0.0, you can apply User-ID mapping information directly to the firewall using the API with the type=*user-id* parameter. Additionally you can also register a Dynamic Address Group using this API request. It takes the input file containing the User-ID mapping information.

>wget --post-file *filename* "http(s)://firewall/api/?type=*user-id* &client=*wget* &file-name=*filename*"

>curl --form file=@*filename* "http(s)://firewall/api/? type=*user-id*"

2.9.1 Map users and groups

When providing a User-ID mapping for a login event, logout event, or for groups, the input file format is as shown below.

```
<uid-message>
    <version>1.0</version>
    <type>update</type>
    <payload>
        <login>
            <entry name="domain\uid1" ip="10.1.1.1" timeout="20">
                <hip-report>
                        .....

                </hip-report>
            </entry>
        </login>
        <groups>
            <entry name="group1">
                <members>
                    <entry name=" user1"/>
                    <entry name=" user2"/>
                </members>
            </entry>
            <entry name="group2">
                <members>
                    <entry name="user3"/>
                </members>
            </entry>
        </groups>
    </payload>
```

```
</uid-message>
```

2.9.2 Create a multi-user system entry on the firewall

Use the following input file format to set up a terminal server entry on the firewall and to specify the port range and block size of ports that will be assigned per user. If you are using the default port range (1025 to 65534) and block size (200) you do not need to send a multiusersystem setup message; the firewall will automatically create the terminal server object when it receives the first login message.

```
<uid-message>
    <payload>
        <multiusersystem>
            <entry ip="10.1.1.2" startport="xxxxx" endport="xxxxx" blocksize="xxx">
        </multiusersystem>
    </payload>
    <type>update</type>
    <version>1.0</version>
</uid-message>
```

2.9.3 Create IP address-port-username mappings from multi-user system login and logout events

The following shows the input file format for a User-ID XML multiusersystem login event. Note that a login event payload that the terminal server sends to the firewall can contain multiple login events. The firewall uses the information in the information in the login message to populate its user mapping table. For example, if the firewall received a packet with a source address and port of 10.1.1.23:20101, it would map the request to user jparker for policy enforcement.

```
 <uid-message>
    <payload>
        <login>
            <entry name="acme\jparker" ip="10.1.1.23" blockstart="20100">
        </login>
    </payload>
    <type>update</type>
    <version>1.0</version>
</uid-message>
```

The following shows the input file format for a User-ID XML multiusersystem logout event. Upon receipt of a logout event message with a blockstart parameter, the firewall removes the corresponding IP address-port-user mapping. If the logout message contains a username and IP address, but no blockstart parameter, the firewall removes all mappings for the user. If the logout message contains an IP address only, the firewall removes the multi-user system and all associated mappings.

```
<uid-message>
    <payload>
        <logout>
            <entry user="domain\uid2" ip="10.1.1.2" blockstart="xxxxx">
        </logout>
    </payload>
    <type>update</type>
    <version>1.0</version>
```

```
</uid-message>
```

2.9.4 Register an IP address for a Dynamic Address Group

When registering an IP address for a Dynamic Address Group, the input file format is as shown below.

```
<uid-message>
    <version>1.0</version>
    <type>update</type>
    <payload>
       <register>
          <entry ip="10.1.1.1">
           <tag>
                 <member>CBB09C3D-3416-4734-BE90-0395B7598DE3</member>
           </tag>
          </entry>
       </register>
        <unregister>
           <entry ip="10.1.1.3"/>
            <tag>
                 <member>CBB09C3D-3416-4734-BE90-0395B7598DE5</member>
            </tag>
           </entry>
       </unregister>
    </payload>
</uid-message>
```

3 Panorama to device redirection

You can use the API on the Panorama to redirect the queries to a specific firewall device managed by the Panorama using the target parameter. The target parameter takes the device serial number as a value. For instance, to run a Panorama query that directs an operational command to a firewall device, you can use. http(s)://panorama/api/?type=op&cmd=<show><system><info></info></system></show>&target=device-serial-number

4 Targeting a specific Virtual System

Use the vsys parameter to target the API request to a specific Virtual System. You can use the vsys parameter for all Operational commands, Dynamic reports, Custom reports, and User-ID. For configuration commands, the xpath for virtual system specific objects includes the virtual system. For example, the xpath for an address group object in vsys1 is /config/devices/entry/vsys/entry[@name='vsys1']/address-group/entry[@name='test'].

5 Error Codes

The API response XML contains a status field and additionally an error field. The different error codes returned by the API in the error field are listed in the table below.

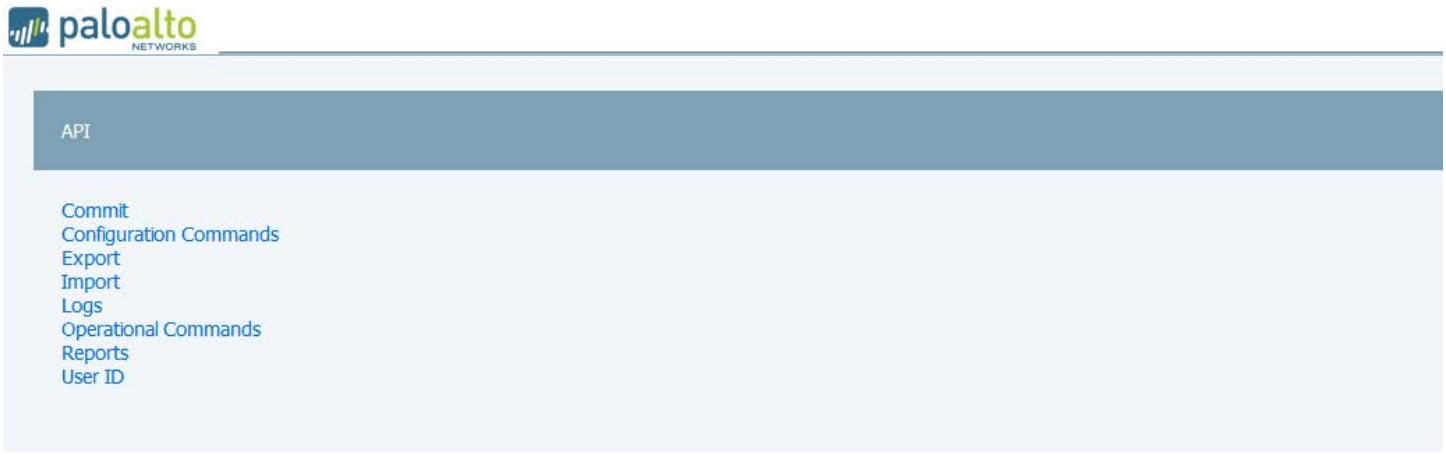
Error code	Name	Description
400	Bad request	Returned when a required parameter is missing, an illegal parameter value is used.
403	Forbidden	Returned for authentication or authorization errors including invalid key, insufficient admin access rights.

1	Unknown command	The specific config or operational command is not recognized.
2-5	Internal errors	Check with technical support when seeing these errors.
6	Bad Xpath	The xpath specified in one or more attributes of the command is invalid. Check the API browser for proper xpath values.
7	Object not present	Object specified by the xpath is not present. For example, entry[@name='value'] where no object with name 'value' is present.
8	Object not unique	For commands that operate on a single object, the specified object is not unique.
9	Internal error	Check with technical support when seeing these errors.
10	Reference count not zero	Object cannot be deleted as there are other objects that refer to it. For example, address object still in use in policy.
11	Internal error	Check with technical support when seeing these errors.
12	Invalid object	Xpath or element values provided are not complete.
13	Operation failed	A descriptive error message is returned in the response.
14	Operation not possible	Operation is not possible. For example, moving a rule up one position when it is already at the top.
15	Operation denied	For example, Admin not allowed to delete own account, Running a command that is not allowed on a passive device.
16	Unauthorized	The API role does not have access rights to run this query.
17	Invalid command	Invalid command or parameters.
18	Malformed command	The XML is malformed.
19-20	Success	Command completed successfully.
21	Internal error	Check with technical support when seeing these errors.
22	Session timed out	The session for this query timed out.

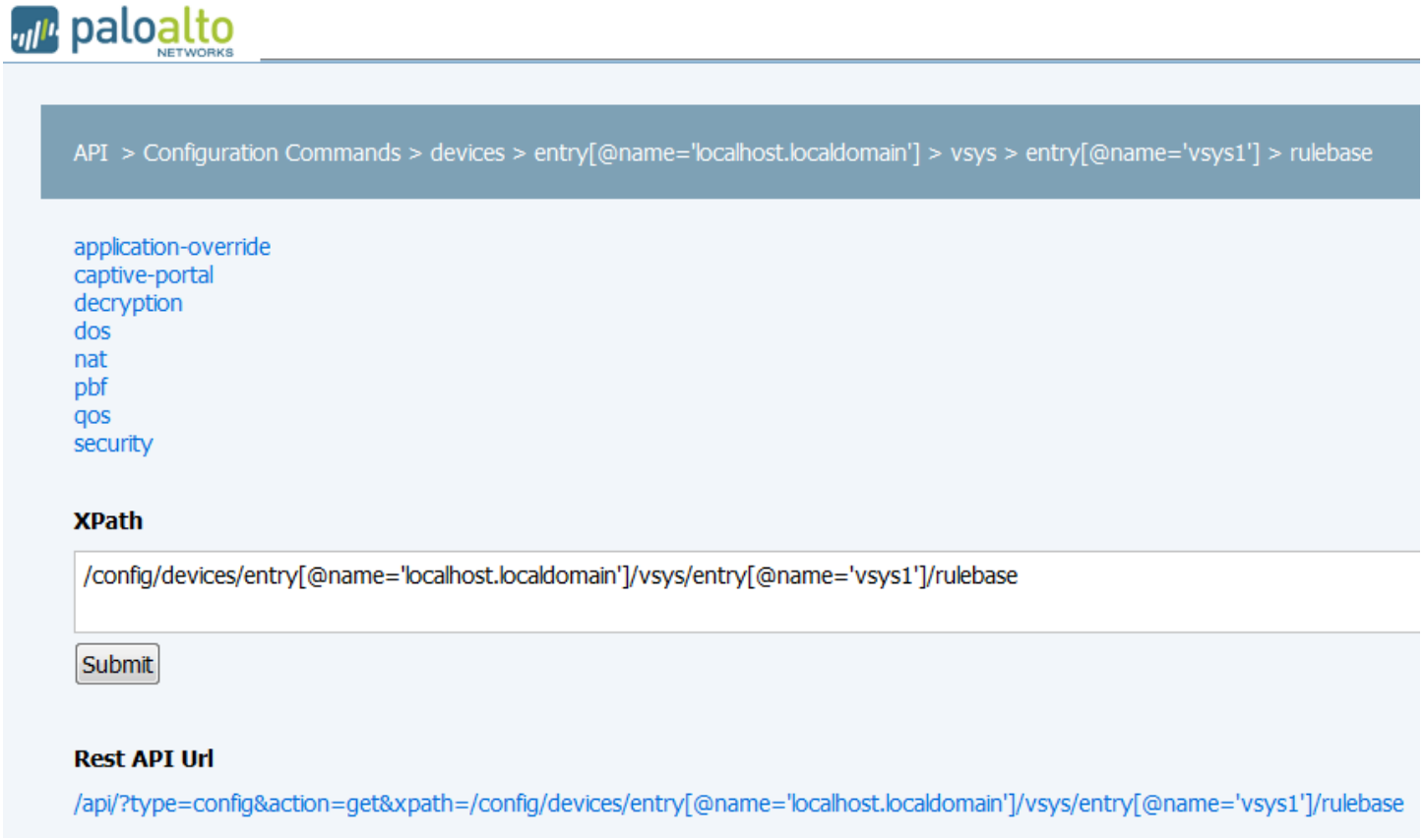
6 API Browser

The API browser is available at **http(s)://hostname/api.** You need to be logged in to the device's WebUI to be able to view the API browser.

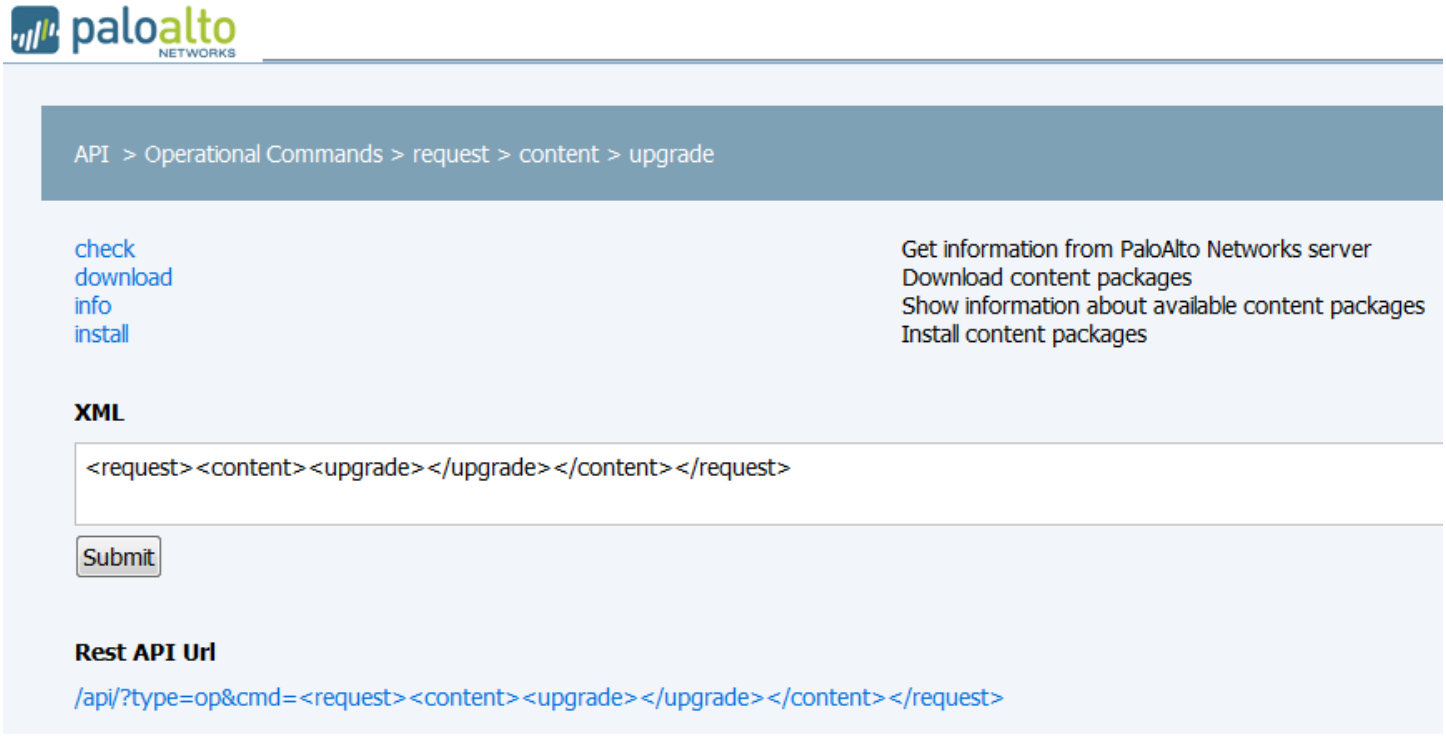
You can use API browser to navigate different API requests that are available for use. For configuration commands, you can navigate to any path and view the corresponding xpath and API URL on the browser.



For Configuration commands, you can navigate to a specific command to see its xpath.



For Operational commands and Commit commands, you can navigate to a specific command to see the xml body to use for the *cmd* parameter.



For reports, you can view the report names for all the supported dynamic and predefined reports.

7 Frequently Asked Questions

1 *How do I discover the xpath for the configuration object I am interested in?*
Use the API browser at http(s)://hostname/api to see all the available configuration commands along with their xpaths shown on the bottom of the screen. Alternatively, you can use the XML response for API request to show the entire running config, to navigate and discover the xpath for any element in the config. http(s)://hostname/api/?type=config&action=show

2 *How do I build an xpath when there are multiple entries in a node in the config path to the element I am interested in?*
When there are multiple entries in any node in the path, you can specify the entry you are interested via the name of the entry, like so entry[@name='value']. For instance, the xpath to the address objects in vsys1 is /config/devices/entry/vsys/entry[@name='vsys1']/address

3 *How do I build the the xml body for the cmd parameter to be used in Operational and Commit commands?*
Use the API browser to navigate to a specific command and view the xml body to be used with the cmd parameter.

4 *Do I need to use URL/percent-encoding?*
You need to use URL encoding when using tools like cURL or wget. When using the browser, most browsers automatically do the URL encoding.

5 *What if my API request is too long?*
When the API request is 2K or longer, you should use HTTP POST instead of GET to avoid errors from the webserver. If you are using scripts and not a browser, you can use cURL or wget. Examples usages are shown below. Refer to their respective man pages for additional usage information.

Wget provides the --post-data and the --post-file options to do a HTTP POST.
> wget --post-data "*query-parameters*" http(s)://hostname/api/?query-parameters
> wget --post-file *input-filename* http(s)://hostname/api/?more-query-parameters, where the *input-filename* contains additional query paramaters for the API request.

Curl provides the --data options to do a HTTP POST.
> curl --data "*query-parameters*" http(s)://hostname/api/?*more-query-parameters*
> curl --data @input-filename http(s)://hostname/api/?*more-query-parameters*, where input-filename contains additional query parameters for the API request.

6 *How do I retrieve Panorama-pushed shared configuration from a firewall device?*
Use the Config Get API with xpath=/config/panorama. One example of this is if you want to retrieve pre- and post-rules from security policy.

7 *What are the xpaths and API queries for some sample configuration objects on the Firewall and Panorama?*
Creating a new URL filtering profile with a block action for www.badsite.com:
http(s)://hostname/api/?type=config&action=set&xpath=/config/devices/entry/vsys/entry[@name='vsys1']/profiles/url-filtering/entry[@name='xml test']&element=<description>xml api test</description><dynamic-url>yes</dynamic-url><action>block</action><block-list><member>www.badsite.com</member></block-list>

Adding a url to block list in an existing url profile:
http(s)://hostname/api/?type=config&action=set&xpath=/config/devices/entry/vsys/entry[@name='vsys1']/profiles/url-filtering/entry[@name='xml test']/block-list&element=<member>www.badsite.com</member>

Creating a new custom URL category:
http(s)://hostname/api/?type=config&action=set&xpath=/config/devices/entry/vsys/entry[@name='vsys1']/profiles/custom-url-category/entry[@name='xmltest urlcat']&element=<description>testing xml api</description><list><member>www.somesite.com</member></list>

Adding a URL to a custom URL category:
http(s)://hostname/api/?type=config&action=set&xpath=/config/devices/entry/vsys/entry[@name='vsys1']/profiles/custom-url-category/entry[@name='xmltest urlcat']/list&element=<member>www.somesite.com</member>

Adding an address object:
http(s)://hostname/api/?type=config&action=set&xpath=/config/devices/entry/vsys/entry[@name='vsys1']/address-group/entry[@name='xmltest addr']&element=<static> <ip-netmask>1.2.3.4/32</ip-netmask></static><description>xml testing</description>

8 *How to pull Application and Threat Content information from the Firewall?*
Get a list of all the applications:
http(s)://hostname/api/?type=config&action=get&xpath=/config/predefined/application
Get a list of all the vulnerabilities:
http(s)://hostname/api/?type=config&action=get&xpath=/config/predefined/threats/vulnerability
Get information on a specific vulnerability by Threat-ID:
http(s)://hostname/api/?type=config&action=get&xpath=/config/predefined/threats/vulnerability/entry[@name='30003']