PAN-OS® and Panorama™
XML API Usage Guide

Version 7.1
Contact Information

Corporate Headquarters:
Palo Alto Networks
4401 Great America Parkway
Santa Clara, CA 95054
https://www.paloaltonetworks.com/company/contact-support

About this Guide

This API reference guide covers the features and usage of the PAN-OS XML API. For additional information, refer to the following resources:

- For information on how to configure other components in the Palo Alto Networks Next-Generation Security Platform, go to the Technical Documentation portal: https://docs.paloaltonetworks.com or search the documentation.
- For access to the knowledge base and community forums, refer to https://live.paloaltonetworks.com.
- For contacting support, for information on support programs, to manage your account or devices, or to open a support case, refer to https://www.paloaltonetworks.com/support/tabs/overview.html.
- For the most current PAN-OS and Panorama 7.1 release notes, go to https://docs.paloaltonetworks.com/pan-os/7-1/pan-os-release-notes

To provide feedback on the documentation, please write to us at: documentation@paloaltonetworks.com.
PAN-OS XML API Request Types

The following topics provide common request examples that you can use to better understand the PAN-OS XML API.

- PAN-OS XML API Request Types and Actions
- Asynchronous and Synchronous Requests to the PAN-OS XML API
- Configuration (API)
- Commit Configuration (API)
- Run Operational Mode Commands (API)
- Get Reports (API)
- Export Files (API)
- Import Files (API)
- Retrieve Logs (API)
- Apply User-ID Mapping and Populate Dynamic Address Groups (API)
- Get Version Info (API)
PAN-OS XML API Request Types and Actions

Use PAN-OS XML API to run various requests depending on the request type that you specify:

- Request Types
- Configuration Actions

Request Types

You can currently use the following request types:

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type=keygen</td>
<td>Generate API keys for authentication.</td>
</tr>
<tr>
<td>type=config</td>
<td>Modify the configuration.</td>
</tr>
<tr>
<td>type=commit</td>
<td>Commit firewall configuration, including partial commits.</td>
</tr>
<tr>
<td>type=op</td>
<td>Perform operational mode commands, including checking system status and validating configurations.</td>
</tr>
<tr>
<td>type=report</td>
<td>Get reports, including predefined, dynamic, and custom reports.</td>
</tr>
<tr>
<td>type=log</td>
<td>Get logs, including traffic, threat, and event logs.</td>
</tr>
<tr>
<td>type=import</td>
<td>Import files including configurations and certificates.</td>
</tr>
<tr>
<td>type=export</td>
<td>Export files including packet captures, certificates, and keys.</td>
</tr>
<tr>
<td>type=user-id</td>
<td>Update User-ID mappings.</td>
</tr>
<tr>
<td>type=version</td>
<td>Show the PAN-OS version, serial number, and model number.</td>
</tr>
</tbody>
</table>

Configuration Actions

In addition to the request type that you specify, use available actions to modify or read configurations using type=config:

- Actions for Modifying a Configuration
- Actions for Reading a Configuration
Actions for Modifying a Configuration

<table>
<thead>
<tr>
<th>Configuration Action Type</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set candidate configuration</td>
<td>action=set</td>
</tr>
<tr>
<td>Edit candidate configuration</td>
<td>action=edit</td>
</tr>
<tr>
<td>Delete candidate object</td>
<td>action=delete</td>
</tr>
<tr>
<td>Rename a configuration object</td>
<td>action=rename</td>
</tr>
<tr>
<td>Clone a configuration object</td>
<td>action=clone</td>
</tr>
<tr>
<td>Move a configuration object</td>
<td>action=move</td>
</tr>
<tr>
<td>Override a template setting</td>
<td>action=override</td>
</tr>
<tr>
<td>Move multiple objects in a device group or virtual system</td>
<td>action=multi-move</td>
</tr>
<tr>
<td>Clone multiple objects in a device group or virtual system</td>
<td>action=multi-clone</td>
</tr>
<tr>
<td>Show available subnode values and XPaths for a given XPath.</td>
<td>action=complete</td>
</tr>
</tbody>
</table>

Set and edit actions differ in two important ways:
- Set actions add, update, or merge configuration nodes, while edit actions replace configuration nodes.
- Set actions are non-destructive and are only additive, while edit actions can be destructive.

Actions for Reading a Configuration

<table>
<thead>
<tr>
<th>Configuration Action Type</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get active configuration</td>
<td>action=show</td>
</tr>
<tr>
<td>Get candidate configuration</td>
<td>action=get</td>
</tr>
</tbody>
</table>

Show and get actions differ in three important ways:
- Show actions retrieve the active configuration, while get actions retrieve the candidate, uncommitted configuration.
- Show actions only work when the provided XPath specifies a single node. Get actions work with single and multiple nodes.
- Show actions can use relative XPath, while get actions require absolute XPath.
Asynchronous and Synchronous Requests to the PAN-OS XML API

Most PAN-OS XML API requests are synchronous, meaning the response immediately provides the requested data. For example, when you Make Your First API Call and request system information, the API response is immediate and contains information such as the IP address, hostname, and model of your firewall.

However, there are some Request Types that require more time to process and are asynchronous, meaning they require more than one request to get final results. These API requests include the following:

- Get Reports (API)
- Retrieve Logs (API)
- Export Technical Support Data
- Some requests to Run Operational Mode Commands (API), including download, upgrade, and installation requests

With asynchronous requests, you first initiate a request. The API responds with a job ID while it processes your request. In your subsequent requests, you use this job ID to check on the results of your original request.
Configuration (API)

The requests examples in these topics illustrate how you can use the PAN-OS XML API to configure your firewall.

- Get Active Configuration
- Get Candidate Configuration
- Set Configuration
- Edit Configuration
- Delete Configuration
- Rename Configuration
- Clone Configuration
- Move Configuration
- Override Configuration
- Multi-Move or Multi-Clone Configuration
- View Configuration Node Values for XPath

Get Active Configuration

- Use XPath to Get Active Configuration
- Use XPath to Get ARP Information
Use XPath to Get Active Configuration

Use `action=show` with no additional parameters to retrieve the entire active configuration.

**Use XPath to Get Active Configuration**

**Step 1** Use the `xpath` parameter to target a specific portion of the configuration. For example, to retrieve just the security rulebase:

```
https://firewall/api/?type=config&action=show&key=apikey&xpath=/config/devices/entry/vsys/entry/rulebase/security
```

*There is no trailing backslash character at the end of the XPath.*

**Step 2** Confirm that the XML response for the query looks similar to the following (truncated):

```
<response status="success">
  <result>
    <security>
      <rules>
        <entry name="IT DNS Services">
          <profile-setting>
            <group>
              <member>best-practice</member>
            </group>
          </profile-setting>
          <to>
            <member>untrust</member>
          </to>
          <from>
            <member>trust</member>
          </from>
          <source>
            <member>any</member>
          </source>
          <destination>
            <member>Data Center</member>
          </destination>
          <source-user>
            <member>any</member>
          </source-user>
          <category>
            <member>any</member>
          </category>
          <application>
            <member>dns</member>
          </application>
          <service>
            <member>application-default</member>
          </service>
          <hip-profiles>
            <member>any</member>
          </hip-profiles>
          <action>allow</action>
          <tag>
            <member>Best Practice</member>
          </tag>
          <log-start>no</log-start>
          <log-setting>default</log-setting>
        </entry>
        ...
      </rules>
    </security>
  </result>
</response>
```
Use XPath to Get ARP Information

Step 1  Use the following request to retrieve ARP information:

https://firewall//api/?type=op&command=<show><arp><entry name='all'/></arp></show>

Step 2  Confirm that the XML response for the query looks like the following (truncated):

<response status="success">
  <result>
    <max>3000</max>
    <total>16</total>
    <timeout>1800</timeout>
    <dp>dp0</dp>
    <entries>
      <entry>
        <status>c</status>
        <ip>10.47.0.1</ip>
        <mac>00:1b:17:00:2f:13</mac>
        <ttl>1743</ttl>
        <interface>ethernet1/1</interface>
        <port>ethernet1/1</port>
      </entry>
      <entry>
        <status>c</status>
        <ip>10.47.0.10</ip>
        <mac>00:50:56:93:68:6f</mac>
        <ttl>386</ttl>
        <interface>ethernet1/1</interface>
        <port>ethernet1/1</port>
      </entry>
    </entries>
  </result>
</response>
Get Candidate Configuration

Get the candidate configuration from a firewall by specifying the portion of the configuration to get. Use the following request, including the `xpath` parameter to specify the portion of the configuration to get.

https://firewall/api/?type=config&action=get&xpath=path-to-config-node

<table>
<thead>
<tr>
<th>Configuration Node</th>
<th>API Request</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address objects in a VSYS.</td>
<td><a href="https://firewall/api/?type=config&amp;action=get&amp;xpath=/config/devices/entry/vsys/entry%5B@name='vsys1'%5D/address">https://firewall/api/?type=config&amp;action=get&amp;xpath=/config/devices/entry/vsys/entry[@name='vsys1']/address</a></td>
</tr>
<tr>
<td>Pre-rules pushed from Panorama.</td>
<td><a href="https://firewall/api/?type=config&amp;action=get&amp;xpath=/config/panorama/vsys/entry%5B@name='vsys'%5D/pre-rulebase/security">https://firewall/api/?type=config&amp;action=get&amp;xpath=/config/panorama/vsys/entry[@name='vsys']/pre-rulebase/security</a></td>
</tr>
<tr>
<td>Detailed information on Applications and Threats from the firewall.</td>
<td><a href="https://firewall/api/?type=config&amp;action=get&amp;xpath=/config/predefined/threats/vulnerability/entry%5B@name='30003">https://firewall/api/?type=config&amp;action=get&amp;xpath=/config/predefined/threats/vulnerability/entry[@name='30003</a>']</td>
</tr>
<tr>
<td>Full list of all applications.</td>
<td><a href="https://firewall/api/?type=config&amp;action=get&amp;xpath=/config/predefined/application">https://firewall/api/?type=config&amp;action=get&amp;xpath=/config/predefined/application</a></td>
</tr>
<tr>
<td>Details on the specific application.</td>
<td><a href="https://firewall/api/?type=config&amp;action=get&amp;xpath=/config/predefined/application/entry%5B@name='hotmail">https://firewall/api/?type=config&amp;action=get&amp;xpath=/config/predefined/application/entry[@name='hotmail</a>']</td>
</tr>
</tbody>
</table>

The response looks similar to the following:

```xml
<response status="success" code="19">
  <result total-count="1" count="1">
    <address admin="name" dirtyId="8" time="2015/10/20 15:32:36">
      <entry name="testobject">
        <ip-netmask>2.2.2.2</ip-netmask>
      </entry>
      <entry name="test1">
        <ip-netmask>1.1.1.1</ip-netmask>
      </entry>
      ...
    </address>
  </result>
</response>
```
Set Configuration

Use **action=set** to 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.

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**).

---

**Set Configuration**

**Step 1** Create a new rule called rule1 in the security policy:

```
https://firewall/api/?type=config&action=set&key=keyvalue&xpath=xpath-value&element=element-value
```

where the xpath-value is:

```
/config/devices/entry/vsys/entry/rulebase/security/rules/entry[@name='rule1']
```

and the 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>
```

**Step 2** Use the response from the config show API request to create the XML body for the element.

```
https://firewall/api/?type=config&action=show
```

**Step 3** 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:

```
https://firewall/api/?type=config&action=set&xpath=/config/devices/entry/vsys/entry[@name='vsys1']/address-group/entry[@name='test']&element=<static><member>abc</member></static>
```
Edit Configuration

Use `action=edit` to 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`).

### Edit Configuration

**Step 1** Replace the application(s) currently used in a rule `rule1` with a new application:

```
https://firewall/api/?type=config&action=edit&key=apikey&xpath=xpath-value&element=element-value
```

where
```
xpath=/config/devices/entry/vsys/entry/rulebase/security/rules/entry[@name='rule1']/application
```
```
extelement=

where
```
xpath=/config/devices/entry/vsys/entry/rulebase/security/rules/entry[@name='rule1']/application
```
```
extelement=
```

**Step 2** Use the response from the config show API request to create the XML body for the element.

```
https://firewall/api/?type=config&action=show
```

**Step 3** Optionally replace all members in a node with a new set of members using 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:

```
https://firewall/api/?type=config&action=edit&xpath=/config/devices/entry/vsys/entry[vsys1]/address-group/entry[test]@name='test']/<entry name='test'><member>abc</member><member>xyz</member></entry></static>
```

Delete Configuration

Use `action=delete` to delete an object at a specified location in the configuration. Use the `xpath` parameter to specify the location of the object to be deleted.

### Delete Configuration

- **Delete a rule named `rule1` in the security policy:**

```
https://firewall/api/?type=config&action=delete&xpath=/config/devices/entry/vsys/entry/rulebase/security/rules/entry[@name='rule1']
```

- **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 following `xpath`:**

```
https://firewall/api/?type=config&action=delete&xpath=/config/devices/entry/vsys/entry[vsys1]/address-group/entry[=test]/static/member[text()='abc']
```
Rename Configuration

Use `action=rename` to 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.

---

**Step 1**

Use the following API query to rename an address object called `old_address` to `new_address`:

```
https://firewall/api/?type=config&action=rename&xpath=/config/devices/entry/vsys/entry[@name='vsys1']/address/entry[@name='old_address']&newname=new_address
```

**Step 2**

Confirm that the XML response for the request looks like the following:

```
<response status="success" code="20">msg>command succeeded</msg></response>
```

Clone Configuration

Use `action=clone` to 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 new name for the cloned object.

---

**Step 1**

Use the following API query to clone a security policy called `rule1` to `rule2`:

```
https://firewall/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
```

**Step 2**

Confirm that the XML response for the request looks like the following:

```
<response status="success" name="rule2"/>
```

A corresponding success log is recorded in the Configuration 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
```
Move Configuration

Use `action=move` to 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 path.

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

Move Configuration

Step 1  Use the following API query to move a security policy called `rule1` to come after `rule2`:

```
https://firewall/api/?type=config&action=move&xpath=/config/devices/entry/vsys/entry[@name='vsys1']/rulebase/security/rules/entry[@name='rule1']&where=after&dst=rule2
```

Step 2  Confirm that the XML response for the request looks like the following:

```
<response status="success" code="20"><msg>command succeeded</msg></response>
```

Override Configuration

Use `action=override` to override a setting that was pushed to a firewall from a template. Use the `xpath` parameter to specify the location of the object to override.

Override Configuration

Step 1  Override the SNMP Trap profile configuration settings that were pushed to the firewall using a template:

```
https://firewall/api/?type=config&action=override&xpath=/config/shared/log-settings/snmtrapprofile/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>
```

Step 2  Confirm that the XML response for the request looks like the following:

```
<response status="success" code="20"><msg>command succeeded</msg></response>
```
Multi-Move or Multi-Clone Configuration

Use the action=multi-move and action=multi-clone actions 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.

### Multi-Move or Multi-Clone Configuration

- **Move addresses** `addr1`, `addr2`, to device group `norcal` from device group `socal`:
  
  ```
  https://firewall/api/?type=config&action=multimove&xpath=/config/devices/entry[@name='localhost.localdomain']/devicegroup/entry[@name='norcal']/address&element=<selected-list><source xpath="/config/devices/entry[@name='localhost.localdomain']/devicegroup/entry[@name='socal']/address"><member>addr1</member><member>addr2</member></source></selected-list><all-errors>no</all-errors>
  ```

- **Clone addresses** `addr1`, `addr2`, to device group `norcal` from device group `socal`:
  
  ```
  https://firewall/api/?type=config&action=multiclone&xpath=/config/devices/entry[@name='localhost.localdomain']/devicegroup/entry[@name='norcal']/address&element=<selected-list><source xpath="/config/devices/entry[@name='localhost.localdomain']/devicegroup/entry[@name='socal']/address"><member>addr1</member><member>addr2</member></source></selected-list><all-errors>no</all-errors>
  ```

### View Configuration Node Values for XPath

Use action=complete action along with an XPath to see possible values that are available with the XPath node.

### View Configuration Node Values for XPath

**Step 1** View the possible values, such as network interfaces, for multi-vsys firewalls, use the following command:

```
https://firewall/api/?type=config&action=complete&xpath=/api/?type=config&action=get&xpath=/config/devices/entry[@name='localhost.localdomain']/vsys&key=apikey
```

**Step 2** Confirm that the XML response for the request looks like the following:

```
<response status="success" code="19">
  <completions>
    <completion value="vsys1" vxpath="/config/devices/entry[@name='localhost.localdomain']/vsys/entry[@name='vsys1']" current="yes" help-string="vsys1"/>
  </completions>
</response>
```
Commit Configuration (API)

You can use the commit API request to commit a candidate configuration to a firewall.

- Commit
- Commit-all

Commit

Use the API Browser to find different options available for use with force and partial commits. Replace the body element in the cmd parameter with the XML element for the corresponding commit operation.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Use one of the following requests to commit a configuration:</td>
</tr>
<tr>
<td></td>
<td>- Commit—</td>
</tr>
<tr>
<td></td>
<td><a href="https://firewall/api/?type=commit&amp;cmd=">https://firewall/api/?type=commit&amp;cmd=</a>&lt;commit&gt;&lt;/commit&gt;</td>
</tr>
<tr>
<td></td>
<td>- Force Commit—</td>
</tr>
<tr>
<td></td>
<td><a href="https://firewall/api/?type=commit&amp;cmd=">https://firewall/api/?type=commit&amp;cmd=</a>&lt;commit&gt;&lt;force&gt;&lt;/force&gt;&lt;/commit&gt;</td>
</tr>
<tr>
<td></td>
<td>- Partial commit—</td>
</tr>
<tr>
<td></td>
<td><a href="https://firewall/api/?type=commit&amp;cmd=">https://firewall/api/?type=commit&amp;cmd=</a>&lt;commit&gt;&lt;/commit&gt;</td>
</tr>
<tr>
<td>2</td>
<td>Confirm that the XML response for the request looks like one of the following:</td>
</tr>
<tr>
<td></td>
<td>- No pending changes to commit:</td>
</tr>
<tr>
<td></td>
<td>&lt;response status=&quot;success&quot; code=&quot;19&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;msg&gt;There are no changes to commit.&lt;/msg&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/response&gt;</td>
</tr>
<tr>
<td></td>
<td>- Pending changes:</td>
</tr>
<tr>
<td></td>
<td>&lt;response status=&quot;success&quot; code=&quot;19&quot;&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;result&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;msg&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;line&gt;Commit job enqueued with jobid 4&lt;/line&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/msg&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;job&gt;4&lt;/job&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/result&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/response&gt;</td>
</tr>
<tr>
<td>3</td>
<td>Query the status of the job using the job ID:</td>
</tr>
<tr>
<td></td>
<td><a href="https://firewall/api/?type=op&amp;cmd=">https://firewall/api/?type=op&amp;cmd=</a>&lt;show&gt;&lt;jobs&gt;&lt;id&gt;4&lt;/id&gt;&lt;/jobs&gt;&lt;/show&gt;</td>
</tr>
</tbody>
</table>
Commit Configuration (API)

Commit (Continued)

Step 4  Confirm that the XML response for the request looks like the following:

```
<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>
    </job>
  </result>
</response>
```

Commit-all

To centrally manage firewalls from Panorama, 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.

<table>
<thead>
<tr>
<th>Commit Type</th>
<th>API Request</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-commit policy validation.</td>
<td><a href="https://panorama/api/?type=commit&amp;action=all&amp;cmd=">https://panorama/api/?type=commit&amp;action=all&amp;cmd=</a>&lt;commit-all&gt;&lt;shared-policy&gt;&lt;validate-only/&gt;&lt;/validate-only&gt;</td>
</tr>
<tr>
<td>Device group commit.</td>
<td><a href="https://panorama/api/?type=commit&amp;action=all&amp;cmd=">https://panorama/api/?type=commit&amp;action=all&amp;cmd=</a>&lt;commit-all&gt;&lt;shared-policy&gt;&lt;device-group&gt;&lt;entry name=&quot;device-group-name&quot;/&gt;&lt;/device-group&gt;&lt;/shared-policy&gt;</td>
</tr>
<tr>
<td>VSYS commit.</td>
<td><a href="https://panorama/api/?type=commit&amp;action=all&amp;cmd=">https://panorama/api/?type=commit&amp;action=all&amp;cmd=</a>&lt;commit-all&gt;&lt;shared-policy&gt;&lt;device-group&gt;&lt;entry name=&quot;device-group-name&quot;/&gt;&lt;devices&gt;&lt;entry name=&quot;serial_number&quot;&gt;vsys-name&lt;/entry&gt;&lt;/devices&gt;&lt;/device-group&gt;&lt;/shared-policy&gt;</td>
</tr>
<tr>
<td>Specific firewall commit.</td>
<td><a href="https://panorama/api/?type=commit&amp;action=all&amp;cmd=">https://panorama/api/?type=commit&amp;action=all&amp;cmd=</a>&lt;commit-all&gt;&lt;shared-policy&gt;&lt;device-group&gt;&lt;entry name=&quot;device-group-name&quot;/&gt;&lt;devices&gt;&lt;entry name=&quot;serial_number&quot;&gt;vsys-name&lt;/entry&gt;&lt;/devices&gt;&lt;/device-group&gt;&lt;/shared-policy&gt;</td>
</tr>
</tbody>
</table>

Use the API Browser to find other options available for granular commit operations on Panorama. In the cmd parameter, you must replace the XML element for the corresponding commit-all operation.
Run Operational Mode Commands (API)

Use any of the operational mode commands available on the command line interface with the following API request:

https://firewall/api/?type=op&cmd=xml-body

Use the API Browser to explore operational mode commands and a complete listing of all the options available for the xml-body and their corresponding operation.

<table>
<thead>
<tr>
<th>Operational Command</th>
<th>API Request</th>
</tr>
</thead>
<tbody>
<tr>
<td>System restart.</td>
<td><a href="https://firewall/api/?type=op&amp;cmd=">https://firewall/api/?type=op&amp;cmd=</a>&lt;request&gt;&lt;restart&gt;&lt;system&gt;&lt;/system&gt;&lt;/restart&gt;</td>
</tr>
<tr>
<td>System software version installation.</td>
<td><a href="https://firewall/api/?type=op&amp;cmd=">https://firewall/api/?type=op&amp;cmd=</a>&lt;request&gt;&lt;system&gt;&lt;software&gt;&lt;install&gt;&lt;version&gt;version_number&lt;/version&gt;&lt;/install&gt;&lt;software&gt;&lt;/system&gt;&lt;/request&gt;</td>
</tr>
<tr>
<td>Full configuration validation.</td>
<td><a href="https://firewall/api/?type=op&amp;cmd=">https://firewall/api/?type=op&amp;cmd=</a>&lt;validate&gt;&lt;full&gt;&lt;/full&gt;&lt;/validate&gt;</td>
</tr>
<tr>
<td>Configuration saving.</td>
<td><a href="https://firewall/api/?type=op&amp;cmd=">https://firewall/api/?type=op&amp;cmd=</a>&lt;save&gt;&lt;config&gt;&lt;to&gt;filename&lt;/to&gt;&lt;/config&gt;&lt;/save&gt;</td>
</tr>
<tr>
<td>Configuration loading.</td>
<td><a href="https://firewall/api/?type=op&amp;cmd=">https://firewall/api/?type=op&amp;cmd=</a>&lt;load&gt;&lt;config&gt;&lt;from&gt;filename&lt;/from&gt;&lt;/config&gt;&lt;/load&gt;</td>
</tr>
</tbody>
</table>

Some requests operational mode commands, including download, upgrade, and installation requests, are asynchronous, meaning they require more than one request to get final results. Learn more about Asynchronous and Synchronous Requests to the PAN-OS XML API.
Get Reports (API)

The XML API provides a way to quickly pull the results of any report defined in the system using the `type=report` parameter.

You can access three kinds of reports:
- 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:

```
https://firewall/api/?type=report&reporttype=dynamic|predefined|custom&reportname=name
```

When you request a report, the API responds asynchronously with a job ID, which you can use to retrieve the reports. Learn more about Asynchronous and Synchronous Requests to the PAN-OS XML API.

- Dynamic Reports
- Predefined Reports
- Custom Reports

Dynamic Reports

You can use the API to view a number of dynamic reports, such as `top-applications-summary`, `top-blocked-url-summary`, and `top-spyware-threats-summary`. For dynamic reports, provide either a specific period using the `period` or a time frame using `starttime` and `endtime` options (use a + instead of a space between the date and timestamp). Use `topn` to determine the number of rows.

<table>
<thead>
<tr>
<th>Dynamic Report Type</th>
<th>API Request</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full dynamic report list.</td>
<td><code>https://firewall/api/?type=report&amp;reporttype=dynamic</code></td>
</tr>
<tr>
<td>Last 60 seconds.</td>
<td><code>https://firewall/api/?type=report&amp;reporttype=dynamic&amp;reportname=top-app-summary&amp;period=last-60-seconds&amp;topn=5</code></td>
</tr>
<tr>
<td>Last hour.</td>
<td><code>https://firewall/api/?type=report&amp;reporttype=dynamic&amp;reportname=top-app-summary&amp;period=last-hour&amp;topn=5</code></td>
</tr>
<tr>
<td>Last 12 hours.</td>
<td><code>https://firewall/api/?type=report&amp;reporttype=dynamic&amp;reportname=top-app-summary&amp;period=last-12-hrs&amp;topn=5</code></td>
</tr>
<tr>
<td>Last calendar day.</td>
<td><code>https://firewall/api/?type=report&amp;reporttype=dynamic&amp;reportname=top-app-summary&amp;period=last-calendar-day&amp;topn=5</code></td>
</tr>
<tr>
<td>Last 7 days</td>
<td><code>https://firewall/api/?type=report&amp;reporttype=dynamic&amp;reportname=top-app-summary&amp;period=last-7-days&amp;topn=5</code></td>
</tr>
</tbody>
</table>
### Predefined Reports

Predefined reports always return data for the last 24-hour period. You can also get this list by following the link for predefined reports, such as `top-applications`, `top-attackers`, and `bandwidth-trend` on the API browser.

<table>
<thead>
<tr>
<th>Dynamic Report Type</th>
<th>API Request</th>
</tr>
</thead>
<tbody>
<tr>
<td>Last 7 calendar days</td>
<td><a href="https://firewall/api/?type=report&amp;reporttype=dynamic&amp;reportname=top-app-summary&amp;period=last-hour&amp;topn=5">https://firewall/api/?type=report&amp;reporttype=dynamic&amp;reportname=top-app-summary&amp;period=last-hour&amp;topn=5</a></td>
</tr>
<tr>
<td>Last calendar week.</td>
<td><a href="https://firewall/api/?type=report&amp;reporttype=dynamic&amp;reportname=top-app-summary&amp;period=last-calendar-week&amp;topn=5">https://firewall/api/?type=report&amp;reporttype=dynamic&amp;reportname=top-app-summary&amp;period=last-calendar-week&amp;topn=5</a></td>
</tr>
<tr>
<td>Last 30 days</td>
<td><a href="https://firewall/api/?type=report&amp;reporttype=dynamic&amp;reportname=top-app-summary&amp;period=last-30-days&amp;topn=5">https://firewall/api/?type=report&amp;reporttype=dynamic&amp;reportname=top-app-summary&amp;period=last-30-days&amp;topn=5</a></td>
</tr>
</tbody>
</table>

### Custom Reports

For custom reports, the selection criteria, such as time frame, group-by, and sort-by are part of the report definition. The API returns any shared custom reports. Note that quotes are not required around the report name 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.

### Get a Custom Dynamic Report

**Step 1**  
Retrieve the report definition from the configuration:

```
https://firewall/api/?type=config&action=get&xpath=/config/devices/entry/vsys/entry[@name='vsys1']/reports/entry[@name='report-abc']
```
Get a Custom Dynamic Report (Continued)

Step 2  Create a job to retrieve a dynamic report using `reporttype=dynamic`, `reportname=custom-dynamic-report`, and `cmd=report-definition` where `report-definition` is the XML definition retrieved in the previous query:

```
https://firewall/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>
```

The response includes the job ID you can use to view the results:

```
<response status="success">
  <result>
    <msg>
      <line>Report job enqueued with jobid 6</line>
    </msg>
    <job>6</job>
  </result>
</response>
```

Step 3  View the dynamic report:

```
https://firewall/api/?type=report&action=get&job-id=jobid
```
Export Files (API)

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
- **Technical support data**—category=tech-support
- **Device State**—category=device-state

Use cURL tools to export the file from the firewall and save locally with a local file name:

```bash
curl -o filename "https://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.

- ▲ Export Packet Captures
- ▲ Export Certificates and Keys
- ▲ Export Technical Support Data

Export Packet Captures

You can export packet captures from the firewall by specifying the PCAP type using the category parameter:

- ▲ Export Application PCAPS
- ▲ Export Threat, Filter, and Data Filtering PCAPs

Export Application PCAPS

Application PCAPs are organized by a directory/filename structure where the directory is a date in yyyyMMdd format. Filenames for application pcaps use a SourceIP-SourcePort-DestinationIP-DestinationPort-SessionID.pcap format.

<table>
<thead>
<tr>
<th>Application PCAP Type</th>
<th>API Request</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application PCAP directory list.</td>
<td><a href="https://firewall/api/?type=export&amp;category=application-pcap">https://firewall/api/?type=export&amp;category=application-pcap</a></td>
</tr>
<tr>
<td>List of files under a directory using the from parameter to indicate date.</td>
<td><a href="https://firewall/api/?type=export&amp;category=application-pcap&amp;from=yyyyMMdd">https://firewall/api/?type=export&amp;category=application-pcap&amp;from=yyyyMMdd</a></td>
</tr>
</tbody>
</table>
Export Threat, Filter, and Data Filtering PCAPs

To export threat PCAPs, you need to provide the 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 a `pcapID.pcap` format.

<table>
<thead>
<tr>
<th>PCAP Type</th>
<th>API Request</th>
</tr>
</thead>
<tbody>
<tr>
<td>Threat PCAP using PCAP ID and search</td>
<td><code>https://firewall/api/?type=export&amp;category=threat-pcap&amp;pcap-id=id&amp;search-time=yyyy/mm/dd hr:min:sec</code></td>
</tr>
<tr>
<td>List of filtered PCAPs</td>
<td><code>https://firewall/api/?type=export&amp;category=filters-pcap</code></td>
</tr>
<tr>
<td>Specific filtered PCAP file</td>
<td><code>https://firewall/api/?type=export&amp;category=filters-pcap&amp;from=filename</code></td>
</tr>
<tr>
<td>List of data filtering PCAP file names</td>
<td><code>https://firewall/api/?type=export&amp;category=dlp-pcap&amp;dlp-password=password</code></td>
</tr>
<tr>
<td>Specific data filtering PCAP file</td>
<td><code>https://firewall/api/?type=export&amp;category=dlp-pcap&amp;dlp-password=password&amp;from=filename&amp;to=localfile</code></td>
</tr>
</tbody>
</table>

Export Certificates and Keys

To export certificates and keys, specify query parameters `certificate-name`, `format`, and `passphrase`:

```
https://firewall/api/?type=export&category=certificate&certificate-name=certificate_name&passphrase=passphrase&format=pkcs12 | pem&include-key=yes | no&vsys=vsys | omit this parameter to import it into a shared location
```

- `certificate-name`—name of the certificate object on the firewall
- `passphrase`—required when including the certificate key
- `format`—certificate format, pkcs12 or pem
- `include-key`—yes or no parameter to include or exclude the key
- `vsys`—virtual system where the certificate object is used. Ignore this parameter if the certificate is a shared object.
Export Technical Support Data

Debug log data sizes are large, so the API uses an asynchronous job scheduling approach to retrieve technical support data. Learn more about Asynchronous and Synchronous Requests to the PAN-OS XML API. The values for the action parameter are:

- **action=null**—When an action parameter is not specified, the system creates a new job to retrieve tech support data. The initial query creates a job ID that you can then use to check on the status of the job, retrieve results, or delete the job.
- **action=status**—Check the status of the job. This returns an XML response with a status element; when the status text data is **FIN** the job is completed and the tech support file can be retrieved. Example:
  ```
  https://firewall/api/?type=export&category=tech-support&action=status&job-id=299
  ```
- **action=get**—Retrieve the tech support file as an attachment. The response contains a `application/octet-stream` content-type and a content-disposition header with a suggested filename; for example:
  ```
  Content-Type: application/octet-stream
  Content-Length: 19658186
  Content-Description: File Transfer
  Content-Transfer-Encoding: binary
  Content-Disposition: attachment; filename=techsupport-8469.tgz
  ```
- **action=finish**—Stop an active job.

Export Certificates and Keys (Continued)

Step 2 Confirm that the XML response includes the certificate:

```
-----BEGIN CERTIFICATE-----
MIIDXTCCAkWgAwIBAgIJAJC1HiIAZAlIMA0GCSqGSIb3DfBAYTakFVMRwEQYDVQQIDApTb211LVN0XRIeMSEwHwYDVxawRnaXZrIIFBoeSBmdGQwHhcNMTEyMjMxMDg1OTQ0WhcNMT
<!-- TRUNCATED -->
-----END CERTIFICATE-----
```
**Export Technical Support Data (Continued)**

**Step 2** Check on the status of the job.

Use the job ID returned in the previous response as the job-id parameter:

```
https://firewall/api/?type=export&category=tech-support&action=get&job-id=id
```

A status value of **FIN** indicates the data is ready to be retrieved.

```xml
<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>0K</result>
      <tfin>10:12:39</tfin>
      <progress>10:12:39</progress>
      <details />
      <warnings />
      <resultfile>///tmp/techsupport.tgz</resultfile>
    </job>
  </result>
</response>
```

**Step 3** Retrieve the tech support data.

```
https://firewall/api/?type=export&category=tech-support&action=get&job-id=id
```

When using cURL, you can specify the output file name as an option to cURL (`-o`). After a successful retrieval of the job data, the job is automatically deleted by the system.

**Step 4** (Optional) Stop the active job in case of error.

If there is an error or issue with the export job, it may not complete. In cases like this, stop the active job:

```
https://firewall/api/?type=export&category=tech-support&action=finish&job-id=id
```

The response includes a success message:

```xml
<response status="success">
  <msg>Job 2 removed.</msg>
</response>
```
Import Files (API)

You can import certain types of files, including as software, content, licenses, and configurations into the firewall using the `type=import` parameter in the API request.

Use `type=import` and specify the category to import these types of files:

- **Software**—`category=software`
- **Content**—`category=<anti-virus | content | url-database | signed-url-database>`
- **Licenses**—`category=license`
- **Configuration**—`category=configuration`
- **Certificates/key**—`category=<certificate | high-availability-key | key-pair>`
- **Clients**—`category=global-protect-client`
- **Custom logo**—`category=custom-logo`

## Importing Basics

Use cURL to import files to the firewall.

### Import Files to a Firewall or Panorama

- **Import files to a firewall:**
  
  ```bash
  curl --form file=@filename "https://firewall/api/?query-parameters"
  ```

- **Import files to a firewall via Panorama.** First import the file to Panorama, then run a request batch upload-install op command:

  ```bash
  http://panorama/api/?type=op&cmd=<request><batch><anti-virus><upload-install><uploaded-file>your-file-name-here</uploaded-file><devices>serialnumber</devices></upload-install></anti-virus></batch></request>
  ```
Import Files

Use the API Browser to see a full list of import categories.

---

**Import Certificates, Keys, Response Pages, or Custom Logos**

- Import a certificate or key by specifying the type of the certificate or key file using the `category` parameter:
  - `category=certificate`
  - `category=keypair`
  - `category=high-availability-key`

- *(category=certificate or category=keypair only)* Specify these additional parameters for the certificate file and keypair imports:
  - `certificate-name`—name of the certificate object on the firewall
  - `format`—certificate 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.

  ```
  https://firewall/api/?type=import&category=certificate&certificate-name=certificate_name&format=pkcs12 | pem&passphrase=text&vsys=vsys
  ```

- Import a GlobalProtect response pages using an additional parameter for the security profile in which the page should be imported:
  - `profile=profilename`

- Import custom logos to different locations based on the `where` parameter:
  - `where=<login-screen | main-ui | pdf-report-footer | pdf-report-header>`
Retrieve Logs (API)

Retrieve logs from a firewall using the API.

▲ API Log Retrieval Parameters
▲ Example: Use the API to Retrieve Traffic Logs

API Log Retrieval Parameters

Specify the log type with additional optional parameters to retrieve logs from a firewall.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>log-type</td>
<td>The type of logs to retrieve:</td>
</tr>
<tr>
<td></td>
<td>• log-type-traffic—Traffic logs</td>
</tr>
<tr>
<td></td>
<td>• log-type-threat—Threat logs</td>
</tr>
<tr>
<td></td>
<td>• log-type-config—Config logs</td>
</tr>
<tr>
<td></td>
<td>• log-type-system—System logs</td>
</tr>
<tr>
<td></td>
<td>• log-type-hipmatch—HIP logs</td>
</tr>
<tr>
<td></td>
<td>• log-type-wildfire—WildFire logs</td>
</tr>
<tr>
<td></td>
<td>• log-type-url—URL filtering logs</td>
</tr>
<tr>
<td></td>
<td>• log-type-data—Data filtering logs</td>
</tr>
<tr>
<td></td>
<td>• log-type-corr—Correlated event logs as seen in the user interface within</td>
</tr>
<tr>
<td></td>
<td>Monitor &gt; Automated Correlated Engine &gt; Correlated Events.</td>
</tr>
<tr>
<td></td>
<td>• log-type-corr-detail—Correlated event details as seen in the user interface</td>
</tr>
<tr>
<td></td>
<td>when you select an event within Monitor &gt; Automated Correlated Engine</td>
</tr>
<tr>
<td></td>
<td>&gt; Correlated Events.</td>
</tr>
<tr>
<td></td>
<td>• log-type-corr-categ—Correlated events by category, currently compromised</td>
</tr>
<tr>
<td></td>
<td>hosts seen within ACC &gt; Threat Activity &gt; Compromised Hosts.</td>
</tr>
<tr>
<td>query</td>
<td>(Optional) Specify the match criteria for the logs. This is similar to the</td>
</tr>
<tr>
<td></td>
<td>query provided in the web interface under the Monitor tab when viewing the</td>
</tr>
<tr>
<td></td>
<td>logs. The query must be URL encoded.</td>
</tr>
<tr>
<td>nlogs</td>
<td>(Optional) Specify the number of logs to retrieve. The default is 20 when</td>
</tr>
<tr>
<td></td>
<td>the parameter is not specified. The maximum is 5000.</td>
</tr>
<tr>
<td>skip</td>
<td>(Optional) Specify the number of logs to skip when doing a log retrieval.</td>
</tr>
<tr>
<td></td>
<td>The default is 0. This is useful when retrieving logs in batches where you</td>
</tr>
<tr>
<td></td>
<td>can skip the previously retrieved logs.</td>
</tr>
<tr>
<td>dir</td>
<td>(Optional) Specify whether logs are shown oldest first (forward) or newest</td>
</tr>
<tr>
<td></td>
<td>first (backward). Default is backward.</td>
</tr>
</tbody>
</table>

Copyright © 2007-2017 Palo Alto Networks
PAN-OS XML API Request Types

Retrieve Logs (API)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>(Optional) Log data sizes can be large so the API uses an asynchronous job scheduling approach to retrieve log data. The initial query returns a Job ID ((job-id)) that you can then use for future queries with the (\text{action}) parameter:</td>
</tr>
<tr>
<td></td>
<td>(\text{action}=\text{get}) — Check status of an active job or retrieve the log data when the status is (\text{FIN}) (finished). This is slightly different than the asynchronous approach to retrieve tech support data where a separate status action is available.</td>
</tr>
<tr>
<td></td>
<td>(\text{action}=\text{finish}) — Stop an active job.</td>
</tr>
<tr>
<td></td>
<td>(\text{Not specified}) — When not specified, such as during an initial query, the system creates a new job to retrieve log data.</td>
</tr>
</tbody>
</table>

Learn more about Asynchronous and Synchronous Requests to the PAN-OS XML API.

Example: Use the API to Retrieve Traffic Logs

Use the API to Retrieve Traffic Logs

**Step 1**  Create a job to retrieve all traffic logs that occurred after a certain time:
https://firewall/api/?type=log&log-type=traffic&query=(receive_time geq '2012/06/22 08:00:00')

A web-browser will automatically URL encode the parameters, but when using cURL, the query parameter must be URL encoded.

**Response:**

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

**Step 2**  Retrieve traffic log data using the following request using the job ID as the value returned in the previous response:
https://firewall/api/?type=log&action=get&job-id=18
Use the API to Retrieve Traffic Logs (Continued)

**Step 3** Confirm that the XML response looks similar to the following:

```xml
<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>
          <segno>6784588</segno>
          <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>
        </entry>
      </logs>
    </log>
  </result>
</response>
```

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

**Step 4** *(Optional)* Delete and active log retrieval job. To delete an active log retrieval job, run the following query:

```plaintext
https://firewall/api/?type=log&action=finish&job-id=id
```

A successful completion returns a job ID.
Apply User-ID Mapping and Populate Dynamic Address Groups (API)

Use the `type=user-id` parameter to apply User-ID mapping information directly to the firewall. If you are using a third-party VPN solution or have users who are connecting to an 802.1x enabled wireless network, the User-ID API enables you to map users to groups so that you can capture log-in events and send them to the User-ID agent or directly to the firewall. Additionally, you can use the API to register the IP-to-user mapping information from the input file to populate the members of a Dynamic Address Group on the firewall.

```
curl -F key=apikey --form file=@filename "https://firewall/api/?type=user-id"
```
or

```
curl --data-urlencode key=apikey --data-urlencode "cmd=xml-document" https://firewall/api/
```

With your User-ID API requests, you can use the following optional parameters:
- `vsys=vsys_id`—Specify the vsys where you want to apply User-ID mapping.
- `target=serialnumber`—Specify the firewall by serial number when redirecting through Panorama.

Use the information in the following table to apply User-ID mapping information to a firewall:

<table>
<thead>
<tr>
<th>Mapping or Registration Action</th>
<th>API Request</th>
</tr>
</thead>
</table>
| User-ID mapping for a login, logout, or groups. | Use this input file format when providing a User-ID mapping for a login event, logout event, or for groups:  

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

You can include a HIP report by including a `<hip-report></hip-report>` XML container within an `<entry>` parent element.
<table>
<thead>
<tr>
<th>Mapping or Registration Action</th>
<th>API Request</th>
</tr>
</thead>
</table>
| **Multi-User System Entry**   | 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.  
```xml
<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>
``` |
| **User-ID XML multiuser system login event** | When the terminal servers sends a login event payload to the firewall, it 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.  
```xml
<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>
``` |
| **User-ID XML multiuser system logout** | 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.  
```xml
<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>
``` |
<table>
<thead>
<tr>
<th>Mapping or Registration Action</th>
<th>API Request</th>
</tr>
</thead>
</table>
| Dynamic Address Group IP address registration | <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> |
Get Version Info (API)

Use the `type=version` request type to show the PAN-OS version for a firewall or Panorama. In addition to the PAN-OS version, this request provides a direct way to obtain the serial number and model number.

### Get Version Info (API)

**Step 1** Make a request to the PAN-OS XML API and with `type=version` along with your API key:

https://firewall/api/?type=version&key=apikey

**Step 2** The XML response contains the software version, model, serial number, and whether multi-vsys mode is on:

```xml
<response status="success">
  <result>
    <sw-version>7.1.0</sw-version>
    <multi-vsys>off</multi-vsys>
    <model>pa-vm</model>
    <serial>007000001222</serial>
  </result>
</response>
```