Compliance XCH API

At the moment we support four different API calls for NCCM and compliance:

- **nccm_run:** force an NCCM poll
- **nccm_submit:** Push a configuration to the NCCM
- **cmpl_run:** force a compliance check
- **cmpl_report:** retrieve a report for compliance on a policy, node group, shown per policy or per node

Forcing an NCCM poll

You can also force an NCCM poll through the exchange server. A sample exchange XML call looks like this:

```
<task>
  <head
    userid="--your login name--"
    passwd="--your (encrypted) password--"
    task_type="xml_request"
    task_name="nccm_run"
  />
  <request
    node_name="switch13"
    fqdn="192.168.60.113"
  />
</task>
```

The parameters you can send are simple:

- **node_name:** the node's hostname. This can be either in the YCE or CMDB database
- **fqdn:** the node's fqdn. If no node_name is provided, we try to find the node based on its fqdn, which can be an ip or string.

These nodes will be scheduled for an NCCM poll and they will be picked up on the nccmd daemon's next cycle (if load permits).

Submit a manual NCCM configuration

The configurations are normally retrieved from the nodes (jobs, nccm poll). But sometimes it could be desired to upload a configuration directly into the NCCM. For example when a node configuration cannot be retrieved directly and a NCCM report or Compliance check is required anyway.

The **`nccm_submit`** API call allows you to create an NCCM entry for a node as the 'latest' configuration. To submit a configuration for a node it must exist as either a CMDB node or as an YCE node.

The configuration can be submitted with an optional attribute, `nccm_polltime`, to insert a manual timestamp of the configuration instead of the current date and time. However, as with GIT and other

'diff' based storage engines, the NCCM cannot process submitted configurations out-of-order. The provided polling timestamps are mostly administrative and the configuration is still considered the 'latest' superseding the previous configuration. The option is useful mostly to submit a series of configurations taken at different historical moments. Their order is NOT verified or enforced although a transaction with an invalid or badly formatted nccm_polltime is rejected. The nccm_polltime must be formatted as “YYYY-MM-DD hh:mm:ss”.

As the configuration will be embedded in the XML-formatted API call, precautions must be taken to prevent conflicting XML characters in the configuration. Two options exists to achieve this.

First the configuration can be **encoded** using HTML codes. The < and > will then be encoded as `<` and `>` respectively and some other characters will be treated likewise. The use of encoding must be explicitly indicated in the request by adding `xml_decode=“yes”` in the “head” and `<xml_decode>config</xml_decode>` in the “request” part of the API call. This informs the API that the field “config” must be decoded.

An example of this call using encoding:

```
<task>
  <head
    userid="username"
    passwd="xxxxxxxxxxxxx"
    log_level="0"
    task_type="xml_request"
    task_name="nccm_submit"
    xml_decode="yes" />
  <request >
    <node_name>asd--cr01001</node_name>
    <nccm_polltime>2022-09-27 09:20:00</nccm_polltime>
    <xml_decode>config</xml_decode>
    <config>
#
# This configuration is automatically generated at 2022-06-09 16:59:00
#
hostname &lt;asd--cr01001&gt;

snmp-server localhost
#
interface loopback
  address 127.0.0.1 255.255.255.255
#
end
    </config>
  </request>
</task>
```

The second option is to insert the configuration as **CDATA**. This encapsulates the configuration using the header `<![CDATA[` and footer `]]>`, which informs the XML decoder to ignore any xml characters within this section. The use of CDATA does not require any variables in the API request.

The same example using CDATA for the configuration:

```
<task>
  <head
    userid="username"
    passwd="xxxxxxxxxxxxx"
    log_level="0"
    task_type="xml_request"
    task_name="nccm_submit" />
  <request node_name="asd--cr01001">
<config><![CDATA[
#
# This configuration is automatically generated at 2020-06-09 16:59:00
#
hostname <asd--cr01001>

snmp-server localhost
#
interface loopback
  address 127.0.01
#
end
]]></config>
  </request>
</task>
```

The response to these calls:

```
<task>
  <head> ... </head>
  <request> ... </request>
  <response
    nccm_status="configuration unchanged, not added to nccm"
    request_error="0"
    request_status="completed">
    <log>configuration has '14' lines</log>
    <log>configuration unchanged, not added to nccm</log>
    <nccm_data
      action="upload"
      job_descr="nccm upload"
      node_domain="DOM013400"
      node_fqdn="asd--cr01001.acme.com"
      node_name="asd--cr01001"
      node_vendor="HP_C5"
      operator="username"
      session_type="mgmt"
      state="manual"
      verbose="1"/>
  </response>
</task>
```

If a configuration was determined as unchanged, the response `nccm_status` will say as much. When a new entry is created in the NCCM, the message will read “created new nccm diff config: 65”, where

the number refers to the Nccm_id where it is stored.

The response will also return the node details it used to create the NCCM entry like the fqdn, vendor and domain name.

Forcing a Compliance check

You can also force a Compliance check through the exchange server. A sample exchange XML call looks like this:

```
<task>
  <head
    userid="--your login name--"
    passwd="--your (encrypted) password--"
    task_type="xml_request"
    task_name="cmpl_run"
  />
  <request
    node_name="switch13"
    fqdn="192.168.60.113"
  />
</task>
```

The parameters you can send are simple:

- **node_name:** the node's hostname. This can be either in the YCE or CMDB database
- **fqdn:** the node's fqdn. If no node_name is provided, we try to find the node based on its fqdn, which can be an ip or string.

These nodes will be scheduled for compliance and they will be picked up on the nccmd daemon's next cycle (if load permits).

Requesting reports

You can request reports with the same search filters as you can in the reports form (for more information, refer to the Reports section. A sample request looks like this:

```
<task response="">
    <head
        userid="--your login name--"
  passwd="--your (encrypted) password--"
  task_type="xml-request"
  task_name="cmpl_report"
    />
    <request
        type="rules"
        report_template_id="1001"
  hostname="ASD-CR01001"
```

```
        policy_id="1005"
        policy_name="Compliance Check 1"
        rule_name="Compliance Check 2"
        vendor_type="Cisco_IOS"
        group_name="Cisco_IOS"
  detail_level="3"
        status="0"
        severity="Critical"
        report_summary="is not compliant"
    />
</task>
```

- **type:** The report type. Mandatory
  - 'policies': get results based on policies
  - 'rules': get results based on rules
- **report_template_id:** A report template id. Optional. When a value is provided it will load all filters in that template. Any other parameter values in this request will overwrite them if they exist.

Either one of these following is mandatory, to prevent overloading the system

- **policy_id:** a policy's ID from the netYCE database
- **group_name:** a node group name

Additional parameters to set as filters:

- **hostname:** A (part of a) hostname of a node. So both “switch1” and “swi” match “switch1”. Optional.
- **policy_name:** A (part of a) policy name. Optional.
- **rule_name:** A (part of a) rule name. Optional.
- **vendor_type:** A vendor type. Optional.
- **detail_level:** The level of detail to show the report in:
  - 0: Only only the basic policy or node information
  - 1: Up to conditions - default
  - 2: Up to condition details
  - 3: Up to condition full details - this will return everything
- **status:** 1 (“compliant”) or 0 (“not compliant”). Optional
- **severity:** The string version of the severity as set in the NCCM Lookup. Optional.
- **report_summary:** The report summary. Can be part of this string. Optional.

A sample return is as follows:

```
<task>
    <head abort_on_error="1" error="0" log_level="0" passwd="--encrypted
password--" req_host="eth0gate.netyce.org" status="completed"
task_id="0127_0005" task_level="2" task_name="cmpl_report" task_type="xml-
request" userid="NetYCE">
    <logs> </logs>
</head>
<request Group_name="1234" hostname="swi" request_id="1" type="rules">
</request>
    <response request_error="0" request_status="completed">
```

```
    <reports Compliance="Not compliant" Hostname="switch1" Policy_id="523"
Policy_name="Test Policy" Report_details="Rule 'test 3' compliance failure:"
Report_id="138835" Report_rule_id="156547" Rule_id="5015" Rule_name="test 3"
Severity="1" Severity_color="#cc9977" Severity_text="Minor" Status="0"
Vendor_type="HP_C5"/>
    <reports Compliance="Compliant" Hostname="switch1" Policy_id="523"
Policy_name="Test Policy" Report_details="Rule 'test' compliance success. "
Report_id="138835" Report_rule_id="156535" Rule_id="2961" Rule_name="test"
Severity="1" Severity_color="#cc9977" Severity_text="Minor" Status="1"
Vendor_type="HP_C5"/>
    <reports Compliance="Compliant" Hostname="switch1" Policy_id="523"
Policy_name="Test Policy" Report_details="Rule 'test 2' compliance success.
" Report_id="138835" Report_rule_id="156539" Rule_id="5013" Rule_name="test
2" Severity="1" Severity_color="#cc9977" Severity_text="Minor" Status="1"
Vendor_type="HP_C5"/>
    </response>
</task>
```