TEMPLATE-OPS

VERSION HISTORY

version	author	notes	date
v1	Juniper Networks, Inc		2024-11-12

TABLE OF CONTENTS

TEMPLATE-OPS SYNOPSIS

```
arguments:

  --template-vars          pointer to variable gen function in template_ops_vars.py
  --template               name of j2 template receiving variables from --template-vars
  --input                  template seeding data, e.g., sequence number 1-n (or data enclosed in " ")
  --diff-target            device name to retrieve diff between candidate and running config
  --push-target            device name for config template push
  --exec-target            device name for processing template as Python code
  --profile                push to multiple SRX devices using profile [profile-name|# from list]
  --mprofile               multi-profile execution defined by profile [mprofile-name|# from list]
  --list-profile           list push target profiles from template_ops_conf.py
  --list-mprofile          list multi-profiles from template_ops_conf.py
  --show-profile           show details of push target profile [all|profile-name|# from list]
  --show-mprofile          show multi-profile details [all|mprofile-name|# from list]
  --list-template          list available Jinja2 template files
  --show-template          show contents of specific template [template-name|# from list ]
  --debug                  [on] enable backtraces to stdout (SYSLOG by default)
```

TEMPLATE-OPS DESCRIPTION

`template-ops` is a Python script folding minimalistic framework built atop of PyEZ for human interaction with Junos devices in terms of configuration and execution of RPCs (actions, data retrieval). Running both Linux off-box and Junos/Junos-EVO on-box. Initially designed for operations in Juniper MX/PTX/SRX scale-out architectures, but totally generic meanwhile. Licensed under Apache 2.0 license.

Features:

- Conduct Junos config changes from Linux off-box and on-box from Junos devices like MX/PTX by using PyEZ libraries and SSH Netconf sessions with key authentication
- Uses Jinja2 templates for rendering both configuration and Python code with RPCs
- For configurations designed to operate in Junos groups using set commands with explicit delete as a first operation
- Have a configuration profile(s) driven bulk configuration push
- Optional workflow using prototype device for validation and verification prior bulk change. The idea of prototype prior rollout comes from real-life when changes can cause unforeseen consequences. Say one or more devices in the scale-out system can be designated for testing changes including longevity test prior roll out to whole system.
    - Preview Junos set commands for prototype device
    - Do diff of running/candidate config on prototype device
    - Load configuration to prototype device
    - Finally conduct bulk change across multiple devices
- Able to operate on multiple devices using input modifier, either in configuration profile or part of CLI. For example, the devices having sequentially lined up interface IPs, next /30 prefix, BGP ASNs, etc. Tasks can be quite complex like configure using one push of a button multiple SRX MN-HA pairs
- Multi-profile operations when pre-defined configuration/execution profiles are sequentially executed with optional passing variables in-between code execution templates. Includes optional profile pre/post delays.
- Profiles able to mix different device types with different templates and input variable processing
- Control what device types are eligible for configuration push and/or only for viewing rendered set commands from template (safety and operator review prior Junos load set)
- Easy to expand by new template variables inside of the code for given device type, simple expansion for template data from external data sources (e.g., YAML)
- Thorough logging, including template rendering and execution debug
- Ability to archive committed/executed templates and result of rendering for audit and roll-back purposes
- Option to include/exclude profiles from archival operations depending on global on/off
- Ability to operate locally Junos on-box without SSH
- Supports Junos Ephemeral Database with set/text/XML/JSON formats for loading configurations
- Convenience CLIs for listing and displaying templates / profiles.
- For code execution templates retrieving data, either use default tabular view or allow custom header and/or column formatted contents

EXAMPLE USE

This section gives overview of **template-ops** functionality, the 1st example contains both Junos on-box and Linux off-box sample, the rest will be focused on off-box execution. Operations are consistent except local device reference specific to Junos on-box operation.

RENDER SET COMMANDS

ON-BOX (Junos/Junos EVO)

```
> op template-ops template-vars mx template cgn_mx_add input 2
```

```
delete groups scale-out-srx-02
set groups scale-out-srx-02 interfaces ge-0/0/4 unit 0 family inet address 100.65.2.1/24
set groups scale-out-srx-02 interfaces ge-0/0/5 unit 0 family inet address 100.64.2.1/24
set groups scale-out-srx-02 routing-instances vr-trust protocols bgp group vsrx-trust neighbor 100.65.2.10 peer-as 65002
set groups scale-out-srx-02 routing-instances vr-trust interface ge-0/0/4.0
set groups scale-out-srx-02 routing-instances vr-untrust protocols bgp group vsrx-untrust neighbor 100.64.2.10 peer-as 65002
set groups scale-out-srx-02 routing-instances vr-untrust interface ge-0/0/5.0
set apply-groups scale-out-srx-02
```

OFF-BOX (Linux)

```
# ./template-ops.py --template-vars mx --template cgn_mx_add --input 2
```

```
delete groups scale-out-srx-02
set groups scale-out-srx-02 interfaces ge-0/0/4 unit 0 family inet address 100.65.2.1/24
set groups scale-out-srx-02 interfaces ge-0/0/5 unit 0 family inet address 100.64.2.1/24
set groups scale-out-srx-02 routing-instances vr-trust protocols bgp group vsrx-trust neighbor 100.65.2.10 peer-as 65002
set groups scale-out-srx-02 routing-instances vr-trust interface ge-0/0/4.0
set groups scale-out-srx-02 routing-instances vr-untrust protocols bgp group vsrx-untrust neighbor 100.64.2.10 peer-as 65002
set groups scale-out-srx-02 routing-instances vr-untrust interface ge-0/0/5.0
set apply-groups scale-out-srx-02
```

DIFF RUNNING VS TEMPLATE

```
# ./template-ops.py --template-vars vsrx --template cgn_srx_add --diff-target vsrx-01 --input 2
```

```
[edit groups scale-out-srx nat source pool pool-1 address]
+       2.2.0.0/29;
-       2.1.0.0/29;
[edit groups scale-out-srx routing-instances vr-cgn routing-options static]
+       route 2.2.0.0/29 discard;
-       route 2.1.0.0/29 discard;
```

SINGLE DEVICE CONFIGURATION

```
# ./template-ops.py --template-vars vsrx --template cgn_srx_add --push-target vsrx-01 --input 2
```

```
-------------------------------------------------------------------------------------------------------------
|    device    |                                  template operation output                                 |
-------------------------------------------------------------------------------------------------------------
|   vsrx-01    |                                       cgn_srx_add template commit completed, j2+set-cmd archived |
-------------------------------------------------------------------------------------------------------------
```

SINGLE DEVICE RPC EXECUTION

```
# ./template-ops.py --template-vars exec1 --template sessions --exec-target vsrx-01 --input 0/0
```

```
-------------------------------------------------------------------------------------
|    device    |  Total sessions  |   TCP sessions   |   UDP sessions   |  ICMP sessions   |
-------------------------------------------------------------------------------------
|   vsrx-01    |             3453 |             2525 |              841 |               87 |
-------------------------------------------------------------------------------------
```

DEVICE CONFIGURATION PROFILES

```
# ./template-ops.py --profile add_srx_1
```

```
-------------------------------------------------------------------------------------------------------------------
|    device    |                                    template operation output                                     |
-------------------------------------------------------------------------------------------------------------------
|    vmx-01    |                                         cgn_mx_add template commit completed, j2+set-cmd archived |
-------------------------------------------------------------------------------------------------------------------
|   vsrx-01    |                                        cgn_srx_add template commit completed, j2+set-cmd archived |
-------------------------------------------------------------------------------------------------------------------
```

```
# ./template-ops.py --profile update_srx_all
```

```
-------------------------------------------------------------------------------------------------------------------
|    device    |                                    template operation output                                     |
-------------------------------------------------------------------------------------------------------------------
|   vsrx-01    |                                        cgn_srx_add template commit completed, j2+set-cmd archived |
-------------------------------------------------------------------------------------------------------------------
|   vsrx-02    |                                        cgn_srx_add template commit completed, j2+set-cmd archived |
-------------------------------------------------------------------------------------------------------------------
|   vsrx-03    |                                        cgn_srx_add template commit completed, j2+set-cmd archived |
-------------------------------------------------------------------------------------------------------------------
|   vsrx-04    |                                        cgn_srx_add template commit completed, j2+set-cmd archived |
-------------------------------------------------------------------------------------------------------------------
```

RPC EXECUTION PROFILE

```
# ./template-ops.py --profile sessions
```

```
-------------------------------------------------------------------------------------
|    device    |  Total sessions  |   TCP sessions   |   UDP sessions   |  ICMP sessions   |
-------------------------------------------------------------------------------------
|   vsrx-01    |                4 |                2 |                2 |                0 |
-------------------------------------------------------------------------------------
|   vsrx-02    |                4 |                2 |                2 |                0 |
-------------------------------------------------------------------------------------
|   vsrx-03    |                4 |                2 |                2 |                0 |
-------------------------------------------------------------------------------------
|   vsrx-04    |                4 |                2 |                2 |                0 |
-------------------------------------------------------------------------------------
```

```
./template-ops.py --profile sessions --input 100.65.2.0/24
```

```
-------------------------------------------------------------------------------------
|    device    |  Total sessions  |   TCP sessions   |   UDP sessions   |  ICMP sessions   |
-------------------------------------------------------------------------------------
|   vsrx-02    |                2 |                1 |                1 |                0 |
-------------------------------------------------------------------------------------
```

MULTI PROFILE

```
# ./template-ops.py --mprofile add_srx_all
```

```
-------------------------------------------------------------------------------------------------------------
|    device    |                                  template operation output                                 |
-------------------------------------------------------------------------------------------------------------
|    vmx-01    |                                    cgn_mx_add template commit completed, j2+set-cmd archived |
-------------------------------------------------------------------------------------------------------------
|   vsrx-01    |                                   cgn_srx_add template commit completed, j2+set-cmd archived |
-------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------
|    device    |                                  template operation output                                 |
-------------------------------------------------------------------------------------------------------------
|    vmx-01    |                                    cgn_mx_add template commit completed, j2+set-cmd archived |
-------------------------------------------------------------------------------------------------------------
|   vsrx-02    |                                   cgn_srx_add template commit completed, j2+set-cmd archived |
-------------------------------------------------------------------------------------------------------------

-------------------------------------------------------------------------
|    device    |          BGP Inet           |           BGP Gi            |
-------------------------------------------------------------------------
|   vsrx-01    |     100.64.1.1: Established |     100.65.1.1: Established |
-------------------------------------------------------------------------
|   vsrx-02    |     100.64.2.1: Established |     100.65.2.1: Established |
-------------------------------------------------------------------------

-----------------------------------------------------------------------------------------
|    device    |  Total sessions  |   TCP sessions   |   UDP sessions   |  ICMP sessions   |
-----------------------------------------------------------------------------------------
|   vsrx-01    |                4 |                2 |                2 |                0 |
-----------------------------------------------------------------------------------------
|   vsrx-02    |                4 |                2 |                2 |                0 |
-----------------------------------------------------------------------------------------
|   Summary    |               16 |                8 |                8 |                0 |
-----------------------------------------------------------------------------------------
```

LIST PROFILES

```
#./template-ops.py --list-profile all
```

```
-----------------------------------------------------------------------------------------
|   # |     push profile      |                            comment                          |
-----------------------------------------------------------------------------------------
|   1 |            add_srx_1 |                                          add MX/SRX config 1 |
-----------------------------------------------------------------------------------------
|   2 |            add_srx_2 |                                          add MX/SRX config 2 |
-----------------------------------------------------------------------------------------
|   3 |            del_srx_1 |                                          del MX/SRX config 1 |
-----------------------------------------------------------------------------------------
|   4 |            del_srx_2 |                                          del MX/SRX config 2 |
-----------------------------------------------------------------------------------------
|   5 |             sessions |                     retrieve sessions for specific source IP |
-----------------------------------------------------------------------------------------
|   6 |                 load |                                            collect load data |
-----------------------------------------------------------------------------------------
```

SHOW PROFILES

```
# ./template-ops.py --show-profile sessions
```

```
-------------------------------------------------------------------------------------------------------------------------------------------
|       profile       |    device     | template-vars  |         template          |          input          | exec |   eph-instance:fmt   |  archival override  |
-------------------------------------------------------------------------------------------------------------------------------------------
|             sessions |      vsrx-01 |          exec1 |                 sessions |                    0/0 |    Y |                    N |                   N |
|                      |      vsrx-02 |          exec1 |                 sessions |                    0/0 |    Y |                    N |                   N |
|                      |      vsrx-03 |          exec1 |                 sessions |                    0/0 |    Y |                    N |                   N |
|                      |      vsrx-04 |          exec1 |                 sessions |                    0/0 |    Y |                    N |                   N |
-------------------------------------------------------------------------------------------------------------------------------------------
```

```
# ./template-ops.py --show-profile all
```

profile	device	template-vars	template	input	exec	eph-instance:fmt	archival override
add_srx_1	vmx-01	mx	cgn_mx_add	1	N	N	N
	vsrx-01	vsrx	cgn_srx_add	1	N	N	N
add_srx_2	vmx-01	mx	cgn_mx_add	2	N	N	N
	vsrx-02	vsrx	cgn_srx_add	2	N	N	N
del_srx_1	vmx-01	mx	cgn_mx_del	1	N	N	N
	vsrx-01	vsrx	cgn_srx_del	1	N	N	N
del_srx_2	vmx-01	mx	cgn_mx_del	2	N	N	N
	vsrx-02	vsrx	cgn_srx_del	2	N	N	N
update_srx_all	vsrx-01	vsrx	cgn_srx_add	1	N	N	N
	vsrx-02	vsrx	cgn_srx_add	2	N	N	N
	vsrx-03	vsrx	cgn_srx_add	3	N	N	N
	vsrx-04	vsrx	cgn_srx_add	4	N	N	N
sessions	vsrx-01	exec1	sessions	0/0	Y	N	N
	vsrx-02	exec1	sessions	0/0	Y	N	N
	vsrx-03	exec1	sessions	0/0	Y	N	N
	vsrx-04	exec1	sessions	0/0	Y	N	N
pl_1_add_json	vsrx-01	vsrx	pl_1_add_json	1	N	pl:json	rendered:N, J2:N
	vsrx-02	vsrx	pl_1_add_json	2	N	pl:json	rendered:N, J2:N
	vsrx-03	vsrx	pl_1_add_json	3	N	pl:json	rendered:N, J2:N
	vsrx-04	vsrx	pl_1_add_json	4	N	pl:json	rendered:N, J2:N
version	vsrx-01	exec1	version		Y	N	N
	vsrx-02	exec1	version		Y	N	N
	vsrx-03	exec1	version		Y	N	N
	vsrx-04	exec1	version		Y	N	N
mx_local	local	exec1	version		Y	N	N
shutdown	vmx-01	exec1	shutdown		Y	N	N
	vsrx-01	exec1	shutdown		Y	N	N
	vsrx-02	exec1	shutdown		Y	N	N
	vsrx-03	exec1	shutdown		Y	N	N
	vsrx-04	exec1	shutdown		Y	N	N
bgp	vsrx-01	exec1	bgp		Y	N	N
	vsrx-02	exec1	bgp		Y	N	N
	vsrx-03	exec1	bgp		Y	N	N
	vsrx-04	exec1	bgp		Y	N	N
load	vsrx-01	exec1	load		Y	N	N
	vsrx-02	exec1	load		Y	N	N
	vsrx-03	exec1	load		Y	N	N
	vsrx-04	exec1	load		Y	N	N
mp_load_sum	vsrx-01	exec1	mp_load_sum		Y	N	N

SHOW MULTI-PROFILES

```
# ./template-ops.py --show-mprofile all
```

multi profile	profile	pre-delay[s]	post-delay[s]
sessions	sessions		
	mp_sessions_sum		2
status	version	1	
	alarm		
add_srx_all	add_srx_1		2
	add_srx_2		2
	add_srx_3		2
	add_srx_4		2
	bgp	10	
	sessions		
	mp_sessions_sum		
del_srx_all	del_srx_1		
	del_srx_2		
	del_srx_3		
	del_srx_4		
	bgp		
load	load		
	mp_load_sum		

LIST TEMPLATES

```
# ./template-ops.py --list-template   all
```

```
-------------------------------------------------------------------------------
|   # | /scripts/template-ops-138/xtemplate/*.j2 |               md5               |
-------------------------------------------------------------------------------
|   2 |                                      bgp |   32602ac2d928a41561fb79fef74c7bbe |
-------------------------------------------------------------------------------
|   3 |                              cgn_mx_add |   7d14feaa86dcd09ad84fec36fb6193a1 |
-------------------------------------------------------------------------------
|   4 |                              cgn_mx_del |   f5284d57b56fd109b7f83838a8147c46 |
-------------------------------------------------------------------------------
|   5 |                             cgn_srx_add |   69d709a20357d5793beb486fafb7bc50 |
-------------------------------------------------------------------------------
|   6 |                             cgn_srx_del |   4c9a8a0d40e42ff3b10c065c1c732716 |
-------------------------------------------------------------------------------
|   7 |                                  filter |   9d838d5b9de7e6a06f64309653677fcd |
-------------------------------------------------------------------------------
|  10 |                                    load |   2b87c56f557f00605e5213994050dc2b |
-------------------------------------------------------------------------------
|  11 |                               mnha_flip |   d7915268587e9c7446ced335530f04cd |
-------------------------------------------------------------------------------
|  12 |                             mnha_status |   92c9c23f9a07348ed1008be3ebdc38d2 |
-------------------------------------------------------------------------------
|  13 |                             mp_load_sum |   f5af5c1539cbf4629ac514b3ec2a83af |
-------------------------------------------------------------------------------
|  14 |                             mp_sessions |   97a91783173a0bb117c1519fadf7390f |
-------------------------------------------------------------------------------
|  15 |                         mp_sessions_sum |   ea03616468d545e93eb42e44ca506705 |
-------------------------------------------------------------------------------
|  16 |                              mx_add_srx |   fe530b6643375c5f98ab7994d7b3c249 |
-------------------------------------------------------------------------------
|  21 |                                  reboot |   542237df6dea34cbc325c9288ae95f5f |
-------------------------------------------------------------------------------
|  22 |                                sessions |   97a91783173a0bb117c1519fadf7390f |
-------------------------------------------------------------------------------
|  23 |                                shutdown |   422f783ce98832bf4ad9ce9ac7bab732 |
-------------------------------------------------------------------------------
```

SHOW TEMPLATE

```
# ./template-ops.py --show-template sessions
```

```
#
######################################## BEGIN sessions ########################################
#
global result
global result_adv
global header

header = "{:^16} | {:^16} | {:^16} | {:^16}".format(
    "Total sessions", "TCP sessions", "UDP sessions", "ICMP sessions"
)

ip="{{ ip }}"

rpc_output = dev.rpc.get_flow_session_information(
    normalize=True, source_prefix=ip, summary=True
)
total = rpc_output.findtext(".//displayed-session-count")

if int(total) > 0:

{% for proto in ['tcp', 'udp', 'icmp', 'icmp6'] %}
    rpc_output = dev.rpc.get_flow_session_information(
        normalize=True, source_prefix=ip, summary=True, protocol="{{ proto }}"
    )
    {{ proto }} = rpc_output.findtext(".//displayed-session-count")
{% endfor %}

    icmp = int(icmp) + int(icmp6)

    result = "{:>16} | {:>16} | {:>16} | {:>16}".format(total, tcp, udp, icmp)
    result_adv = [ total, tcp, udp, icmp ]
```

```
elif ip == "0/0":
    result = "{:>16} | {:>16} | {:>16} | {:>16}".format(0, 0, 0, 0)
    result_adv = [ 0, 0, 0, 0 ]

else:
    result = ""

#
##################################### END  sessions ########################################
```

DEBUG IN-LINE

```
# ./template-ops.py --template-vars exec1 --template sessions --exec-target vsrx-01 --input 1.1.1.256
```

```
-------------------------------------------------------------------------------------------
|    device    |  Total sessions  |   TCP sessions   |   UDP sessions   |  ICMP sessions   |
-------------------------------------------------------------------------------------------
|   vsrx-01    | Error executing code sessions, use debug on/see log                       |
-------------------------------------------------------------------------------------------
```

```
# ./template-ops.py --template-vars exec1 --template sessions --exec-target vsrx-01 --input 1.1.1.256 --debug on
```

```
vsrx-01 error executing code sessions.j2, traceback: Traceback (most recent call last):
  File "/usr/lib/python3/dist-packages/jnpr/junos/device.py", line 768, in execute
    rpc_rsp_e = self._rpc_reply(rpc_cmd_e,
                ^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/jnpr/junos/decorators.py", line 169, in wrapper
    raise ex
  File "/usr/lib/python3/dist-packages/jnpr/junos/decorators.py", line 116, in wrapper
    rsp = function(self, *args, **kwargs)
          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/jnpr/junos/device.py", line 1318, in _rpc_reply
    return self._conn.rpc(rpc_cmd_e)._NCElement__doc
           ^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/ncclient/manager.py", line 246, in execute
    return cls(self._session,
           ^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/ncclient/operations/third_party/juniper/rpc.py", line 52, in request
    return self._request(rpc)
           ^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/ncclient/operations/rpc.py", line 375, in _request
    raise self._reply.error
ncclient.operations.rpc.RPCError: invalid value '256' in ip address: '1.1.1.256'
```

DEBUG LOG

Off-box

```
# journalctl
```

```
template-ops.py[188084]: [root]/root/scripts/template-ops-138/template-ops.py[188084] vsrx-01 code execution start /root/scripts/template-ops-138/xtemplate/sessions.j2 (md5:
97a91783173a0bb117c1519fadf7390f)

template-ops.py[188084]: [root]/root/scripts/template-ops-138/template-ops.py[188084] vsrx-01 error executing code sessions.j2, traceback: Traceback (most recent call last):
                                                    File "/usr/lib/python3/dist-packages/jnpr/junos/device.py", line 768, in execute
                                                      rpc_rsp_e = self._rpc_reply(rpc_cmd_e,
                                                                  ^^^^^^^^^^^^^^^^^^^^^^^^^^
                                                    File "/usr/lib/python3/dist-packages/jnpr/junos/decorators.py", line 169, in wrapper
                                                      raise ex
                                                    File "/usr/lib/python3/dist-packages/jnpr/junos/decorators.py", line 116, in wrapper
                                                      rsp = function(self, *args, **kwargs)
                                                            ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
                                                    File "/usr/lib/python3/dist-packages/jnpr/junos/device.py", line 1318, in _rpc_reply
                                                      return self._conn.rpc(rpc_cmd_e)._NCElement__doc
                                                             ^^^^^^^^^^^^^^^^^^^^^^^^^
                                                    File "/usr/lib/python3/dist-packages/ncclient/manager.py", line 246, in execute
                                                      return cls(self._session,
                                                             ^^^^^^^^^^^^^^^^^^
                                                    File "/usr/lib/python3/dist-packages/ncclient/operations/third_party/juniper/rpc.py", line 52, in request
                                                      return self._request(rpc)
                                                             ^^^^^^^^^^^^^^^^^^
                                                    File "/usr/lib/python3/dist-packages/ncclient/operations/rpc.py", line 375, in _request
                                                      raise self._reply.error
                                                  ncclient.operations.rpc.RPCError: invalid value '256' in ip address: '1.1.1.256'
```

On-box

> show log messages

```
vmx-01 cscript[79600]: CSCRIPT_SECURITY_WARNING: unsigned python script '/var/db/scripts/op/template-ops.pyc' without checksum is executed

vmx-01 cscript[79600]: [root]/var/db/scripts/op/template-ops.pyc[79600] vsrx-01 code execution start /var/db/scripts/op/xtemplate/sessions.j2 (md5:
97a91783173a0bb117c1519fadf7390f)

vmx-01 cscript[79600]: [root]/var/db/scripts/op/template-ops.pyc[79600] vsrx-01 error executing code sessions.j2, traceback: Traceback (most recent call
last):   File "../../../../../../../../src/dist/python-add-ons/junos-eznc/lib/jnpr/junos/device.py", line 774, in execute   File
"../../../../../../../../src/dist/python-add-ons/junos-eznc/lib/jnpr/junos/decorators.py", line 169, in wrapper   File
"../../../../../../../../src/dist/python-add-ons/junos-eznc/lib/jnpr/junos/decorators.py", line 116, in wrapper   File
"../../../../../../../../src/dist/python-add-ons/junos-eznc/lib/jnpr/junos/device.py", line 1336, in _rpc_reply   File
"../../../../../../../../src/dist/python-add-ons/ncclient/ncclient/manager.py", line 215, in execute   File "../../../../../../../../src/dist/python-add-
ons/ncclient/ncclient/operations/third_party/juniper/rpc.py", line 46, in request   File "../../../../../../../../src/dist/python-add-
ons/ncclient/ncclient/operations/rpc.py", line 341, in _request ncclient.operations.rpc.RPCError: invalid value '256' in ip address: '1.1.1.256'  During
handling of the above exception, another exception occurred:  Traceback (most recent call last):   File "/var/db/scripts/op/template-ops.py", line 1118, in
template_thread     exec(template_output)   File "<string>", line 12, in <module>   File "../../../../../../../../src/dist/python-add-ons/junos-
eznc/lib/jnpr/junos/rpcmeta.py", line 345, in _exec_rpc   File "../../../../../../../../src/dist/python-add-ons/junos-eznc/lib/jnpr/junos/decorators.py", line
63, in wrapper   File "../../../../../../../../src/dist/python-add-ons/junos-eznc/lib/jnpr/junos/decorators.py", line 31, in wrapper   File
"../../../../../../../../src/dist/python-add-ons/junos-eznc/lib/jnpr/junos/device.py", line 794, in execute jnpr.junos.exception.RpcError: RpcError(severity:
error, bad_element: 1.1.1.256, message: invalid value '256' in ip address: '1.1.1.256')
```

INSTALLATION

TEMPLATE-OPS FILES

Junos on-box /var/db/scripts/op/ or any folder on Linux	Function
`template-ops.py`	Main script
`template_ops_conf.py`	Configuration file
`template_ops_vars.py`	Generator functions for template rendering
`id_rsa`	Default SSH key location, configurable in _conf.py
`xarchive/`	Default folder for archiving templates and rendered outputs, configurable in _conf.py
`xtemplate/`	Default folder for storing configuration and code execution templates, configurable in _conf.py
`xdata/`	Folder for input data, e.g., yaml formatted
`sync_to_mx.sh`	Sample off-box shell script for uni-directional sync of template ops to Junos
`run-profiles.sh`	Sample off-box shell script for executing sequentially profiles

JUNOS ON-BOX

For on-box operations place the extracted files into **`/var/db/scripts/op/`** folder and insert into Junos configuration the commands from **`JUNOS_SET_CMD`** file – those activate the op script and provide contextual help/CLI completion. Pretty much any Junos / Junos EVO 21.4+ should work. Ideally insert the commands into separate Junos group as in the provided example.

Tip - **`> op template-ops [commands]`** Junos CLI can be changed to pretty much anything using

`set command [command]`

from sample **`JUNOS_SET_CMD`** file, e.g., could be eventually executed **`> op t [commands]`** when desired (also see WRAPPER TO TEMPLATE-OPS in ADVANCED CONCEPTS section)

LINUX OFF-BOX

For off-box installation on Debian/Ubuntu either:

`apt-install python3-junos-eznc` (as of Debian 12 and Ubuntu 24.04 installs fairly old 2.1.7)

Some platforms may need to explicitly enable Python 3, for Debian/Ubuntu:

`apt-install python-is-python3`

Alternatively install latest PyEZ:

https://www.juniper.net/documentation/us/en/software/junos-pyez/junos-pyez-developer/topics/task/junos-pyez-server-installing.html

Then extract the **template-ops** files into folder of choice, multiple instances in different folders can exist (handy is to use symlinks to share common settings).

The location of the configuration files is tackled using short code at beginning of **template_ops_conf.py** – sets absolute location for on-box operation and relative for off-box operation (see below).

The minimum tested Python version with **template-ops** is 3.7 (present Junos on-box version).

SSH KEY GENERATION

Note - for on-box operation where no remote device is reached to, no SSH key is needed – special device **"local"** can be used.

For operation both on-box and off-box with remote devices use RSA keys as Ed25519 keys are working only with latest off-box libraries.

To generate private **id_rsa** and **id_rsa.pub** public RSA SSH keys:

(off-box relatively to **template-ops** folder, on-box **cd /var/db/scripts/op/**), no passphrase by pressing Enter twice:

```
ssh-keygen -t rsa -b 2048 -m pem -f id_rsa
```

where **-m pem** is legacy format as older on-box PyEZ/Paramiko may have issues with the new OpenSSH key format.

SSH private key must be protected from unauthorized access. Different usernames and/or SSH keys can be used for individual devices.

TARGET JUNOS SYSTEM

Then on target Junos devices setup specific user for **template-ops**, for example:

```
set system login user template-ops class super-user
set system login user template-ops authentication ssh-rsa "public SSH key" from [list of allowed hosts]
```

Example:

```
set system login user template-ops authentication ssh-rsa "ssh-rsa AAA<SNIP>vnK adm@vmx-01" from 10.0.0.1
```

Defining list of permitted hosts using **from** stanza is optional but recommended as the user has elevated privileges and SSH key with passphrase on host with **template-ops** is not really practical.

Finally, enable Netconf over SSH (TCP port 830):

```
set system services netconf ssh
```

IMPORTANT NOTE – obviously system with deployed **template-ops** as above becomes full administrator platform for the target devices.

TEMPLATE SEEDING FUNCTIONS FILE

Included by main code, **template_ops_vars.py** serves the purpose of retrieving Python dictionaries used in Jinja2 template rendering, both configuration and code execution.

1st is definition of Python lists containing mapping **template-vars** input (entered from CLI or profile), typically referring to device types/groups or purpose, with colors for easier references:

```
TEMPLATE_VARS1_LIST = ["vsrx"]
TEMPLATE_VARS2_LIST = ["ptx", "mx"]
TEMPLATE_VARS3_LIST = ["exec1"]
TEMPLATE_VARS4_LIST = ["srx_yaml1"]
```

2nd - lists are then referring to specific function for generating configuration/code using Jinja2, where **template_vars_arg** and **_input** parameters to **template_vars_get** function are effectively the **--template-vars** and **--input** arguments from one device operation CLI or profile discussed later.

```
def template_vars_get(template_vars_arg, _input):
    if template_vars_arg in TEMPLATE_VARS1_LIST:
        return template_vars1(_input)
    elif template_vars_arg in TEMPLATE_VARS2_LIST:
        return template_vars2(_input)
    elif template_vars_arg in TEMPLATE_VARS3_LIST:
        return template_vars3(_input)
    elif template_vars_arg in TEMPLATE_VARS4_LIST:
        return template_vars4(_input)
    else:
        raise ValueError("UNKNOWN device-type")
```

Returned dictionary data from **template_vars_get** are passed to Jinja for rendering configuration or Python code for execution.

SIMPLE INPUT

sample function referred by above providing to Jinja2 calculated AS number and sequence in vSRX scale-out lineup:

```
def template_vars1(_input):
    seq = int(_input)
    aut_sys = 65000 + seq
    template_vars = {
        "seq": str(seq),
        "aut_sys": str(aut_sys),
    }
    return template_vars
```

another simple function used in conjunction with code execution profile below

```
def template_vars3(_input):
    template_vars = {
```

```
        "ip": str(_input),
    }
    return template_vars
```

MULTI-VALUE INPUT

Sample function to receiving on input list of IP addresses formatted as **`--input "1.1.1.2, 1.1.1.2"`** and hand over it to template processing as a list (string formatted, **`eval()`** or similar is needed for processing as Python list in exec templates)

```
def template_vars2(_input):
    template_vars = {
        "ips": _input.split(","),
    }
    return template_vars
```

YAML INPUT

sample function extracting from YAML file data, used for generating set commands or Python code:

```
def template_vars4(_input):
    import yaml
    with open(_input, 'r') as file:
      data = yaml.safe_load(file)

    template_vars = {
        "list1": data['list1']
    }
    return template_vars
```

corresponding YAML file which would be pointed to by **`--input`** parameter from CLI or profile

```
list1:
- 1.1.1.1
- 1.1.1.2
- 1.1.1.3
```

ADDITIONAL SETTINGS

This variable defines list of devices which can be subjected to any other operation than just rendering set commands, mostly for safety purposes when administrator wants to review set commands prior using Junos **`load set`**:

```
DIFF_PUSH_ELIGIBLE_LIST = ["mx", "vsrx", "exec1"]
```

This parameter is string included in Linux help, for purposes of listing top used **`template_vars`**:

```
TEMPLATE_VARS_STR = "[vsrx|srx4600|mx|ptx|exec1]"
```

CONFIGURATION FILE

Configuration file **template_ops_conf.py** is effectively by **template-ops.py** included Python code with all the Pythonic programmability.

COMMON SETTINGS

1st section just checks for on-box / off-box operation for purposes of looking up path

```
try:
   from junos import Junos_Context
   PATH = '/var/db/scripts/op/'
except:
   from os import getcwd
   PATH = getcwd() + '/'
```

SSH_KEY variable sets path to RSA key, multiple variables with different SSH keys can be defined

```
SSH_KEY = PATH + 'id_rsa'
```

Following variable ensures that if configuration template operations starting with delete of the particular group do not have yet the syntax to delete (like initial push), the logic upon loading error removes delete commands and tries again

```
REMOVE_DEL_CMDS_DURING_PUSH_DIFF_ERR_ENABLE = 1
```

This is a global enable/disable control whether Python code can be generated by Jinja and subjected to execution of by **exec()** method. This could impose a security risk if unauthorized person got access to templates.

```
TEMPLATE_EXEC_ENABLE = 1
```

Defines path for looking up Jinja2 templates

```
TEMPLATE_SEARCH_PATH = PATH + 'xtemplate'
```

This variable sets very conservative maximum of devices in profile (covered later), essentially how many multi-threaded operations will be happening simultaneously. When overrunning the capacity, then the next option is sequential multi profile execution

```
MAX_PROFILE_DEV = 16
```

Defines commit timeout, default is 30 seconds

```
COMMIT_TIMEOUT = 30
```

Controls whether to save into archive folder defined below committed Jinja2 template

Archival of Jinja2 template

```
SAVE_COMMIT_J2_ENABLE = 1
```

Archival of resulting Junos config:

```
SAVE_COMMIT_CFG_ENABLE = 1
```

Archival of Jinja2 template used to prepare Python code for **exec()**

```
SAVE_EXEC_J2_ENABLE = 1
```

Archival of resulting Python code after rendering Jinja2 template

```
SAVE_EXEC_PY_ENABLE = 1
```

Self-explanatory paths for archiving exec/configuration templates and rendered results

```
SAVE_PATH_COMMIT_J2 =  PATH + 'xarchive'
SAVE_PATH_COMMIT_CFG = PATH + 'xarchive'
SAVE_PATH_EXEC_J2 = PATH + 'xarchive'
SAVE_PATH_EXEC_PY = PATH + 'xarchive'
```

DEVICE AUTHNETICATION PROFILES

Authentication profile defines credentials and network parameters for connection to devices. There is also an option to use default profile with common settings, where specific device settings are merged to (example bellow configures IP address). Merged profile overrides settings in default profile.

General syntax:

```
auth_profiles = {
  'default':{ 'user':['junos-username'], 'port':[Netconf/SSH port], 'ssh_key':['SSH KEY PATH'] },
  'device1':{ 'host':['IP address'] },
  'device2':{ 'host':['IP address'] },
}
```

Sample:

```
auth_profiles = {
  'default':{ 'user':['template-ops'], 'port':[830], 'ssh_key':[SSH_KEY] },
   'vmx-01':{ 'host':['10.0.0.10'],},
  'vsrx-01':{ 'host':['10.0.0.11'] },
  'vsrx-02':{ 'host':['10.0.0.12'] },
  'vsrx-03':{ 'host':['10.0.0.13'] },
  'vsrx-04':{ 'host':['10.0.0.14'] },
}
```

PROFILE LIST

Push profile structure defines all available profiles for bulk operation by single profile, format is following, comment is for purpose of listing profiles. Sample below.

```
push_profiles = {
  'add_srx_1':     { 'comment':['add MX/SRX config 1'] },
  'add_srx_2':     { 'comment':['add MX/SRX config 2'] },
  'del_srx_1':     { 'comment':['del MX/SRX config 1'] },
  'del_srx_2':     { 'comment':['del MX/SRX config 2'] },
  'update_srx_all':{ 'comment':['update srx config all'] },
  'sessions':      { 'comment':['retrieve sessions for specific source IP'] },
  'pl_1_add_json': { 'comment':['loads prefix list pl-1 into eph instance pl, json fmt'] },
  'pl_1_add_set':  { 'comment':['loads prefix list pl-1 into eph instance pl, set fmt'] },
  'pl_1_del':      { 'comment':['empty prefix list pl-1 in ephemeral instance pl'] },
  'filter':        { 'comment':['displays counters from FF counters'] },
  'version':       { 'comment':['show Junos versions'] },
  'mx_local':      { 'comment':['show Junos versions, mx local'] },
  'alarm':         { 'comment':['show system alarms'] },
  'shutdown':      { 'comment':['shutdown all'] },
  'bgp':           { 'comment':['show bgp peers'] },
  'load':          { 'comment':['collect load data'] },
  'mp_load_sum':   { 'comment':['aggregate load data'] },
}
```

DEVICE GROUPS

Device groups are Python dictionaries which are merged with profiles described in the next section, this would be sample group:

```
vsrx = { 'vsrx-01':{}, 'vsrx-02':{}, 'vsrx-03':{}, 'vsrx-04':{}, }
```

PROFILES

Generic syntax is following, more explanation in sample profiles below.

```
PROFILE_NAME = {
   "APPLICATBLE DEVICE from auth_profiles OR default OR local": {
       "template_vars": ["REFERAL TO TEMPLATE VARS"],
       "template": ["TEMPLATE"],
       "input": ["INPUT"],
       "eph_inst": ["EPHEMERAL INSTANACE NAME", "INPUT DATA FORMAT – TEXT OR XML OR JSON OR SET"],
       "save_rendered_j2": [[True OR False TO SAVE RESULT OF RENDER, True OR False TO SAVE J2 TEMPLATE]],
       "exec":[True OR False TO TREAT RESULT OF TEMLATE RENDER AS PYTHON CODE FOR EXECUTION],
   },
}
```

IMPORTANT NOTES RELATED TO EPHEMERAL DB

- When dealing with Ephemeral Database, avoid set syntax if there are large lists (e.g. prefix-list) crossing approximately 8000 elements which need to be deleted first. Instead use JSON/XML/text format for which template-ops always uses overwrite operation (not supported with set format). That's a known Junos PR with no resolution to date. Issue manifest as high `mgd` load and commit typically times out.

- To avoid growing Ephemeral Database until maximum size, configure rotation, for example:
  `set system configuration-database ephemeral purge-on-version 5`

SAMPLE PROFILE - CONFIGURATION

Basic profile called **update_srx_all** for bulk configuration of 4 vSRX instances (01-04) with **default** common settings, similarly to device authentication profile device specific settings override **default**. When executed, the operation is multi-threaded reaching out to the devices in parallel.

Note - this is an essential example for understanding **template-ops** workflow.

```
update_srx_all = {
  'default':{ 'template_vars':['vsrx'], 'template':['cgn_srx_add'] },

  'vsrx-01':{ 'input':['1'] },
  'vsrx-02':{ 'input':['2'] },
  'vsrx-03':{ 'input':['3'] },
  'vsrx-04':{ 'input':['4'] },
}
```

template_vars above is a pointer to **template_vars1** function by using **vsrx** value defined in the **TEMPLATE_VARS1_LIST** list below, receiving defined **input** values (1, 2, 3, 4), unique per device as defined above.

```
TEMPLATE_VARS1_LIST = ["vsrx"]

def template_vars_get(template_vars_arg, _input):
    if template_vars_arg in TEMPLATE_VARS1_LIST:
        return template_vars1(_input)

def template_vars1(_input):
    seq = int(_input)
    aut_sys = 65000 + seq
    template_vars = {
        "seq": str(seq),
        "aut_sys": str(aut_sys),
    }
    return template_vars
```

Jinja2 template (seeded by **template_vars** dictionary with **seq** and **aut_sys** keys/values from above) is effectively the **cgn_srx_add.j2** file, located by default in **xtemplate** folder. Template with set commands in group would delete the group first, and re-establish setting from beginning including **apply-groups** as in example below:

```
# ./template-ops.py --show-template cgn_srx_add

delete groups scale-out-srx
set groups scale-out-srx security log stream pod-host source-address 100.65.{{ seq }}.10
set apply-groups scale-out-srx

# ./template-ops.py --template-vars vsrx --template cgn_srx_add --input 1

delete groups scale-out-srx
set groups scale-out-srx security log stream pod-host source-address 100.65.1.10
set apply-groups scale-out-srx
```

Resulting configuration profile pushed to devices would be variable in source-address for the vsrx-01 to vsrx-04 devices (1-4 in 3rd octet).

SAMPLE PROFILES – CODE EXECUTION

Basic code execution template may not need any input, for example gathering version information, template sets **exec** flag to **True**.

vSRX group described in DEVICE GROUPS is below referred to by using Python dictionary merging (older syntax compatible with Python 3.7).

```
version = {
  'default':{ 'template_vars':['exec1'], 'template':['version'], 'input':[''], 'exec':[True] },
}

version = {**version, **vsrx }
```

Contents of simple code execution template with no Jinja code altering the code prior execution, simple text is passed to **result** variable, and by default outputs are printed:

```
global result

rpc_output = dev.rpc.get_software_information({"format": "text"})

result = " ".join(
    [line for line in rpc_output.text.splitlines() if "Junos" in line or "ike" in line]
)
```

More advanced profile and template passing IP address from **template_vars** to retrieve firewall filter counter (named as IP addresses), besides results also when **header** variable is returned, custom header is printed instead of default:

```
filter = {
    "default": {
        "template_vars": ["exec1"],
        "template": ["filter"],
        "input": ["accept"],
        "exec": [True],
        "save_rendered_j2": [True, False],
    },
    "vsrx-01": {},
    "vsrx-02": {},
    "vsrx-03": {},
    "vsrx-04": {},
}
```

Template vars:

```
def template_vars3(_input):
    template_vars = {
        "ip": str(_input),
    }
    return template_vars
```

Template:

```
global result
global header
ip = "{{ ip }}"
rpc_output = dev.rpc.get_firewall_counter_information(
    normalize=True, countername=ip, filter="protect-1"
)
counter = rpc_output.findtext(".//counter-name")
bytes = rpc_output.findtext(".//byte-count")
packets = rpc_output.findtext(".//packet-count")
header = "{:^16} | {:^16} | {:^16}".format("Counter", "Bytes", "Packets")
if counter:
    result = "{:>16} | {:>16} | {:>16}".format(counter, bytes, packets)
else:
    result = "{:>16} | {:>16} | {:>16}".format(ip, "N/A", "N/A")
```

Finally, simple profile for on-box operation using **local** keyword instead of device name (avoiding Netconf over SSH).

```
mx_local = {
  'default':{ 'template_vars':['exec1'], 'template':['version'], 'input':[''], 'exec':[True] },
  'local':{ },
}
```

Template:

```
global result
rpc_output = dev.rpc.get_software_information({"format": "text"})
result = " ".join(
    [line for line in rpc_output.text.splitlines() if "Junos" in line or "ike" in line]
)
```

MULTI-PROFILES

Multi-profiles (mprofile) do basically chain regular profiles for configuration and execution in one go with an option to introduce pre and post delays. Typical use would be configuration of MX when adding new scaled-out SRX, each profile in multi-profile would cover one SRX and relevant MX side in one go. Use-case for execution is post-processing retrieved data from regular profiles.

Generic syntax is following:

```
multi_profiles = {
    "MULTI-PROFILE NAME": {
        "comment": ["COMMENT TEXT"],
        "push_profiles": [
            {"PROFILE 1": {}},
            {"PROFILE 2": {"post-delay": SECONDS (DEFAULT 0), "pre-delay": SECONDS (DEFAULT 0)}},
        ],
    },
}
```

Sample multi-profile configuration, **sessions** retrieve and prints data from individual devices, **mp_sesssions_sum** profile leverages retrieved data for printing summary (see ADVANCED CONCEPTS). Mprofile **add_srx_all** adds one at a time SRX and MX configurations in scale-out swarm with **post-delay** of 2s between operations, retrieves BGP status with **pre-delay** of 10s and displays session counts like above multi-profile does.

Note – when operating off-box, the post-processing profile (e.g., the mentioned **mp_sesssions_sum)** needs to be executed on some selected Junos device. In the current software local off-box code execution is not present.

```
multi_profiles = {
    "sessions": {
        "comment": ["retrieve session info"],
        "push_profiles": [
            {"sessions": {}},
            {"mp_sessions_sum": {"post-delay": 2, "pre-delay": 0}},
        ],
    },
    "add_srx_all": {
        "comment": ["add all srx and mx"],
        "push_profiles": [
            {"add_srx_1": {"post-delay": 2}},
            {"add_srx_2": {"post-delay": 2}},
            {"add_srx_3": {"post-delay": 2}},
            {"add_srx_4": {"post-delay": 2}},
            {"bgp": {"pre-delay":10}},
            {"sessions": {}},
            {"mp_sessions_sum": {}},
        ],
    },
}
```

DEBUG AND AUDIT

LOGGING

By default, **template-ops** is on both off-box Linux and on-box Junos logging actions taken. The idea behind recording MD5 checksum is ability to trace back template in conjunction with archiving described below. Current MD5 checksums of templates are shown in **list-templates** command. MD5 has been chosen for readability as collision attacks are not likely in a secure environment.

Sample logs:

```
vsrx-01 template diff start /var/db/scripts/op/xtemplate/cgn_srx_add.j2 (md5: 69d709a20357d5793beb486fafb7bc50)
vsrx-01 no diff between candidate and current config /var/db/scripts/op/xtemplate/cgn_srx_add.j2 (md5: 69d...)
vsrx-01 diff exist between candidate and current config /var/db/scripts/op/xtemplate/cgn_srx_add.j2 (md5: 69d7...)
vsrx-01 template push start /var/db/scripts/op/xtemplate/cgn_srx_add.j2 (md5: 69d709a20357d5793beb486fafb7bc50)
vsrx-01 commit completed /var/db/scripts/op/xtemplate/cgn_srx_add.j2 (md5: 69d709a20357d5793beb486fafb7bc50)
vsrx-04 code execution start /var/db/scripts/op/xtemplate/filter.j2 (md5: 9d838d5b9de7e6a06f64309653677fcd)
vsrx-04 template filter, exec code archived
vsrx-02 template sessions, exec j2+code archived
```

In addition to above sample context, logs are prefixed by **[username][path][PID]**, for example:

```
[root]/var/db/scripts/op/template-ops.pyc[91308]
```

```
[root]/root/scripts/template-ops/template-ops.py[168450]
```

For off-box operation this is meant to track operations when multiple instances are deployed to various folders.

DEBUGGING

By default, without enabling **debug on** parameter/value, minimum errors get backtrace right to the terminal. However, backtraces are logged to SYSLOG on-box/off-box automatically.

As in EXAMPLE USE section above, in case of debugging, either logs can be looked up or by enabling debug, also error outputs appear directly on console.

Debug log with traceback is 1st place where to look for troubleshooting configuration syntax issues, template syntax and **template-ops** runtime / related libraries errors.

General syntax on-box:

```
> op template-ops [commands] debug on
```

Off-box:

```
./template-ops [commands] --debug on
```

ARCHIVING

The purpose of archiving controlled globally by **`SAVE_*`** settings as described in CONFIGURATION FILE section are audit and roll-back. For both configuration and code execution, both Jinja2 template and result of rendering can be saved. Where globally enabled archiving can be overridden on individual profile level by **`"save_rendered_j2": [bool, bool]`** control.

Default file format of saved templates:

`YYYYMMDD-HHMMSS__TEMPLATE_NAME__DEVICE.[set|.j2 |.py |.py.j2]`

Where file suffixes determine type:

`.set`	result of configuration render
`.j2`	Jinja2 used for rendering set commands
`.py`	is executed Python code
`.py.j2`	is Jinja2 template used for rendering Python code

Sample files in **`xarchive`** folder:

```
20241005-225014__cgn_srx_add__vsrx-01.j2
20241005-225014__cgn_srx_add__vsrx-01.set
20241006-094116__sessions__vsrx-01.py.j2
20241006-094116__sessions__vsrx-01.py
```

IMPORTANT NOTE – administrators need to take care of the archival folder clean-up

ADVANCED CONCEPTS

INPUT VARIABLES IN CODE EXECUTION

Mostly relevant variables and sample values available for use in exec templates:

```
'_input': '1',
'dev': Device(10.0.0.11),
'profile': None,
'profile_operation': False,
'push_target': 'vsrx-01',
'template_abs': '/root/scripts/template-ops-138/xtemplate/version.j2',
'template_file': 'version.j2',
'template_md5': '64a5ac2bbe81164b9d129a2fe4d87a22',
'template_vars': 'exec1',
'template_vars_for_render': {'ip': '1'},
'timestamp': '20241007-082157'
```

For complete listing of global and local variables, following can be used in a template:

```
from pprint import pprint
print(20 * "#" + "GLOBALS")
pprint(globals())
print(20 *"#" + "LOCALS")
pprint(locals())
```

EXEC TEMPLATE RETURN VALUES

Following rules apply for return values from executed Python code templates:

- returned variables in template must be global (e.g., **`global result`**)
- internally **`result`** variable is appended to **`template_thread_data`** list as 2nd column (1st is device name)
- value of **`result`** of type string is printed in tabular view for individual device
- multi-line **`result`** string has special handling in printing
- **`result`** set to **`list`**, **`dict`**, **`None`** or empty string is not printed
- string value of **`header`** displaces default tabular value header during printing
- setting **`header`** to **`None`** prevents printing of header
- setting **`header`** to **`"default"`** string restores default header value
- after every printing, **`result`** variable other than **`list`** or **`dict`** type is set to empty string to avoid repeated printing in multi-profile

Using **`None`** is handy for printing custom outputs. Aligning **`result`** and **`header`** is useful for simple embedded printing where use of vertically aligned **"|"** character can mimic tabular view.

DATA SHARING IN MULTI- PROFILE

The typical use of data sharing between profiles is post-processing and printing summaries of gathered outputs from individual devices in multi-profile operation. Either global **result** of type **list** or **dict** or **result_adv** (result advanced) variable can be used. In the latter one any data structures can be passed. The use of both **result** and **result_adv** is to support situations when **result** of type **string** is used for simple embedded tabular printing and **result_adv** used to

produce aggregate data. If returned from template, **result_adv** is appended to **template_thread_data** list as 3rd column where from it can be accessed typically by sub-sequent exec templates in multi-profile operation.

PYTHON EXEC() CONTEXT

When writing exec templates in a following way:

```
def msg(msg):
  print(msg)
def main():
  msg('test')
main()
```

This error may appear due to Python **exec()** context handling:

```
NameError: name 'msg' is not defined
```

Probably the most elegant solution is to use approach with arbitrary class, for example:

```
class DynamicExecutor:
    def msg(self, msg):
        print(msg)

    def main(self):
        self.msg('test')

executor = DynamicExecutor()
executor.main()
```

WRAPPER TO TEMPLATE-OPS

Following is a sample wrapper example for on-box and off-box operation which would make the command more purpose specific, for example – to make turn template-ops into tool for operating SRX:

```
scale-out --srx-load                ->  template-ops --mprofile load
scale-out --srx-sessions            ->  template-ops --mprofile sessions
scale-out --srx-sessions 100.64/10  ->  template-ops --mprofile sessions –-input 100.64/10
```

Skeleton code rewriting CLI parameters:

```
#!/usr/bin/python
import runpy, sys

if len(sys.argv) > 1:
    if sys.argv[1] in ["--srx-load"]:
        sys.argv = ["", "--mprofile", "load"]
    elif sys.argv[1] in ["--srx-sessions"]:
        if len(sys.argv) == 3:
            sys.argv = ["", "--mprofile", "sessions", "--input", sys.argv[2]]

        else:
            sys.argv = ["", "--mprofile", "sessions"]
    else:
        print("Unknown parameters")
        sys.exit()
    runpy.run_module("template-ops", run_name="__main__")
```

JUNOS EVENT-OPTIONS TRIGGER

TEMPLATE AS A JUNOS DAEMON

SYNC FROM OFF-BOX TO ON-BOX

MULTIPLE INSTANCES OFF-BOX