Nokia SR OS Napalm Driver Summary of Methods

This document details all the napalm methods supported by the Nokia SR OS driver. We've provided details of which arguments, options and response parameter mapping to SR OS' Yang data model objects or model driven (MD)-CLI command responses, are supported for each method. While the SR OS Yang model is extensive, there are a few parameters which are currently only retrievable by CLI for this version of SR OS.

We've also included any known considerations or limitations per method. If SR OS does not support an expected response parameter, we've returned the following:

- String = ""
- Boolean = false
- Integer = -1
- Float = -1.0

We welcome suggestions and contributions to the driver. Please contact the Nokia owners of this repository for how to contribute.

Nokia SR OS supported Napalm methods:

Configuration Methods:

cli(commands):

- CLI commands provided as input to this method must be in Nokia SR OS MD-CLI format and will be executed on the target SR OS as written, and any response or errors will be returned.
- Considerations – use of this method opens a single ssh session to the SR OS device. If a configuration command is used, it is recommended to do so in "exclusive" mode to lock the candidate config from changes by other users. If this mode is used though, the user will need to "commit" the changes within the same ssh session to avoid all changes to the exclusive candidate config being lost once the session is closed.

Method Name	Output Parameters with Datatype	SR OS Path
cli(commands)	Command dependent	MD-CLI will be used for output of [commands]

commit_config(message=""):

- This command executes a commit on the target SR OS in the context of which type of candidate config the user is changing. Any response or errors will be returned.
- Considerations – use of this method atomically would mean the candidate configuration context is the "global" candidate, which may have many changes by many users in it. For this reason, this method's use atomically should be done with caution. Our recommendation is to use this as

part of a script containing execution of multiple commands such that this commit would be in the same session to the device, and hence in the same “exclusive” candidate config context.

- Limitations: ○ SR OS does not support to commit the config with the message parameter passed to the method.

Method Name	Output Parameters with Datatype	SR OS Path
commit_config(message=””)		Text -> MD-CLI commands [“commit”] XML -> ncclient commit()

compare_config(optional_args=None):

- This command executes a compare on the target SR OS in the context of the candidate config is in, versus the running config. Any response or errors will be returned.
- Considerations – use of this method atomically would mean the candidate configuration context is the “global” candidate, which may have many changes by many users in it. Our recommendation is to use this as part of a script containing execution of multiple commands such that this compare would be in the same session to the device, and hence in the same “exclusive” candidate config context.
- Optional_args takes arguments as “json_format”: True/False , by default it is false. It is used when config is in XML format, to find the diff between two configs and whether to print it in json format or not.
- Limitations:
    - When paired with load_merge_candidate() / load_replace_candidate() methods:
        - ✦ When the configuration passed to these methods is in MD-CLI format, compare_config() would be able to return the difference. This is because the same “exclusive” config session will remain open in the MD-CLI for multiple commands to be run.
        - ✦ When the configuration passed to these methods is in XML format, compare config would return the difference. This is accomplished by using a “dictdiffer” comparison library executed on the host executing the Napalm command. Results are returned for all add, remove, and changes to the configuration. It should be noted that for compare results of type change for objects in list form in SR OS, the returned result will provide a numeric value for the position in the list of the change, and not the ID of the object. The ID of the object is stored as a separate leaf object in the SR OS Yang models.

Method Name	Output Parameters with Datatype	SR OS Path

compare_config()	Config dependent	Text -> MD-CLI commands [“environment more false”, “compare”] XML -> Supported with dictdiffer library

discard_config():

- This command executes a discard on the target SR OS in the context of the candidate config context of the session. Any response or errors will be returned.
- Considerations – use of this method atomically would mean the candidate configuration context is the “global” candidate, which may have many changes by many users in it. Our recommendation is to use this as part of a script containing execution of multiple commands such that this discard would be in the same session to the device, and hence in the same “exclusive” candidate config context.

Method Name	Output Parameters with Datatype	SR OS Path
discard_config()		Text -> MD-CLI commands [“discard”] XML -> ncclient discard_changes()

Load_merge_candidate(filename=None, config=None)

- This method adds the provided configuration to the candidate config. This can be formatted as MD-CLI or XML. Any response or errors will be returned.
- Limitation
  - When the configuration passed to these methods is in MD-CLI format, compare_config() would be able to return the difference. This is because the same “exclusive” config session will remain open in the MD-CLI for multiple commands to be run.
  - When the configuration passed to these methods is in XML format, compare config would return the difference. This is accomplished by using a “dictdiffer” comparison library executed on the host executing the Napalm command. Results are returned for all add, remove, and changes to the configuration. It should be noted that for compare results of type change for objects in list form in SR OS, the returned result will provide a numeric value for the position in the list of the change, and not the ID of the object. The ID of the object is stored as a separate leaf object in the SR OS Yang models.

Method Name	Output Parameters with Datatype	SR OS Path
load_merge_candidate(filename=None, config=None)		MD-CLI to merge the configuration XML -> ncclient edit-config()

Load_replace_candidate(filename=None, config=None)

- This method adds the provided configuration to the candidate config. This can be formatted as MD-CLI or XML. Any response or errors will be returned.
- Limitation

- When the configuration passed to these methods is in MD-CLI format, compare_config() would be able to return the difference. This is because the same “exclusive” config session will remain open in the MD-CLI for multiple commands to be run.
- When the configuration passed to these methods is in XML format, compare config would return the difference. This is accomplished by using a “dictdiffer” comparison library executed on the host executing the Napalm command. Results are returned for all add, remove, and changes to the configuration. It should be noted that for compare results of type change for objects in list form in SR OS, the returned result will provide a

numeric value for the position in the list of the change, and not the ID of the object. The ID of the object is stored as a separate leaf object in the SR OS Yang models.

Method Name	Output Parameters with Datatype	SR OS Path
load_replace_candidate(filename=None, config=None)		MD-CLI to replace the configuration XML -> ncclient edit-config()

ping(destination, source="", ttl=128, timeout=2, size=100, count=5, vrf="")

- This method executes a ping command from the base routing or VPN context on the device via MD-CLI
- Limitation: ○ Input parameter ttl – Should be in the range of 1...128

Method Name	Output Parameters with Datatype	SR OS Path
ping(destination, source="", ttl=128, timeout=2, size=100, count=5, vrf="")	"success": {	
	probes_sent – int	MD-CLI command "ping {destination} timeout {timeout} ttl {ttl} source-address {source} size {size} count{count} router-instance {vrf}" Packets transmitted
	packet_loss – int	MD-CLI command "ping {destination} timeout {timeout} ttl {ttl} source-address {source} size {size} count{count} router-instance {vrf}" Packets transmitted – Packets received
	rtt_min – float	MD-CLI command "ping {destination} timeout {timeout} ttl {ttl} source-address {source} size {size} count{count} router-instance {vrf}" round-trip min
	rtt_max – float	MD-CLI command "ping {destination} timeout {timeout} ttl {ttl} source-address {source} size {size} count{count} router-instance {vrf}" round-trip max
	rtt_avg – float	MD-CLI command "ping {destination} timeout {timeout} ttl {ttl} source-address {source} size {size} count{count} router-instance {vrf}" round-trip avg
	rtt_stddev – float	MD-CLI command "ping {destination} timeout {timeout} ttl {ttl} source-address {source} size {size} count{count} router-instance {vrf}" round-trip stddev
	results – list [	
	ip_address – String	MD-CLI command "ping {destination} timeout {timeout} ttl {ttl} source-address {source} size {size}count{count}routerinstance{vrf}"❼# bytes from ip_address
	rtt – float] }	MD-CLI command "ping {destination} timeout {timeout} ttl {ttl} source-address {source} size {size}count{count}router-instance{vrf}" ❼ #bytes from ip_address: time

❼

❼

❼

	“error”	“Unknown host {destination}”

rollback()

- This method executes a rollback on the SR OS device via MD-CLI. It will rollback to the last configuration before the last “commit” was executed.
- Considerations: ○ It should be noted that SR OS does support storing and rolling back to older snapshots, but the napalm method does not.
    - This method should also be used with caution to ensure the changes of other users are not erased.

Method Name	Output Parameters with Datatype	SR OS Path
rollback()		MD-CLI command [“quit-config”, ”configure exclusive”, “rollback 1”, “commit”, “exit”]

traceroute(destination, source=””, ttl=255, timeout=10, vrf=””)

- This method executes a traceroute command from the base routing or VPN context onthe device via MD-CLI
- Limitation:
    - Input parameter timeout – Should be in the range of 10…60000

Method Name	Output Parameters with Datatype	SR OS Path
traceroute(destination, source=””, ttl=255, timeout=10, vrf=””)	“success”: {	
	rtt – float	MD-CLI command “traceroute {destination} wait {timeout} ttl {ttl} source-address {source} router-instance{vrf}”❼#ip_address(host_name)rtt
	ip_address – String	MD-CLI command “traceroute {destination} wait {timeout} ttl {ttl} source-address {source} router-instance {vrf}” ❼#ip_address

	host_name – String }	MD-CLI command “traceroute {destination} wait {timeout} ttl {ttl} source-address {source} router-instance {vrf}” ❼#ip_address(host_name)
	“error”	“Unknown host {destination}”

Get Methods:

get_arp_table(vrf=””):

- This method returns the arp table in the context of the vrf passed in the method
- All parameters are retrieved for this method via the SR OS Yang models from the running config.
- Limitations ○ Retrieval is only for IPv4 entries in the current driver

Method Name	Output Parameters with Datatype	SR OS Path
get_arp_table(vrf=””)	interface – String	state/router/interface/interface-name and state/service/vprn/interface/interface-name
	mac – String	state/router/interface/ipv4/neighbor-discovery/neighbor/mac-address and state/service/vprn/interface/ipv4/neighbor-discovery/neighbor/mac-address
	ip – String	state/router/interface/ipv4/neighbor-discovery/neighbor/ipv4-address and state/router/interface/ipv4/neighbor-discovery/neighbor/ipv4-address
	age – float	state/router/interface/ipv4/neighbor-discovery/neighbor/timer and state/router/interface/ipv4/neighbor-discovery/neighbor/timer

get_bgp_config(group=””, neighbor=””):

- This method returns the bgp config in the context of the group and neighbor passed in the method for both base and VRFs.
- All parameters are retrieved for this method via the SR OS Yang models for the running config.

Method Name	Output Parameters with Datatype	SR OS Path
get_bgp_config(group=””, neighbor=””)	type – String	“internal” if local_as==remote_as or remote_as is not configured and configure/router/bgp/group/type or configure/service/vprn/bgp/group/type are set to “internal”; “external” if remote_as is configured, else “no-type”
	description – String	configure/router/bgp/group/description and configure/service/vprn/bgp/group/description

	apply_groups – String list	configure/router/bgp/group/apply-groups and configure/service/vprn/bgp/group/apply-groups
	multihop_ttl – int	configure/router/bgp/group/multihop and configure/service/vprn/bgp/group/multihop
	multipath – Boolean	configure/router/bgp/group/multipath/{ibgp,ebgp} > 1 and configure/service/vprn/bgp/group/multipath/{ibgp,ebgp} > 1 Note: Does not check if ecmp is > 1
	local_address – String	configure/router/bgp/group/local-address and configure/service/vprn/bgp/group/local-address
	local_as – int	configure/router/bgp/group/local-as/as-number and configure/service/vprn/bgp/group/local- as/as-number
	remote_as – int	configure/router/bgp/group/peer-as and configure/service/vprn/bgp/group/peer-as
	import_policy – String list	configure/router/bgp/group/import/policy and configure/service/vprn/bgp/group/import/policy

	export_policy – String list	configure/router/bgp/group/export/policy and configure/service/vprn/bgp/group/export/policy
	remove_private_as – Boolean	configure/router/bgp/group/remove- private/limited and configure/service/vprn/bgp/group/remove- private/limited
	prefix_limit – dict	configure/router/bgp/group/prefix-limit/family, limit, threshold, maximum and configure/service/vprn/bgp/group/ prefix- limit/family, limit, threshold, maximum
	neighbors: {	configure/router/bgp/neighbor/ip-address and configure/service/vprn/bgp/neighbor/ip-address
	description – String	configure/router/bgp/neighbor/description and configure/service/vprn/bgp/neighbor/description
	import_policy – Sting list	configure/router/bgp/neighbor/import/policy and configure/service/vprn/bgp/neighbor/import/policy
	export_policy – String list	configure/router/bgp/neighbor/export/policy and configure/service/vprn/bgp/neighbor/export/policy
	local_address – String	configure/router/bgp/neighbor/local-address and configure/service/vprn/bgp/neighbor/local- address
	local_as – int	configure/router/bgp/neighbor/local-as/as- number and configure/service/vprn/bgp/neighbor/local-as/as- number

	remote_as – int	configure/router/bgp/neighbor/peer-as and configure/service/vprn/neighbor/peer-as
	authentication_key – String	configure/router/bgp/neighbor/authentication- keychain and configure/service/vprn/neighbor/authentication-keychain Note: not taken from group
	prefix_limit – dict	configure/router/bgp/neighbor/prefix- limit/family, limit, threshold, maximum and configure/service/vprn/bgp/neighbor/ prefix- limit/family, limit, threshold, maximum
	route_reflector_client – Boolean	configure/router/bgp/{neighbor,group,global}/client-reflect + cluster/cluster-id and configure/service/vprn/bgp/{neighbor,group,global}/client-reflect + cluster/cluster-id
	nhs – Boolean	configure/router/bgp/{neighbor,group}/next-hop-self and configure/service/vprn/{neighbor,group}/next-hop-self

get_bgp_neighbors():

- This method returns the bgp neighbors information from the SR OS config and state Yang datastores for the running config for base routing (excluding management) and all VRFs.
- All parameters are retrieved for this method via the SR OS Yang models.

Method Name	Output Parameters with Datatype	SR OS Path
get_bgp_neighbors()	router_id – String	state/router/oper-router-id and state/service/vprn/oper-router-id
	“peers” : {	
	ip-address : {	state/router/bgp/neighbor/ip-address and state/service/vprn/ bgp/neighbor/ip-address
	local_as – int	configure/router/bgp/{neighbor,group}/local-as/as-number and configure/service/vprn/bgp/{neighbor,group}/local-as/as- number
	remote_as – int	configure/router/bgp/{neighbor,group}/peer-as and configure/service/vprn/{neighbor,group}/peer-as
	is_up – Boolean	state/router/bgp/neighbor/statistics/session- state and state/service/vprn/ bgp/neighbor/statistics/session-state
	is_enabled – Boolean	configure/router/bgp/neighbor/admin-state and configure/service/vprn/neighbor/admin-state
	description – String	configure/router/bgp/neighbor/description and configure/service/vprn/neighbor/description

	uptime – int	state/system/current-time – state/router/bgp/neighbor/statistics/last-established-time
	“address_family”: {	
	ipv4/ipv6 : {	
	received_prefixes – int	state/router/bgp/neighbor/statistics/family-prefix/ipv4/received and state/service/vprn/bgp/neighbor/statistics/family -prefix/ipv4/received
	sent_prefixes – int	state/router/bgp/neighbor/statistics/family-prefix/ipv4/sent and state/service/vprn/ bgp/neighbor/statistics/family-prefix/ ipv4/sent
	accepted_prefixes – int	state/router/bgp/neighbor/statistics/family-prefix/ipv4/active and state/service/vprn/ bgp/neighbor/statistics/family-prefix/ ipv4/active

get_bgp_neighbors_detail(neighbor_address=””):

- This method returns the bgp neighbor information details from the SR OS config and state Yang datastores for the running config for the provided neighbor address (default:all) in base routing or VRFs.
- Parameters are retrieved for this method via the SR OS Yang models.

Method Name	Output Parameters with Datatype	SR OS Path
get_bgp_neighbors_detail (neighbor_address=””)	router_name – String	state/router/router-name and state/service/vprn/service-name
	“peer_as”: [{	configure/router/bgp/neighbor/peer-as and configure/service/vprn/neighbor/peer- as
	is_up – Boolean	True if state/router/bgp/neighbor/statistics/session-state and state/service/vprn/bgp/neighbor/statistics/session-state == “established”
	local_as – int	configure/router/bgp/neighbor/local-as/as-number and configure/service/vprn/bgp/neighbor/local-as/as-number
	remote_as – int	configure/router/bgp/neighbor/peer-as and configure/service/vprn/neighbor/peer-as
	router_id – String	state/router/peer-identifier and state/service/vprn/peer-identifier

	local_address – String	state/router/bgp/neighbor/operational-local-address and state/service/vprn/neighbor/operational-local-address
	routing_table – String	configure/router/router-name and configure/service/vprn/service-name
	local_address_configured – Boolean	True if (configure/router/bgp/{neighbor,group}/local-address and configure/service/vprn/{neighbor,group}/local-address) is not empty
	local_port – int	state/router/bgp/neighbor/statistics/local-port and state/service/vprn/bgp/neighbor/statistics/local-port
	remote_address – String	configure/router/bgp/neighbor/ip-address and configure/service/vprn/bgp/neighbor/ip-address
	remote_port – int	state/router/bgp/neighbor/statistics/peer-port and

		state/service/vprn/bgp/neighbor/statistics/peer- port
	multihop – Boolean	True if (configure/router/bgp/{neighbor,group,global}/multihop and configure/service/vprn/bgp/{neighbor,group,global}/multihop) > 0
	multipath – Boolean	configure/router/bgp/multipath/{ibgp,ebgp} > 1 and configure/service/vprn/bgp/multipath/{ibgp,ebgp} > 1
	remove_private_as – Boolean	configure/router/bgp/{neighbor,group,global}/remove-private/limited and configure/service/vprn/bgp/{neighbor,group,global}/remove-private/limited
	import_policy – String list	configure/router/bgp/{neighbor,group,global}/import/policy and configure/service/vprn/bgp/{neighbor,group,global}/import/policy (only first level match is returned, i.e. if neighbor level policy is set then group/global levels are not returned)
	export_policy – String list	configure/router/bgp/{neighbor,group,global}/export/policy and configure/service/vprn/bgp/{neighbor,group,global}/export/policy (first match is returned)

	input_messages – int	state/router/bgp/neighbor/statistics/received/me ssages and state/service/vprn/bgp/neighbor/statistics/receiv ed/messages
	output_messages – int	state/router/bgp/neighbor/statistics/sent/messa ges and state/service/vprn/bgp/neighbor/statistics/sent/ messages
	input_updates – int	state/router/bgp/neighbor/statistics/received/up dates and state/service/vprn/bgp/neighbor/statistics/receiv ed/updates
	output_updates – int	state/router/bgp/neighbor/statistics/sent/update s and state/service/vprn/bgp/neighbor/statistics/sent/u pdates
	messages_queued_out – int	state/router/bgp/neighbor/statistics/sent/queues and state/service/vprn/bgp/neighbor/statistics/sent/q ueues
	connection_state – String	state/router/bgp/neighbor/statistics/session- state and state/service/vprn/bgp/neighbor/statistics/sessio n-state
	previous_connection_state – String	state/router/bgp/neighbor/statistics/last-state and

		state/service/vprn/bgp/neighbor/statistics/last- state
	last_event – String	state/router/bgp/neighbor/statistics/last-event and state/service/vprn/bgp/neighbor/statistics/last- event
	suppress_4byte_as – Boolean	configure/router/bgp/{neighbor,group,global}/asn-4-byte and configure/service/vprn/bgp/{neighbor,group,global}/asn-4-byte == “false”
	local_as_prepend – Boolean	configure/router/bgp/{neighbor,group,global}/local-as/prepend-global-as and configure/service/vprn/bgp/{neighbor,group,global}/local-as/prepend-global-as
	holdtime – int	state/router/bgp/{neighbor,group,global}/hold- time-interval and state/service/vprn/bgp/{neighbor,group,global}/hold- time-interval

	configured_holdtime – int	configure/router/bgp/neighbor/hold- time/seconds and configure/service/vprn/bgp/neighbor/hold- time/seconds
	keepalive – int	state/router/bgp/neighbor/keep-alive-interval and state/service/vprn/bgp/neighbor/keep-alive-interval
	configured_keepalive – int	configure/router/bgp/neighbor/keepalive and configure/service/vprn/bgp/neighbor/keepalive
	active_prefix_count – int	state/router/bgp/neighbor/statistics/family-prefix/ipv[4+6]/active and state/service/vprn/bgp/neighbor/statistics/family - prefix/ipv[4+6]/active
	received_prefix_count – int	state/router/bgp/neighbor/statistics/family-prefix/ipv[4+6]/received and state/service/vprn/bgp/neighbor/statistics/family - prefix/ipv[4+6]/received
	accepted_prefix_count – int	received - rejected
	supressed_prefix_count – int	state/router/bgp/neighbor/statistics/family-prefix/ipv[4+6]/suppressed and state/service/vprn/bgp/neighbor/statistics/family - prefix/ipv[4+6]/suppressed
	advertised_prefix_count – int	state/router/bgp/neighbor/statistics/family- prefix/ipv[4+6]/sent and state/service/vprn/bgp/neighbor/statistics/family - prefix/ipv[4+6]/sent
	flap_count – int	state/router/bgp/neighbor/statistics/established-transitions and state/service/vprn/bgp/neighbor/statistics/established-transitions

get_config(retrieve="all", full=False, sanitized=False, optional_args={"format":"cli"}):

- This method returns the running configuration from the SR OS device.
- Parameters are retrieved for this method via ncclient for Yang model objects and ssh for MD-CLI
- Parameter optional_args is a dictionary used to get config in different formats as following: ○ "xml" – It is the default format
    - "cli" – If you want to get config in MD-CLI format, use this flag ○ If not passed, by default it will return in XML format
- Limitations:
    - The Nokia SR OS napalm driver currently ignores the "full" and "sanitized" optional arguments and will always return the full config.

Method Name	Output Parameters with Datatype	SR OS Path

get_config(retrieve="all", full=False, sanitized=False)	running – String	MD-CLI command "admin show configuration | no-more" XML -> configure(source=running)
	candidate – String	MD-CLI command ["configure global", "info| no-more"] XML -> configure(source=candidate)
	startup – String	MD-CLI command "admin show configuration | no-more" XML -> configure(source=running)

get_ environment():

- This method returns the environment information details from the SR OS config and state Yang datastores for the running config
- Limitations: ○ If the device is virtual, limited information will be available.

Method Name	Output Parameters with Datatype	SR OS Path
get_environment()	"fans": {	
	fan-slot – String :	state/chassis/fan/fan-slot
	status – Boolean }	state/chassis/hardware-oper-state
	"temperature": {	
	temperature – float	state/cpm/hardware-temperature and state/card/hardware-temperature and state/card/mda/hardware-temperature
	is_alert - Boolean	True if temperature >= 80% of (state/cpm/hardware-temperature-threshold and state/card/hardware-temperature-threshold and state/card/mda/hardware-temperature-threshold)
	is_critical – Boolean }	True if temperature >= (state/cpm/hardware-temperature-threshold and state/card/hardware-temperature-threshold and

		state/card/mda/hardware-temperature-threshold)
	“power”: {	
	status – Boolean	True if state/chassis/power-shelf/power-module/hardware-oper-state == “in-service”
	capacity – float	state/chassis/power-shelf/power-module/available-wattage
	output – float }	MD-CLI command “show chassis power-management utilization detail” ❼Current Util (output)/ total_power_modules
	“cpu” : {	
	cpu-usage – float	state/system/cpu/summary/usage/cpu-usage
	“memory” : {	
	available_ram – int	state/system/memory-pools/summary/available-memory
	used_ram – int	state/system/memory-pools/summary/total-in-use

get_ facts():

- This method returns the basic facts from the SR OS config Yang datastore for the running config of the device
- Consideration – this method will always return “Nokia” as the vendor string

Method Name	Output Parameters with Datatype	SR OS Path
get facts	uptime – float	state/system/up-time
	vendor - String	Nokia
	model – String	state/system/platform
	hostname – String	state/system/oper-name
	fqdn – String	state/system/oper-name
	os_version – String	state/system/version/version-number
	serial number - String	state/chassis/hardware-serial-number
	interface_list - List	state/router/interface/interface-name

get_ interfaces():

- This method returns the physical port/interface (layer 2) AND logical interface (layer 3) information from the SR OS config and state Yang datastores for the running config of the device.
- Limitations ○ SR OS currently does not support the “last_flapped” parameter for physical ports, so the Nokia SR OS napalm driver will return a -1 value.

Method Name	Output Parameters with Datatype	SR OS Path
get_interfaces	is_up – Boolean	state/port/oper-state and state/router/interface/if-oper-status

	is_enabled – Boolean	configure/port/admin-state and configure/router/interface/admin-state
	description – String	configure/port/description and configure/router/interface/description
	last_flapped – float	For port this is not currently supported by SR OS and will return -1.0 and state/router/interface/last-oper-change
	speed – int	state/port/ethernet/oper-speed and 1. configure/router/interface/port 2. state/port/ethernet/oper-speed
	MTU – Bytes	configure/port/ethernet/mtu and state/router/interface/oper-ip-mtu
	mac_address – String	state/port/hardware-mac-address and configure/router/interface/mac

get_ interfaces_counters():

- This method returns the physical port/interface (layer 2) AND logical interface (layer 3) stats from the SR OS state Yang datastore for the device
- Limitations:
    - For logical interfaces, not all fields are supported. The table provided identifies which fields are not supported for logical interfaces and will return the default values mentioned at the beginning of this document.

Method Name	Output Parameters with Datatype	SR OS Path
get_interfaces_counters	tx_errors – int	state/port/statistics/out-errors and for router/interface, -1 will be returned
	rx_errors – int	state/port/statistics/in-errors and for router/interface, -1 will be returned
	tx_discards – int	state/port/statistics/out-discards and state/router/interface/statistics/ip/out-discard-packets
	rx_discards – int	state/port/statistics/in-discards and for router/interface, -1 will be returned
	tx_octets – int	state/port/statistics/out-octets and state/router/interface/statistics/ip/out-octets
	rx_octets – int	state/port/statistics/in-octets and state/router/interface/statistics/ip/in-octets
	tx_unicast_packets – int	state/port/statistics/out-unicast-packets and for router/interface, not supported
	rx_unicast_packets – int	state/port/statistics/in-unicast-packets and for router/interface, not supported
	tx_multicast_packets – int	state/port/statistics/out-multicast-packets and for router/interface, not supported

	rx_multicast_packets – int	state/port/statistics/in-multicast-packets and for router/interface, not supported
	tx_broadcast_packets – int	state/port/statistics/out-broadcast-packets and
		for router/interface, not supported
	rx_broadcast_packets – int	state/port/statistics/in-broadcast-packets and for router/interface, not supported

get_interfaces_ip():

- This method returns the logical IP4 and IPv6 interface details from the SR OS config and state Yang datastore of the base routing context and all configured L3 VPNs of the device

Method Name	Output Parameters with Datatype	SR OS Path
get_interfaces_ip()	interface_name – String {	state/router/interface/interface-name and state/service/vprn/interface/interface-name
	"ipv4/ipv6" : {	
	ip_address – String {	state/router/interface/ipv4/primary/address and state/router/interface/ipv4/secondary/address and state/router/interface/ipv6/address/ipv6-address and state/service/vprn/interface/ipv4/primary/address and state/service/vprn/interface/ipv4/secondary/address and state/service/vprn /interface/ ipv6/address/ipv6-address
	prefix_length – int	state/router/interface/ipv4/primary/prefix-length and state/router/interface/ipv4/secondary/prefix-length and state/router/interface/ipv6/address/prefix-length and state/service/vprn/interface/ipv4/primary/prefix-length and state/service/vprn/interface/ipv4/secondary/prefix-length and state/service/vprn/interface/ ipv6/address/prefix-length

get_ ipv6_neighbors_table():

- This method returns the IPv6 neighbor details from the SR OS MD-CLI for the base routing context and all configured VPNs of the device

Method Name	Output Parameters with Datatype	SR OS Path

get_ipv6_neighbors_table()	interface – String	MD-CLI command “show router router_name neighbor” ❼ Interface
	mac – String	MD-CLI command “show router router_name neighbor” ❼MAC Address
	ip – String	MD-CLI command “show router router_name neighbor” ❼Ipv6 Address
	age – float	MD-CLI command “show router router_name neighbor” ❼ Expiry
	state – String	MD-CLI command “show router router_name neighbor” ❼ State

get_lldp_neighbors():

- This method returns a list of lldp neighbors from the SR OS Yang state datastores for the device

Method Name	Output Parameters with Datatype	SR OS Path
get_lldp_neighbors()	port_id – String: {	state/port/port-id
	hostname – String	state/port/ethernet/lldp/dest-mac/remote-system/system-name
	port – String	state/port/ethernet/lldp/dest-mac/remote-system/remote-port-id

get_lldp_neighbors():

- This method returns lldp neighbor details from the SR OS Yang state datastores for the device
- Limitation ○ SR OS does not currently support the “parent_interface” string field, and will return a “” value

Method Name	Output Parameters with Datatype	SR OS Path
get_lldp_neighbors_detail (interface=””)	interface – String: [{	state/port/port-id
	parent_interface – String	Not supported
	remote_port – String	state/port/ethernet/lldp/dest-mac/remote-system/remote-port-id
	remote_port_description – String	state/port/ethernet/lldp/dest-mac/remote-system/port-description
	remote_chassis_id – String	state/port/ethernet/lldp/dest-mac/remote-system/chassis-id
	remote_system_name – String	state/port/ethernet/lldp/dest-mac/remote-system/system-name
	remote_system_description – String	state/port/ethernet/lldp/dest-mac/remote-system/system-description
	remote_system_capab – String list	state/port/ethernet/lldp/dest-mac/remote-system/sytem-supported-capabilities

	remote_system_enabled_capab – String list	state/port/ethernet/lldp/dest-mac/remote-system/sytem-enabled-capabilities

get_mac_address_table():

- This method returns a list of learnt MACs from the SR OS device via retrieval by MD-CLI
- Limitation ○ The SR OS object model does not store an "active" Boolean field per MAC, so we will return a "null"
    - SR OS currently does not support the "moves" integer field, and will return a "-1". ○ SR OS currently does not support the "last_move" float field, and will return a "-1.0"

Method Name	Output Parameters with Datatype	SR OS Path
get_mac_address_table()	mac – String	MD-CLIcommand"showservicefdb-mac"❼ MAC
	interface – String	MD-CLIcommand"showservicefdb-mac"❼ Source_Identifier
	vlan – int	MD-CLIcommand"showservicefdb-mac"❼ Source_Identifier
	active – Boolean	""
	static – Boolean	True if MD-CLI command "show service fdb-mac" ❼ Type == "static"
	moves – int	Not supported
	last_move – float	Not supported

get_network_instances(name=""):

- This method returns a list of all L2 (vpls) and L3 (vprn) network instances (including base and mgmt) for the SR OS device from the Yang configuration and state datastores • Limitation:

Method Name	Output Parameters with Datatype	SR OS Path
get_network_instances(name="")	name – String {	state/router/router-name and state/service/vprn/service-name and state/service/vpls/service-name
	name – String	state/router/router-name and state/service/vprn/service-name and state/service/vpls/service-name
	type – String	DEFAULT_INSTANCE/MGMT/L3VRF/VPLS
	"state" {	
	route-distinguisher –String	state/service/vprn/oper-router-distinguisher
	"interfaces" {	
	"interface" {	

	interface-name – Dict	state/router/interface/interface-name and state/service/vprn/interface/interface-name and state/service/vpls/interface/interface-name

get_ntp_peers():

- This method returns a list of all ntp peers for the SR OS device from the Yang state datastores

Method Name	Output Parameters with Datatype	SR OS Path
get_ntp_peers()	ip-address – String	state/system/time/ntp/peer/ip-address

get_ntp_servers():

- This method returns a list of all ntp servers for the SR OS device from the Yang state datastores

Method Name	Output Parameters with Datatype	SR OS Path
get_ntp_servers()	ip-address – String	state/system/time/ntp/server/ip-address

get_ntp_stats():

- This method returns all ntp stats for the SR OS device from the Yang config and state datastores and via MD-CLI objects.
- Limitations
    - SR OS currently does not support the “when” string field, and as such will return a “” ○ SR OS currently does not support the “reachability” integer field, and as such will return a -1
    - SR OS currently does not support the “delay” float field, and as such will return a-1.0 ○ SR OS currently does not support the “jitter” float field, and as such will return a-1.0

Method Name	Output Parameters with Datatype	SR OS Path
get_ntp_stats()	remote – String	MD-CLIcommand“show systemntpservers” ❼ Remote MD-CLIcommand“show system ntppeers” Remote
	referenceid – String	MD-CLIcommand“show systemntpservers” ❼ Reference ID MD-CLIcommand“show system ntppeers” Reference ID
	synchronized – Boolean	Trueif(MD-CLIcommand“showsystem ntp servers” ❼ State MD-CLIcommand“show system ntppeers” State) == “chosen”
	stratum – int	MD-CLIcommand“show systemntpservers” ❼ St MD-CLIcommand“showsystemntppeers”❼
	type – String	MD-CLIcommand“show systemntpservers” ❼ Type MD-CLIcommand“show system ntppeers” Type
	when – String	Not supported

❼

❼

❼

St

❼

	hostpoll – int	MD-CLIcommand“show systemntpservers” ❼ Poll MD-CLIcommand“show system ntppeers” Poll

❼

	reachability – int	Not supported
	delay – float	Not supported
	offset – float	MD-CLIcommand“show systemntpservers” ❼ Offset MD-CLIcommand“show system ntppeers” Offset
	jitter – float	Not supported

❼

get_optics():

- This method returns all optics state and stats for the SR OS device from the Yang config and state datastores.
- Limitations ○ SR OS currently does not support the input_power “avg”, ”min”, and “max” float fields, and as such will return a -1.0
    - SR OS currently does not support the output_power “avg”, ”min”, and “max” float fields, and as such will return a “-1.0”
    - SR OS currently does not support the laser_bias_current “avg”, ”min”, and “max” float fields, and as such will return a “-1.0”

Method Name	Output Parameters with Datatype	SR OS Path
get_optics()	intf_name – String {	state/port/port-id
	“physical_channels”	
	“channels”: [	
	index - int	state/port/transceiver/digital-diagnostic-monitoring/lane/lane-id
	“state” {	
	“input_power” {	
	“instant” – float	state/port/transceiver/digital-diagnostic-monitoring/lane/received-optical-power/current
	“avg” – float	Not supported
	“min” – float	Not supported
	“max” – float }	Not supported
	“output_power” {	
	“instant” – float	state/port/transceiver/digital-diagnostic-monitoring/lane/transmit-output-power/current
	“avg” – float	Not supported
	“min” – float	Not supported
	“max” – float }	Not supported
	“laser_bias_current” {	
	“instant” – float	state/port/transceiver/digital-diagnostic-monitoring/lane/transmit-bias-current/current
	“avg” – float	Not supported
	“min” – float	Not supported
	“max” – float }	Not supported

get_probes_config():

- This method returns the continuously run probe configuration for the SR OS device from the Yang config datastore.
- Limitations ○ SR OS currently only supports an ICMP ping test via the Yang object model

Method Name	Output Parameters with Datatype	SR OS Path
get_probes_config()	probe_name – String {	configure/saa/owner/owner-name
	test_name – String {	configure/saa/owner/test
	probe_type - String	"icmp-ping"
	target – String	configure/saa/owner/type/icmp-ping/destination_address
	source – String	configure/saa/owner/type/icmp-ping/source_address
	probe_count – int	configure/saa/owner/type/icmp-ping/count
	test_interval – int	configure/saa/owner/type/icmp-ping/interval

get_probes_results():

- This method returns the continuously run probe results for the SR OS device from the Yang config and state datastores and MD-CLI.
- Limitations
    - SR OS currently only supports an ICMP ping test via the Yang object mode ○ SR OS currently does not support the "global_test_min_delay", "global_test_max_delay", and "global_test_avg_delay" float fields, and as such will return a -1.0

Method Name	Output Parameters with Datatype	SR OS Path
get_probes_results()	probe_name – String {	configure/saa/owner/owner-name
	test_name – String {	configure/saa/owner/test
	target – String	configure/saa/owner/type/icmp-ping/destination_address
	source – String	configure/saa/owner/type/icmp-ping/source_address
	probe_type - String	“icmp-ping”
	probe_count – int	configure/saa/owner/type/icmp-ping/count
	rtt – float	MD-CLIcommand“show saa{test_name}” current_test roundtrip[Average]
	round_trip_jitter – float	MD-CLIcommand“show saa{test_name}” current_test roundtrip[Jitter]
	last_test_loss – float	MD-CLIcommand“show saa{test_name}” last_test Number of requests that failed to be sent out / Total number of attempts
	current_test_min_delay – float	MD-CLIcommand“show saa{test_name}” current_test roundtrip[Min]
	current_test_max_delay – float	MD-CLIcommand“show saa{test_name}” current_test roundtrip[Max]
	current_test_avg_delay – float	MD-CLIcommand“show saa{test_name}” current_test roundtrip[Average]
	last_test_min_delay – float	MD-CLIcommand“show saa{test_name}” last_test roundtrip[Min]
	last_test_max_delay – float	MD-CLIcommand“show saa{test_name}” last_test roundtrip[Max]
	last_test_avg_delay – float	MD-CLIcommand“show saa{test_name}” last_test roundtrip[Average]
	global_test_min_delay – float	Not supported
	global_test_max_delay – float	Not supported
	global_test_avg_delay – float	Not supported

❼

get_route_to(destination="", protocol="", longer=False)

- This method returns the get_route_to results based on the provided inpts, for the SR OS device from the Yang config and state datastores and MD-CLI.
- Limitations
    - SR OS currently does not support the "metric 2" integer field, and as such will return a -1 ○ SR OS currently does not support the "inactive_reason" string field, and as such will return a ""

Method Name	Output Parameters with Datatype	SR OS Path and Response
get_route_to(destination="", protocol="", longer=False)	destination – String {	MD-CLI command "show router {router_name} route-table {destination} protocol {protocol} extensive all" ❼ Dest Prefix
	protocol – String	protocol
	current_active – Boolean	MD-CLI command "show router {router_name} route-table {destination} protocol {protocol} extensive all" ❼Active
	last_active – Boolean	False
	age – int	MD-CLI command "show router {router_name} route-table {destination} protocol {protocol} extensive all" ❼ Age
	next_hop – String	MD-CLI command "show router {router_name} route-table {destination} protocol {protocol} extensiveall"❼Next-HoporIndirectNext-Hop
	outgoing_interface – String	MD-CLI command "show router {router_name} route-table {destination} protocol {protocol} extensive all" ❼Interface
	selected_next_hop – Boolean	True if next_hop exists
	preference – int	MD-CLI command "show router {router_name} route-table {destination} protocol {protocol} extensive all" ❼Preference
	inactive_reason – String	Not supported
	routing_table – String	state/router/router-name and state/service/vprn/oper-service-id
	"protocol_attributes": {	
	If BGP:	

	local_as – int	MD-CLI command “show router {router-name} bgp routes {destination} detail” ❼ LocalAS
	remote_as – int	state/router/bgp/neighbor/statistics/peer-as and state/service/vprn/bgp/neighbor/peer-as
	peer_id – String	state/router/bgp/neighbor/statistics/peer-identifier and state/service/vprn/bgp/neighbor/peer-identifier
	as_path – String	MD-CLI command “show router {router-name} bgproutes{destination} detail” ❼AS-Path
	communities – String list	MD-CLI command “show router {router-name} bgproutes{destination}detail” ❼Community
	local_preference – int	MD-CLI command “show router {router-name} bgproutes{destination} detail” ❼Loca lPref.
	preference2 – int	MD-CLI command “show router {router_name} route-table {destination} protocol {protocol} extensive all” ❼Preference
	metric – int	MD-CLI command “show router {router_name} route-table {destination} protocol {protocol} extensive all” ❼Metric
	metric2 – int	Not supported
	If ISIS:	
	level – int	MD-CLI command “show router {router_name} isis routes ip-prefix-prefix-length {destination}” ❼ level
	If OSPF:	
	cost – int	MD-CLI command “show router {router_name} ospf routes {destination}” ❼ cost

get_snmp_information()

- This method returns the snmp configuration for the SR OS device from the Yang config datastore.
- Consideration – the community acl parameter will be returned as a hashed value

Method Name	Output Parameters with Datatype	SR OS Path
get_snmp_information()	chassis_id – String	configure/system/name
	“community”: {	
	acl – String	configure/system/security/snmp/community/source-access-list
	mode – String	configure/system/security/snmp/community/access-permissions
	contact – String	configure/system/contact
	location – String	configure/system/location

get_users()

- This method returns the configured users for the SR OS device from the Yang config datastore.
- Considerations:
    - SR OS does not support the “level” parameter but does support users being members of groups inheriting the same user privileges. A mapping table is provided in the SR OS napalm driver for operators to map the privilege group names they configure in SR OS at /configure/system/security/aaa/local-profiles/profile to 0-15 integer levels. The default is 0.
    - The method maps the level parameter based on parsing the profile name, looking for an integer value between 0-15 and maps this integer accordingly. eg Profile name Level-7 maps to 7 ○ The password will always be returned in hashed form for security purposes.

Method Name	Output Parameters with Datatype	SR OS Path
get_users()	username – String {	/configure/system/security/user-params/local-user/user/user-name
	level – int	/configure/system/security/user-params/local-user/user/console/member
	password – String	configure/system/security/user-params/local-user/user/password
	sshkeys – String list	configure/system/security/user-params/local-user/user/public-keys/rsa/rsa-key/key-value

is_alive()

- This method returns True if NETCONF connection is open or SSH connection is open for the SR OS device.

Method Name	Output Parameters with Datatype	SR OS Path
is_alive()		Returns True if NETCONF connection is open as well as SSH connection is open

This concludes the documentation of the Nokia SR OS napalm methods.

Any questions or concerns can be directed to the owners of the repository and we will do our best to respond quickly.

Thanks for your support, from the Nokia ION team.