Lenovo Network

Reference Guide

for Lenovo Network Ruby Application Programming Interface, Version 1.0

Lenovo™

Note: Before using this information and the product it supports, read the general information in the *Safety information and Environmental Notices and User Guide* documents on the Lenovo *Documentation* CD and the *Warranty Information* document that comes with the product.

First Edition (January 2018)

Preface

The *Lenovo Network Ruby Application Programming Interface Reference Guide* describes how to use the Lenovo Network Ruby Application Programming Interface.

Intended Audience for This Guide

This guide is intended for network installers and system administrators engaged in configuring and maintaining a network. The administrator is expected to be familiar with Ethernet concepts, IP addressing, and Spanning Tree Protocol. The user of this book is also expected to be familiar with the Ruby programming language.

Typographic Conventions

The following table describes the typographic styles used in this book.

Table 1. *Typographic Conventions*

Typeface or Symbol	Meaning	Example
`ABC123`	This type is used for names of commands, files, and directories used within the text. It also depicts on-screen computer output and prompts.	View the `readme.txt` file. `Main#`
`ABC123`	This bold type appears in command examples. It shows text that must be typed in exactly as shown.	`Main#` **`sys`**
<ABC123>	This italicized type appears in command examples as a parameter placeholder. Replace the indicated text with the appropriate real name or value when using the command. Do not type the brackets. This also shows book titles, special terms, or words to be emphasized.	To establish a Telnet session, enter: `host#` **`telnet`** *<IP address>* Read your *User's Guide* thoroughly.
[ ]	Command items shown inside brackets are optional and can be used or excluded as the situation demands. Do not type the brackets.	`host#` **`ls`** [**`-a`**]
|	The vertical bar (|) is used in command examples to separate choices where multiple options exist. Select only one of the listed options. Do not type the vertical bar.	`host#` **`set left|right`**
AaBbCc123	This block type depicts menus, buttons, and other controls that appear in Web browsers and other graphical interfaces.	Click the **Save** button.

Contents

Overview

This document describes the classes available in the Lenovo Network Ruby Application Programming Interface. You install this interface automatically when you install a Lenovo Network Development Toolkit such as the Lenovo Network Development Toolkit for Chef or the Lenovo Network Development Toolkit for Puppet.

This API works with Ruby 2.3.1 or later.

The following classes are available:

- "Connection Management" on page 8
- "Address Resolution Protocol Management" on page 11
- "IP Management" on page 15
- "Internet Group Management Protocol Management" on page 19
- "Link Aggregation Control Protocol Management" on page 23
- "Link Aggregation Group Management" on page 25
- "Link Layer Discovery Protocol Management" on page 35
- "Multiple Spanning Tree Protocol Management" on page 42
- "Spanning Tree Protocol Management" on page 50
- "System Management" on page 54
- "Telemetry Management" on page 72
- "Virtual Link Aggregation Group Management" on page 83
- "Virtual Local Area Network Management" on page 95
- "VLAN Interface Management" on page 99
- "Virtual Router Redundancy Protocol Management" on page 102
- "REpresentational State Transfer Management" on page 111

Connection Management

The `Connect` class contains methods for establishing a connection with the switch.

These methods are in the file `connect.rb`.

new(file)

Initialize node reachability parameters and establish a connection with the node (switch).

Syntax

`Connect.new(`*<parameters>*`)`

where *parameters* is a dictionary with the following key-value pairs::

```
{
 "transport"=>"<protocol>",
 "port"=>"<port>",
 "ip"=>"<IP>",
 "user"=>"<user>",
 "password"=>"<password>"
}
```

where:

Parameter	**Description**
protocol	Either `http` or `https`.
port	The port being used for the connection; either `8090` or `443`.
IP	The switch IP address (String).
user	The user name (String).
password	The password for the user (String).

Returns

A connection object.

Example

Given the file `config.yml` that contains:

```
transport : 'http'
port : '443'
ip : '10.240.177.120'
user : 'admin'
password : 'admin'
```

the following Ruby code returns the connection object `#<Connect:0x000000026ec010>`:

```
require 'cnos-rbapi/connect'
require 'yaml'

params = YAML.load_file('./config.yml')
conn = Connect.new(params)
```

getCookie()

Gets a cookie for the current session.

Syntax

```
Connect.getCookie()
```

Returns

A string; the cookie for the current node connection.

getHdr()

Gets header information for the current node connection.

Syntax

```
Connect.getHeader()
```

Returns

A string; the header for the current node connection.

getIp()

Gets the IP address for the current node connection.

Syntax

```
Connect.getIp()
```

Returns

A string; the IP address for the current node connection.

getPassword()

Gets the password for the current node connection.

Syntax

```
Connect.getPassword()
```

Returns

A string; the password for the current node connection.

getPort()

Gets the port for the current node connection.

Syntax

```
Connect.getPort()
```

Returns

A string; the port for the current node connection.

getTransport()

Gets the transport protocol for the current node connection.

Syntax

```
Connect.getTransport()
```

Returns

A string; the transport protocol for the current node connection.

getUser()

Gets the user for the current node connection.

Syntax

```
Connect.getUser()
```

Returns

A string; the user name for the current node connection.

Address Resolution Protocol Management

The `Arp` class methods manage ARP on the node.

These methods are in the file `arp.rb`.

get_arp_intf_prop(conn, intf)

Gets the ARP properties of the specified interface.

Syntax

`Arp.get_arp_intf_prop(<conn>, <intf>)`

where:

Parameter	**Description**
conn	A connection object for the node.
intf	The Layer 3 interface name; a string.

Returns

The ARP properties of the interface as a JSON response:

Response Body
 	
(JSON)
	`[`
	`  {`
	`    "if_name": "<if_name>",`
	`    "ageout_time": "<ageout_time>"`
	`  }`
	`]`

where:

Element	**Description**
if_name	IP interface name (String). **Note:** The interface must exist.
ageout_time	The global ARP entry age-out time, in seconds; an integer from 60-28800. Default value: 1500 seconds.

get_arp_sys_prop(conn)

Gets the ARP properties for the node.

Syntax

`Arp.get_arp_sys_prop(<conn>)`

where *conn* is a connection object for the node.

Returns

The ARP properties of the system as a JSON response:

Response Body
 	
(JSON)
	`[`
	`  {`
	`    "if_name": "`*<if_name>*`",`
	`    "ageout_time": "`*<ageout_time>*`"`
	`  }`
	`]`

where:

Element	**Description**
if_name	IP interface name (String). **Note:** The interface must exist.
ageout_time	The global ARP entry age-out time, in seconds; an integer from 60-28800. Default value: 1500 seconds.

set_arp_intf_prop(conn, intf)

Sets the ARP properties of the specified interface.

Syntax

```
Arp.get_arp_intf_prop(<conn>, <intf>, <parameters>)
```

where:

Parameter	**Description**
conn	A connection object for the node.
intf	The Layer 3 interface name; a string.

parameters	A dictionary containing the following JSON key-value pairs:
	`[`
	`  {`
	`    "if_name": "`*<if_name>*`",`
	`    "ageout_time": "`*<ageout_time>*`"`
	`  }`
	`]`

if_name	The IP interface name (Str). **Note:** The interface must exist.
ageout_time	The global ARP entry age-out time, in seconds; an integer from 60-28800. Default value: 1500 seconds.

Returns

The ARP properties of the interface as a JSON response:

Response Body
 	
(JSON)
	[
	  {
	    "if_name": "*<if_name>*",
	    "ageout_time": "*<ageout_time>*"
	  }
	]

where:

Element	**Description**
if_name	IP interface name (String). **Note:** The interface must exist.
ageout_time	The global ARP entry age-out time, in seconds; an integer from 60-28800. Default value: 1500 seconds.

set_arp_sys_prop(conn, parameters)

Sets the ARP properties of the node.

Syntax

```
Arp.set_arp_intf_prop(<conn>, <parameters>)
```

where:

Parameter	**Description**
conn	A connection object for the node.

parameters	A dictionary containing the following JSON key-value pairs:
	[
	  {
	    "if_name": "*<if_name>*",
	    "ageout_time": "*<ageout_time>*"
	  }
	]

if_name	The IP interface name (String). **Note:** The interface must exist.
ageout_time	The global ARP entry age-out time, in seconds; an integer from 60-28800. Default value: 1500 seconds.

Returns

The ARP properties of the node as a JSON response:

Response Body
 	
(JSON)
	`[`
	`  {`
	`    "if_name": "<if_name>",`
	`    "ageout_time": "<ageout_time>"`
	`  }`
	`]`

where:

Element	**Description**
if_name	IP interface name (String). **Note:** The interface must exist.
ageout_time	The global ARP entry age-out time, in seconds; an integer from 60-28800. Default value: 1500 seconds.

IP Management

The `Ipintf` class methods manage the IP interfaces on the node.

These methods are in the file `ip_intf.rb`.

get_ip_prop_all(conn)

Get the IP properties of all interfaces.

Syntax

`Ipintf.get_ip_prop_all(`*<conn>*`)`

where *conn* is a connection object.

Returns

A JSON response containing the IP properties of all interfaces on the node:

Response Body
 	
(JSON)
	`[`
	`  {`
	`    "if_name": "<if_name>",`
	`    "bridge_port": "<bridge_port>",`
	`    "mtu": {mtu},`
	`    "ip_addr": "<ip_addr>",`
	`    "ip_prefix_len": "<ip_prefix_len>",`
	`    "vrf_name": "<vrf_name>",`
	`    "admin_state": "<admin_state>"`
	`  }`
	`]`

where:

Element	**Description**
if_name	IP interface name (String). **Note:** The interface must exist.
bridge_port	Whether or not the port is a bridge port; one of `yes` (default), `no`.
mtu	The maximum transmission unit, in bytes; an integer from 64-9216. Default value: 1500.
ip_addr	IP address for the interface.
ip_prefix_len	IP address mask; a positive integer from 1-32.
vrf_name	The name of the VRF to which the interface belongs (String). **Note:** The named VRF must exist.
admin_state	The admin status; one of `up`, `down`.

get_ip_prop_int(conn, intf)

Get the IP properties of the specified interface.

Syntax

`Ipintf.get_ip_prop_all(<conn>, <intf>)`

where:

Parameter	Description
conn	A connection object for the node.
intf	The Layer 3 interface name; a string.

Returns

A JSON response containing the IP properties of the specified node:

Response Body
 	
(JSON)
	`[`
	`  {`
	`    "if_name": "<if_name>",`
	`    "bridge_port": "<bridge_port>",`
	`    "mtu": {mtu},`
	`    "ip_addr": "<ip_addr>",`
	`    "ip_prefix_len": "<ip_prefix_len>",`
	`    "vrf_name": "<vrf_name>",`
	`    "admin_state": "<admin_state>"`
	`  }`
	`]`

where:

Element	Description
if_name	IP interface name (String). **Note:** The interface must exist.
bridge_port	Whether or not the port is a bridge port; one of `yes` (default), `no`.
mtu	The maximum transmission unit, in bytes; an integer from 64-9216. Default value: 1500.
ip_addr	IP address for the interface.
ip_prefix_len	IP address mask; a positive integer from 1-32.
vrf_name	The name of the VRF to which the interface belongs (String). **Note:** The named VRF must exist.
admin_state	The admin status; one of `up`, `down`.

update_ip_prop_intf(conn, intf, parameters)

Update the IP properties of the specified interface.

Syntax

`Ipintf.update_ip_prop_intf(<conn>, <intf>, <parameters>)`

where:

Parameter	Description
conn	A connection object for the node.
intf	The IP interface name (String). **Note:** The interface must exist.

parameters	A dictionary containing the following JSON key-value pairs:
	`{`
	`  "if_name": "<if_name>",`
	`  "bridge_port": "<bridge_port>",`
	`  "mtu": <mtu>,`
	`  "ip_addr": "<ip_addr>",`
	`  "ip_prefix_len": "<ip_prefix_len>",`
	`  "vrf_name": "<vrf_name>",`
	`  "admin_state": "<admin_state>"`
	`}`

if_name	IP interface name (String). **Note:** The interface must exist.
bridge_port	Whether or not the port is a bridge port; one of `yes` (default), `no`.
mtu	The maximum transmission unit, in bytes; an integer from 64-9216. Default value: 1500.
ip_addr	IP address for the interface.
ip_prefix_len	IP address mask; a positive integer from 1-32.
vrf_name	The name of the VRF to which the interface belongs (String). **Note:** The named VRF must exist.
admin_state	The admin status; one of `up`, `down`.

Returns

A JSON response containing the updated IP properties:

Response Body
 	
(JSON)
	`{`
	`  "if_name": "<if_name>",`
	`  "bridge_port": "<bridge_port>",`
	`  "mtu": {mtu},`
	`  "ip_addr": "<ip_addr>",`
	`  "ip_prefix_len": "<ip_prefix_len>",`
	`  "vrf_name": "<vrf_name>",`
	`  "admin_state": "<admin_state>"`
	`}`

where:

Element	**Description**
if_name	IP interface name (String). **Note:** The interface must exist.
bridge_port	Whether or not the port is a bridge port; one of `yes` (default), `no`.
mtu	The maximum transmission unit, in bytes; an integer from 64-9216. Default value: 1500.
ip_addr	IP address for the interface.
ip_prefix_len	IP address mask; a positive integer from 1-32.
vrf_name	The name of the VRF to which the interface belongs (String). **Note:** The named VRF must exist.
admin_state	The admin status; one of `up`, `down`.

Internet Group Management Protocol Management

The `Igmp` class methods manage IGMP on the node.

These methods are in the file `igmp.rb`.

get_igmp_snoop_prop(conn)

Gets IGMP snooping properties of the system.

Syntax

`Igmp.get_igmp_snoop_prop(`*<conn>*`)`

where *conn* is the connection object of the node.

Returns

A JSON response containing IGMP snooping properties:

Response Body
 	
(JSON)
	`{`
	  `"ena_igmp_snoop": "`*<ena_igmp_snoop>*`"`
	`}`

where:

Parameter	**Description**
ena_igmp_ snoop	Enables IGMP snooping globally on all VLANs; one of *yes* (default), `no`. If disabled globally, IGMP snooping is disabled on all VLANs, regardless of the per-VLAN setting of IGMP snooping. If IGMP snooping is enabled globally, the per-VLAN setting of IGMP snooping takes effect.

get_igmp_vlan_prop(conn, vlan_id)

Gets IGMP Snooping properties for the specified VLAN.

Syntax

`Igmp.get_igmp_vlan_prop(`*<conn>*`, `*<vlan_id>*`)`

where:

Parameter	**Description**
conn	connection object to the node
vlan_id	VLAN ID number.

Returns

A JSON response containing the IGMP snooping properties for the VLAN:

Response Body
 	
(JSON)
	`[`
	`  {`
	`    "vlan_id": "<vlan_id>",`
	`    "ena_igmp_snoop": "<ena_igmp_snoop>"`
	`  }`
	`]`

where:

Parameter	**Description**
vlan_id	VLAN number.
ena_igmp_ snoop	Enables IGMP snooping on a VLAN; one of `yes` (default), `no`.

set_igmp_snoop_prop(conn, parameters)

Updates the global IGMP snooping properties of the system.

Syntax

`Igmp.set_igmp_snoop_prop(<conn>, <parameters>)`

where:

Parameter	**Description**
conn	A connection object for the node

parameters	A dictionary that requires the following key-value pair:
	`{`
	`  "ena_igmp_snoop": "<ena_igmp_snoop>"`
	`}`

ena_igmp_ snoop	Enables IGMP snooping globally on all VLANs; one of *yes* (default), `no`. If disabled globally, IGMP snooping is disabled on all VLANs, regardless of the per-VLAN setting of IGMP snooping. If IGMP snooping is enabled globally, the per-VLAN setting of IGMP snooping takes effect.

Returns

A JSON response containing the updated IGMP snooping properties:

Response Body
 	
(JSON)
	`{`
	`  "ena_igmp_snoop": "<ena_igmp_snoop>"`
	`}`

set_igmp_vlan_prop(conn, vlan_id, parameters)

Updates the IGMP snooping properties of the specified VLAN.

Syntax

`Igmp.set_igmp_vlan_prop(<conn>, <vlan_id>, <parameters>)`

where:

Parameter	**Description**
conn	A connection object for the node.
vlan_id	VLAN ID number. **Note:** The VLAN must exist.

parameters	A dictionary that requires the following key-value pairs:
	`{`
	`    "vlan_id": "<vlan_id>",`
	`    "ena_igmp_snoop": "<ena_igmp_snoop>",`
	`    "fast_leave": "<fast_leave>",`
	`    "query_interval": "<query_interval>",`
	`    "version": "<version>",`
	`}`

ena_igmp_snoop	Enables IGMP snooping globally on all VLANs; one of *yes* (default), `no`. If disabled globally, IGMP snooping is disabled on all VLANs, regardless of the per-VLAN setting of IGMP snooping. If IGMP snooping is enabled globally, the per-VLAN setting of IGMP snooping takes effect.
ena_igmp_snoop	(Optional) Whether to enable IGMP snooping on a VLAN; one of `yes`, `no`. Default value: `yes`.
fast_leave	One of `yes`, `no`. Default value: `no`.
query_interval	(Optional) IGMP query interval, in seconds; an integer from 1-18000. Default value: 125.
version	(Optional) IGMP Snooping version number; one of 2, 3. Default value: 3.

Returns

A JSON response containing the updated IGMP snooping properties of the specified VLAN:

Response Body
 	
(JSON)
	`{`
	`    "vlan_id": "<vlan_id>",`
	`    "ena_igmp_snoop": "<ena_igmp_snoop>",`
	`    "fast_leave": "<fast_leave>",`
	`    "query_interval": "<query_interval>",`
	`    "version": "<version>",`
	`}`

where:

Element	**Description**
vlan_id	VLAN number. **Note:** The VLAN must exist.
ena_igmp_ snoop	(Optional) Whether to enable IGMP snooping on a VLAN; one of `yes`, `no`. Default value: `yes`.
fast_leave	One of `yes`, `no`. Default value: `no`.
query_interval	(Optional) IGMP query interval, in seconds; an integer from 1-18000. Default value: 125.
version	(Optional) IGMP Snooping version number; one of 2, 3. Default value: 3.

Link Aggregation Control Protocol Management

The `Lacp` class methods manage LACP on the node.

These methods are in the file `lacp.rb`.

get_lacp(conn)

Gets the LACP properties of the node.

Syntax

`Lacp.get_lacp(<conn>)`

where *conn* is a connection object for the node.

Returns

A JSON response containing the LACP properties of the node:

Response Body	`{`
	`"sys_prio": "<sys_prio>",`
(JSON)	`"max_bundle": "<max_bundle>",`
	`"interfaces": [`
	`  {`
	`    "if_name": "<if_name>",`
	`    "lag_mode": "<lag_mode>",`
	`    "lacp_prio": "<lacp_prio>",`
	`    "lacp_timeout": "<lacp_timeout>"`
	`  }`
	`}`

where:

Element	Description
sys_prio	LACP system priority.; a positive integer from 1-65535.
max_bundle	The supported maximum number of links per LAG.; a positive integer.
if_name	Ethernet interface name (String).
lag_mode	LAG mode; one of `lacp_active`, `lacp_passive`, `no_lacp`
lacp_prio	LACP priority for the physical port a positive integer from 1-65535.
lacp_timeout	LACP timeout for the physical port; one of `short`, `long`.

update_lacp(conn, parameters)

Updates the LACP properties of the node.

Syntax

`Lacp.update_lacp(<conn>, <sys_prio>)`

where:

Element	**Description**
conn	A connection object for the node.

parameters	A dictionary that requires the following key-value pair:
	`{`
	`  "sys_prio": "<sys_prio>",`
	`}`

sys_prio	LACP system priority; a positive integer from 1-65535. Default value: 32768.

Returns

A JSON response containing the updated LACP properties:

Response Body
 	
(JSON)
	`{`
	`  "sys_prio": "<sys_prio>",`
	`}`

where:

Element	**Description**
sys_prio	LACP system priority; a positive integer from 1-65535.

Link Aggregation Group Management

The `Lag` class provides methods for managing LAG configuration on the node.

These methods are in the file `lag.rb`.

create_lag(conn, parameters)

Creates a LAG.

Syntax

```
Lag.create_lag(<conn>, <parameters>)
```

where:

Parameter	Description
conn	A connection object for the node.

parameters	A dictionary of the following key-value pairs:
	`{`
	`"lag_id": "<lag_id>",`
	`"interfaces": [`
	`  {`
	`    "if_name": "<if_name>",`
	`    "lag_mode": "<lag_mode>",`
	`    "lacp_prio": "<lacp_prio>",`
	`    "lacp_timeout": "<lacp_timeout>"`
	`  }`
	`]`
	`}`
	Up to 32 interfaces can be added.

lag_id	LAG identifier; a positive integer from 1-4096.

`interfaces`	Physical interface members of the LAG; a dictionary of the following key-value pairs:
	`{`
	`"lag_id": "<lag_id>",`
	`"interfaces": [`
	`  {`
	`    "if_name": "<if_name>",`
	`    "lag_mode": "<lag_mode>",`
	`    "lacp_prio": "<lacp_prio>",`
	`    "lacp_timeout": "<lacp_timeout>"`
	`  }`
	` ]`
	`}`
	Up to 32 interfaces can be added.

if_name	Ethernet interface name (String). **Note:** The interface must exist.
lag_mode	LAG mode; one of `lacp_active`, `lacp_passive`, `no_lacp`.

Parameter	Description
lacp_prio	(Optional) LACP priority for the physical port; a positive integer from 1-65535. Default value: 32768.
lacp_timeout	(Optional) LACP timeout for the physical port; one of `short`, `long`. Default value: `long`.

Returns

A JSON response containing the parameters of the created LAG:

Response Body (JSON)	`{ "lag_id": "<lag_id>", "lag_name": "<lag_name>", "interfaces": [ { "if_name": "<if_name>", "lag_mode": "<lag_mode>", "lacp_prio": "<lacp_prio>", "lacp_timeout": "<lacp_timeout>" } ] }`

where:

Parameter	Description
conn	A connection object for the node.
lag_id	LAG identifier; a positive integer from 1-4096.

delete_lag(conn, lag_id)

Deletes a LAG.

Syntax

```
Lag.delete_lag(<conn>, <lag_id>)
```

where:

Parameter	Description
lag_name	LAG name.
lag_id	LAG identifier; a positive integer from 1-4096.
`interfaces`	Physical interface members of the LAG. Up to 32 interfaces can be added.
if_name	Ethernet interface name (String). **Note:** The interface must exist.
lag_mode	LAG mode; one of `lacp_active`, `lacp_passive`, `no_lacp`.

Parameter	**Description**
lacp_prio	LACP priority for the physical port; a positive integer from 1-65535. Default value: 32768.
lacp_timeout	LACP timeout for the physical port; one of `short`, `long`. Default value: `long`.

get_all_lag(conn)

Gets the properties of all LAGs.

Syntax

`Lag.get_all_lag(<`*conn*`>)`

where *conn* is a connection object for the node.

Returns

A JSON response containing the properties of all LAGs:

Response Body

(JSON)

```
[
  {
    "lag_name":  "<lag_name>",
    "lag_id":  "<lag_id>",
    "interfaces": [
      {
        "if_name":  "<if_name>",
        "lag_mode":  "<lag_mode>",
        "lacp_prio":  "<lacp_prio>",
        "lacp_timeout":  "<lacp_timeout>"
      }
    ],
    "suspend_individual":  "<status>",
  }
]
```

where:

Element	**Description**
lag_name	The name of the LAG; a string.
lag_id	LAG identifier; an integer from 1-65535
interfaces	Physical interface members of the LAG; an integer from 1-32.
if_name	Ethernet interface name. **Note:** The interface must exist.
lag_mode	LAG mode; one of `lacp_active`, `lacp_passive`, `no_lacp`.
lacp_prio	LACP priority for the physical port; an integer from 1-65535. Default value: 32768.
lacp_timeout	LACP timeout for the physical port; one of `short`, `long`. Default value: `long`.

Element	Description
suspend_ individual	If the LAG does not get the LACP BPUD from peer ports the port aggregation, the result is one of the following: ● `Yes` - LACP on the the ports is suspended rather than put into individual state. ● `No`: LAG on the ports is put into individual state. Default value: `No`.

get_lag_prop(conn, lag_id)

Gets properties of the specified LAG.

Syntax

`Lag.get_lag_prop(<conn>, <lag_id>)`

where:

Element	Description
conn	A connection object for the node.
lag_id	LAG identifier; an integer from 1-65535.

Returns

A JSON response containing the properties of the specified LAG:

Response Body
 	
(JSON)
	[
	  {
	    "lag_name": "<lag_name>",
	    "lag_id": "<lag_id>",
	    "interfaces": [
	      {
	        "if_name": "<if_name>",
	        "lag_mode": "<lag_mode>",
	        "lacp_prio": "<lacp_prio>",
	        "lacp_timeout": "<lacp_timeout>"
	      }
	    ],
	    "suspend_individual": "<status>",
	  }
	]

where:

Element	Description
lag_name	The name of the LAG; a string.
lag_id	LAG identifier; an integer from 1-65535
interfaces	Physical interface members of the LAG; an integer from 1-32.

Element	Description
if_name	Ethernet interface name. **Note:** The interface must exist.
lag_mode	LAG mode; one of `lacp_active`, `lacp_passive`, `no_lacp`.
lacp_prio	LACP priority for the physical port; an integer from 1-65535. Default value: 32768.
lacp_timeout	LACP timeout for the physical port; one of `short`, `long`. Default value: `long`.
suspend_ individual	If the LAG does not get the LACP BPUD from peer ports the port aggregation, the result is one of the following: ● `Yes` - LACP on the the ports is suspended rather than put into individual state. ● `No`: LAG on the ports is put into individual state. Default value: `No`.

get_load_balance(conn)

Gets load balance properties for port aggregations.

Syntax

`Lag.get_load_balance(`*<conn>*`)`

where *conn* is a connection object for the node.

Returns

A JSON response containing the load balance properties for port aggregations:

Response Body
 	
(JSON)
	`{`
	`  "destination-ip"   : "<destination-ip>"`
	`  "destination-mac"  : "<destination-mac>"`
	`  "destination-port" : "<destination-port>"`
	`  "source-dest-ip"   : "<source-dest-ip>"`
	`  "source-dest-mac"  : "<source-dest-mac>"`
	`  "source-dest-port" : "<source-dest-port>"`
	`  "source-interface" : "<source-interface>"`
	`  "source-ip"        : "<source-ip>"`
	`  "source-mac"       : "<source-mac>"`
	`  "source-port”      : "<source-port>"`
	`}`

where:

Element	Description
destination-ip	Load distribution on the destination IP address.
destination-mac	Load distribution on the destination MAC address.

Element	**Description**
destination-port	Load distribution on the destination TCP/UDP port.
source-dest-ip	Load distribution on the source and destination IP address.
source-dest-mac	Load distribution on the source and destination MAC address.
source-dest-port	Load distribution on the source and destination TCP/UDP port.
source-interface	Load distribution on the source ethernet interface.
source-ip	Load distribution on the source IP address.
source-mac	Load distribution on the source MAC address.
source-port	Load distribution on the source TCP/UDP port.

update_lag(conn, parameters)

Updates properties of a LAG.

Syntax

`Lag.update_lag(<conn>, <lag_id>, <parameters>)`

where:

Parameter	**Description**
conn	A connection object for the node.
lag_id	LAG identifier; a positive integer from 1-4096.

parameters	A dictionary of the following key-value pairs:
	`{`
	`"lag_name": "<lag_name>",`
	`"lag_id": "<lag_id>",`
	`"interfaces": [`
	`  {`
	`    "if_name": "<if_name>",`
	`    "lag_mode": "<lag_mode>",`
	`    "lacp_prio": "<lacp_prio>",`
	`    "lacp_timeout": "<lacp_timeout>"`
	`  }`
	` ]`
	`"suspend_individual":  "<status>",`
	`"min_links":  "<min_links>",`
	`}`
	`}`

lag_name	LAG name; a string.
lag_id	LAG identifier; a positive integer from 1-4096.

Parameter	Description

`interfaces`	Physical interface members of the LAG; a dictionary of the following key-value pairs:
	`{`
	`  "if_name": "<if_name>",`
	`  "lag_mode": "<lag_mode>",`
	`  "lacp_prio": "<lacp_prio>",`
	`  "lacp_timeout": "<lacp_timeout>"`
	`}`
	Up to 32 interfaces can be added.

if_name	Ethernet interface name (String). **Note:** The interface must exist.
lag_mode	LAG mode; one of `lacp_active`, `lacp_passive`, `no_lacp`.
lacp_prio	(Optional) LACP priority for the physical port; a positive integer from 1-65535. Default value: 32768.
lacp_timeout	(Optional) LACP timeout for the physical port; one of `short`, `long`. Default value: `long`.
suspend_ individual	(Optional) If the LAG does not get the LACP BPUD from peer ports the port aggregation, the result is one of the following: ● `Yes` - LACP on the the ports is suspended rather than put into individual state. ● `No`: LAG on the ports is put into individual state. Default value: `No`.
min_links	The minimum number of LACP links; an integer from 1-65535. Default value: 1.

Returns

A JSON response containing the updated LAG properties:

Response Body
 	
(JSON)
	`[`
	`  {`
	`    "lag_id":  "<lag_id>",`
	`    "lag_name":  "<lag_name>",`
	`    "interfaces": [`
	`      {`
	`        "if_name":  "<if_name>",`
	`        "lag_mode":  "<lag_mode>",`
	`        "lacp_prio":  "<lacp_prio>",`
	`        "lacp_timeout":  "<lacp_timeout>"`
	`      }`
	`    ],`
	`    "suspend_individual":  "<status>",`
	`    "min_links":  "<min_links>",`
	`  }`
	`]`

where:

Parameter	Description
lag_id	LAG identifier; a positive integer from 1-4096.
lag_name	LAG name; a string.

`interfaces`	Physical interface members of the LAG; a dictionary of the following key-value pairs:
	`{`
	`"if_name": "<if_name>",`
	`"lag_mode": "<lag_mode>",`
	`"lacp_prio": "<lacp_prio>",`
	`"lacp_timeout": "<lacp_timeout>"`
	`}`

if_name	Ethernet interface name (String).
lag_mode	LAG mode; one of `lacp_active`, `lacp_passive`, `no_lacp`.
lacp_prio	(Optional) LACP priority for the physical port; a positive integer from 1-65535.
lacp_timeout	(Optional) LACP timeout for the physical port; one of `short`, `long`.
suspend_ individual	(Optional) If the LAG does not get the LACP BPUD from peer ports the port aggregation, the result is one of the following: ● `Yes` - LACP on the the ports is suspended rather than put into individual state. ● `No`: LAG on the ports is put into individual state.
min_links	The minimum number of LACP links; an integer from 1-65535.

update_lag_load_balance(conn, parameters)

Updates the load balance properties for the port aggregation.

Syntax

```
Lag.update_lag_load_balance(<conn>, <parameters>)
```

where:

Parameter	**Description**
conn	A connection object for the node.
parameters	A dictionary of the following key-value pairs: `{` `"destination-ip" : "`*<destination-ip>*`"` `"destination-mac" : "`*<destination-mac>*`"` `"destination-port" : "`*<destination-port>*`"` `"source-dest-ip" : "`*<source-dest-ip>*`"` `"source-dest-mac" : "`*<source-dest-mac>*`"` `"source-dest-port" : "`*<source-dest-port>*`"` `"source-interface" : "`*<source-interface>*`"` `"source-ip" : "`*<source-ip>*`"` `"source-mac" : "`*<source-mac>*`"` `"source-port” : "`*<source-port>*`"` `}`
destination-ip	Load distribution on the destination IP address.
destination-mac	Load distribution on the destination MAC address.
destination-port	Load distribution on the destination TCP/UDP port.
source-dest-ip	Load distribution on the source and destination IP address.
source-dest-mac	Load distribution on the source and destination MAC address.
source-dest-port	Load distribution on the source and destination TCP/UDP port.
source-interface	Load distribution on the source ethernet interface.
source-ip	Load distribution on the source IP address.
source-mac	Load distribution on the source MAC address.
source-port	Load distribution on the source TCP/UDP port.

Returns

A JSON response containing the updated load balance properties:

Response Body
 	
(JSON)
	{
	  "destination-ip" : "*<destination-ip>*"
	  "destination-mac" : "*<destination-mac>*"
	  "destination-port" : "*<destination-port>*"
	  "source-dest-ip" : "*<source-dest-ip>*"
	  "source-dest-mac" : "*<source-dest-mac>*"
	  "source-dest-port" : "*<source-dest-port>*"
	  "source-interface" : "*<source-interface>*"
	  "source-ip" : "*<source-ip>*"
	  "source-mac" : "*<source-mac>*"
	  "source-port” : "*<source-port>*"
	}

where:

Element	**Description**
destination-ip	Load distribution on the destination IP address.
destination-mac	Load distribution on the destination MAC address.
destination-port	Load distribution on the destination TCP/UDP port.
source-dest-ip	Load distribution on the source and destination IP address.
source-dest-mac	Load distribution on the source and destination MAC address.
source-dest-port	Load distribution on the source and destination TCP/UDP port.
source-interface	Load distribution on the source ethernet interface.
source-ip	Load distribution on the source IP address.
source-mac	Load distribution on the source MAC address.
source-port	Load distribution on the source TCP/UDP port.

Link Layer Discovery Protocol Management

The `Lldp` class methods manage LLDP on the node.

These methods are in the file `lldp.rb`.

get_lldp_all_intf(conn)

Gets the LLDP properties of all interfaces.

Syntax

`Lldp.get_lldp_all_intf(<conn>)`

where *conn* is a connection object for the node.

Returns

A JSON response containing the global LLDP properties of all interfaces:

Response Body
 	
(JSON)
	`{`
	`  "reinit delay": "<reinit delay>",`
	`  "transit interval": "<transmit interval>",`
	`  "transmit delay": "<transmit delay>"`
	`}`

where:

Element	**Description**
if_name	Ethernet interface name (String).
ena_lldp_rx	Enables or disables LLDP frame reception on a physical interface; one of `yes`, `no`.
ena_lldp_tx	Enables or disable sLLDP frame transmission on a physical interface; one of `yes`, `no`.

get_lldp_intf(conn, intf)

Gets the LLDP properties for a specified Ethernet interface.

Syntax

`Lldp.get_lldp_intf(<conn>, <intf>)`

where:

Parameter	**Description**
conn	A connection object for the node.
intf	The Ethernet interface name (String). **Note:** The interface must exist.

Returns

A JSON response containing the global LLDP properties for the specified Ethernet interface:

Response Body
 	
(JSON)
	`{`
	`  "if_name": "<if_name>",`
	`  "ena_lldp_rx": "<ena_lldp_rx>",`
	`  "ena_lldp_tx": "<ena_lldp_tx>"`
	`}`

where:

Element	**Description**
if_name	Ethernet interface name (String).
ena_lldp_rx	Enables or disables LLDP frame reception on a physical interface; one of `yes`, `no`.
ena_lldp_tx	Enables or disable sLLDP frame transmission on a physical interface; one of `yes`, `no`.

get_lldp_intf_neighbor(conn, intf)

Gets LLDP neighbor information for a specified interface.

Syntax

`Lldp.get_lldp_intf_neighbor(<conn>, <intf>)`

where:

Parameter	**Description**
conn	A connection object for the node.
intf	The Ethernet interface name (String). **Note:** The interface must exist.

Returns

A JSON response containing the LLDP information for the specified interface:

Response Body
 	
(JSON)
	`{`
	`  "if_name": "<if_name>",`
	`  "capability": "<capability>",`
	`  "rx ttl": "<rx ttl>",`
	`  "system name": "<system name>",`
	`  "system description": "<system description>"`
	`}`

where:

Element	**Description**
if_name	Ethernet interface name (String).
capability	Remote switch capability; one of `(B) – Bridge`, `(R) – Router`.
rx ttl	The TTL.
system name	Remote system name.
system description	Remote system description.

get_lldp_intf_neighbor_all(conn)

Gets LLDP neighbor information for all interfaces.

Syntax

`Lldp.get_lldp_intf_neighbor_all(<conn>)`

where *conn* is the connection object to the node.

Returns

A JSON response containing LLDP neighbor information for all interfaces:

Response Body
 	
(JSON)
	`[`
	`{`
	`"if_name": "<if_name>",`
	`"capability": "<capability>",`
	`"rx ttl": "<rx ttl>",`
	`"system name": "<system name>",`
	`"system description": "<system description>"`
	`}`
	`]`

where:

Element	**Description**
if_name	Ethernet interface name (Str). **Note:** The Ethernet interface must exist.
capability	Remote switch capability; one of `(B) – Bridge`, `(R) – Router`.
rx ttl	The TTL.
system name	Remote system name.
system description	Remote system description.

get_lldp_intf_stats(conn, intf)

Gets LLDP interface statistics for the specified interface.

Syntax

`Lldp.get_lldp_intf_stats(<conn>, <intf>)`

where:

Parameter	Description
conn	A connection object for the node.
intf	The Ethernet interface name (String). **Note:** The interface must exist.

Returns

A JSON response containing the interface statistics:

Response Body (JSON)	`{ "total frames": "<total_frames>", "total tlvs discarded": "<total_tlvs_discarded>", "total frames transmitted": "<total_frames_transmitted>", "total errrored frames": "<total_errrored_frames>", "total frames discarded": "<total_frames_discarded>", "total entries aged": "<total_entries_aged>", "total tlvs unrecognized": "<total_tlvs_unrecognized>" }`

where:

Element	Description
total_frames	The total number of LLDP frames received.
total_tlvs_ discarded	The total number of LLDP TLVs discarded.
total_frames_trans mitted	The total number of LLDP frames transmitted.
total_errrored_ frames	The total number of frames received with errors.
total_frames_ discarded	The total number of discarded frames.
total_entries_aged	The total number of entries aged out.
total_tlvs_ unrecognized	The total number of unrecognized LLDP TLVs.

get_lldp_prop(conn)

Gets global LLDP properties of the system.

Syntax

`Lldp.get_lldp_prop(<conn>)`

where *conn* is the connection object for the node.

Returns

A JSON response containing the global LLDP properties:

Response Body
(JSON)
	`{`
	`  "reinit delay": "<reinit delay>",`
	`  "transit interval": "<transmit interval>",`
	`  "transmit delay": "<transmit delay>"`
	`}`

where:

Element	**Description**
reinit delay	The number of seconds until LLDP re-initialization is attempted on an interface; an integer from 1-10.
transmit interval	The time interval, in seconds, between transmissions of LLDP messages; an integer from 5-32768.
transmit delay	The number of seconds for transmission delay; an integer from 1-8192.

update_lldp_intf(conn, intf, parameters)

Updates the LLDP properties for a specified interface.

Syntax

`Lldp.update_lldp_intf(<conn>, <intf>, <parameters>)`

where:

Parameter	**Description**
conn	A connection object for the node.
intf	The IP interface name (String). **Note:** The interface must exist.

parameters	A dictionary containing the following JSON key-value pairs:
	`{`
	`  "if_name": "<if_name>",`
	`  "ena_lldp_rx": "<ena_lldp_rx>",`
	`  "ena_lldp_tx": "<ena_lldp_tx>"`
	`}`

if_name	Ethernet interface name (String). **Note:** The Ethernet interface must exist.

Parameter	**Description**
ena_lldp_rx	Enables or disables LLDP frame reception on a physical interface; one of `yes` (default), `no`.
ena_lldp_tx	Enables or disables LLDP frame transmission on a physical interface; one of `yes` (default), `no`.

Returns

A JSON response containing the updated LLDP interface properties:

Response Body
 	
(JSON)
	`{`
	`  "if_name": "<if_name>",`
	`  "ena_lldp_rx": "<ena_lldp_rx>",`
	`  "ena_lldp_tx": "<ena_lldp_tx>"`
	`}`

where:where:

Element	**Description**
if_name	Ethernet interface name (String).
ena_lldp_rx	Enables or disables LLDP frame reception on a physical interface; one of `yes`, `no`.
ena_lldp_tx	Enables or disables LLDP frame transmission on a physical interface; one of `yes`, `no`.

update_lldp_prop(conn, parameters)

Updates the global LLDP properties.

Syntax

```
Lldp.update_lldp_prop(<conn>, <parameters>)
```

where:

Parameter	**Description**
conn	A connection object for the node.

parameters	A dictionary containing the following JSON key-value pairs:
	`{`
	`  "reinit delay": "<reinit delay>",`
	`  "transit interval": "<transmit interval>",`
	`  "transmit delay": "<transmit delay>"`
	`}`

reinit delay	The number of seconds until LLDP re-initialization is attempted on an interface; an integer from 1-10. Default value: 2 seconds.

Parameter	Description
transmit interval	The time interval, in seconds, between transmissions of LLDP messages; an integer from 5-32768. Default value: 30 seconds.
transmit delay	The number of seconds for transmission delay; an integer from 1-8192. Default value: 2 seconds.

Returns

A JSON response containing the updated global LLDP properties:

Response Body
 	
(JSON)
	`{`
	  `"reinit delay": "<`*reinit delay*`>",`
	  `"transit interval": "<`*transmit interval*`>",`
	  `"transmit delay": "<`*transmit delay*`>"`
	`}`

where:

Element	Description
reinit delay	The number of seconds until LLDP re-initialization is attempted on an interface; an integer from 1-10.
transmit interval	The time interval, in seconds, between transmissions of LLDP messages; an integer from 5-32768.
transmit delay	The number of seconds for transmission delay; an integer from 1-8192.

Multiple Spanning Tree Protocol Management

The `Mstp` class methods manage MSTP on the node.

These methods are in the file `mstp.rb`.

create_mstp_inst(conn, parameters)

Creates an MSTP instance.

Syntax

`Mstp.create_mstp_inst(<conn>, <parameters>)`

where:

Parameter	**Description**
conn	A connection object for the node.

parameters	A dictionary containing the following JSON key-value pairs:
	`{`
	`  "instance_id": "<instance_id>",`
	`  "instance_prio": "<instance_prio>",`
	`  "vlans": [`
	`    {`
	`      "vlan_id": "<vlan_id>"`
	`    }`
	`  ]`
	`}`

instance_id	MST instance ID; an integer from 0-64. Instance 0 refers to the CIST.
instance_prio	Sets the instance bridge priority; an integer from 0-61440. Default value: 32768.
vlan_id	Maps a range of VLANs to a multiple spanning tree instance (MSTI); an integer from 1-4094.

Returns

A JSON response containing information about the MSTP instance:

Response Body
 	
(JSON)
	`{`
	`  "instance_id": "<instance_id>",`
	`  "instance_prio": "<instance_prio>",`
	`  "vlans": [`
	`    {`
	`      "vlan_id": "<vlan_id>"`
	`    }`
	`  ]`
	`}`

where:

Element	Description
instance_id	MST instance ID; an integer from 0-64. Instance 0 refers to the CIST.
instance_prio	Sets the instance bridge priority; an integer from 0-61440.
vlan_id	Maps a range of VLANs to a multiple spanning tree instance (MSTI); an integer from 1-4094.

del_mstp_inst(conn, instance_id)

Deletes an MSTP instance.

Syntax

`Mstp.del_mstp_inst(`*<conn>*`, `*<instance_id>*`)`

where:

Parameter	Description
conn	A connection object for the node.
instance_id	The MST instance ID; an integer from 0-64. Instance 0 refers to the CIST.

get_mstp_inst(conn, instance_id)

Gets properties of an MSTP instance.

Syntax

`Mstp.get_mstp_inst(`*<conn>*`, `*<instance_id>*`)`

where:

Parameter	Description
conn	A connection object for the node.
instance_id	The MST instance ID; an integer from 0-64. Instance 0 refers to the CIST.

Returns

A JSON response containing the properties of the MSTP instance:

Response Body
 	
(JSON)
	`{`
	`"instance_id": "<instance_id>",`
	`"instance_prio": "<instance_prio>",`
	`"vlans": [`
	`  {`
	`    "vlan_id": "<vlan_id>"`
	`  }`
	`]`
	`}`

where:

Element	**Description**
instance_id	MST instance ID; an integer from 0-64. Instance 0 refers to the CIST.
instance_prio	Sets the instance bridge priority; an integer from 0-61440.
vlan_id	Maps a range of VLANs to a multiple spanning tree instance (MSTI); an integer from 1-4094.

get_mstp_inst_all(conn)

Gets properties of all MSTP instances.

Syntax

```
Mstp.get_mstp_inst_all(<conn>)
```

where *conn* is the connection object for the node.

Returns

A JSON response containing the properties of all MSTP instances:

Response Body
 	
(JSON)
	`[`
	`  {`
	`  "instance_id": "<instance_id>",`
	`  "instance_prio": "<instance_prio>",`
	`  "vlans": [`
	`    {`
	`      "vlan_id": "<vlan_id>"`
	`    }`
	`    ]`
	`  }`
	`]`

where:

Element	Description
instance_id	MST instance ID; an integer from 0-64. Instance 0 refers to the CIST.
instance_prio	Sets the instance bridge priority; an integer from 0-61440.
vlan_id	Maps a range of VLANs to a multiple spanning tree instance (MSTI); an integer from 1-4094.

get_mstp_inst_intf(conn, instance_id, intf)

Gets interface properties of a specified MSTP instance.

Syntax

```
Mstp.get_mstp_inst_intf(<conn>, <instance_id>, <intf>)
```

where:

Parameter	Description
conn	A connection object for the node.
instance_id	The MST instance ID; an integer from 0-64. Instance 0 refers to the CIST.
intf	The interface in the MSTP instance.

Returns

A JSON response containing the properties of the specified instance:

Response Body (JSON)	`{ "instance_id": "<instance_id>", "instance_prio": "<instance_prio>", "vlans": [ { "vlan_id": "<vlan_id>" } ] }`

where:

Element	Description
instance_id	MST instance ID; an integer from 0-64. Instance 0 refers to the CIST.
instance_prio	Sets the instance bridge priority; an integer from 0-61440.
vlan_id	Maps a range of VLANs to a multiple spanning tree instance (MSTI); an integer from 1-4094.

get_mstp_sys_prop(conn)

Gets global MSTP properties of the system.

Syntax

`Mstp.get_mstp_sys_prop(`*<conn>*`)`

where *conn* is the connection object for the node.

Returns

A JSON response containing the global MSTP properties of the system:

Response Body
 	
(JSON)
	`{`
	`  "region_name": "<region_name>"`
	`  "revision": "<revision>"`
	`}`

where:

Element	**Description**
region_name	Region name; a string up to 32 characters long.
revision	Revision number; an integer from 0-65535.

update_mstp_inst(conn, instance_id, parameters)

Updates the properties of an MSTP instance.

Syntax

`Mstp.update_mstp_inst(`*<conn>*`, `*<instance_id>*`, `*<parameters>*`)`

where:

Parameter	**Description**
conn	A connection object for the node.
instance_id	The MST instance ID; an integer from 0-64. Instance 0 refers to the CIST.

parameters	A dictionary containing the following JSON key-value pairs:
	`{`
	`  "instance_id": "<instance_id>",`
	`  "instance_prio": "<instance_prio>",`
	`  "vlans": [`
	`    {`
	`      "vlan_id": "<vlan_id>"`
	`    }`
	`  ]`
	`}`

instance_id	MST instance ID; an integer from 0-64. Instance 0 refers to the CIST.

Parameter	Description
instance_prio	Sets the instance bridge priority; an integer from 0-61440. Default value: 32768.
vlan_id	Maps a range of VLANs to a multiple spanning tree instance (MSTI); an integer from 1-4094.

Returns

A JSON response containing the updated MSTP properties of the system:

Response Body
 	
(JSON)
	`{`
	  `"instance_id": "<instance_id>",`
	  `"instance_prio": "<instance_prio>",`
	  `"vlans": [`
	    `{`
	      `"vlan_id": "<vlan_id>"`
	    `}`
	   `]`
	`}`

where:

Element	Description
instance_id	MST instance ID; an integer from 0-64. Instance 0 refers to the CIST.
instance_prio	Sets the instance bridge priority; an integer from 0-61440.
vlan_id	Maps a range of VLANs to a multiple spanning tree instance (MSTI); an integer from 1-4094.

update_mstp_inst_intf(conn, instance_id, intf, parameters)

Updates interface properties of a specified MSTP instance.

Syntax

```
Mstp.update_mstp_inst_intf(<conn>, <instance_id>, <intf>,
<parameters>)
```

where:

Parameter	Description
conn	A connection object for the node.
instance_id	The MST instance ID; an integer from 0-64. Instance 0 refers to the CIST.
intf	The interface in the MSTP instance

Parameter	**Description**

parameters	A dictionary containing the following JSON key-value pairs:
	`{`
	`  "if_name": "<if_name>",`
	`  "path_cost": "<path_cost>",`
	`  "port_prio": "<port_prio>"`
	`}`

if_name	Interface name. **Note:** The interface must exist.
path_cost	The port path-cost value on the specified MST instance; either an integer from 1-200000000 or `auto` (default) to base the path-cost on port speed.
port_prio	The port priority, in increments of 32, on the specified MST instance; a multiple of 32 from 0-224. Default value: 128.

Returns

A JSON response containing the updated interface properties of the MSTP instance:

Response Body
 	
(JSON)
	`{`
	`  "if_name": "<if_name>",`
	`  "path_cost": "<path_cost>",`
	`  "port_prio": "<port_prio>"`
	`}`

where:

Element	**Description**
if_name	Interface name.
path_cost	The port path-cost value on the specified MST instance; either an integer from 1-200000000 or `auto` to base the path-cost on port speed.
port_prio	The port priority, in increments of 32, on the specified MST instance; a multiple of 32 from 0-224.

update_mstp_sys_prop(conn, parameters)

Updates MSTP properties of the node.

Syntax

```
Mstp.update_mstp_sys_prop(<conn>, <parameters>)
```

where:

conn	A connection object for the node.

parameters	A dictionary containing the following JSON key-value pairs:
	`{`
	`  "instance_id": "<instance_id>",`
	`  "instance_prio": "<instance_prio>",`
	`  "vlans": [`
	`    {`
	`      "vlan_id": "<vlan_id>"`
	`    }`
	`  ]`
	`}`

instance_id	MST instance ID; an integer from 0-64. Instance 0 refers to the CIST.
instance_prio	Sets the instance bridge priority; an integer from 0-61440. Default value: 32768.
vlan_id	Maps a range of VLANs to a multiple spanning tree instance (MSTI); an integer from 1-4094.

Returns

A JSON response containing the global MSTP properties of the system:

Response Body
 	
(JSON)
	`{`
	`  "region_name": "<region_name>"`
	`  "revision": "<revision>"`
	`}`

where:

Element	**Description**
region_name	Region name; a string up to 32 characters long.
revision	Revision number; an integer from 0-65535.

Spanning Tree Protocol Management

The `Stp` class methods manage STP on the node.

These methods are in the file `stp.rb`.

get_all_stp(conn)

Gets STP properties of all interfaces.

Note: These properties are supported by all STP nodes.

Syntax

`Stp.get_all_stp(`*<conn>*`)`

where *conn* is the connection object for the node.

Returns

A JSON response containing the STP properties of all interfaces:

Response Body
 	
(JSON)
	`{`
	`  "if_name":  "<if_name>",`
	`  "edge_port":  "<edge_port>",`
	`  "bpdu_guard":  "<bpdu_guard>",`
	`  "loop_guard":  "<loop_guard>",`
	`  "root_guard":  "<root_guard>"`
	`}`

where:

Element	**Description**
if_name	The IP interface name; a string.
edge_port	Whether the interface is configured as an edge port, which allows the port to automatically transition to the STP forwarding state; one of `yes`, `no`.
bpdu_guard	(Optional) Whether BPDU guard is enabled on a port, which automatically shuts down the interface upon receipt of a BPDU; one of `enable`, `disable`.
loop_guard	(Optional) Whether look guard is enabled on a port for additional checks for preventing STP looping; one of `enable`, `disable`.
root_guard	(Optional) Whether guard mode is set to root guard on interface.

get_stp_intf(conn, intf)

Gets STP properties for a specified interface.

Syntax

`Stp.get_stp_intf(`*<conn>*`, `*<intf>*`)`

where:

Parameter	Description
conn	A connection object for the node.
intf	The IP interface name (String). **Note:** The interface must exist.

Returns

A JSON response containing the STP properties of the specified interface:

Response Body
 	
(JSON)
	{
	  "if_name": "<if_name>",
	  "edge_port": "<edge_port>",
	  "bpdu_guard": "<bpdu_guard>",
	  "loop_guard": "<loop_guard>",
	  "root_guard": "<root_guard>"
	}

where:

Element	Description
if_name	The IP interface name; a string.
edge_port	Whether the interface is configured as an edge port, which allows the port to automatically transition to the STP forwarding state; one of `yes`, `no`.
bpdu_guard	(Optional) Whether BPDU guard is enabled on a port, which automatically shuts down the interface upon receipt of a BPDU; one of `enable`, `disable`.
loop_guard	(Optional) Whether look guard is enabled on a port for additional checks for preventing STP looping; one of `enable`, `disable`.
root_guard	(Optional) Whether guard mode is set to root guard on interface.

update_stp(conn, intf, parameters)

Updates STP properties for the specified interface.

Syntax

```
Stp.update_stp(<conn>, <intf>, <parameters>)
```

where:

Parameter	Description
conn	A connection object for the node.
intf	The IP interface name (String). **Note:** The interface must exist.

Parameter	**Description**

parameters	A dictionary containing the following JSON key-value pairs:
	`{`
	`  "if_name":  "<if_name>",`
	`  "edge_port":  "<edge_port>",`
	`  "bpdu_guard":  "<bpdu_guard>",`
	`  "loop_guard":  "<loop_guard>",`
	`  "root_guard":  "<root_guard>"`
	`}`

if_name	The IP interface name; a string. **Note:** The interface must exist.
edge_port	Whether the interface is configured as an edge port, which allows the port to automatically transition to the STP forwarding state; one of `yes`, `no`. Default value: `yes`.
bpdu_guard	(Optional) Whether BPDU guard is enabled on a port, which automatically shuts down the interface upon receipt of a BPDU; one of `enable`, `disable`. Default value: `disable`.
loop_guard	(Optional) Whether look guard is enabled on a port for additional checks for preventing STP looping; one of `enable`, `disable`. Default value: `disable`.
root_guard	(Optional) Whether guard mode is set to root guard on interface.

Returns

A JSON response containing the updated STP properties:

Response Body
 	
(JSON)
	`{`
	`  "if_name":  "<if_name>",`
	`  "edge_port":  "<edge_port>",`
	`  "bpdu_guard":  "<bpdu_guard>",`
	`  "loop_guard":  "<loop_guard>",`
	`  "root_guard":  "<root_guard>"`
	`}`

where:

Element	**Description**
if_name	The IP interface name; a string.
edge_port	Whether the interface is configured as an edge port, which allows the port to automatically transition to the STP forwarding state; one of `yes`, `no`.
bpdu_guard	(Optional) Whether BPDU guard is enabled on a port, which automatically shuts down the interface upon receipt of a BPDU; one of `enable`, `disable`.

Element	**Description**
loop_guard	(Optional) Whether look guard is enabled on a port for additional checks for preventing STP looping; one of `enable`, `disable`.
root_guard	(Optional) Whether guard mode is set to root guard on interface.

System Management

The `System` class methods manage and set up images and configuration files on the node.

These methods are in the file `system.rb`.

download_boot_img(conn, parameters)

Downloads a boot image to the switch.

Syntax

`System.download_boot_img(`*<conn>*`, `*<parameters>*`)`

where:

Parameter	**Description**
conn	A connection object for the node.

parameters	A dictionary containing the following JSON key-value pairs:
	`{`
	`  "protocol":"<protocol>",`
	`  "serverip":"<serverip>",`
	`  "srcfile":"<srcfile>",`
	`  "imgtype":"<imgtype>",`
	`  "username":"<username>",`
	`  "passwd":"<passwd>",`
	`  "vrf_name":"<vrf_name>"`
	`}`

protocol	Protocol name; one of "`tftp`", "`sftp`".
serverip	Server IP address.
srcfile	Source file; up to 256 characters long.
imgtype	System image type; one of "`all`", "`boot`", "`onie`", "`os`".
username	(Optional) Username for the server. Not required for TFTP.
passwd	(Optional) Password for the server username. Not required for TFTP.
vrf_name	(Optional) VRF name; an alphabetic string up to 64 characters long.

Returns

A JSON response containing the transfer status:

Response Body
 	
(JSON)
	`{`
	`  "status": "<status>",`
	`}`

where *status* is one of one of “`transferring`”, “`installing`”, “`successful`”, “`failed`”.

download_sw_config(conn, parameters)

Downloads a configuration to the switch.

Syntax

`System.download_sw_config(<conn>, <parameters>)`

where:

Parameter	**Description**
conn	A connection object for the node.

parameters	A dictionary containing the following JSON key-value pairs:
	`{`
	`  "protocol":"<protocol>",`
	`  "serverip":"<serverip>",`
	`  "srcfile":"<srcfile>",`
	`  "imgtype":"<imgtype>",`
	`  "username":"<username>",`
	`  "passwd":"<passwd>",`
	`  "vrf_name":"<vrf_name>"`
	`}`

protocol	Protocol name; one of "`tftp`", "`sftp`".
serverip	Server IP address.
srcfile	Source file; up to 256 characters long.
imgtype	System image type; one of `all`, `boot`, `onie`, `os`.
username	(Optional) Username for the server.
passwd	(Optional) Password for the server username.
vrf_name	(Optional) VRF name; an alphabetic string up to 64 characters long.

Returns

A JSON response:

Response Body
 	
(JSON)
	`{`
	`  "status": "<status>",`
	`  "details": "<details>",`
	`  "filename": "<filename>"`
	`}`

where:

Element	Description
status	Transfer status; one of "`transferring`", "`installing`", "`successful`", "`failed`".
details	Detailed description of the status; one of: ● `Transferring running-config` ● `Transferring startup-config` ● `Installing image` ● *image* `installation succeeded` ● `Copy success` ● `VRF` *vrf_name* `doesn't exist` ● `Another image installation is in progress` ● `Host` *serverip* `is unreachable` ● `ONIE feature in not enabled on this switch` ● `File not found` ● `SFTP authentication failure` ● *image* `installation failed` ● `Copy failed`
filename	Configuration filename.

get_clock(conn)

Gets the clock information from the node.

Syntax

`System.get_clock(<conn>)`

where *conn* is the connection object for the node.

Returns

A JSON response containing the system date and time:

Response Body
 	
(JSON)
	`{`
	  `"date": "<date>",`
	`}`

where:

Element	Description
date	System date in the format: HH:MM:SS xM ZZZ Wkd Mon Dy YEAR where: ● HH - hour ● MM - minutes ● SS - seconds ● xM - one of "`AM`", "`PM`" ● Wkd - three-letter weekday abbreviation ● Mon - three-letter month abbreviation ● Dy - one or two-digit day ● YEAR - four-digit year For example: `10:55:58 AM UTC Mon Jul  4 2016`

get_device_contact(conn)

Gets the device contact from the node.

Syntax

`System.get_device_contact(<conn>)`

where *conn* is the connection object for the node.

Returns

A JSON response containing the device contact information:

Response Body
(JSON)
	`{`
	`  "contact": <contact>,`
	`}`

where *contact* is the device contact; a string up to 256 characters long.

get_device_descr(conn)

Gets the device description.

Syntax

`System.get_device_descr(<conn>)`

where *conn* is the connection object for the node.

Returns

A JSON response containing the device description:

Response Body
 	
(JSON)
	`{`
	`  "descr": <descr>,`
	`}`

where *descr* is the device contact; a string up to 256 characters long.

get_hostname(conn)

Gets the hostname of the system

Syntax

`System.get_hostname(<conn>)`

where *conn* is the connection object for the node.

Returns

A JSON response containing the hostname:

Response Body
 	
(JSON)
	`{`
	`  "hostname": "<hostname>",`
	`}`

where *hostname* is the hostname of the system; a string from 1-64 characters long.

get_startup_sw(conn)

Gets the system startup image.

Syntax

`System.get_startup_sw(<conn>)`

where *conn* is the connection object for the node.

Returns

A JSON response containing the system startup image setting:

Response Body
 	
(JSON)
	`{`
	`  “boot software” : “<setting>”`
	`}`

where *setting* is the next startup image setting; one of `active` or `standby`.

reset_switch(conn)

Resets the switch.

Syntax

`System.reset_switch(<conn>)`

where *conn* is the connection object for the node.

Returns

`True` if the operation succeeded; otherwise `False`.

get_transfer_status(conn, type, content)

Gets the status of a file transfer.

Syntax

`System.get_transfer_status(<conn>, <type>, <content>)`

where:

Parameter	**Description**
conn	A connection object for the node.
content	A string; one of `image`, `running_config`, `startup_config`.
type	The type of transfer; one of `upload`, `download`.

Returns

A JSON response containing the file transfer status:

Response Body
 	
(JSON)
	`{`
	`  "status": "<status>",`
	`  "details": "<details>",`
	`  "filename": "<filename>"`
	`}`

put_startup_sw(conn, parameters)

Updates the system startup image

Syntax

`System.put_startup_sw(<conn>, <parameters>)`

where:

Parameter	**Description**
conn	A connection object for the node.

parameters	A dictionary containing the following JSON key-value pair:
	`{`
	`  “boot software” : “`*<setting>*`”`
	`}`

setting	Next startup image setting; one of `active`, `standby`.

Returns

A JSON response containing the next system image startup setting:

Response Body
 	
(JSON)
	`{`
	`  “boot software” : “`*<setting>*`”`
	`}`

where *setting* is one of `active`, `standby`.

reset_switch(conn)

Resets the switch.

Syntax

`System.reset_switch(`*<conn>*`)`

where *conn* is the connection object for the node.

Returns

`True` if the operation succeeded; otherwise `False`.

save_config(conn)

Saves the running configuration to the flash memory

Syntax

`System.save_config(`*<conn>*`)`

where *conn* is the connection object for the node.

Returns

`True` if the operation succeeded; otherwise `False`.

set_clock(conn, parameters)

Sets the system date and time.

Syntax

`System.set_clock(`*<conn>*`, `*<parameters>*`)`

where:

Parameter	Description
conn	A connection object for the node.

parameters	A dictionary containing the following JSON key-value pairs:
	`{`
	`  "time": "<time>" ,`
	`  "day": <day>,`
	`  "month": <month> ,`
	`  "year": <year>`
	`}`

time	System time in the format "HH:MM:SS".
day	The day of the month; an integer from 1-31.
month	The month; one of the following case-insensitive strings: ● `january` ● `february` ● `march` ● `april` ● `may` ● `june` ● `july` ● `august` ● `september` ● `october` ● `november` ● `december`
`year`	The year; an integer from 2000-2030.

Returns

A JSON response containing the system date:

Response Body
 	
(JSON)
	`{`
	`  "date": "<date>",`
	`}`

where *date* is the system date and time in the format:

HH:MM:SS xM ZZZ Wkd Mon Dy YEAR

where:

- HH - hour
- MM - minutes
- SS - seconds
- xM - one of “`AM`”, “`PM`”
- Wkd - three-letter weekday abbreviation
- Mon - three-letter month abbreviation
- Dy - one or two-digit day
- YEAR - four-digit year

For example:

```
10:55:58 AM UTC Mon Jul  4 2016
```

set_clock_format(conn, parameters)

Sets the clock format to 12 hour or 24 hour format.

Syntax

`System.set_clock_format(`*<conn>*`, `*<parameters>*`)`

where:

Parameter	**Description**
conn	A connection object for the node.

parameters	A dictionary containing the following JSON key-value pair:
	`{`
	`  "format": <format>,`
	`}`

format	An integer; one of `12`, `24`.

Returns

A JSON response containing the system clock format:

Response Body
 	
(JSON)
	`{`
	`  "format": <format>,`
	`}`

where *format* is one of `12`, `24`.

set_clock_protocol(conn, parameters)

Sets the protocol to either manual or Network Time Protocol.

Syntax

`System.set_clock_protocol(`*<conn>*`, `*<parameters>*`)`

where:

Parameter	Description
conn	A connection object for the node.

parameters	A dictionary containing the following JSON key-value pair:
	`{`
	`  "protocol": <protocol>,`
	`}`

`protocol`	One of `none`, `ntp`. Default value: `ntp`.

Returns

A JSON response containing the clock protocol:

Response Body
 	
(JSON)
	`{`
	`  "protocol": "<protocol>",`
	`}`

where *protocol* is one of `none`, `ntp`.

set_clock_summertime(conn, parameters)

Set the transition to and from a summer time zone adjustment.

Syntax

```
System.set_clock_summertime(<conn>, <parameters>)
```

where:

Parameter	Description
conn	A connection object for the node.

parameters	A dictionary containing the following JSON key-value pairs:
	`{`
	`  "timezone": <time_zone>,`
	`  "startweek": <startweek>,`
	`  "startweekday": <startweekday>,`
	`  "startmonth": <startmonth>,`
	`  "starttime" :  "<starttime>",`
	`  "endweek"   : <endweek>,`
	`  "endweekday": <endweekday>,`
	`  "endmonth"  : <endmonth>,`
	`  "endtime"   :  "<endtime>",`
	`  "offsetmin" : <minutes>`
	`}`

time	System time in the format "HH:MM:SS".
timezone	Local time zone of the system; a three to five character string such as "PST", "MST", "CST", or "EST".

Parameter	**Description**
startweek	Week number in the month in which to start Daylight Saving time; an integer from 1-5 (first week=1, last week=5).
startweekday	Weekday on which to start DST; one of the following case-insensitive strings: • `monday` • `tuesday` • `wednesday` • `thursday` • `friday` • `saturday` • `sunday`
startmonth	Month to start DST; one of the following case-insensitive strings: • `january` • `february` • `march` • `april` • `may` • `june` • `july` • `august` • `september` • `october` • `november` • `december`
starttime	Time to start DST; a string in the format "HH:MM".
endweek	Week number in which to end DST; an integer from 1-5 (first week=1, last week=5).
endweekday	Weekday on which to end DST; one of the following case-insensitive strings: • `monday` • `tuesday` • `wednesday` • `thursday` • `friday` • `saturday` • `sunday)`

Parameter	Description
endmonth	Month in which DST ends; one of the following case-insensitive strings: ● `january` ● `february` ● `march` ● `april` ● `may` ● `june` ● `july` ● `august` ● `september` ● `october` ● `november` ● `december`
endtime	Time to end DST; a string in the format "HH:MM"
offsetmin	Offset to add, in minutes; an integer from 1-1440.

Returns

A JSON response containing the system date:

Response Body
 	
(JSON)
	`{`
	`  "date": "<date>",`
	`}`

where *date* is the system date and time in the format:

HH:MM:SS xM ZZZ Wkd Mon Dy YEAR

where:

- HH - hour
- MM - minutes
- SS - seconds
- xM - one of "`AM`", "`PM`"
- Wkd - three-letter weekday abbreviation
- Mon - three-letter month abbreviation
- Dy - one or two-digit day
- YEAR - four-digit year

For example:

```
10:55:58 AM UTC Mon Jul  4 2016
```

set_clock_timezone(conn, parameters)

Sets the clock time zone for the switch

Syntax

```
System.set_clock_timezone(<conn>, <parameters>)
```

where:

Parameter	Description
conn	A connection object for the node.

parameters	A dictionary containing the following JSON key-value pairs:
	`{`
	`  "timezone": "<timezone>",`
	`  "offsethour": "<offsethour>",`
	`  "offsetmin": "<lag_mode>",`
	`}`

timezone	One to five letter string denoting the local system time zone, such as `PST`, `GMT`.
offsethour	Hours offset from UTC; an integer from -23 through 23.
offsetmin	Minutes offset from UTC; an integer from 0-59.

Returns

A JSON response containing the system date:

Response Body
 	
(JSON)
	`{`
	`  "date": "<date>",`
	`}`

where *date* is the system date and time in the format:

HH:MM:SS xM ZZZ Wkd Mon Dy YEAR

where:

- HH - hour
- MM - minutes
- SS - seconds
- xM - one of “`AM`”, “`PM`”
- Wkd - three-letter weekday abbreviation
- Mon - three-letter month abbreviation
- Dy - one or two-digit day
- YEAR - four-digit year

For example:

```
10:55:58 AM UTC Mon Jul  4 2016
```

set_device_contact(conn, parameters)

Updates the device contact.

Syntax

`System.set_device_contact(<conn>, <parameters>)`

where:

Parameter	**Description**
conn	A connection object for the node.

parameters	A dictionary containing the following JSON key-value pair:
	`{`
	`  "contact": "<contact>",`
	`}`

contact	Device contact; a string up to 256 characters long.

Returns

A JSON response containing the device contact:

Response Body
 	
(JSON)
	`{`
	`  "contact": <contact>,`
	`}`

where *contact* is the device contact; a string up to 256 characters long.

set_device_descr(conn, parameters)

Updates the device description.

Syntax

`System.set_device_descr(<conn>, <parameters>)`

where:

Parameter	**Description**
conn	A connection object for the node.

parameters	A dictionary containing the following JSON key-value pair:
	`{`
	`  "descr": <descr>,`
	`}`

descr	Device description; a string up to 256 characters long.

Returns

A JSON response containing the device description:

Response Body
 	
(JSON)
	`{`
	  `"descr": <descr>,`
	`}`

where *descr* is the device description; a string up to 256 characters long.

set_hostname(conn, parameters)

Sets the system hostname.

Syntax

```
System.set_hostname(<conn>, <parameters>)
```

where:

Parameter	**Description**
conn	A connection object for the node.

parameters	A dictionary containing the following JSON key-value pair:
	`{`
	  `"hostname": <hostname>,`
	`}`

hostname	The hostname of the system; a string from 1-64 characters long.

Returns

A JSON response containing the system hostname:

Response Body
 	
(JSON)
	`{`
	  `"hostname": "<hostname>",`
	`}`

where *hostname* is hostname of the system; a string from 1-64 characters long.

upload_sw_config(conn, parameters)

Uploads a configuration file from the switch to the server.

Syntax

```
System.upload_sw_config(<conn>, <parameters>)
```

where:

Parameter	Description
conn	A connection object for the node.

parameters	A dictionary containing the following JSON key-value pairs:
	`{`
	`  "protocol":"<protocol>",`
	`  "serverip":"<serverip>",`
	`  "srcfile":"<srcfile>",`
	`  "dstfile":"<dstfile>",`
	`  "username":"<username>",`
	`  "passwd":"<passwd>",`
	`  "vrf_name":"<vrf_name>"`
	`}`

protocol	Protocol name; one of `tftp`, `sftp`.
serverip	Server IP address.
srcfile	Source file; up to 256 characters long.
dstfile	Destination file; one of `running_config`, `startup_config`.
username	(Optional) Username for the server.
passwd	(Optional) Password for the server username.
vrf_name	(Optional) VRF name; an alphabetic string up to 64 characters long.

Returns

A JSON response containing the transfer status, details, and the configuration file:

Response Body
 	
(JSON)
	`{`
	`  "status": "<status>",`
	`  "details": "<details>",`
	`  "filename": "<filename>"`
	`}`

where:

Element	Description
status	Transfer status; one of `transferring`, `installing`, `successful`, `failed`.
details	Detailed description of the status; one of: ● `Transferring running-config` ● `Transferring startup-config` ● `Installing image` ● *image* `installation succeeded` ● `Copy success` ● `VRF` *vrf_name* `doesn't exist` ● `Another image installation is in progress` ● `Host` *serverip* `is unreachable` ● `ONIE feature in not enabled on this switch` ● `File not found` ● `SFTP authentication failure` ● *image* `installation failed` ● `Copy failed`
filename	Configuration filename.

upload_tech_support(conn, parameters)

Uploads technical support information from the switch to the server.

Syntax

`System.upload_tech_support(`*<conn>*`, `*<parameters>*`)`

where:

Parameter	Description
conn	A connection object for the node.

parameters	A dictionary containing the following JSON key-value pair:s
	`{`
	`"protocol":"`*<protocol>*`",`
	`"serverip":"`*<serverip>*`",`
	`"dstfile":"`*<dstfile>*`",`
	`"username":"`*<username>*`",`
	`"passwd":"`*<passwd>*`",`
	`"vrf_name":"`*<vrf_name>*`"`
	`}`

protocol	Protocol name; one of `tftp`, `sftp`.
serverip	Server IP address.
dstfile	Destination file; one of `running_config`, `startup_config`.
username	(Optional) Username for the server.

Parameter	**Description**
passwd	(Optional) Password for the server username.
vrf_name	(Optional) VRF name; an alphabetic string up to 64 characters long.

Returns

A JSON response containing the transfer status:

Response Body
 	
(JSON)
	`{`
	`  "status": "<status>",`
	`}`

where:

Element	**Description**
status	Transfer status; one of `transferring`, `installing`, `successful`, `failed`.

Telemetry Management

The Telemetry class methods manage telemetry features on the node.

These methods are in the file `telemetry.rb`.

clear_bst_cgsn_ctrs(conn)

Clears BST congestion drops.

Syntax

`Telemetry.clear_bst_cgsn_ctrs(`<*conn*>`)`

where *conn* is the connection object for the node.

Returns

`True` if the operation succeeded; otherwise `False`.

clear_bst_stats(conn)

Clears BST statistics.

Syntax

`Telemetry.clear_bst_stats(`<*conn*>`)`

where *conn* is the connection object for the node.

Returns

`True` if the operation succeeded; otherwise `False`.

clear_bst_threshold(conn)

Clears BST threshold.

Syntax

`Telemetry.clear_bst_threshold(`<*conn*>`)`

where *conn* is the connection object for the node.

Returns

`True` if the operation succeeded; otherwise `False`.

get_bst_feature(conn)

Gets BST information.

Syntax

`Telemetry.get_bst_feature(`<*conn*>`)`

where *conn* is the connection object for the node.

Returns

A JSON response containing BST feature settings:

Response Body
 	
(JSON)
	`{`
	`  "bst-enable": <bst-enable>,`
	`  "send-async-reports": <send-async-reports>,`
	`  "collection-interval": <collection-interval>,`
	`  "stats-in-percentage": <stats-in-percentage>,`
	`  "stat-units-in-cells": <stat-units-in-cells>,`
	`  "trigger-rate-limit": <trigger-rate-limit>,`
	`  "trigger-rate-limit-interval": <trigger-rate-limit-interval>,`
	`  "send-snapshot-on-trigger": <send-snapshot-on-trigger>`
	`}`

where:

Element	**Description**
bst-enable	Set to 1 to enable BST, 0 to disable it. Enabling BST allows the switch to track buffer utilization statistics.
send-async-reports	Set to 1 to enable the transmission of periodic asynchronous reports, 0 to disable this feature.
collection-interval	The collection interval, in seconds. This defines how frequently periodic reports will be sent to the configured controller.
trigger-rate-limit	The trigger rate limit, which defines the maximum number of threshold-driven triggered reports that the agent is allowed to send to the controller per `trigger-rate-limit-interval`; an integer from 1-5.
trigger-rate-limit-interval	The trigger rate limit interval, in seconds; an integer from 10-60.
send-snapshot-on-trigger	Set to 1 to enable sending a complete snapshot of all buffer statistics counters when a trigger happens, 0 to disable this feature.
async-full-report	Set to 1 to enable the async full report feature, 0 to disable it. When this feature is enabled, the agent sends full reports containing data related to all counters. When the feature is disabled, the agent sends incremental reports containing only the counters that have changed since the last report.

Element	Description
stat-units-in-cells	Whether the buffer statistics are reported in units of bytes or cells. **Note:** This value is always set to 0. It cannot be modified and it is always ignored because `stats-in-percentage` is always set to 1.
stats-in-percentage	When set to 1, the buffer usage statistics are reported as percentages and the parameter `stat-units-in-cells` is ignored while reporting the statistics. This variable is applicable for statistics and threshold reporting. **Note:** This variable is always set to 1 and cannot be modified. The percentage values in the BST/trigger report are an approximation of buffer utilization, not an exact value.

get_bst_report(conn, parameters)

Gets BST report information.

Syntax

`Telemetry.get_bst_report(`*<conn>*`,` *<parameters>*`)`

where:

Parameter	Description
conn	A connection object for the node.

parameters	A dictionary containing the following JSON key-value pair:s
	`{`
	`  "include-ingress-port-priority-group" :` *<include-ingress-port-priority-group>*`,`
	`  "include-ingress-port-service-pool" :` *<include-ingress-port-service-pool>*`,`
	`  "include-ingress-service-pool" :` *<include-ingress-service-pool>*`,`
	`  "include-egress-port-service-pool" :` *<include-egress-port-service-pool>*`,`
	`  "include-egress-service-pool" :` *<include-egress-service-pool>*`,`
	`  "include-egress-rqe-queue" :` *<include-egress-rqe-queue>*`,`
	`  "include-egress-uc-queue" :` *<include-egress-uc-queue>*`,`
	`  "include-egress-mc-queue" :` *<include-egress-mc-queue>*`,`
	`  "include-egress-cpu-queue":` *<include-egress-cpu-queue>*`,`
	`  "include-device" :` *<include-device>*
	`}`

include-ingress-port-priority-group	Ingress port priority group; 1 to enable, 0 to disable.
include-ingress-port-service-pool	Ingress port service pool; 1 to enable, 0 to disable.
include-ingress-service-pool	Ingress service pool; 1 to enable, 0 to disable.

Parameter	**Description**
include-egress-port-service-pool	Egress port service pool; 1 to enable, 0 to disable.
include-egress-service-pool	Egress service pool; 1 to enable, 0 to disable.
include-egress-rqe-queue	Egress RQE queue; 1 to enable, 0 to disable.
include-egress-uc-queue	Egress unicast queue buffers. Set to 1 to enable in BST report, 0 to disable it.
include-egress-cpu-queue	Egress CPU queue buffers. Set to 1 to enable in BST report, 0 to disable it.
include-egress-mc-queue	Egress multicast queue buffers. Set to 1 to enable in BST report, 0 to disable it.
include-device	Device; 1 to enable, 0 to disable.

Returns

A JSON response containing BST report information:

Response Body

(JSON)

```
{
  {"time-stamp": "<time-stamp>",
     "report": [
     {"realm": "device",
      "data": "46"},
     {
      "realm": "ingress-port-priority-group",
      "data": [
        {
         "interface": " Ethernet1/2",
         "data": [[5, "100", "100"]]
         },
        {
         "interface": " Ethernet1/3",
         "data": [[5, "100", "100"]]
         } ]
    },
    { "realm": "ingress-port-service-pool",
      "data": [
       {
         "interface": "Ethernet1/2",
         "data": [[5, "100"]]
       },
       {
         "interface ": "Ethernet1/3",
         "data": [[6, "100"]]
      }]
   },
   {
    "realm": "ingress-service-pool",
    "data": [[1, "100"], [2, "100"]]
   },
   {
     "realm": "egress-cpu-queue",
     "data": [[3, "100"]]
   },
   { "realm": "egress-uc-queue",
     "data": [[3, "100"]]
   },
   { "realm": "egress-mc-queue",
     "data": [[3, "100"]]
   },
   { "realm": "egress-port-service-pool",
     "port-service-pool-ctr": [{
       "interface": " Ethernet1/2",
       "data": [["5","10", "10", "30"]]
    },
    {  "interface": " Ethernet1/3",
       "data": [["60", "30", "36","45"]]
        }]
   },
   { "realm": "egress-rqe-queue",
     "data": [[2, "33"], [5, "25"]]
   },
   { "realm": "egress-service-pool",
     "data": [[1,"20", "10", "10", "32"],
             [3, "3660", 0, 0]]  },
]
```

where:

Index	**Description**
time-stamp	The date and time the request was made in 24-hour format: `YYYY-MM-DD - HH:MM:SS`
rqe-threshold	Threshold values in percentages; an integer from 1-100.
cpu-threshold	Threshold values in percentages; an integer from 1-100.
um-threshold	Threshold values in percentages; an integer from 1-100.
uc-threshold	Threshold values in percentages; an integer from 1-100.
mc-threshold	Threshold values in percentages; an integer from 1-100.
queue	The queue number with respect to the realm; an integer.
service-pool	Service pool; either 0 (disabled) or 1 (enabled).
priority-group	Priority group; an integer from 0-7.
interface	Physical port interface of the device; a string.

The following realms apply:

Realm	**Index # 1**	**Index # 2**	**Statistics**
`ingress-port-priority-group`	*interface* (such as `Ethernet1/7`)	`priority-group`	`um-share-buffer-count` `um-head room-buffer-count`
`ingress-port-service-pool`	*interface* (such as `Ethernet1/7`)	`service-pool`	`um-share-buffer-count`
`ingress-service-pool`	`service-pool`		`um-share-buffer-count`
`egress-port-service-pool`	*interface* (such as `Ethernet1/7`)	`service-pool`	`uc-share-buffer-count, um-share-buffer-count, mc-share-buffer-count,`
`egress-cpu-queue`	`queue`		`cpu-buffer-count`
`egress-uc-queue`	`queue`		`uc-buffer-count`
`egress-mc-queue`	`queue`		`mc-buffer-count`
`egress-service-pool`	`service-pool`		`um-share-buffercount, mc-share-buffer-count`

Realm	Index # 1	Index # 2	Statistics
`egress-rqe-queue`	`queue`		`rqe-buffer-count`
`device`			`data`

Note: For more information on realm parameters and indexes, see the *CNOS Application Guide*.

get_bst_threshold(conn, parameters)

Gets the BST threshold.

Syntax

`Telemetry.get_bst_threshold(`*<conn>*`,` *<parameters>*`)`

where:

Parameter	Description
conn	A connection object for the node.

parameters	A dictionary containing the following JSON key-value pair:s
	`{`
	`  "include-ingress-port-priority-group" :` *<include-ingress-port-priority-group>*`,`
	`  "include-ingress-port-service-pool" :` *<include-ingress-port-service-pool>*`,`
	`  "include-ingress-service-pool" :` *<include-ingress-service-pool>*`,`
	`  "include-egress-port-service-pool" :` *<include-egress-port-service-pool>*`,`
	`  "include-egress-service-pool" :` *<include-egress-service-pool>*`,`
	`  “include-egress-cpu-queue" :` *<include-egress-cpu-queue>*`,`
	`  “include-egress-uc-queue" :` *<include-egress-uc-queue>*`,`
	`  “include-egress-mc-queue" :` *<include-egress-mc-queue>*
	`}`

include-ingress-port-priority-group	Ingress port priority group; 1 to enable, 0 to disable.
include-ingress-port-service-pool	Ingress port service pool; 1 to enable, 0 to disable.
include-ingress-service-pool	Ingress service pool; 1 to enable, 0 to disable.
include-egress-port-service-pool	Egress port service pool; 1 to enable, 0 to disable.
include-egress-service-pool	Egress service pool; 1 to enable, 0 to disable.

Parameter	**Description**
include-egress-rqe-queue	Egress CPU queue buffers. Set to 1 to enable in BST threshold report, 0 to disable it.
include-egress-uc-queue	Egress unicast queue buffers. Set to 1 to enable in BST report, 0 to disable it.
include-egress-cpu-queue	Egress CPU queue buffers. Set to 1 to enable in BST report, 0 to disable it.
include-egress-mc-queue	Egress multicast queue buffers. Set to 1 to enable in BST report, 0 to disable it.

Returns

A JSON response containing BST threshold information:

<table><tr><td>Response Body

(JSON)</td><td>

```
{
    "report": [
        {
            "realm": "ingress-port-service-pool",
            "data": [
                {
                    "interface": "<interface>",
                    "data": [
                        [
                            <service-pool>,
                            "<um-share-threshold>"
                        ]
                    ]
                },
...
        {
            "realm": "ingress-service-pool",
            "data": [
                [
                    0,
                    "100"
                ]
            ]
        }
...
```

</td></tr></table>

where:

ParameterDescription

`realm`The realm (see table)

*interface*The interface (for example, `Ethernet1/1`)

*service-pool*The service pool for this feature.

um-share-thresholdThe

Realm	Index # 1	Index # 2	Thresholds
`ingress-port-priority-group`	*interface* (such as `Ethernet1/7`)	`priority-group`	`um-share-threshold` `um-head room-threshold`
`ingress-port-service-pool`	*interface* (such as `Ethernet1/7`)	`service-pool`	`um-share-threshold`
`ingress-service-pool`	`service-pool`		`um-share-threshold`
`egress-port-service-pool`	*interface* (such as `Ethernet1/7`)	`service-pool`	`uc-share-threshold`, `um-share-threshold` `mc-share-threshold`
`egress-service-pool`	`service-pool`		`um-share-threshold` `mc-share-threshold`
`egress-rqe-queue`	`queue`		`rqe-threshold`
`egress-cpu-queue`	`queue`		`cpu-threshold`
`egress-uc-queue`	`queue`		`uc-threshold`
`egress-mc-queue`	`queue`		`mc-threshold`
`device`			`threshold`

Note: For more information on realm parameters and indexes, see the *CNOS Application Guide*.

set_bst_tracking(conn, parameters)

Sets BST trackers and the tracking mode on the ASIC.

Syntax

`Telemetry.upload_tech_support(`*<conn>*`,` *<parameters>*`)`

where:

Parameter	**Description**
conn	A connection object for the node.

parameters	A dictionary containing the following JSON key-value pair:s
	`{`
	` "track-peak-stats": `*<track-peak-stats>*`,`
	` "track-ingress-port-priority-group" : `*<track-ingress-port-priority-group>*`,`
	` "track-ingress-port-service-pool" : `*<track-ingress-port-service-pool>*`,`
	` "track-ingress-service-pool" : `*<track-ingress-service-pool>*`,`
	` "track-egress-port-service-pool" : `*<track-egress-port-service-pool>*`,`
	` "track-egress-service-pool" : `*<track-egress-service-pool>*`,`
	` "track-egress-rqe-queue" : `*<track-egress-rqe-queue>*`,`
	` "track-egress-cpu-queue" : `*<track-egress-cpu-queue>*`,`
	` "track-egress-uc-queue" : `*<track-egress-uc-queue>*
	` "track-egress-mc-queue" : `*<track-egress-mc-queue>*
	` "track-device" : `*<track-device>*
	`}`

track-peak-stats	Set to 1 to peak statistics tracking, 0 to disable this feature
track-ingress-port-priority-group	Set to 1 to enable ingress port priority group tracking, 0 to disable this feature
track-ingress-port-service-pool	Set to 1 to enable ingress port service pool tracking, 0 to disable this feature
track-ingress-service-pool	Set to 1 to enable ingress service pool tracking, 0 to disable this feature
track-egress-port-service-pool	Set to 1 to enable egress port service pool tracking, 0 to disable this feature
track-egress-service-pool	Set to 1 to enable egress service pool tracking, 0 to disable this feature
track-egress-rqe-queue	Set to 1 to enable egress RQE queue tracking, 0 to disable this feature
track-egress-cpu-queue	Set to 1 to enable egress CPU queue tracking, 0 to disable this feature
track-egress-uc-queue	Set to 1 to enable egress unicast queue tracking, 0 to disable this feature
track-egress-mc-queue	Set to 1 to enable egress multicast queue tracking, 0 to disable this feature
track-device	Set to 1 to enable tracking of this device, 0 to disable this feature

Returns

`True` if the operation succeeded; otherwise `False`.

set_sys_feature(conn, parameters)

Sets system feature.

Syntax

`Telemetry.set_sys_feature(`*<conn>*`, `*<parameters>*`)`

where:

Parameter	Description
conn	A connection object for the node.

parameters	A dictionary containing the following JSON key-value pair:s
	`{`
	`  "heartbeat-enable" : <heartbeat-enable>,`
	`  "msg-interval" : <msg-interval>,`
	`}`

heartbeat-enable	When enabled, the Agent asynchronously sends the registration and heartbeat message to the collector. One of: ● 0: disable heartbeat ● 1: enable heartbeat (default value)
msg-interval	Determines the interval with which the registration and heartbeat messages are sent to the collector; units of seconds from 1-600. Default value: 5 seconds.

Returns

`True` if the operation succeeded; otherwise `False`.

Virtual Link Aggregation Group Management

The `Vlag` class methods manage vLAG configuration on the node.

These methods are in the file `vlag.rb`.

create_vlag_inst(conn, parameters)

Creates a vLAG instance.

Syntax

`Vlag.create_vlag_inst(`*<conn>*`,` *<parameters>*`)`

where:

Parameter	**Description**
conn	A connection object for the node.

parameters	A dictionary containing the following JSON key-value pair:s
	`{`
	`  "instance_id": "<instance_id>",`
	`  "port_aggregator": "<port_aggregator>",`
	`  "status": "<status>",`
	`}`

instance_id	vLAG instance ID number; an integer from 1-64.
port_ aggregator	LAG identifier; an integer from 1-4096.
status	vLAG status; one of `enable`, `disable`. Default value: `disable`.

Returns

A JSON response containing the vLAG instance:

Response Body
 	
(JSON)
	`{`
	`  "instance_id": "<instance_id>",`
	`  "port_aggregator": "<port_aggregator>",`
	`  "status": "<status>",`
	`}`

where:

Element	**Description**
instance_id	vLAG instance ID number; an integer from 1-64.
port_ aggregator	LAG identifier; an integer from 1-4096.
status	vLAG status; one of `enable`, `disable`.

delete_vlag_inst(conn, instance_id)

Deletes a vLAG Instance.

Syntax

`Vlag.delete_vlag_inst(`*<conn>*`,` *<instance_id>*`)`

where:

Parameter	**Description**
conn	A connection object for the node.
instance_id	The vLAG instance ID number; an integer from 1-4096.

Returns

`True` if the operation succeeded; otherwise `False`.

get_global_vlag(conn)

Gets global VLAG information.

Syntax

`Vlag.get_global_vlag(`*<conn>*`)`

where *conn* is the connection object for the node.

Returns

A JSON response containing the global vLAG information for the node:

Response Body
 	
(JSON)
	{
	  "status": "*<status>*",
	  "system_mac": "*<system_mac>*",
	  "fdb_refresh": "*<fdb_refresh>*",
	  "fdb_synch" : "*<fdb_synch>*",
	  "auto_recovery":
	  {
	    "interval": "*<interval>*"
	    "state": "*<state>*",
	  }
	  "startup_delay":
	  {
	    "interval": "*<interval>*"
	    "state": "*<state>*",
	  }
	  "local":
	  {
	    “tier_id”: "*<tier_id>*",
	    “sys_type”: "*<sys_type>*",
	    “os_version”: "*<os_version>*",
	    “admin_role”: "*<admin_role>*",
	    “oper_role”: "*<oper_role>*",
	    “priority” : "*<priority>*",
	    “system_mac”: "*<system_mac>*",
	    “match”: "*<match>*"
	  }
	  "peer":
	  {
	    “tier_id”: "*<tier_id>*",
	    “sys_type”: "*<sys_type>*",
	    “os_version”: "*<os_version>*",
	    “admin_role”: "*<admin_role>*",
	    “oper_role”: "*<oper_role>*",
	    “priority” : "*<priority>*",
	    “system_mac”: "*<system_mac>*",
	    “match”: "*<match>*"
	  }
	}

where:

Element	**Description**
status	Whether the vLAG is enabled or disabled; one of `enable`, `disable`. Default value: `disable`.
system_mac	Unique vLAG system MAC used for LACP negotiation on the vLAG ports so the access switch forms a single LAG. The vLAG *tier_id* is used to form this vLAG system MAC.
fdb_refresh	Whether FDB refresh is configured; one of `yes`, `no`.
fdb_synch	Whether FDB is synchronized; one of `yes`, `no`.

Element	**Description**
auto_recovery	A dictionary consisting of the following values: ● *interval*: Time interval, in seconds; an integer from 240-3600. Default value: 300. ● *state*: Auto-recovery state; one of `unstarted`, `running`, `finished`.
startup_delay	A dictionary consisting of the following values: ● *interval*: Delay time, in seconds; an integer from 0-3600. Default value: 120. ● *state*: Startup delay state; one of `unstarted`, `running`, `finished`.
local	Dictionary containing the following values: ● *tier_id*: vLAG tier ID of the local switch. ● *sys_type*: Lenovo hardware model number. ● *os_version* CNOS version. ● *admin_role*: One of `Primary`, `Secondary`, `Unselected`. ● *oper_role*: One of `Primary`, `Secondary`, `Unselected`, ● *priority*: The local vLAG priority ● *system_mac*: Local switch MAC. ● *match*: Whether there is an ISL local match or mismatch; one of `Match`, `Mis-Match`.
peer	Dictionary containing the following values: ● *tier_id*: vLAG tier ID of the peer switch. ● *sys_type*: Lenovo hardware model number. ● *os_version* CNOS version. ● *admin_role*: One of `Primary`, `Secondary`, `Unselected`. ● *oper_role*: One of `Primary`, `Secondary`, `Unselected`, ● *priority*: The peer vLAG priority ● *system_mac*: Peer switch MAC. ● *match*: Whether there is an ISL local match or mismatch; one of `Match`, `Mis-Match`.

get_vlag_conf(conn)

Gets VLAG global configuration.

Syntax

`Vlag.get_vlag_conf(`*<conn>*`)`

where *conn* is the connection object for the node.

Returns

A JSON response containing the vLAG configuration for the node:

Response Body
 	
(JSON)
	`{`
	`  "status": "<status>",`
	`  "tier_id": "<tier_id>",`
	`  "priority": "<priority>",`
	`  "auto_recover" : "<auto_recover>",`
	`  "startup_delay": "<startup_delay>",`
	`}`

where:

Element	**Description**
status	Whether the vLAG is enabled or disabled; one of `enable`, `disable`.
tier_id	vLAG tier ID value; an intger from 1-512.
priority	vLAG priority value; an integer from 0-65535.
auto_recover	Time interval, in seconds; an integer from 240-3600.
startup_delay	Delay time, in seconds; an integer from 0-3600.

get_vlag_health(conn)

Gets VLAG health check information.

Syntax

`Vlag.get_vlag_health(<conn>)`

where *conn* is the connection object for the node.

Returns

A JSON response containing vLAG health check information:

Response Body
 	
(JSON)
	`{`
	`  "status" : "<status>",`
	`  "peer_ip": "<peer_ip>",`
	`  "vrf": "<vrf>",`
	`  "local_ip": "<local_ip>",`
	`  "retry_interval": "<retry_interval>",`
	`  "keepalive_attempts" : "<keepalive_attempts>",`
	`  "keepalive_interval" : "<keepalive_interval>",`
	`}`

where:

Element	Description
status	vLAG health check status; one of `up`, `down`.
peer_ip	IP address of peer switch. This can be the management IP address of the peer switch.
vrf	VRF context string.
local_ip	IP address of local switch. This can be the management IP address of the local switch.
retry_interval	Time interval, in seconds; an integer from 1-300. Default value: 30.
keepalive_ attempts	Number of keepalive attempts made before declaring the peer is down; an integer from 1-24. Default value: 3.
keepalive_ interval	Time interval, in seconds; an integer from 2-300. Default value: 5.

get_vlag_inst_confg(conn, instance_id)

Gets configuration paramters for the specified VLAG instance.

Syntax

`Vlag.get_vlag_inst_confg(<conn>, <instance_id>)`

where:

Parameter	Description
conn	A connection object for the node.
instance_id	The vLAG instance ID number; an integer from 1-4096.

Returns

A JSON response containing configuration parameters for the specified vLAG instance:

Response Body
 	
(JSON)
	`{`
	  `"instance_id": "<instance_id>",`
	  `"port_aggregator": "<port_aggregator>",`
	  `"status": "<status>",`
	`}`

where:

Element	Description
instance_id	vLAG instance ID number; an integer from 1-64.
port_ aggregator	LAG identifier; an integer from 1-4096.
status	vLAG status; one of `enable`, `disable`.

get_vlag_inst_info(conn, instance_id)

Get information about a VLAG instance.

Syntax

`Vlag.get_vlag_inst_info(`*<conn>*`, `*<instance_id>*`)`

where:

Parameter	Description
conn	A connection object for the node.
instance_id	The vLAG instance ID number; an integer from 1-4096.

Note: An *instance_id* value of `None` returns configuration parameters for all vLAG instances.

Returns

A JSON response containing the vLAG configuration parameters for the specified vLAG instance:

Response Body
 	
(JSON)
	`{`
	  `"instance_id": "<instance_id>",`
	  `"port_aggregator": "<port_aggregator>",`
	  `"status": "<status>",`
	`}`

get_vlag_isl(conn)

Gets vLAG inter switch link information.

Syntax

`Vlag.get_vlag_isl(`*<conn>*`)`

where *conn* is the connection object for the node.

Returns

A JSON response containing vLAG ISL information for the node:

Response Body
 	
(JSON)
	`{`
	`  "port_aggregator": "<port_aggregator>",`
	`  "if_index": "<if_index>",`
	`  "state": "<state>",`
	`  "prev_state" : "<prev_state>",`
	`}`

where:

Element	**Description**
port_ aggregator	LAG identifier; an integer from 1-4096.
if_index	ISL interface index.
state	ISL state; one of `Down`, `Inactive`, `Active`.
prev_state	Previous ISL state; one of `Down`, `Inactive`, `Active`.

update_vlag_conf(conn, parameters)

Updates VLAG global configuration.

Syntax

```
Vlag.update_vlag_conf(<conn>, <parameters>)
```

where:

Parameter	**Description**
conn	A connection object for the node.

parameters	A dictionary containing the following JSON key-value pairs:
	`{`
	`  "status": "<status>",`
	`  "tier_id": "<tier_id>",`
	`  "priority": "<priority>",`
	`  "auto_recover" : "<auto_recover>",`
	`  "startup_delay": "<startup_delay>",`
	`}`

status	Whether the vLAG is enabled or disabled; one of `enable`, `disable`. Default value: `disable`
tier_id	vLAG tier ID value; an intger from 1-512. Default value: 0.
priority	vLAG priority value; an integer from 0-65535. Default value: 0.
auto_recover	Time interval, in seconds; an integer from 240-3600. Default value: 300.
startup_delay	Delay time, in seconds; an integer from 0-3600. Default value: 120.

Returns

A JSON response containing the updated vLAG configuration:

Response Body
 	
(JSON)
	`{`
	  `"status": "<status>",`
	  `"tier_id": "<tier_id>",`
	  `"priority": "<priority>",`
	  `"auto_recover" : "<auto_recover>",`
	  `"startup_delay": "<startup_delay>",`
	`}`

where:

Element	**Description**
status	Whether the vLAG is enabled or disabled; one of `enable`, `disable`.
tier_id	vLAG tier ID value; an intger from 1-512.
priority	vLAG priority value; an integer from 0-65535.
auto_recover	Time interval, in seconds; an integer from 240-3600.
startup_delay	Delay time, in seconds; an integer from 0-3600.

update_vlag_health(conn, parameters)

Configures the VLAG health check parameters

Syntax

`Vlag.update_vlag_health(<conn>, <parameters>)`

where:

Parameter	**Description**
conn	A connection object for the node.

parameters	A dictionary containing the following JSON key-value pair:s
	`{`
	  `"peer_ip": "<peer_ip>",`
	  `"vrf": "<vrf>",`
	  `“retry_interval": "<retry_interval>",`
	  `"keepalive_attempts" : "<keepalive_attempts>",`
	  `"keepalive_interval" : "<keepalive_interval>",`
	`}`

peer_ip	IP address of peer switch. This can be the management IP address of the peer switch.
vrf	VRF context string.
retry_interval	Time interval, in seconds; an integer from 1-300. Default value: 30.

Parameter	Description
keepalive_ attempts	Number of keepalive attempts made before declaring the peer is down; an integer from 1-24. Default value: 3.
keepalive_ interval	Time interval, in seconds; an integer from 2-300. Default value: 5.

Returns

A JSON response containing the updated health check *Syntax*

```
Vrrp
```

Response Body
 	
(JSON)
	`{`
	`  "peer_ip": "<peer_ip>",`
	`  "vrf": "<vrf>",`
	`  “retry_interval": "<retry_interval>",`
	`  "keepalive_attempts" : "<keepalive_attempts>",`
	`  "keepalive_interval" : "<keepalive_interval>",`
	`}`

where:

Element	Description
peer_ip	IP address of peer switch. This can be the management IP address of the peer switch.
vrf	VRF context string.
retry_interval	Time interval, in seconds; an integer from 1-300. Default value: 30.
keepalive_ attempts	Number of keepalive attempts made before declaring the peer is down; an integer from 1-24. Default value: 3.
keepalive_ interval	Time interval, in seconds; an integer from 2-300. Default value: 5.

update_vlag_inst(conn, instance_id, parameters)

Updates VLAG instance.

Syntax

```
Vlag.update_vlag_inst(<conn>, <instance_id>, <parameters>)
```

where:

Parameter	Description
conn	A connection object for the node.
instance_id	vLAG instance ID number; an integer from 1-64.

Parameter	Description

parameters	A dictionary containing the following JSON key-value pair:s
	`{`
	`  "port_aggregator": "<port_aggregator>",`
	`  "status": "<status>",`
	`}`

port_ aggregator	LAG identifier; an integer from 1-4096.
status	vLAG status; one of `enable`, `disable`. Default value: `disable`.

Returns

A JSON response containing the updated vLAG instance.

Response Body (JSON)	`{`
	`  "instance_id": "<instance_id>",`
	`  "port_aggregator": "<port_aggregator>",`
	`  "status": "<status>",`
	`}`

where:

Element	Description
instance_id	vLAG instance ID number; an integer from 1-64.
port_ aggregator	LAG identifier; an integer from 1-4096.
status	vLAG status; one of `enable`, `disable`.

update_vlag_isl(conn, parameters)

Configures the port aggregator for the VLAG ISL.

Syntax

```
Vlag.update_vlag_isl(<conn>, <parameters>)
```

where:

Parameter	Description
conn	A connection object for the node.

parameters	A dictionary containing the following JSON key-value pair:
	`{`
	`  "port_aggregator" : <port_aggregator>,`
	`}`

Parameter	Description
port_ aggregator	Port aggregator for the vLAG ISL; an integer from 1-4096.
msg-interval	Determines the interval with which the registration and heartbeat messages are sent to the collector; units of seconds from 1-600. Default value: 5 seconds.

Returns

`True` if the operation succeeded; otherwise `False`.

Virtual Local Area Network Management

The `Vlan` class methods manage VLAN configuration on the node.

These methods are in the file `vlan.rb`.

create_vlan(conn, parameters)

Creates a VLAN instance.

Syntax

`Vlan.create_vlan(<conn>, <parameters>)`

where:

Parameter	**Description**
conn	A connection object for the node.

parameters	A dictionary containing the following JSON key-value pairs:
	`{`
	`  "vlan_name": "<vlan_name>",`
	`  "vlan_id": "<vlan_id>",`
	`  "admin_state": "<admin_state>",`
	`}`

vlan_name	VLAN name; a string up to 32 characters long. To create a vlan with the default name, the *vlan_name* field must be null.
vlan_id	VLAN number.; an integer from 1-4093.
admin_state	The admin status; one of `up`, `down`

Returns

A JSON response containing the updated VLAN parameters:

Response Body
 	
(JSON)
	`{`
	`  "vlan_name": "<vlan_name>",`
	`  "vlan_id": "<vlan_id>",`
	`  "admin_state": "<admin_state>",`
	`}`

where:

Element	**Description**
vlan_name	The name of the VLAN.
vlan_id	VLAN number.; an integer from 1-4093.
admin_state	The admin status; one of `up`, `down`.

delete_vlan(conn, vlan_id)

Deletes a VLAN.

Syntax

`Vlan.delete_vlan(<conn>, <vlan_id>)`

where:

Parameter	Description
conn	A connection object for the node.
vlan_id	VLAN number (an integer from 1-4093) or `all` to delete all user-created VLANs.

Returns

`True` if the operation succeeded; otherwise `False`.

get_all_vlan(conn)

Gets properties of all VLANs.

Syntax

`Vlan.get_all_vlan(<conn>)`

where *conn* is the connection object for the node.

Returns

A JSON response containing the properties of all VLANs:

Response Body
 	
(JSON)
	```
	[
	  {
	"vlan_name": "<vlan_name>",
	"vlan_id": "<vlan_id>",
	"admin_state": "<admin_state>",
	"mst_inst_id": "<mst_inst_id>",
	"interfaces": [
	    {   "if_name": "<if_name>",
	        "bridge_port_mode": "<bridge_port_mode>",
	        "pvid": "<pvid>"
	    }
	]
	  }
	]
	```

where:

Element	Description
vlan_name	The name of the VLAN.
vlan_id	VLAN number.; an integer from 2-3999.

Element	Description
admin_state	The admin status; one of `up`, `down`.
mst_inst_id	MST instance ID; an integer from 0-64. Instance 0 refers to the CIST.
interfaces	Interface members of a VLAN.
if_name	Ethernet interface name (Str).

get_vlan_prop(conn, vlan_id)

Gets properties of a VLAN.

Syntax

`Vlan.get_vlan_prop(<conn>, <vlan_id>)`

where:

Parameter	Description
conn	A connection object for the node.
vlan_id	VLAN number.; an integer from 1-4093.

Returns

A JSON response containing the properties of the specified VLAN.

update_vlan(conn, vlan_id, parameters)

Updates properties of a VLAN.

Syntax

`Vlan.update_vlan(<conn>, <vlan_id>, <parameters>)`

where:

Parameter	Description
conn	A connection object for the node.
vlan_id	VLAN number.; an integer from 1-4093.

parameters	A dictionary containing the following JSON key-value pair:
	`{`
	`  "vlan_name": "<vlan_name>",`
	`  "vlan_id": "<vlan_id>",`
	`  "admin_state": "<admin_state>",`
	`}`

Parameter	Description
vlan_name	VLAN name; a string up to 32 characters long. To create a vlan with the default name, the *vlan_name* field must be null.
admin_state	The admin status; one of `up`, `down`

Returns

A JSON response containing the VLAN name and state:

Response Body
 	
(JSON)
	`{`
	`  "vlan_name": "<`*vlan_name*`>",`
	`  "admin_state": "<`*admin_state*`>",`
	`}`

where:

Parameter	Description
vlan_name	VLAN name; a string up to 32 characters long.
admin_state	The admin status; one of `up`, `down`

VLAN Interface Management

The `VlanIntf` class methods manage VLAN interfaces on the node.

These methods are in the file `vlan_intf.rb`.

get_all_vlan_intf(conn)

Gets VLAN properties of all Ethernet interfaces.

Syntax

`VlanIntf.get_all_vlan_intf(`*conn*`)`

where *conn* is the connection object for the node.

Returns

A JSON response containing

get_vlan_prop_intf(conn, intf)

Gets VLAN properties of a specified Ethernet Interface.

Syntax

`VlanIntf.get_vlan_prop_intf(`<*conn*>`, `<*intf*>`)`

where:

Parameter	**Description**
conn	A connection object for the node.
intf	Ethernet interface name (Str). **Note:** The Ethernet interface must exist.

Returns

A JSON response containing the VLAN properties for the specified interface:

Response Body
 	
(JSON)
	`{`
	`"if_name": "<if_name>",`
	`  "bridge_port_mode": "<bridge_port_mode>",`
	`  "pvid": "<pvid>",`
	`  "vlans": ["<vlan_id>"]`
	`}`

where:

Element	**Description**
if_name	Ethernet interface name (Str).

Element	Description
bridge_port_ mode	Bridge port mode; one of `access`, `trunk`.
pvid	Native VLAN for a port (he access VLAN for access ports or the native VLAN for trunk ports); an integer from 1-4093.
vlans	VLAN memberships; either `all`, `none`, or an integer from 1-4093.

update_vlan_intf(conn, intf, parameters)

Updates properties of an Ethernet interface.

Syntax

`VlanIntf.update_vlan_intf(`*<conn>*`,` *<intf>*`,` *<parameters>*`)`

where:

Parameter	Description
conn	A connection object for the node.
intf	The Ethernet interface name (String). **Note:** The interface must exist.

parameters	A dictionary containing the following JSON key-value pairs:
	`{`
	` "if_name": "<if_name>",`
	` “bridge_port_mode”: “<bridge_port_mode>”`
	` “pvid”: "<pvid>",`
	` “vlans”: ["<vlan_id>"]`
	`}`

if_name	Ethernet interface name (String). **Note:** The Ethernet interface must exist.
bridge_port_ mode	Bridge port mode; one of `access`, `trunk`
pvid	Native VLAN for a port (the access VLAN for access ports or the native VLAN for trunk ports); an integer from 1-4093. Default value: 1.
vlans	(Optional) VLAN memberships; `all`, `none`, or an integer from 1-3999.

Returns

A JSON response containing the updated VLAN properties:

Response Body
 	
(JSON)
	`{`
	`  "if_name": "<if_name>",`
	`  "bridge_port": "<bridge_port>",`
	`  "bridge_port_mode": "<bridge_port_mode>",`
	`  "pvid": "<pvid>",`
	`  "vlans": ["<vlans>"]`
	`}`

where:

Parameter	**Description**
if_name	Ethernet interface name (String). **Note:** The Ethernet interface must exist.
bridge_port	Whether the port is a bridge port; one of `yes`, `no`.
bridge_port_ mode	Bridge port mode; one of `access`, `trunk`
pvid	Native VLAN for a port (the access VLAN for access ports or the native VLAN for trunk ports); an integer from 1-4093. Default value: 1.
vlans	(Optional) VLAN memberships; `all`, `none`, or an integer from 1-3999.

Virtual Router Redundancy Protocol Management

The `Vrrp` class methods manage VRRP configuration on the node.

These methods are in the file `vrrp.rb`.

create_vrrp_intf(conn, intf, parameters)

Creates a VRRP VR instance.

Syntax

`Vrrp.create_vrrp_intf(`*<conn>*`,` *<intf>*`,` *<parameters>*`)`

where:

Parameter	Description
conn	A connection object for the node.
intf	The IP interface name (String). **Note:** The interface must exist.

parameters	A dictionary containing the following JSON key-value pairs::
	`{`
	`    "if_name": "<if_name>",`
	`    "vr_id": "<vr_id>",`
	`    "ip_addr": "<ip_addr>",`
	`    "ad_intvl": "<ad_intvl>",`
	`    "preempt": "<preempt>",`
	`    "prio": "<prio>",`
	`    "admin_state": "<admin_state>",`
	`    "track_if": "<track_if>",`
	`    "accept_mode": "<accept_mode>",`
	`    "switch_back_delay": "<switch_back_delay>",`
	`    "v2_compt": "<v2_compt>"`
	`}`

if_name	Interface name. **Note:** The interface must exist.
vr_id	Virtual Router (VR) identifier; an integer from 1-255.
ip_addr	The IP address of the VR; a valid IPv4 address.
ad_intvl	Advertisement interval (The number of centi-seconds between advertisements for VRRPv3); a multiple of 5 from 5-4095. Default value: 100 centi-seconds.
preempt	Enable the preemption of a lower priority master; one of `yes` (default) , `no`.
prio	The priority of the VR on the switch; an integer from 1-254. Default value: 100.
admin_state	Enable the VR one of `up` (default), `down`.
oper_state	The operation state of the VR; one of `master`, `backup`, `init`.

Parameter	**Description**
track_if	The interface to track by this VR. Default value: `none`. **Note:** If an interface is specified, it must exist.
accept_mode	Enables or disables the accept mode for this session; one of `yes` (default), `no`.
switch_back_ delay	The switch back delay interval; an integer from 1-500000, or 0 to disable (default).
v2_compt	Enables backward compatibility for VRRPv2 for the VR; one of `yes`, `no` (default).

Returns

A JSON response containing information about the updated VRRP VR instance:

<table>
<tr><td>Response Body

(JSON)</td><td><pre>
[
  {
    "if_name": "<if_name>",
    "vr_id": "<vr_id>",
    "ip_addr": "<ip_addr>",
    "ad_intvl": "<ad_intvl>",
    "preempt": "<preempt>",
    "prio": "<prio>",
    "admin_state": "<admin_state>",
    "oper_state": "<oper_state>",
    "track_if": "<track_if>",
    "accept_mode": "<accept_mode>",
    "switch_back_delay": "<switch_back_delay>",
    "v2_compt": "<v2_compt>"
  }
]
</pre></td></tr>
</table>

where:

Element	**Description**
if_name	Interface name.
vr_id	Virtual Router (VR) identifier; an integer from 1-255.
ip_addr	The IP address of the VR; a valid IPv4 address.
ad_intvl	Advertisement interval (The number of centi-seconds between advertisements for VRRPv3); a multiple of 5 from 5-4095.
preempt	Enable the preemption of a lower priority master; one of `yes`, `no`.
prio	The priority of the VR on the switch; an integer from 1-254.
admin_state	Enable the VR one of `up`, `down`.
oper_state	The operation state of the VR; one of `master`, `backup`, `init`.
track_if	The interface to track by this VR.

Element	**Description**
accept_mode	Enables or disables the accept mode for this session; one of `yes`, `no`.
switch_back_delay	The switch back delay interval; an integer from 1-500000, or 0 to disable.
v2_compt	Enables backward compatibility for VRRPv2 for the VR; one of `yes`, `no`.

del_vrrp_intf_vrid(conn, intf, vr_id)

Deletes a VRRP VR instance.

Syntax

`Vrrp.del_vrrp_intf_vrid(<conn>, <intf>, <vr_id>)`

where:

Parameter	**Description**
conn	A connection object for the node.
intf	The IP interface name (String). **Note:** The interface must exist.
vr_id	Virtual Router (VR) identifier; either an integer from 1-255 or the string `all` to delete all VRRP entries in the specified interface.

`True` if the operation succeeded; otherwise `False`.

get_vrrp_intf(conn, intf)

Gets properties of all VRRP VRs under the specified interface.

Syntax

`Vrrp.get_vrrp_intf(<conn>, <intf>)`

where:

Parameter	**Description**
conn	A connection object for the node.
intf	The IP interface name (String). **Note:** The interface must exist.

Returns

A JSON response containing the VRRP properties for that interface:

Response Body
 	
(JSON)
	`[`
	`  {`
	`    "if_name": "<if_name>",`
	`    "vr_id": "<vr_id>",`
	`    "ip_addr": "<ip_addr>",`
	`    "ad_intvl": "<ad_intvl>",`
	`    "preempt": "<preempt>",`
	`    "prio": "<prio>",`
	`    "admin_state": "<admin_state>",`
	`    "oper_state": "<oper_state>",`
	`    "track_if": "<track_if>",`
	`    "accept_mode": "<accept_mode>",`
	`    "switch_back_delay": "<switch_back_delay>",`
	`    "v2_compt": "<v2_compt>"`
	`  }`
	`]`

where:

Element	Description
if_name	Interface name.
vr_id	Virtual Router (VR) identifier; an integer from 1-255.
ip_addr	The IP address of the VR; a valid IPv4 address.
ad_intvl	Advertisement interval (The number of centi-seconds between advertisements for VRRPv3); a multiple of 5 from 5-4095.
preempt	Enable the preemption of a lower priority master; one of `yes`, `no`.
prio	The priority of the VR on the switch; an integer from 1-254. Default value: 100.
admin_state	Enable the VR one of `up` (default), `down`.
oper_state	The operation state of the VR; one of `master`, `backup`, `init`.
track_if	The interface to track by this VR. Default value: `none`. **Note:** If an interface is specified, it must exist.
accept_mode	Enables or disables the accept mode for this session; one of `yes` (default), `no`.
switch_back_ delay	The switch back delay interval; an integer from 1-500000, or 0 to disable (default).
v2_compt	Enables backward compatibility for VRRPv2 for the VR; one of `yes`, `no` (default).

get_vrrp_intf_vrid(conn, intf, vr_id)

Gets properties of a VRRP VR.

Syntax

```
Vrrp.get_vrrp_intf_vrid(<conn>, <intf>, <vr_id>)
```

where:

Parameter	Description
conn	A connection object for the node.
intf	The IP interface name (String). **Note:** The interface must exist.
vr_id	Virtual Router (VR) identifier; either an integer from 1-255.

Returns

A JSON response containing the properties of the specified VRRP VR:

Response Body
 	
(JSON)
	`{`
	`    "if_name": "<if_name>",`
	`    "vr_id": "<vr_id>",`
	`    "ip_addr": "<ip_addr>",`
	`    "ad_intvl": "<ad_intvl>",`
	`    "preempt": "<preempt>",`
	`    "prio": "<prio>",`
	`    "admin_state": "<admin_state>",`
	`    "oper_state": "<oper_state>",`
	`    "track_if": "<track_if>",`
	`    "accept_mode": "<accept_mode>",`
	`    "switch_back_delay": "<switch_back_delay>",`
	`    "v2_compt": "<v2_compt>"`
	`}`

where:

Element	Description
if_name	Interface name.
vr_id	Virtual Router (VR) identifier; an integer from 1-255.
ip_addr	The IP address of the VR; a valid IPv4 address.
ad_intvl	Advertisement interval (The number of centi-seconds between advertisements for VRRPv3); a multiple of 5 from 5-4095.
preempt	Enable the preemption of a lower priority master; one of `yes` , `no`.
prio	The priority of the VR on the switch; an integer from 1-254.
admin_state	Enable the VR one of `up`, `down`.
oper_state	The operation state of the VR; one of `master`, `backup`, `init`.

Element	Description
track_if	The interface to track by this VR.
accept_mode	Enables or disables the accept mode for this session; one of `yes`, `no`.
switch_back_ delay	The switch back delay interval; an integer from 1-500000, or 0 to disable.
v2_compt	Enables backward compatibility for VRRPv2 for the VR; one of `yes`, `no`.

get_vrrp_prop_all(conn)

Gets properties of all VRRP VRs of all interfaces.

Syntax

`Vrrp.get_vrrp_prop_all(`*conn*`)`

where *conn* is the connection object for the node.

Returns

A JSON response containing properties of all VRRP VRs of all interfaces:

Response Body
 	
(JSON)
	```
	[
	  {
	    "if_name": "<if_name>",
	    "vr_id": "<vr_id>",
	    "ip_addr": "<ip_addr>",
	    "ad_intvl": "<ad_intvl>",
	    "preempt": "<preempt>",
	    "prio": "<prio>",
	    "admin_state": "<admin_state>",
	    "oper_state": "<oper_state>",
	    "track_if": "<track_if>",
	    "accept_mode": "<accept_mode>",
	    "switch_back_delay": "<switch_back_delay>",
	    "v2_compt": "<v2_compt>"
	  }
	]
	```

where:

Element	Description
if_name	Interface name.
vr_id	The VRRP session Virtual Router (VR) ID; an integer from 1-255.
ip_addr	The IP address of the VR; a valid IPv4 address.
ad_intvl	Advertisement interval (The number of centi-seconds between advertisements for VRRPv3); a multiple of 5 from 5-4095.
preempt	Enable the preemption of a lower priority master; one of `yes` , `no`.

Element	Description
prio	The priority of the VR on the switch; an integer from 1-254.
admin_state	Enable the VR one of `up`, `down`.
oper_state	The operation state of the VR; one of `master`, `backup`, `init`.
track_if	The interface to track by this VR.
accept_mode	Enables or disables the accept mode for this session; one of `yes`, `no`.
switch_back_ delay	The switch back delay interval; an integer from 1-500000, or 0 to disable.
v2_compt	Enables backward compatibility for VRRPv2 for the VR; one of `yes`, `no`.

update_vrrp_intf_vrid(conn, intf, vr_id, parameters)

Updates the properties of a VRRP VR.

Syntax

`Vrrp.update_vrrp_intf_vrid(<conn>, <intf>, <vr_id>, <parameters>)`

where:

Parameter	Description
conn	A connection object for the node.
intf	The IP interface name (String). **Note:** The interface must exist.
vr_id	Virtual Router (VR) identifier; either an integer from 1-255.

parameters	A dictionary containing the following JSON key-value pairs::
	`{`
	`"if_name": "<if_name>",`
	`"vr_id": "<vr_id>",`
	`"ip_addr": "<ip_addr>",`
	`"ad_intvl": "<ad_intvl>",`
	`"preempt": "<preempt>",`
	`"prio": "<prio>",`
	`"admin_state": "<admin_state>",`
	`"track_if": "<track_if>",`
	`"accept_mode": "<accept_mode>",`
	`"switch_back_delay": "<switch_back_delay>",`
	`"v2_compt": "<v2_compt>"`
	`}`

if_name	Interface name. **Note:** The interface must exist.
vr_id	Virtual Router (VR) identifier; an integer from 1-255.
ip_addr	The IP address of the VR; a valid IPv4 address.

Parameter	**Description**
ad_intvl	Advertisement interval (The number of centi-seconds between advertisements for VRRPv3); a multiple of 5 from 5-4095. Default value: 100 centi-seconds.
preempt	Enable the preemption of a lower priority master; one of `yes` (default) , `no`.
prio	The priority of the VR on the switch; an integer from 1-254. Default value: 100.

Returns

A JSON response containing the updated properties:

Response Body

(JSON)

```
{
    "if_name": "<if_name>",
    "vr_id": "<vr_id>",
    "ip_addr": "<ip_addr>",
    "ad_intvl": "<ad_intvl>",
    "preempt": "<preempt>",
    "prio": "<prio>",
    "admin_state": "<admin_state>",
    "oper_state": "<oper_state>",
    "track_if": "<track_if>",
    "accept_mode": "<accept_mode>",
    "switch_back_delay": "<switch_back_delay>",
    "v2_compt": "<v2_compt>"
}
```

where:

Element	**Description**
if_name	Interface name.
vr_id	Virtual Router (VR) identifier; an integer from 1-255.
ip_addr	The IP address of the VR; a valid IPv4 address.
ad_intvl	Advertisement interval (The number of centi-seconds between advertisements for VRRPv3); a multiple of 5 from 5-4095.
preempt	Enable the preemption of a lower priority master; one of `yes` (default) , `no`.
prio	The priority of the VR on the switch; an integer from 1-254.
admin_state	Enable the VR one of `up`, `down`.
oper_state	The operation state of the VR; one of `master`, `backup`, `init`.
track_if	The interface to track by this VR.
accept_mode	Enables or disables the accept mode for this session; one of `yes`, `no`.

Element	Description
switch_back_ delay	The switch back delay interval; an integer from 1-500000, or 0 to disable.
v2_compt	Enables backward compatibility for VRRPv2 for the VR; one of `yes`, `no`.

REpresentational State Transfer Management

The `Rest` class methods implement REST requests and are called by the other Lenovo Network Ruby Application Programming Interface classes. They are included here for reference purposes.

These methods are in the file `rest_utils.rb`.

delete(conn, url, header)

Performs a DELETE request for the specified URL and switch connection.

Syntax

`Rest.delete(`*<conn>*`, `*<url>*`, `*<header>*`)`

where:

Parameter	**Description**
conn	A connection object for the node.
URL	The URL for the DELETE request.
header	The user and password for the connection.

Returns

A valid JSON response or `False` if there is an error.

get(conn, url, header)

Performs a GET request for the specified URL and switch connection.

Syntax

`Rest.get(`*<conn>*`, `*<url>*`, `*<header>*`)`

where:

Parameter	**Description**
conn	A connection object for the node.
URL	The URL for the GET request.
header	The user and password for the connection.

Returns

A valid JSON response or `False` if there is an error.

post(conn, url, header, parameters)

Performs a POST request for the specified URL and switch connection.

Syntax

`Rest.post(`*<conn>*`, `*<url>*`, `*<header>*`, `*<parameters>*`)`

where:

Parameter	**Description**
conn	A connection object for the node.
URL	The URL for the POST request.
header	The user and password for the connection.
parameters	The JSON body for the POST request.

Returns

A valid JSON response or `False` if there is an error:

put(conn, url, header, parameters)

Performs a PUT request for the specified URL and switch connection.

Syntax

`Rest.put(`*<conn>*`, `*<url>*`, `*<header>*`, `*<parameters>*`)`

where:

Parameter	**Description**
conn	A connection object for the node.
URL	The URL for the PUT request.
header	The user and password for the connection.
parameters	The JSON body for the POST request.

Returns

A valid JSON response or `False` if there is an error:

Product Support

This is a free and open source product from Lenovo. There are no support entitlements available for this plugin. Alternatively, you can file an issue or request via the GitHub repository at:

https://github.com/lenovo/rbapi-cnos/issues

Notices

Lenovo may not offer the products, services, or features discussed in this document in all countries. Consult your local Lenovo representative for information on the products and services currently available in your area.

Any reference to a Lenovo product, program, or service is not intended to state or imply that only that Lenovo product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any Lenovo intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any other product, program, or service.

Lenovo may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

Lenovo (United States), Inc.
1009 Think Place - Building One
Morrisville, NC 27560
U.S.A.

Attention: Lenovo Director of Licensing

LENOVO PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. Lenovo may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

The products described in this document are not intended for use in implantation or other life support applications where malfunction may result in injury or death to persons. The information contained in this document does not affect or change Lenovo product specifications or warranties.

Nothing in this document shall operate as an express or implied license or indemnity under the intellectual property rights of Lenovo or third parties. All information contained in this document was obtained in specific environments and is presented as an illustration. The result obtained in other operating environments may vary.

Lenovo may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any references in this publication to non-Lenovo Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this Lenovo product, and use of those Web sites is at your own risk.

Any performance data contained herein was determined in a controlled environment. Therefore, the result obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Trademarks

Lenovo, the Lenovo logo, Flex System, System x, NeXtScale System, and X-Architecture are trademarks of Lenovo in the United States, other countries, or both.

Intel and Intel Xeon are trademarks of Intel Corporation in the United States, other countries, or both.

Internet Explorer, Microsoft, and Windows are trademarks of the Microsoft group of companies.

Linux is a registered trademark of Linus Torvalds.

Other company, product, or service names may be trademarks or service marks of others.

Important Notes

Processor speed indicates the internal clock speed of the microprocessor; other factors also affect application performance.

CD or DVD drive speed is the variable read rate. Actual speeds vary and are often less than the possible maximum.

When referring to processor storage, real and virtual storage, or channel volume, KB stands for 1 024 bytes, MB stands for 1 048 576 bytes, and GB stands for 1 073 741 824 bytes.

When referring to hard disk drive capacity or communications volume, MB stands for 1 000 000 bytes, and GB stands for 1 000 000 000 bytes. Total user-accessible capacity can vary depending on operating environments.

Maximum internal hard disk drive capacities assume the replacement of any standard hard disk drives and population of all hard-disk-drive bays with the largest currently supported drives that are available from Lenovo.

Maximum memory might require replacement of the standard memory with an optional memory module.

Each solid-state memory cell has an intrinsic, finite number of write cycles that the cell can incur. Therefore, a solid-state device has a maximum number of write cycles that it can be subjected to, expressed as total bytes written (TBW). A device that has exceeded this limit might fail to respond to system-generated commands or might be incapable of being written to. Lenovo is not responsible for replacement of a device that has exceeded its maximum guaranteed number of program/erase cycles, as documented in the Official Published Specifications for the device.

Lenovo makes no representations or warranties with respect to non-Lenovo products. Support (if any) for the non-Lenovo products is provided by the third party, not Lenovo.

Some software might differ from its retail version (if available) and might not include user manuals or all program functionality.

Recycling Information

Lenovo encourages owners of information technology (IT) equipment to responsibly recycle their equipment when it is no longer needed. Lenovo offers a variety of programs and services to assist equipment owners in recycling their IT products. For information on recycling Lenovo products, go to:

http://www.lenovo.com/recycling

Particulate Contamination

Attention: Airborne particulates (including metal flakes or particles) and reactive gases acting alone or in combination with other environmental factors such as humidity or temperature might pose a risk to the device that is described in this document.

Risks that are posed by the presence of excessive particulate levels or concentrations of harmful gases include damage that might cause the device to malfunction or cease functioning altogether. This specification sets forth limits for particulates and gases that are intended to avoid such damage. The limits must not be viewed or used as definitive limits, because numerous other factors, such as temperature or moisture content of the air, can influence the impact of particulates or environmental corrosives and gaseous contaminant transfer. In the absence of specific limits that are set forth in this document, you must implement practices that maintain particulate and gas levels that are consistent with the protection of human health and safety. If Lenovo determines that the levels of particulates or gases in your environment have caused damage to the device, Lenovo may condition provision of repair or replacement of devices or parts on implementation of appropriate remedial measures to mitigate such environmental contamination. Implementation of such remedial measures is a customer responsibility..

Contaminant	Limits
Particulate	• The room air must be continuously filtered with 40% atmospheric dust spot efficiency (MERV 9) according to ASHRAE Standard 52.2[1]. • Air that enters a data center must be filtered to 99.97% efficiency or greater, using high-efficiency particulate air (HEPA) filters that meet MIL-STD-282. • The deliquescent relative humidity of the particulate contamination must be more than 60%[2]. • The room must be free of conductive contamination such as zinc whiskers.

Contaminant	**Limits**
Gaseous	• Copper: Class G1 as per ANSI/ISA 71.04-1985[3] • Silver: Corrosion rate of less than 300 Å in 30 days

[1] ASHRAE 52.2-2008 - *Method of Testing General Ventilation Air-Cleaning Devices for Removal Efficiency by Particle Size*. Atlanta: American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc.

[2] The deliquescent relative humidity of particulate contamination is the relative humidity at which the dust absorbs enough water to become wet and promote ionic conduction.

[3] ANSI/ISA-71.04-1985. *Environmental conditions for process measurement and control systems: Airborne contaminants*. Instrument Society of America, Research Triangle Park, North Carolina, U.S.A.

Telecommunication Regulatory Statement

This product may not be certified in your country for connection by any means whatsoever to interfaces of public telecommunications networks. Further certification may be required by law prior to making any such connection. Contact a Lenovo representative or reseller for any questions.

Electronic Emission Notices

When you attach a monitor to the equipment, you must use the designated monitor cable and any interference suppression devices that are supplied with the monitor.

Federal Communications Commission (FCC) Statement

Note: This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference, in which case the user will be required to correct the interference at his own expense.

Properly shielded and grounded cables and connectors must be used to meet FCC emission limits. Lenovo is not responsible for any radio or television interference caused by using other than recommended cables and connectors or by unauthorized changes or modifications to this equipment. Unauthorized changes or modifications could void the user's authority to operate the equipment.

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that might cause undesired operation.

Industry Canada Class A Emission Compliance Statement

This Class A digital apparatus complies with Canadian ICES-003.

Avis de Conformité à la Réglementation d'Industrie Canada

Cet appareil numérique de la classe A est conforme à la norme NMB-003 du Canada.

Australia and New Zealand Class A Statement

Attention: This is a Class A product. In a domestic environment this product may cause radio interference in which case the user may be required to take adequate measures.

European Union - Compliance to the Electromagnetic Compatibility Directive

This product is in conformity with the protection requirements of EU Council Directive 2004/108/EC (until April 19, 2016) and EU Council Directive 2014/30/EU (from April 20, 2016) on the approximation of the laws of the Member States relating to electromagnetic compatibility. Lenovo cannot accept responsibility for any failure to satisfy the protection requirements resulting from a non-recommended modification of the product, including the installation of option cards from other manufacturers.

This product has been tested and found to comply with the limits for Class A equipment according to European Standards harmonized in the Directives in compliance. The limits for Class A equipment were derived for commercial and industrial environments to provide reasonable protection against interference with licensed communication equipment.

CE Lenovo, Einsteinova 21, 851 01 Bratislava, Slovakia

Warning: This is a Class A product. In a domestic environment this product may cause radio interference in which case the user may be required to take adequate measures.

Germany Class A Statement

Zulassungsbescheinigung laut dem Deutschen Gesetz über die elektromagnetische Verträglichkeit von Betriebsmitteln, EMVG vom 20. Juli 2007 (früher Gesetz über die elektromagnetische Verträglichkeit von Geräten), bzw. der EMV EG Richtlinie 2004/108/EC (früher 89/336/EWG), für Geräte der Klasse A.

Dieses Gerät ist berechtigt, in übereinstimmung mit dem Deutschen EMVG das EG-Konformitätszeichen - CE - zu führen. Verantwortlich für die Konformitätserklärung nach Paragraf 5 des EMVG ist die Lenovo (Deutschland) GmbH, Gropiusplatz 10, D-70563 Stuttgart.

Informationen in Hinsicht EMVG Paragraf 4 Abs. (1) 4:

Das Gerät erfüllt die Schutzanforderungen nach EN 55024 und EN 55022 Klasse A.

Nach der EN 55022: “Dies ist eine Einrichtung der Klasse A. Diese Einrichtung kann im Wohnbereich Funkstörungen verursachen; in diesem Fall kann vom Betreiber verlangt werden, angemessene Maßnahmen durchzuführen und dafür aufzukommen.”

Nach dem EMVG: Dieses Produkt entspricht den Schutzanforderungen der EU-Richtlinie 2004/108/EG (früher 89/336/EWG) zur Angleichung der Rechtsvorschriften über die elektromagnetische Verträglichkeit in den EU-Mitgliedsstaaten und hält die Grenzwerte der EN 55022 Klasse A ein.

Um dieses sicherzustellen, sind die Geräte wie in den Handbüchern beschrieben zu installieren und zu betreiben. Des Weiteren dürfen auch nur von der Lenovo empfohlene Kabel angeschlossen werden. Lenovo übernimmt keine Verantwortung für die Einhaltung der Schutzanforderungen, wenn das Produkt ohne Zustimmung der Lenovo verändert bzw. wenn Erweiterungskomponenten von Fremdherstellern ohne Empfehlung der Lenovo gesteckt/eingebaut werden.

Deutschland:

Einhaltung des Gesetzes über die elektromagnetische Verträglichkeit von Betriebsmittein
Dieses Produkt entspricht dem “Gesetz über die elektromagnetische Verträglichkeit von Betriebsmitteln” EMVG (früher “Gesetz über die elektromagnetische Verträglichkeit von Geräten"). Dies ist die Umsetzung der EU-Richtlinie 2004/108/EG (früher 89/336/EWG) in der Bundesrepublik Deutschland.

Zulassungsbescheinigung laut dem Deutschen Gesetz über die elektromagnetische Verträglichkeit von Betriebsmitteln, EMVG vom 20. Juli 2007 (früher Gesetz über die elektromagnetische Verträglichkeit von Geräten), bzw. der EMV EG Richtlinie 2004/108/EC (früher 89/336/EWG), für Geräte der Klasse A.

Dieses Gerät ist berechtigt, in übereinstimmung mit dem Deutschen EMVG das EG-Konformitätszeichen - CE - zu führen. Verantwortlich für die Konformitätserklärung nach Paragraf 5 des EMVG ist die Lenovo (Deutschland) GmbH, Gropiusplatz 10, D-70563 Stuttgart.

Informationen in Hinsicht EMVG Paragraf 4 Abs. (1) 4:
Das Gerät erfüllt die Schutzanforderungen nach EN 55024 und EN 55022 Klasse A.

Nach der EN 55022: “Dies ist eine Einrichtung der Klasse A. Diese Einrichtung kann im Wohnbereich Funkstörungen verursachen; in diesem Fall kann vom Betreiber verlangt werden, angemessene Maßnahmen durchzuführen und dafür aufzukommen."

Nach dem EMVG: “Geräte dürfen an Orten, für die sie nicht ausreichend entstört sind, nur mit besonderer Genehmigung des Bundesministers für Post und Telekommunikation oder des Bundesamtes für Post und Telekommunikation betrieben werden. Die Genehmigung wird erteilt, wenn keine elektromagnetischen Störungen zu erwarten sind.” (Auszug aus dem EMVG, Paragraph 3, Abs. 4). Dieses Genehmigungsverfahren ist nach Paragraph 9 EMVG in Verbindung mit der entsprechenden Kostenverordnung (Amtsblatt 14/93) kostenpflichtig.

Anmerkung: Um die Einhaltung des EMVG sicherzustellen sind die Geräte, wie in den Handbüchern angegeben, zu installieren und zu betreiben.

Japan VCCI Class A Statement

この装置は、クラス A 情報技術装置です。この装置を家庭環境で使用すると電波妨害を引き起こすことがあります。この場合には使用者が適切な対策を講ずるよう要求されることがあります。 VCCI–A

This is a Class A product based on the standard of the Voluntary Control Council for Interference (VCCI). If this equipment is used in a domestic environment, radio interference may occur, in which case the user may be required to take corrective actions.

Japan Electronics and Information Technology Industries Association (JEITA) Statement

高調波ガイドライン適合品

Japan Electronics and Information Technology Industries Association (JEITA) Confirmed Harmonics Guidelines (products less than or equal to 20 A per phase)

高調波ガイドライン準用品

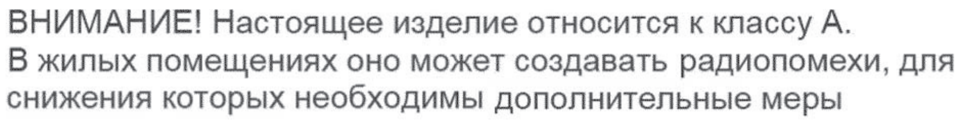
Japan Electronics and Information Technology Industries Association (JEITA) Confirmed Harmonics Guidelines with Modifications (products greater than 20 A per phase).

Korea Communications Commission (KCC) Statement

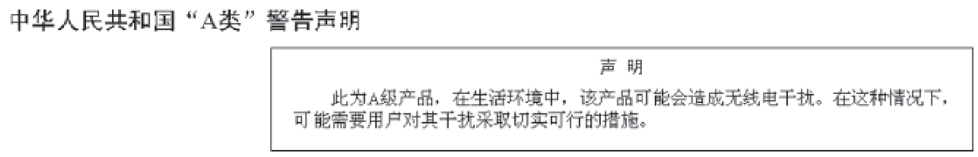
이 기기는 업무용(A급)으로 전자파적합기기로
서 판매자 또는 사용자는 이 점을 주의하시기
바라며, 가정외의 지역에서 사용하는 것을 목
적으로 합니다.

This is electromagnetic wave compatibility equipment for business (Type A). Sellers and users need to pay attention to it. This is for any areas other than home.

Russia Electromagnetic Interference (EMI) Class A statement


ВНИМАНИЕ! Настоящее изделие относится к классу А.
В жилых помещениях оно может создавать радиопомехи, для снижения которых необходимы дополнительные меры

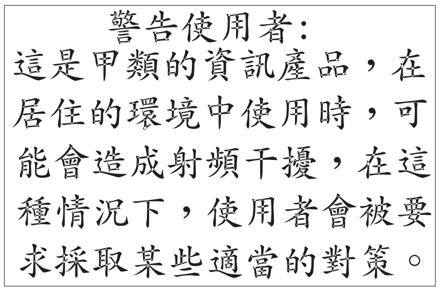
People’s Republic of China Class A electronic emission statement


中华人民共和国“A类”警告声明

声 明

此为A级产品，在生活环境中，该产品可能会造成无线电干扰。在这种情况下，可能需要用户对其干扰采取切实可行的措施。

Taiwan Class A compliance statement


警告使用者:
這是甲類的資訊產品，在居住的環境中使用時，可能會造成射頻干擾，在這種情況下，使用者會被要求採取某些適當的對策。