USER GUIDE

Extreme SLX-OS Puppet User's Guide, 18r. 2.00

Supporting the ExtremeRouting SLX 9850 and SLX 9640 and the ExtremeSwitching SLX 9540 Devices

9035764
November 2018

Legal Notice

Extreme Networks, Inc. reserves the right to make changes in specifications and other information contained in this document and its website without prior notice. The reader should in all cases consult representatives of Extreme Networks to determine whether any such changes have been made.

The hardware, firmware, software or any specifications described or referred to in this document are subject to change without notice.

Trademarks

Extreme Networks and the Extreme Networks logo are trademarks or registered trademarks of Extreme Networks, Inc. in the United States and/or other countries.

All other names (including any product names) mentioned in this document are the property of their respective owners and may be trademarks or registered trademarks of their respective companies/owners.

For additional information on Extreme Networks trademarks, please see: www.extremenetworks.com/company/legal/trademarks

Open Source Declarations

Some software files have been licensed under certain open source or third-party licenses. End-user license agreements and open source declarations can be found at: www.extremenetworks.com/support/policies/software-licensing

Contents

Preface

Document conventions

The document conventions describe text formatting conventions, command syntax conventions, and important notice formats used in Extreme technical documentation.

Notes, cautions, and warnings

Notes, cautions, and warning statements may be used in this document. They are listed in the order of increasing severity of potential hazards.

NOTE
A Note provides a tip, guidance, or advice, emphasizes important information, or provides a reference to related information.

ATTENTION
An Attention statement indicates a stronger note, for example, to alert you when traffic might be interrupted or the device might reboot.

CAUTION
A Caution statement alerts you to situations that can be potentially hazardous to you or cause damage to hardware, firmware, software, or data.

DANGER
A Danger statement indicates conditions or situations that can be potentially lethal or extremely hazardous to you. Safety labels are also attached directly to products to warn of these conditions or situations.

Text formatting conventions

Text formatting conventions such as boldface, italic, or Courier font may be used to highlight specific words or phrases.

Format	Description
bold text	Identifies command names.
	Identifies keywords and operands.
	Identifies the names of GUI elements.
	Identifies text to enter in the GUI.
italic text	Identifies emphasis.
	Identifies variables.
	Identifies document titles.
`Courier font`	Identifies CLI output.

Format	Description
	Identifies command syntax examples.

Command syntax conventions

Bold and italic text identify command syntax components. Delimiters and operators define groupings of parameters and their logical relationships.

Convention	Description
bold text	Identifies command names, keywords, and command options.
italic text	Identifies a variable.
[ ]	Syntax components displayed within square brackets are optional. Default responses to system prompts are enclosed in square brackets.
{ **x** | **y** | **z** }	A choice of required parameters is enclosed in curly brackets separated by vertical bars. You must select one of the options.
x | **y**	A vertical bar separates mutually exclusive elements.
< >	Nonprinting characters, for example, passwords, are enclosed in angle brackets.
...	Repeat the previous element, for example, *member*[*member*...].
\	Indicates a "soft" line break in command examples. If a backslash separates two lines of a command input, enter the entire command at the prompt without the backslash.

Extreme resources

Visit the Extreme website to locate related documentation for your product and additional Extreme resources.

White papers, data sheets, and the most recent versions of Extreme software and hardware manuals are available at www.extremenetworks.com. Product documentation for all supported releases is available to registered users at www.extremenetworks.com/support/documentation.

Document feedback

Quality is our first concern at Extreme, and we have made every effort to ensure the accuracy and completeness of this document. However, if you find an error or an omission, or you think that a topic needs further development, we want to hear from you.

You can provide feedback in two ways:

- Use our short online feedback form at http://www.extremenetworks.com/documentation-feedback-pdf/
- Email us at internalinfodev@extremenetworks.com

Provide the publication title, part number, and as much detail as possible, including the topic heading and page number if applicable, as well as your suggestions for improvement.

Contacting Extreme Technical Support

As an Extreme customer, you can contact Extreme Technical Support using one of the following methods: 24x7 online or by telephone. OEM customers should contact their OEM/solution provider.

If you require assistance, contact Extreme Networks using one of the following methods:

- GTAC (Global Technical Assistance Center) for immediate support
  - Phone: 1-800-998-2408 (toll-free in U.S. and Canada) or +1 408-579-2826. For the support phone number in your country, visit: www.extremenetworks.com/support/contact.
  - Email: support@extremenetworks.com. To expedite your message, enter the product name or model number in the subject line.
- GTAC Knowledge - Get on-demand and tested resolutions from the GTAC Knowledgebase, or create a help case if you need more guidance.
- The Hub - A forum for Extreme customers to connect with one another, get questions answered, share ideas and feedback, and get problems solved. This community is monitored by Extreme Networks employees, but is not intended to replace specific guidance from GTAC.
- Support Portal - Manage cases, downloads, service contracts, product licensing, and training and certifications.

Before contacting Extreme Networks for technical support, have the following information ready:

- Your Extreme Networks service contract number and/or serial numbers for all involved Extreme Networks products
- A description of the failure
- A description of any action(s) already taken to resolve the problem
- A description of your network environment (such as layout, cable type, other relevant environmental information)
- Network load at the time of trouble (if known)
- The device history (for example, if you have returned the device before, or if this is a recurring problem)
- Any related RMA (Return Material Authorization) numbers

About This Document

This document is released in conjunction with SLX-OS 17r.2.01a.

Interface module capabilities

The following table lists the supported capabilities for the following ExtremeRouting SLX 9850 interface modules:

- BR-SLX9850-10Gx72S-M
- BR-SLX9850-100Gx36CQ-M
- BR-SLX9850-10Gx72S-D
- BR-SLX9850-100Gx36CQ-D

TABLE 1 ExtremeRouting SLX 9850 interface modules capabilities

Capability	Modular interface module
MPLS	Yes
Packet Buffers per interface module	12 GB (72x10G) 36 GB (Flex)

Supported hardware

In those instances in which procedures or parts of procedures documented here apply to some devices but not to others, this guide identifies exactly which devices are supported by this release and which are not.

Although many different software and hardware configurations are tested and supported by this release, documenting all possible configurations and scenarios is beyond the scope of this document.

The following hardware platforms are supported by this release:

- ExtremeRouting SLX 9850-4 router
- ExtremeRouting SLX 9850-8 router
- ExtremeRouting SLX 9640 router
- ExtremeSwitching SLX 9540 switch

To obtain information about other releases, refer to the documentation specific to that release.

Interface module capabilities

The following table lists the supported capabilities for the following SLX 9850 interface modules:

- BR-SLX9850-10Gx72S-M
- BR-SLX9850-100Gx36CQ-M
- BR-SLX9850-10Gx72S-D

- BR-SLX9850-100Gx36CQ-D
- BR-SLX9850-100Gx12CQ-M

TABLE 2 SLX 9850 interface modules capabilities

Capability	Modular interface module
MPLS	Yes
Packet buffer memory per interface module	12GB (BR-SLX9850-10Gx72S-M) 36GB (BR-SLX9850-100Gx36CQ-M) 8GB (BR-SLX9850-10Gx72S-D) 24GB (BR-SLX9850-100Gx36CQ-D) 8GB (BR-SLX9850-100Gx12CQ-M)

What's new in this document

On October 30, 2017, Extreme Networks, Inc. acquired the data center networking business from Brocade Communications Systems, Inc. This document has been updated to remove or replace references to Brocade Communications, Inc. with Extreme Networks., Inc., as appropriate .

For the complete list of supported features and the summary of enhancements and configuration notes for this release, refer to the Extreme SLX-OS OS Release Notes.

Extreme Puppet implementation

Extreme Puppet implementation

This document provides information on the third-party virtual machine (TPVM) application, Puppet in the SLX-OS 17r.2.00. Puppet is a scripting language available from Puppet Labs that system administrators can use to automate configuration and management of a data center.

Using Puppet, you can:

- Manage a data center's resources and infrastructure life cycle, from provisioning and configuration to orchestration and reporting.
- Automate repetitive tasks, quickly deploy critical applications, and probatively manage change.
- Scale to thousands of servers, either on location or in the cloud.

Software Requirement

The Extreme Puppet solution requires the following software:

- SLX-OS 17r.2.00 or later
- Puppet Enterprise 4.9 +

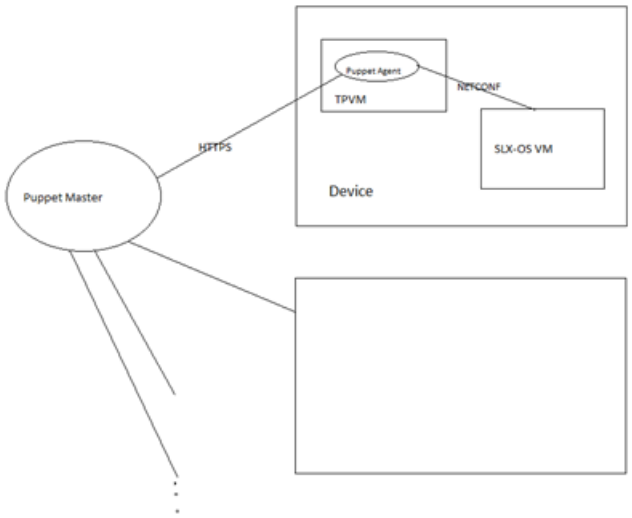
How Puppet works

In the Agent/Master architecture, a Puppet master server controls important configuration info and managed agent nodes request only their own configuration catalogs. Periodically, a Puppet agent sends facts to the Puppet master and requests a catalog. The master compiles and returns the node's catalog. In general, there are two ways of integration with Puppet agent/master architecture.

- Support Puppet agent on a device (router or switch), and let the device be managed by the Puppet master as a node. In this mode types and providers are installed on the master and assigned to a node. The providers, and resource are downloaded to the node and the catalog is executed periodically. In this case, the provider can use any of the switch internal mechanisms (Python, CLI) to configure the switch.
- Implement a provider in such a way that it can run on any puppet agent (that is, where ever there is a puppet agent). In this case, the provider must be capable enough to communicate and configure the router or switch. This can be accomplished by the APIs provided by the device such as , NETCONF and REST API.

Currently, we support a hybrid type of Puppet support. We allows the Provider to run on a Puppet agent installed by the customer on the TPVM. NETCONF is used as the API to communicate from the TPVM to the SLX-OS VM within the Extreme device.

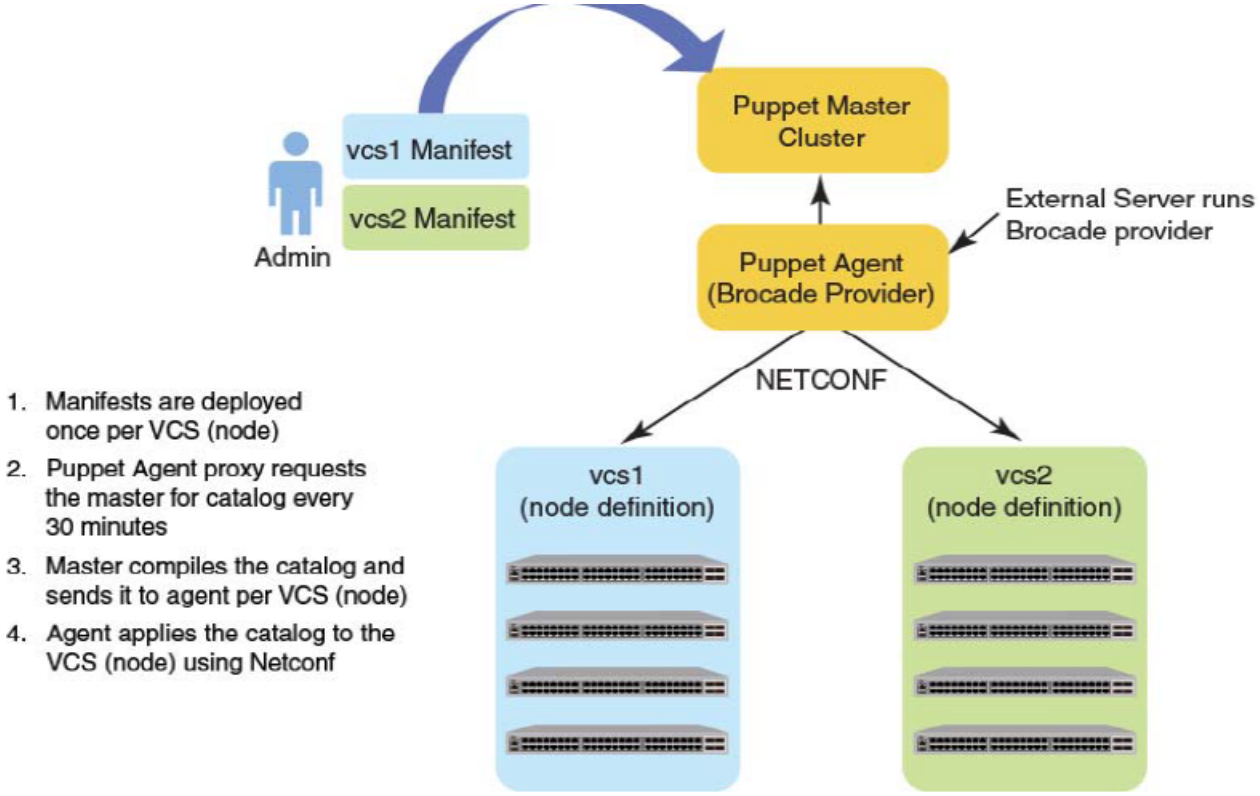
FIGURE 1 Extreme Puppet implementation



Puppet can also run in a stand-alone architecture, where each managed server has its own complete copy of the configuration, and it compiles its own catalog. The managed nodes run the Puppet apply application, as a scheduled task/cron job (or on demand for initial configuration for smaller configuration tasks).

The following illustration depicts how the major Puppet components interact.

FIGURE 2 Extreme Puppet components



Component	Role
Puppet master	In a standard Puppet client-server deployment, the server is known as the master. The Puppet master controls the data center configuration. The puppet master serves configuration catalogs on demand to the puppet agent service that runs on the clients. The master uses an HTTP server to provide catalogs. The master can reside in any location, but the master must have connectivity to the agents. The master contains the Puppet-language file or files known as manifests.
Agent	Puppet agents are Puppet client daemons that receive their configuration information from the Puppet master. A computer running the puppet agent is called an agent node. The Agent nodes must have connectivity to both the Puppet Master and to the Extreme VCS cluster or clusters that it will configure.
Extreme Provider	Extreme Providers implement resource types on a specific type of system, using the system's own tools. The Extreme Provider executes the manifests for the purpose of configuring the associated VCS cluster(s) and switches. The Extreme Provider uses NETCONF calls to configure the switch or switches. These Netconf calls are transparent to the Puppet administrator and user. In general, providers are simple Ruby wrappers around shell commands, so they are usually short and easy to create. Instructions are given later in this chapter and where to obtain the Provider and run it on the Puppet Master.
Manifest	A file containing code written in the Puppet language, and named with the .pp file extension. The manifest usually defines nodes, so that each managed agent node receives a unique catalog.

Component	Role
Catalog	A catalog is a compilation of all the resources that will be applied to a given system and the relationships between those resources.

Limitation and restriction

- Puppet provider does not retry in case NETCONF request times out (switch reboot, failover etc), The script gets run the next time puppet script is scheduled.
- All errors are logged to /var/log/syslog.
- Only the Netdev types mentioned in this document are supported.

Main steps for Using Puppet

The main steps needed to employ Puppet in your environment are:

1. Determining where you want the Puppet master to reside. This can be a server in any location.
2. Determining which node or nodes you want to be Agent nodes. The Agent nodes must have connectivity to both the Puppet Master and the Extreme VCS cluster or clusters that you want an Agent node to configure.
3. Installing the Extreme Provider on the Puppet Master.
4. Setting up the site manifest file.
5. Writing one manifest for each VCS cluster that you want to manage by means of Puppet.
6. Placing the manifest(s) on the Master Puppet server.

Installing the Extreme Provider

Extreme SLX-OS devices running SLX-OS firmware can be configured using this Puppet module. This module supports basic interface, L2 port configuration, LAG and VLAN configurations. This module has been derived from Netdev_stdlib with a modification required to configure the SLX-OS devices.

Follow these steps to install the Extreme Provider onto your Puppet Master:

1. You must obtain the Extreme-supported netdev types and the Extreme Provider from: https://forge.puppetlabs.com. Download them to your Puppet Master.
2. Install the Extreme Provider on the Puppet Master server by running the following command on the server:

puppet module install Extreme-netdev_stdlib_slxos-1.0.0

Puppet agent is not packaged with the SLX-OS software. You must install the puppet Master on a separate server. The puppet Agent can be installed on the TPVM, and must have connectivity to the puppet Master. On chassis-based devices, the virtual-IP can be used to communicate from the TPVM to the SLX-OS VM.

Site manifest

The main “point of entry” manifest is used by the puppet master when compiling a catalog. The location of this manifest is set with the manifest setting in the puppet.conf file, which is downloaded when you install the Provider. The default value of the site manifest location is usually /etc/puppet/manifests/site.pp or /etc/puppetlabs/puppet/manifests/site.pp. A site manifest file must be configured on the Puppet Master to define all the Agent node(s) and switches that must be configured. Refer to Puppet documentation for detailed

information abouthow to configure the site manifest file. The file name is site.pp. You need to create your own site manifest file. The following is an example of a site manifest file. In this example, datacenter10 and datacenter20 are Agent nodes.

```
root@ldap:/etc/puppetlabs/code/environments/production#cat
manifests/site.pp
node tpvm.englab.extreme.com {
   slxos_netdev_interface { "ethernet 0/7":
      ensure        => present,
      admin         => up,
      description   => "Ethernet 0/7",
      mtu           => 1548,
      speed         => "10000",
      target     => "http://admin:password@10.11.12.14:830",
   }
   slxos_netdev_interface { "ethernet 0/8":
      ensure        => present,
      admin         => up,
      description   => "Ethernet 0/8",
      mtu           => 1548,
      speed         => "10000",
      target     => "http://admin:password@10.11.12.14:830",
   }
   slxos_netdev_interface { "ethernet 0/9":
      ensure        => present,
      admin         => up,
      description   => "Ethernet 0/9",
      mtu           => 1548,
      speed         => "10000",
      target     => "http://admin:password@10.11.12.14:830",
   }
   slxos_netdev_vlan { "vlan 2":
      ensure        => present,
      vlan_id       => 2,
      description   => "Vlan 2",
      target     => "http://admin:password@10.11.12.14:830",
   }
   slxos_netdev_vlan { "vlan 3":
```

```
        ensure        => present,
        vlan_id       => 3,
        description   => "Vlan 3",
        target     => "http://admin:password@10.11.12.14:830",
    }
    slxos_netdev_vlan { "vlan 4":
        ensure        => present,
        vlan_id       => 4,
        description   => "Vlan 4",
        target     => "http://admin:password@10.11.12.14:830",
    }
    slxos_netdev_l2_interface { "ethernet 0/7":
        tagged_vlans  => ["2","3"],
        require       => Slxos_netdev_vlan["vlan 2", "vlan 3"],
        target     => "http://admin:password@10.11.12.14:830",
    }
    slxos_netdev_lag { "10":
        minimum_links => 5,
        lacp          => active,
        type          => standard,
        links         => ["ethernet 0/3", "ethernet 0/4"],
        target     => "http://admin:password@10.11.12.14:830",
    }

}
root@ldap:/etc/puppetlabs/code/environments/production#
```

Puppet Documentation

Refer to the following URLs for complete Puppet documentation.

- Main Puppet documentation site: https://docs.puppetlabs.com/
- Information on the main configuration file: https://docs.puppetlabs.com/puppet/latest/reference/config_file_main.html
- Information on the main manifests: https://docs.puppetlabs.com/puppet/4.1/reference/dirs_manifest.html

Additional Information

If you are using Puppet to manage resources, the properties that you configure with Puppet can be changed by using the command line interface (CLI). However, if any such properties are changed from the CLI, the Puppet-managed settings go back into effect when the script is run. This takes place either every 30 minutes (automatically) or when you manually run the script. If you want to change the 30-

minute proxy-request interval, you can change the run interval property of the main configuration file. For more information, refer to https://docs.puppetlabs.com/puppet/latest/reference/config_file_main.html.

Using the Extreme-supported Puppet netdev types

slxos_netdev_interface

The slxos_netdev_interface type allows you to manage the configuration of a physical interface.

Syntax

```
slxos_netdev_interface { "name":
    admin => [up | down],
    mtu => mtu_value,
    speed => "speed_value",
    duplex => $target
    target => $target
    }
```

Properties

TABLE 3 slxos_netdev_interface properties

Property	Description
name (required)	Specifies the name of the physical interface.
admin (optional)	Configures the interface as administratively enabled or disabled. By default, it is up.
mtu_value (optional)	Configures the interface maximum transmission unit (MTU) value.
speed_value (optional)	Configures the interface speed. Possible values are auto, 10m, 100m, 1g, and 10g.The default is auto.
duplex	Configures duplex mode: auto, full, or half. By default, it is set to auto.
target (optional)	Specifies device connection information. For example, http://admin:password@[3001::1]:830 NOTE: The target can also be specified by using a shortcut. An example is: $ip23= "http://admin:password@10.12.13.14:830; you can then use $ip23 subsequently.

Examples

The following Puppet code segment configures several properties for the Ethernet interface.

```
class vcs1 {
$ip23= "http://admin:password@10.12.13.14:830"
slxos_netdev_interface { “eth-1/1":
ensure => present,
admin => up,
description => "this is a storage port",
mtu => 2200,
speed => "10000",
target => $ip23
}
```

Use the **show running interface** command to verify the results of the code segment on the switch.

```
device# sh run interface ethenet 1/1
interface ethernet 1/1
speed 10000
mtu 2200
description this is a storage port
no shutdown
!
```

slxos_netdev_vlan

The slxos_netdev_vlan type allows you to manage VLANS on a switch.

Syntax

```
netdev_vlan { "name":
    vlan_id => VLAN_id,
    description => "vlan-description",
    target => $target
    }
```

Properties

TABLE 4 slxos_netdev_vlan properties

Property	Description
name (required)	Specifies the name of the VLAN. For example “blue”.
description	Assigns the description value to the VLAN. The default is: “Puppet created VLAN: <name>: <vlan-id>” . THe VLAN tag-ID value range is from 1 through 4095.
target (optional)	Specifies device connection information. For example, http://admin:password@[3001::1]:830 NOTE: The target can also be specified by using a shortcut. An example is: $ip23= “http://admin:password@10.24.81.23:830; you can then use $ip23 subsequently.

Examples

The following Puppet code segment configures several VLANs and associated properties for each of these VLANS.

```
class vcs1 {
$ip23= "http://admin:password@10.11.12.13:830"
$vlans= {
'v10' => {ensure => present, vlan_id =>10, description => 'Vlan 10', target
=>$ip23},
'v11' => {ensure => present, vlan_id =>11, description => 'Vlan 11', target
=>$ip23},
'v12' => {ensure => present, vlan_id =>12, description => 'Vlan 12', target
=>$ip23},
'v13' => {ensure => present, vlan_id =>13, description => 'Vlan 13', target
=>$ip23},
'v14' => {ensure => present, vlan_id =>14, target =>$ip23},
}create_resources(slxos_netdev_vlan, $vlans)}
```

Use the **show running interface vlan** command to verify the results of the code segment on the switch.

```
device# show run interfcae vlan
interface Vlan 1
!
interface Vlan 10
description Vlan 10
!
interface Vlan 11
description Vlan 11
!
interface Vlan 12
```

slxos_netdev_l2_interface

The slxos_netdev_l2_interface type allows you to manage assignment of VLANS to ports on a switch.

Syntax

```
slxos_netdev_l2_interface { "name":
    ensure => [present | absent],
    vlan_tagging => [enable | disable],
    description =>  "vlan-description"
    tagged_vlans => (vlan | [vlan1, vlan2, vlan3, ...]),
    untagged_vlan => vlan,
    native_vlans => vlan,

    target => $target
    }
```

Properties

TABLE 5 slxos_netdev_l2_interface properties

Property	Description
name (required)	Specifies the name of the interface.
vlan_tagging	Configures the mode for the given port as access or trunk. A value of enable configures the port in trunk mode, in which tagged packets are processed. A value of disable (the default), configures the port in access mode, in which tagged packets are discarded. If you do not specify a value for this attribute, but you do set the tagged_vlans attribute, the port is configured as a trunk port.
description	The switch interface description.
tagged_vlans	Specifies VLAN IDs for tagged packets. This could be a single value, or an array of values. When this property is set, the vlan_tagging property defaults to enabled.
untagged_vlan (optional)	Specifies VLAN IDs for untagged packets. If the port is also processing tagged packets, this VLAN is the native VLAN.
target (optional)	Specifies device connection information. For example, http://admin:password@[3001::1]:830 NOTE: The target can also be specified by using a shortcut. An example is: $ip23= "http://admin:password@10.11.12.13:830; you can then use $ip23 subsequently.

Examples

The following Puppet code segment configures several properties for the Ethernet interface.

```
class vcs1 {
$ip23= "http://admin:password@10.11.12.13:830"
slxos_netdev_interface { "eth-1/1":
ensure => present,
admin => up,
description => "this is a storage port",
mtu => 2200,
speed => "10000",
target => $ip23
}
```

Use the **show running interface** command to verify the results of the code segment on the switch.

```
device# sh run interface ethenet 1/1
interface ethernet 1/1
speed 10000
mtu 2200
description this is a storage port
no shutdown
!
```

slxos_netdev_lag

The slxos_netdev_LAG type allows you to manage link aggregation groups.

Syntax

```
slxos_netdev_lag { 'name':
    ensure => [present | absent],
    active => [true | false],
    lacp => [active | passive | disabled],
    description => "vlag-description”,
    minimum_links => minimum_links_value,
    links => $lag_ports,
    }
```

Properties

TABLE 6 slxos_netdev_lag properties

Property	Description
name (required)	Specifies the name of the LAG. For example 10.
active (required)	Active configuration. By default, it is set to true.
lacp (optional)	Controls if and how the Link Aggregation Control Protocol (LACP) is used: • On (default)—LACP is not used. • Active —LACP is in the active mode. • Passive—LACP is in the passive mode.
description (optional)	Configures the VLAN description.
minimum_links (optional)	Specifies the number of physical links that must be in the “up” condition to declare the LAG port as being in the “up” condition. By default, this value is set to 0.
links (required)	Specifies a list of physical interfaces that compose the LAG bundle.

Examples

The following Puppet code segment configures a LAG named 10 with several properties.

```
class vcs1 {
$ip23= "http://admin:password@10.11.12.13:830"
$lag_ports = ['te-1/0/2', 'te-1/0/3']
slxos_netdev_lag { '10':
#ensure => absent,
#description => "port-channel 10",
minimum_links => 5,
lacp => active,
links => $lag_ports,
}
```

Use the **show port-channel** command to verify the results of the code segment on the switch.

```
device# show port-channel 10
LACP Aggregator: Po 10
Aggregator type: Standard
Ignore-split is enabled
Admin Key: 0010 - Oper Key 0010
Partner System ID - 0x0000,00-00-00-00-00-00
Partner Oper Key 0000
Link: eth 1/2 (0x10C004000) sync: 0
Link: eth 1/3 (0x10C006000) sync: 0
```

Manifest example

This is an example of a manifest that uses all four supported netdev types.

For ExtremeRouting SLX 9850 Router

```
root@ldap:/etc/puppetlabs/code/environments/production#cat manifests/site.pp
node tpvm-1, tpvm-2 {
   slxos_netdev_interface { "ethernet 4/6":
      ensure        => present,
      admin         => up,
      description   => "Ethernet 4/6",
      mtu           => 1548,
      target     => "http://admin:password@10.11.12.14:830",
   }
   slxos_netdev_interface { "ethernet 4/8":
      ensure        => present,
      admin         => up,
      description   => "Ethernet 4/8",
      mtu           => 1548,
      target     => "http://admin:password@10.11.12.14:830",
   }
   slxos_netdev_interface { "ethernet 4/9":
      ensure        => present,
      admin         => up,
      description   => "Ethernet 4/9",
      mtu           => 1548,
      target     => "http://admin:password@10.11.12.14:830",
   }

   slxos_netdev_vlan { "vlan 2":
      ensure        => present,
      vlan_id       => 2,
      description   => "Vlan 2",
      target     => "http://admin:password@10.11.12.14:830",
   }
   slxos_netdev_vlan { "vlan 3":
```

```
        ensure        => present,
        vlan_id       => 3,
        description   => "Vlan 3",
        target     => "http://admin:password@10.11.12.14:830",
    }
    slxos_netdev_vlan { "vlan 4":
        ensure        => present,
        vlan_id       => 4,
        description   => "Vlan 4",
        target     => "http://admin:password@10.11.12.14:830",
    }
    slxos_netdev_l2_interface { "ethernet 4/6":
        tagged_vlans  => ["2","3"],
        require       => Slxos_netdev_vlan["vlan 2", "vlan 3"],
        target     => "http://admin:password@10.11.12.14:830",
    }
    slxos_netdev_lag { "10":
        minimum_links => 5,
        lacp          => active,
        type          => standard,
        links         => ["ethernet 4/3", "ethernet 4/4"],
        target     => "http://admin:password@10.11.12.14:830",
    }
}
root@ldap:/etc/puppetlabs/code/environments/production#
```

For ExtremeRouting SLX 9140 Router

```
manifests/site.pp
node tpvm.englab.brocade.com {
    slxos_netdev_interface { "ethernet 0/7":
        ensure        => present,
        admin         => up,
        description   => "Ethernet 0/7",
        mtu           => 1548,
        speed         => "10000",
        target     => "http://admin:password@10.11.13.14:830",
    }
```

```
slxos_netdev_interface { "ethernet 0/8":
   ensure        => present,
   admin         => up,
   description   => "Ethernet 0/8",
   mtu           => 1548,
   speed         => "10000",
   target     => "http://admin:password@10.11.13.14:830",
}
slxos_netdev_interface { "ethernet 0/9":
   ensure        => present,
   admin         => up,
   description   => "Ethernet 0/9",
   mtu           => 1548,
   speed         => "10000",
   target     => "http://admin:password@10.11.13.14:830",
}
slxos_netdev_vlan { "vlan 2":
   ensure        => present,
   vlan_id       => 2,
   description   => "Vlan 2",
   target     => "http://admin:password@10.11.13.14:830",
}
slxos_netdev_vlan { "vlan 3":
   ensure        => present,
   vlan_id       => 3,
   description   => "Vlan 3",
   target     => "http://admin:password@10.11.13.14:830",
}
slxos_netdev_vlan { "vlan 4":
   ensure        => present,
   vlan_id       => 4,
   description   => "Vlan 4",
   target     => "http://admin:password@10.11.13.14:830",
}
slxos_netdev_l2_interface { "ethernet 0/7":
   tagged_vlans  => ["2","3"],
```

```
      require       => Slxos_netdev_vlan["vlan 2", "vlan 3"],
      target     => "http://admin:password@10.11.13.14:830",
   }
   slxos_netdev_lag { "10":
      minimum_links => 5,
      lacp          => active,
      type          => standard,
      links         => ["ethernet 0/3", "ethernet 0/4"],
      target     => "http://admin:password@10.11.13.14:830",
   }
}
root@ldap:/etc/puppetlabs/code/environments/production#
```