



Extreme SLX-OS Puppet User's Guide, 20.3.1

Supporting ExtremeRouting and ExtremeSwitching
SLX 9740, SLX 9640, SLX 9540, SLX 9250 and SLX
9150

9036965-00 Rev AA
March 2021



Copyright © 2021 Extreme Networks, Inc. All rights reserved.

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, 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: <https://www.extremenetworks.com/support/policies/open-source-declaration/>



Table of Contents

| | |
|-------------------------------------------------------------|-----------|
| Preface..... | 4 |
| Text Conventions..... | 4 |
| Documentation and Training..... | 5 |
| Getting Help..... | 6 |
| Subscribe to Product Announcements..... | 6 |
| Providing Feedback..... | 6 |
| About This Document..... | 8 |
| What's New in this Document..... | 8 |
| Supported Hardware..... | 8 |
| Extreme Puppet implementation..... | 9 |
| Extreme Puppet implementation..... | 9 |
| Software Requirement..... | 9 |
| How Puppet works..... | 9 |
| Limitation and restriction..... | 12 |
| Main steps for using Puppet..... | 12 |
| Installing the Extreme Provider..... | 12 |
| Site manifest..... | 13 |
| Puppet Documentation..... | 15 |
| Additional Information..... | 15 |
| Using the Extreme-supported Puppet netdev types..... | 16 |
| slxos_netdev_interface | 17 |
| slxos_netdev_vlan | 19 |
| slxos_netdev_l2_interface | 21 |
| slxos_netdev_lag | 23 |
| Manifest example..... | 25 |



Preface

This section describes the text conventions used in this document, where you can find additional information, and how you can provide feedback to us.

Text Conventions

Unless otherwise noted, information in this document applies to all supported environments for the products in question. Exceptions, like command keywords associated with a specific software version, are identified in the text.

When a feature, function, or operation pertains to a specific hardware product, the product name is used. When features, functions, and operations are the same across an entire product family, such as ExtremeSwitching switches or SLX routers, the product is referred to as *the switch* or *the router*.

Table 1: Notes and warnings






| Icon | Notice type | Alerts you to... |
|-------------------------------------------------------------------------------------|-------------|---------------------------------------------------------|
|  | Tip | Helpful tips and notices for using the product |
|  | Note | Useful information or instructions |
|  | Important | Important features or instructions |
|  | Caution | Risk of personal injury, system damage, or loss of data |
|  | Warning | Risk of severe personal injury |

Table 2: Text

| Convention | Description |
|----------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| screen displays | This typeface indicates command syntax, or represents information as it is displayed on the screen. |
| The words <i>enter</i> and <i>type</i> | When you see the word <i>enter</i> in this guide, you must type something, and then press the Return or Enter key. Do not press the Return or Enter key when an instruction simply says <i>type</i> . |
| Key names | Key names are written in boldface, for example Ctrl or Esc . If you must press two or more keys simultaneously, the key names are linked with a plus sign (+). Example: Press Ctrl+Alt+Del |
| Words in italicized type | Italics emphasize a point or denote new terms at the place where they are defined in the text. Italics are also used when referring to publication titles. |
| NEW! | New information. In a PDF, this is searchable text. |

Table 3: Command syntax

| Convention | Description |
|------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| bold text | Bold text indicates command names, keywords, and command options. |
| <i>italic</i> text | Italic text indicates variable content. |
| [] | 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, such as passwords, are enclosed in angle brackets. |
| ... | Repeat the previous element, for example, <i>member</i> [<i>member</i> ...]. |
| \ | In command examples, the backslash indicates a “soft” line break. When a backslash separates two lines of a command input, enter the entire command at the prompt without the backslash. |

Documentation and Training

Find Extreme Networks product information at the following locations:

[Current Product Documentation](#)

[Release Notes](#)

[Hardware and software compatibility](#) for Extreme Networks products

[Extreme Optics Compatibility](#)

[Other resources](#) such as white papers, data sheets, and case studies

Extreme Networks offers product training courses, both online and in person, as well as specialized certifications. For details, visit www.extremenetworks.com/education/.

Getting Help

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

Extreme Portal

Search the GTAC (Global Technical Assistance Center) knowledge base; manage support cases and service contracts; download software; and obtain product licensing, training, and certifications.

The Hub

A forum for Extreme Networks customers to connect with one another, answer questions, and share ideas and feedback. This community is monitored by Extreme Networks employees, but is not intended to replace specific guidance from GTAC.

Call GTAC

For immediate support: (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

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

- Your Extreme Networks service contract number, or serial numbers for all involved Extreme Networks products
- A description of the failure
- A description of any actions 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

Subscribe to Product Announcements

You can subscribe to email notifications for product and software release announcements, Field Notices, and Vulnerability Notices.

1. Go to [The Hub](#).
2. In the list of categories, expand the **Product Announcements** list.
3. Select a product for which you would like to receive notifications.
4. Select **Subscribe**.
5. To select additional products, return to the **Product Announcements** list and repeat steps 3 and 4.

You can modify your product selections or unsubscribe at any time.

Providing Feedback

The Information Development team at Extreme Networks has made every effort to ensure the accuracy and completeness of this document. We are always striving to improve our documentation and help you work better, so we want to hear from you. We welcome all feedback, but we especially want to know about:

- Content errors, or confusing or conflicting information.

- Improvements that would help you find relevant information in the document.
- Broken links or usability issues.

If you would like to provide feedback, you can do so in three ways:

- In a web browser, select the feedback icon and complete the online feedback form.
- Access the feedback form at <https://www.extremenetworks.com/documentation-feedback/>.
- Email us at documentation@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.



About This Document

[What's New in this Document](#) on page 8

[Supported Hardware](#) on page 8

What's New in this Document

No changes were made to this document for the SLX-OS 20.3.1 software release. For more information about the release, see the Extreme SLX-OS Release Notes.

Supported Hardware

For instances in which a topic or part of a topic applies to some devices but not to others, the topic specifically identifies the devices.

SLX-OS 20.3.1 supports the following hardware platforms.

- Devices based on the Broadcom XGS® chipset family:
 - ExtremeSwitching SLX 9250
 - ExtremeSwitching SLX 9150
- Devices based on the Broadcom DNX® chipset family:
 - ExtremeRouting SLX 9740
 - ExtremeRouting SLX 9640
 - ExtremeSwitching SLX 9540



Note

Although many software and hardware configurations are tested and supported for this release, documenting all possible configurations and scenarios is beyond this document's scope.

For information about other releases, see the documentation for those releases.



Extreme Puppet implementation

[Extreme Puppet implementation](#) on page 9

Extreme Puppet implementation

This document provides information on Puppet, a third-party virtual machine (TPVM) application that runs on SLX-OS platforms. 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 information and manages agent nodes requests for 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. 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, wherever 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.

We support a hybrid Puppet setup which 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 in the Extreme device.

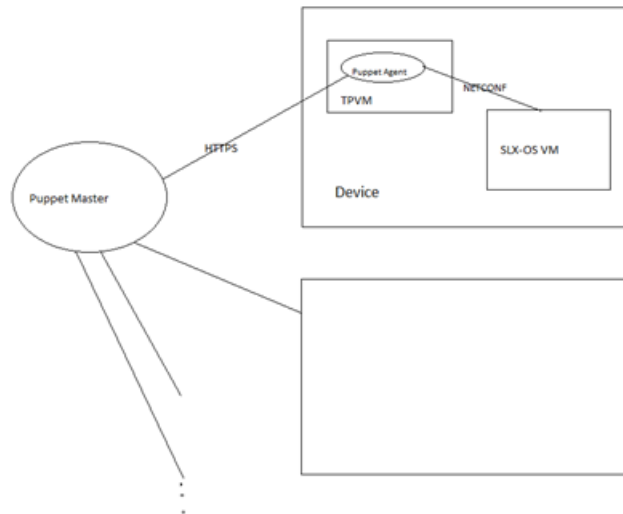


Figure 1: Extreme Puppet implementation

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.

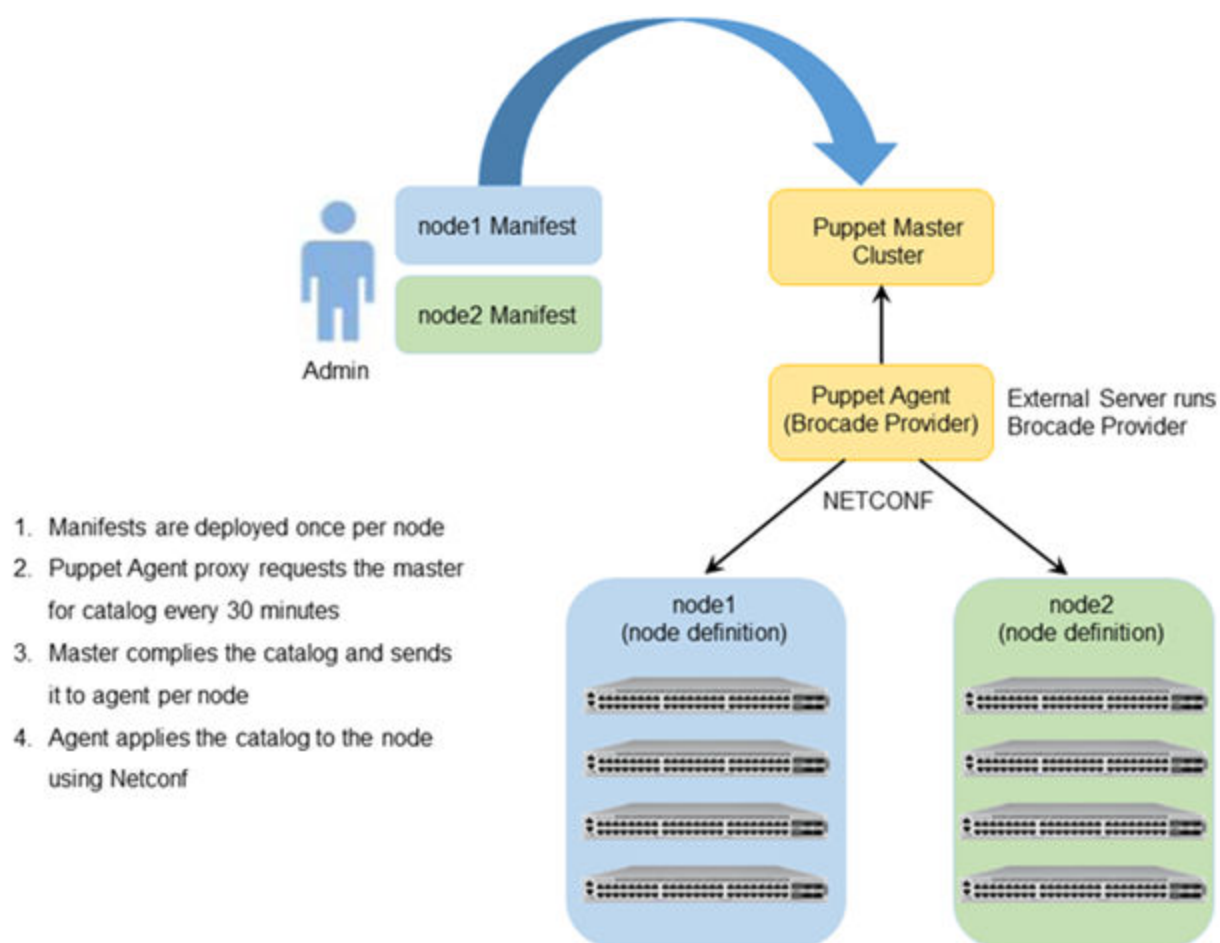


Figure 2: Extreme Puppet components

| 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 node cluster or clusters it is configuring. |

| Component | Role |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 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 node 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. |
| Catalog | A catalog is a compilation of all the resources that are 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 runs 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 node cluster or clusters that you want the 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 node 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 the Puppet module. This module supports basic interface, L2 port configuration, LAG and VLAN configurations. This module is derived from Netdev_stdlib with a modification required to configure the SLX-OS devices.

Follow these steps to install the Puppet Agent and Master:

1. Download the netdev types from <https://github.com/brocade/netdev-stdlib-slxos> and compress it to tar.gz format.
2. Install the Extreme Provider by running the following command on the server: **puppet module install netdev_stdlib_slxos-1.0.0.tar.gz**
3. Puppet Master is not packaged with the SLX-OS software. Download the Puppet master from <https://apt.puppetlabs.com/>. You must install the Puppet Master on a separate server.
4. 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 about how 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](#) on page 17

[slxos_netdev_vlan](#) on page 19

[slxos_netdev_l2_interface](#) on page 21

[slxos_netdev_lag](#) on page 23

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 4: 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, <code>http://admin:password@[3001::1]:830</code> NOTE: The target can also be specified by using a shortcut. An example is: <code>\$ip23= "http://admin:password@10.12.13.14:830"</code> ; you can then use <code>\$ip23</code> subsequently. |

Examples

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

```
class node1 {
  $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 ethernet 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 5: 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, <code>http://admin:password@[3001::1]:830</code> . NOTE: The target can also be specified by using a shortcut. An example is: <code>\$ip23= "http://admin:password@10.24.81.23:830"</code> ; you can then use <code>\$ip23</code> subsequently. |

Examples

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

```
class node1 {
  $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 6: 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 <code>enable</code> configures the port in trunk mode, in which tagged packets are processed. A value of <code>disable</code> (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 <code>tagged_vlans</code> 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 <code>vlan_tagging</code> 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, <code>http://admin:password@[3001::1]:830</code> NOTE: The target can also be specified by using a shortcut. An example is: <code>\$ip23= "http://admin:password@10.11.12.13:830"</code> ; you can then use <code>\$ip23</code> subsequently. |

Examples

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

```
class node1 {
  $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 ethernet 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 7: 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: <ul style="list-style-type: none"> 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 node1 {
  $ip23= "http://admin:password@10.11.12.13:830"
  $lag_ports = ['eth 0/2', 'eth 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.

```
root@ldap:/etc/puppetlabs/code/environments/production#cat manifests/site.pp

node tpvm-1, tpvm-2 {

  slxos_netdev_interface { "ethernet 0/4":

    ensure      => present,

    admin       => up,

    description => "Ethernet 0/4",

    mtu         => 1548,

    target      => "http://admin:password@10.11.12.14:830",

  }

  slxos_netdev_interface { "ethernet 0/4":

    ensure      => present,

    admin       => up,

    description => "Ethernet 0/4",

    mtu         => 1548,

    target      => "http://admin:password@10.11.12.14:830",

  }

  slxos_netdev_interface { "ethernet 0/4":

    ensure      => present,

    admin       => up,

    description => "Ethernet 0/4",

    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 0/4":
    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 0/4", "ethernet 0/4"],
    target        => "http://admin:password@10.11.12.14:830",
  }
}

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

```