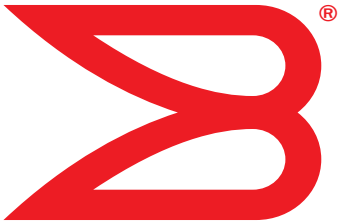
53-1003847-01
25 June 2015



Network OS

Puppet User's Guide

Supporting Network OS v6.0.1

BROCADE

Brocade Communications Systems, Incorporated

Corporate and Latin American Headquarters
Brocade Communications Systems, Inc.
130 Holger Way
San Jose, CA 95134
Tel: 1-408-333-8000
Fax: 1-408-333-8101
E-mail: info@brocade.com

Asia-Pacific Headquarters
Brocade Communications Systems China HK, Ltd.
No. 1 Guanghua Road
Chao Yang District
Units 2718 and 2818
Beijing 100020, China
Tel: +8610 6588 8888
Fax: +8610 6588 9999
E-mail: china-info@brocade.com

European Headquarters
Brocade Communications Switzerland Sàrl
Centre Swissair
Tour B - 4ème étage
29, Route de l'Aéroport
Case Postale 105
CH-1215 Genève 15
Switzerland
Tel: +41 22 799 5640
Fax: +41 22 799 5641
E-mail: emea-info@brocade.com

Asia-Pacific Headquarters
Brocade Communications Systems Co., Ltd. (Shenzhen WFOE)
Citic Plaza
No. 233 Tian He Road North
Unit 1308 – 13th Floor
Guangzhou, China
Tel: +8620 3891 2000
Fax: +8620 3891 2111
E-mail: china-info@brocade.com

Document History

Title	Publication number	Summary of changes	Date
Brocade Network OS Puppet User's Guide	53-1003847-01	New document	25 June 2015

Contents

About This Document

How this document is organized

This document is organized to help you find the information that you want as quickly and easily as possible.

The document contains the following components:

- Chapter 1, "Using the Brocade Puppet Implementation," provides an overview of the Brocade Puppet solution as well as instructions in implementing Puppet into your environment.
- Chapter 2, "Using the Brocade-supported Puppet netdev types," explains how to use Puppet netdev types to build a Puppet-language file to manage system resources.

Document conventions

This section describes text formatting conventions and important notice formats used in this document.

Text formatting

The narrative-text formatting conventions that are used are as follows:

bold text	Identifies command names Identifies the names of user-manipulated GUI elements Identifies keywords and operands Identifies text to enter at the GUI or CLI
italic text	Provides emphasis Identifies variables Identifies paths and Internet addresses Identifies document titles
`code` text	Identifies CLI output Identifies command syntax examples

For readability, command names in the narrative portions of this guide are presented in mixed lettercase: for example, **switchShow**. In actual examples, command lettercase is all lowercase.

Command syntax conventions

Command syntax in this manual follows these conventions:

Convention	Description
[ ]	Keywords or arguments that appear within square brackets are optional. For example: **command** [**active** | **standby** | **disabled**] = One (and only one) of this set of keywords may be used. **command** [**active**] [**standby**] [**disabled**] = Three independent options, and one or more may be used on the same command line.
{x | y | z}	A choice of required keywords appears in braces separated by vertical bars. You must select one. For example: **command** {**active** | **standby** | **disabled**} = One (and only one) of this set of keywords must be used.
`screen font`	Examples of information displayed on the screen.
< >	Nonprinting characters, for example, passwords, appear in angle brackets.
[ ]	Default responses to system prompts appear in square brackets.
italic text	Identifies variables.
bold text	Identifies literal command options and keywords.

NOTE
In standalone mode, interfaces are identified using *slot/port* notation. In Brocade VCS technology® mode, interfaces are identified using *switch/slot/port* notation.

Nesting square brackets and curly braces

When reading a command entry, optional keywords are surrounded by square brackets and mandatory keywords are surrounded by curly braces. Refer to "Command syntax conventions" on page 4 for complete details.

In some cases, these brackets can be nested. In the following example, **rbridge-id** is optional as denoted by the square brackets, but if you use it, then you must follow it with either a specific *rbridge-id* or the word "all."

command [**rbridge-id** {*rbridge-id* | **all**}]

However, square brackets can appear within curly braces, showing that while a keyword is mandatory, supporting operands may be optional, as shown in the following example:

command {**security** [**active**] [**standby**] [**disabled**]}

command {**security** [**active** | **standby** | **disabled**]}

Notes, cautions, and warnings

The following notices and statements are used in this manual. They are listed below in 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 potential damage to hardware or data.

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.

Key terms

For definitions specific to Brocade and Fibre Channel, see the technical glossaries on MyBrocade. Refer to "Brocade resources" on page 6 for instructions on accessing MyBrocade.

For definitions of SAN-specific terms, visit the Storage Networking Industry Association online dictionary at:

http://www.snia.org/education/dictionary

Notice to the reader

This document may contain references to the trademarks of the following corporations. These trademarks are the properties of their respective companies and corporations.

These references are made for informational purposes only.

Corporation	Referenced Trademarks and Products
Microsoft Corporation	Windows, Windows NT, Internet Explorer
Oracle Corporation	Oracle, Java
Netscape Communications Corporation	Netscape
Red Hat, Inc.	Red Hat, Red Hat Network, Maximum RPM, Linux Undercover

Additional information

This section lists additional Brocade and industry-specific documentation that you might find helpful.

Brocade resources

To get up-to-the-minute information, go to http://my.brocade.com to register at no cost for a user ID and password.

White papers, online demonstrations, and data sheets are available through the Brocade website at:

http://www.brocade.com/products-solutions/products/index.page

For additional Brocade documentation, visit the Brocade website:

http://www.brocade.com

Release notes are available on the MyBrocade website.

Other industry resources

For additional resource information, visit the Technical Committee T11 website. This website provides interface standards for high-performance and mass storage applications for Fibre Channel, storage management, and other applications:

http://www.t11.org

For information about the Fibre Channel industry, visit the Fibre Channel Industry Association website:

http://www.fibrechannel.org

Getting technical help

Contact your switch support supplier for hardware, firmware, and software support, including product repairs and part ordering. To expedite your call, have the following information available:

1. General Information
    - Switch model
    - Switch operating system version
    - Software name and software version, if applicable
    - Error numbers and messages received
    - Detailed description of the problem, including the switch or behavior immediately following the problem, and specific questions
    - Description of any troubleshooting steps already performed and the results
    - Serial console and Telnet session logs
    - syslog message logs
2. Switch Serial Number

    The switch serial number and corresponding bar code are provided on the serial number label, as illustrated below:

    The serial number label is located on the switch ID pull-out tab located on the bottom of the port side of the switch.
3. World Wide Name (WWN)

    Use the **show license id** command to display the WWN of the chassis.

    If you cannot use the **show license id** command because the switch is inoperable, you can get the WWN from the same place as the serial number, except for the Brocade DCX. For the Brocade DCX, access the numbers on the WWN cards by removing the Brocade logo plate at the top of the nonport side of the chassis.

Document feedback

Quality is our first concern at Brocade 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. Forward your feedback to:

documentation@brocade.com

Provide the title and version number of the document and as much detail as possible about your comment, including the topic heading and page number and your suggestions for improvement.

Chapter

1 Using the Brocade Puppet Implementation

In this chapter

What is Puppet

Puppet is a scripting language available from Puppet Labs that system administrators can use to automate configuration and management of a data center. Puppet allows you to:

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

Software Requirements

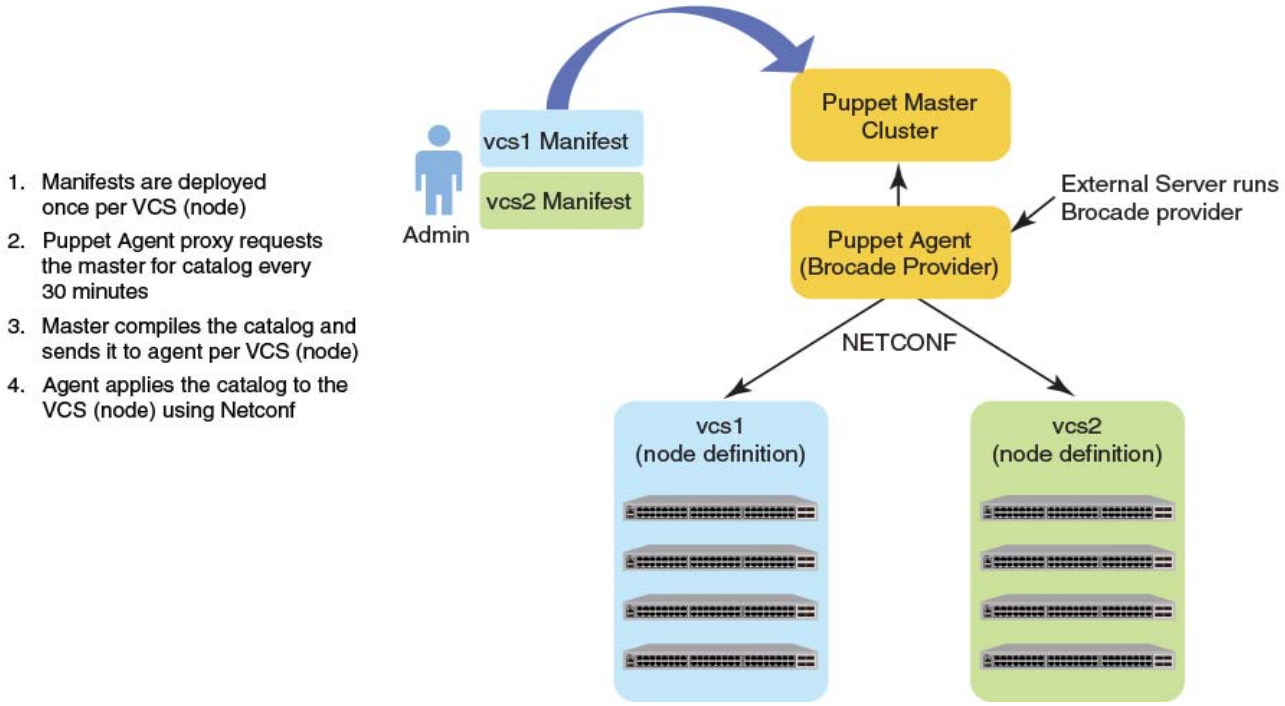
The Brocade Puppet solution requires the following software:

- Network OS 6.0.1 or later
- Puppet software (obtained from www.puppetlabs.com):
  - Puppet Enterprise 3.7 +
  - Puppet Community Edition
  - Netdev 0.9.0

Puppet Overview

The following illustration depicts how the major Puppet components interact.

FIGURE 1 Brocade Puppet components



The following table describes the main components that are shown in the previous figure.

TABLE 1 Puppet components and descriptions of roles

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* (see below in this table).
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 Brocade VCS cluster or clusters that it will configure. **NOTE:** Puppet Agents do not run on Brocade VDX switches.
Brocade Provider	Brocade Providers implement resource types on a specific type of system, using the system's own tools. The Brocade Provider executes the manifests for the purpose of configuring the associated VCS cluster(s) and switches. The Brocade Provider uses Netconf calls to configure the switch or switches. These Netconf calls are transparent to the Puppet administrator and user. Typically, 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 will be applied to a given system and the relationships between those resources.

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

Main steps for Using Puppet

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

1. Determine where you want the Puppet master to reside. This can be a server in any location.
2. Determine which node or nodes you want to be Agent nodes. The Agent nodes must have connectivity to both the Puppet Master and the Brocade VCS cluster or clusters that you want an Agent node to configure.
3. Install the Brocade Provider on the Puppet Master. Refer to the "Installing the Brocade Provider" on page 5.
4. Set up the site manifest file. Refer to the "Site Manifest" on page 6.
5. Write one manifest for each VCS cluster that you want to manage by means of Puppet. Refer to Using the Brocade-supported Puppet netdev types 9.
6. Place the manifest(s) on the Master Puppet server.

Installing the Brocade Provider

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

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

   puppet module install netdev_stdlib_nos

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.

```
Node "puppet.englab.brocade.com" {
   # Maps to $modulepath/netdev_stdlib_nos/manifests/datacenter10.pp
   include netdev_stdlib_nos::datacenter10
   # Maps to $modulepath/netdev_stdlib_nos/manifests/datacenter20.pp
   include netdev_stdlib_nos::datacenter20
}
node "web1.englab.brocade.com" {
   # Maps to $modulepath/netdev_stdlib_nos/manifests/datacenter30.pp
   include netdev_stdlib_nos::datacenter30
   # Maps to $modulepath/netdev_stdlib_nos/manifests/datacenter40.pp
   include netdev_stdlib_nos::datacenter40
}
```

An example of a manifest file for datacenter10 is show below. The manifest file for an Agent provides a blueprint of the data center.

```
class netdev_stdlib_nos::datacenter10 {
$vlan10= "Green"
$if2501= "te-25/0/1"
$if2601= "te-26/0/1"
$ip= "http://admin:password@10.24.81.26:830"
```

Manually Running the Script

If you want to run the manifest manually at any time, instead of waiting for the default 30-minute interval when the Agent sends a proxy request to the Puppet Master, you can run the **puppet apply** command, as shown in the following example where the name of the manifest file is *vcs1.pp*:

```
[root@puppet tests]# puppet apply vcs1.pp
Notice: Compiled catalog for pupfvt.englab.brocade.com in environment production
in 0.40 seconds
Notice: Nos_netdev_device[sw1](provider=nos): nos_netdev_device: CREATE sw1
Notice: /Stage[main]/Vcs1/Nos_netdev_device[sw1]/ensure: created
Notice: Nos_netdev_device[sw2](provider=nos): nos_netdev_device: CREATE sw2
Notice: /Stage[main]/Vcs1/Nos_netdev_device[sw2]/ensure: created
Notice: Nos_netdev_interface[1/0/1](provider=nos): nos_netdev_interface: CREATE
1/0/1
Notice: /Stage[main]/Vcs1/Nos_netdev_vlan[v10]/description: description changed
'' to 'Vlan 10'
```

```
Notice: /Stage[main]/Vcs1/Nos_netdev_vlan[v10]/vlan_id: vlan_id changed '' to
'10'
Notice: /Stage[main]/Vcs1/Nos_netdev_vlan[v10]/target: target changed '' to
'http://admin:password@10.24.81.23:830'
Notice: Nos_netdev_l2_interface[te-2/0/1](provider=nos): nos_netdev_l2_interface:
CREATE te-2/0/1
Notice: /Stage[main]/Vcs1/Nos_netdev_l2_interface[te-2/0/1]/ensure: created
Notice: Nos_netdev_l2_interface[te-2/0/2](provider=nos): nos_netdev_l2_interface:
CREATE te-2/0/2
Notice: /Stage[main]/Vcs1/Nos_netdev_l2_interface[te-2/0/2]/ensure: created
Notice: Nos_netdev_l2_interface[te-2/0/3](provider=nos): nos_netdev_l2_interface:
CREATE te-2/0/3
Notice: /Stage[main]/Vcs1/Nos_netdev_l2_interface[te-2/0/3]/ensure: created
Notice: /Stage[main]/Vcs1/Nos_netdev_lag[10]/ensure: created
Notice: Finished catalog run in 9.21 seconds
```

Additional Information

If you are using Puppet to manage resources, the properties that you configure with Puppet should not be changed by using the command line interface. 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 *runinterval* property of the main configuration file. For more information, refer to https://docs.puppetlabs.com/puppet/latest/reference/config_file_main.html.

NOTE
All events from the Brocade Provider are logged at: /var/log/syslog

1 Additional Information

Chapter

Using the Brocade-supported Puppet netdev types 2

In this chapter

netdev types

The Brocade Puppet solution supports the following five netdev types:

- "nos_netdev_device"
- "nos_netdev_interface"
- "nos_netdev_VLAN"
- "nos_netdev_l2_interface"
- "nos_netdev_LAG"

nos_netdev_device

Syntax

```
nos_netdev_device {"name":
                vcs_id          => vcs_id_value,
                rbridge_id      => rbridge_id_value,
                target          => target
        }
```

Description

You can use the netdev_device type to configure the VCS and RBridge IDs of a switch.

Properties

TABLE 2 netdev_device properties

Property	Description
name	Specifies the name Puppet uses to identify the device resource.
vcs_id_value (required	Configures the VCS ID of the switch.
rbridge_id_value (required	Configures the RBridge ID of the switch.
target (required	Specifies the IP address, username and password of the switch or Rbridge; 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.`

Example

The following Puppet code segments show an example of using the netdev_device type to configure the VCS and RBridge ID of a switch:

```
class vcs1 {
$ip23= "http://admin:password@10.24.81.23:830"
$ip24= "http://admin:password@10.24.81.24:830"


nos_netdev_device {"name":
                vcs_id          => 25,
                rbridge_id      => 1,
                target          => $ip23
        }
nos_netdev_device {"sw2":

                  vcs_id => 25,
                  rbridge_id => 2,
                  target => $ip24
```

To verify the results of the code segment on the switch, you can run the **sh vcs** command:

```
device# sh vcs
Config Mode : Distributed
VCS Mode : Logical Chassis
```

```
VCS ID : 25
VCS GUID : 2de5478c-46ca-44af-a841-edf1b3ae60ee
Total Number of Nodes : 2
Rbridge-Id WWN Management IP VCS Status Fabric Status
HostName
-------------------------------------------------------------------------------
-------------------------
1 >10:00:00:27:F8:9B:9D:12* 10.24.81.23 Online Online sw0
2 10:00:00:27:F8:D0:EC:47 10.24.81.24 Online Online sw0
```

nos_netdev_interface

Syntax

```
nos_netdev_interface { "name":
       ensure          => [present | absent],
       admin           => [up | down],
       description     => "interface description",
       mtu             => mtu_value,
       speed           => "speed_value",
       target          => $target
                     }
```

Description

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

Properties

TABLE 3 nos_netdev_interface properties

Property	Description
name (required)	Specifies the name of the physical interface. An example of a 10-Gb ethernet interface would be “te-1/0/1”.
ensure (optional)	Specifies whether to create or delete the configuration. A value of *present* (the default) creates the configuration. A value of *absent* is ignored.
admin (optional)	Configures the interface as administratively enabled or disabled. A value of *up* (the default) enables the interface, and a value of *down* disables the interface.
interface description (optional)	Assigns the description value to the interface. The default is: “Puppet created interface: <name>”.
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.
target (optional)	Specifies the IP address, username and password of the switch or Rbridge; 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.`

Example

The following Puppet code segment configures several properties for the 10-Gb ethernet interface te-1/0/1:

```
class vcs1 {
$ip23= "http://admin:password@10.24.81.23:830"


nos_netdev_interface { “te-1/0/1":
    ensure => present,
    admin => up,
    description => "this is a storage port",
    mtu => 2200,
    speed => "10000",
    target => $ip23
}
```

To verify the results of the code segment on the switch, you can run the **sh run int** command, as shown below:

```
device# sh run int te 1/0/1
interface TenGigabitEthernet 1/0/1
speed 10000
mtu 2200
description this is a storage port
fabric isl enable
fabric trunk enable
no shutdown
!
```

nos_netdev_VLAN

Syntax

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

}
```

Description

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

Properties

TABLE 4 nos_netdev_VLAN properties

Property	Description
name (required)	Specifies the name of the VLAN; for example “Blue”.
VLAN_id (required)	Sets the VLAN tag-ID value. The range is 1 to 4095.
vlan-description (optional)	Assigns the description value to the VLAN. The default is: “Puppet created VLAN: <name>: <vlan-id>”
target (required)	Specifies the IP address, username and password of the switch or Rbridge; 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.`

NOTE
VLANs are assigned to ports using the netdev_l2_interface type, described later in this chapter.

Example

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

```
class vcs1 {
$ip23= "http://admin:password@10.24.81.23:830"


$vlans= {
    'v10' => {ensure => present, vlan_id =>10, description => 'Vlan 10', target
=>$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(nos_netdev_vlan, $vlans)}
```

To verify the results of the code segment on the switch, you can run the **sh run int vlan** command, as shown below:

```
device# sh run int vlan
interface Vlan 1
!
interface Vlan 10
  description Vlan 10
!
interface Vlan 11
  description Vlan 11
!
interface Vlan 12
```

```
  description Vlan 12
!
interface Vlan 13
  description Vlan 13
!
interface Vlan 14
```

nos_netdev_l2_interface

Syntax

```
netdev_l2_interface { "name":
    ensure => [present | absent],
    vlan_tagging => [enable | disable],
    tagged_vlans => (vlan | [vlan1, vlan2, vlan3, ...]),
    untagged_vlan => vlan,
    native_vlans => vlan,
    require => Nos_netdev_vlan[v10,v11,v12,v13,v14,v15,v16]
    target => $target
}
```

Description

The nos_netdev_l2_interface type allows you to manage assignment of VLANS to ports on a switch.

Properties

The following table provides property descriptions for the nos_netdev_l2_interface type.

TABLE 5 nos_netdev_l2_interface properties

Property	Description
name (required)	Specifies the name of the interface; for example “gi-0/0/0”. **NOTE:** The name does not include the unit number.
ensure (optional)	Specifies whether to create or delete the configuration. A value of *present* (the default) creates the configuration. A value of *absent* deletes the configuration.
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.
tagged_vlans (optional)	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, then this VLAN is the native VLAN.
native_vlan (optional)	Specifies VLAN IDs for native VLAN.
require (optional)	When specified, ensures that the required resource are created first. **Example**: require => Nos_netdev_vlan[v10,v11,v12] This ensures that vlans with names v10, v11, v12 are created before running this netdev type.
target (required)	Specifies the IP address, username and password of the switch or Rbridge; 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.`

Example

The following Puppet code segment shows an example of configuring tagged, untagged, and native VLAN tagging:

```
class vcs1 {
$ip23= "http://admin:password@10.24.81.23:830"$l2_ifs= {
'te-2/0/1' => {tagged_vlans =>[10,11,12], require =>
Nos_netdev_vlan[v10,v11,v12], target =>
$ip23},
'te-2/0/2' => {untagged_vlan => 13, require => Nos_netdev_vlan[v13], target =>
$ip23},
'te-2/0/3' => {native_vlan => 14, require => Nos_netdev_vlan[v14], target =>
$ip23}
}create_resources(nos_netdev_l2_interface, $l2_ifs)
}
```

To verify the results of the code segment on the switch, you can run the following **sh run int** commands on your switch for each physical interface:

```
device# sh run int te 2/0/1
interface TenGigabitEthernet 2/0/1
fabric isl enable
fabric trunk enable
switchport
switchport mode trunk
switchport trunk allowed vlan add 10-12
switchport trunk tag native-vlan
spanning-tree shutdown
no shutdown
!

device# sh run int te 2/0/2
interface TenGigabitEthernet 2/0/2
fabric isl enable
fabric trunk enable
switchport
switchport mode access
switchport access vlan 13
spanning-tree shutdown
no shutdown
!

device# sh run int te 2/0/3
interface TenGigabitEthernet 2/0/3
fabric isl enable
fabric trunk enable
switchport
switchport mode trunk
switchport trunk tag native-vlan
switchport trunk native-vlan 14
spanning-tree shutdown
no shutdown
!
```

nos_netdev_LAG

Syntax

```
nos_netdev_lag { 'name':
    ensure => [present | absent],
    description => "vlag-description”,
    minimum_links => minimum_links_value,
    admin => [up | down],
    lacp => [active | passive | on],
    type => [brocade | standard],
    links => $lag_ports,
    target=> "http://admin:password@10.24.81.23:830"
}
```

Description

The nos_netdev_LAG type allows you to manage link aggregation groups.

Properties

TABLE 6 nos_netdev_LAG properties

Property	Description
name (required)	Specifies the name of the LAG; for example 10.
ensure (optional)	Specifies whether to create or delete the configuration. A value of *present* (the default) creates the configuration. A value of *absent* deletes the LAG.
vlag-description (optional)	Assigns the description value to the VLAN.
minimum_links_value (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 not set and there is no minimum-link requirement.
admin	Sets the port-channel admin state: • Up (default) = No shut. • Down = Shut.
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
type (required)	• Brocade • Standard (default)
links (required)	Specifies a list of physical interfaces that compose the LAG bundle.
target (required)	Specifies the IP address, username and password of the switch or Rbridge; 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.`

Example

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

```
class vcs1 {
$ip23= "http://admin:password@10.24.81.23:830"
$lag_ports = ['te-1/0/2', 'te-1/0/3']


nos_netdev_lag { '10':
    #ensure => absent,
    #description => "port-channel 10",
    minimum_links => 5,
    #admin => down,
    lacp => active,
    type => standard,
    links => $lag_ports,
    target=> "http://admin:password@10.24.81.23:830"
}
```

To verify the results of the code segment on the switch, you can run the **show port-channel** command:

```
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
Member ports on rbridge-id 1:
Link: Te 1/0/2 (0x10C004000) sync: 0
Link: Te 1/0/3 (0x10C006000) sync: 0
```

Manifest example

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

```
# Configure VCS and Rbridge Id
        nos_netdev_device {"sw1":
                vcs_id          => 25,
                rbridge_id      => 1,
                target          => $ip23
        }

        nos_netdev_device {"sw2":
                vcs_id          => 25,
                rbridge_id      => 2,
                target          => $ip24
        }

# Configure Physical Interface
        nos_netdev_interface { "1/0/1":
                ensure          => present,
                admin           => up,
                description     => "this is a storage port",
                mtu             => 2200,
                speed           => "10000",
                target          => $ip23
        }

# Configure VLANs
```

```
        $vlans= {
                'v10'         => {ensure => present, vlan_id =>10, description =>
'Vlan 10', target =>$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 =>$ip23},
                'v14'         => {ensure => present, vlan_id =>14, description =>
'Vlan 14', target =>$ip23},
        }

   create_resources(nos_netdev_vlan, $vlans)

# Configure L2 Interfaces
        $l2_ifs= {
                'te-2/0/1'          => {tagged_vlans => [ 10, 11, 12], require =>
Nos_netdev_vlan[v10,v11,v12], target =>
$ip23},
                 'te-2/0/2'          => {untagged_vlan => 13, require =>
Nos_netdev_vlan[v13],  target => $ip23},
                 'te-2/0/3'          => {native_vlan => 14, require =>
Nos_netdev_vlan[v14], target => $ip23},
        }
        }

   create_resources(nos_netdev_l2_interface, $l2_ifs)

# Configure LAG ports
   $lag_ports = ['te-1/0/2', 'te-1/0/3']

  nos_netdev_lag { '10':
     #ensure => absent,
     #description => "port-channel 10",
     minimum_links => 5,
     #admin => down,
     lacp => active,
     type => standard,
     links => $lag_ports,
     target      => "http://admin:password@10.24.81.23:830"
  }

}

node "puppet.englab.brocade.com" {
   include vcs1
}
```

2 Manifest example