



Junos[®] OS

Chef for Junos OS Getting Started Guide

Release
11.10



Modified: 2016-06-27

Juniper Networks, Inc.
1133 Innovation Way
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

Copyright © 2016, Juniper Networks, Inc. All rights reserved.

Juniper Networks, Junos, Steel-Belted Radius, NetScreen, and ScreenOS are registered trademarks of Juniper Networks, Inc. in the United States and other countries. The Juniper Networks Logo, the Junos logo, and JunosE are trademarks of Juniper Networks, Inc. All other trademarks, service marks, registered trademarks, or registered service marks are the property of their respective owners.

Juniper Networks assumes no responsibility for any inaccuracies in this document. Juniper Networks reserves the right to change, modify, transfer, or otherwise revise this publication without notice.

Junos® OS Chef for Junos OS Getting Started Guide

11.10

Copyright © 2016, Juniper Networks, Inc.

All rights reserved.

The information in this document is current as of the date on the title page.

YEAR 2000 NOTICE

Juniper Networks hardware and software products are Year 2000 compliant. Junos OS has no known time-related limitations through the year 2038. However, the NTP application is known to have some difficulty in the year 2036.

END USER LICENSE AGREEMENT

The Juniper Networks product that is the subject of this technical documentation consists of (or is intended for use with) Juniper Networks software. Use of such software is subject to the terms and conditions of the End User License Agreement ("EULA") posted at <http://www.juniper.net/support/eula.html>. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

Table of Contents

	About the Documentation	ix
	Documentation and Release Notes	ix
	Documentation Conventions	ix
	Documentation Feedback	xi
	Requesting Technical Support	xii
	Self-Help Online Tools and Resources	xii
	Opening a Case with JTAC	xii
Part 1	Overview	
Chapter 1	Introduction	3
	Chef for Junos OS Overview	3
	Understanding Cookbooks, Recipes, Resources, and Providers	3
	Chef for Junos OS Features	4
	Providers for the netdev Cookbook Resources	4
	Native Chef Client	5
	Native Ohai	5
	Ruby Interpreter and junos-ez-stdlib	5
	Components of a Chef for Junos OS Deployment	5
	Deploying Chef for Junos OS Overview	7
	Running the Chef Client	7
Part 2	Installation	
Chapter 2	Installing the Chef Client	11
	Installing or Removing the Chef Client on Juniper Networks Devices Running Junos OS	11
	Installing or Upgrading the Chef Client	11
	Removing the Chef Client from the Juniper Networks Device	13
Part 3	Configuration	
Chapter 3	Client Configuration	17
	Configuring the Chef Client on Juniper Networks Devices Running Junos OS	17
Chapter 4	Example	19
	Example: Using Chef for Junos OS to Configure Ethernet Switching on EX Series, OCX Series, and QFX Series Switches	19
	Example: Using Chef for Junos OS to Configure Ethernet Switching on MX Series Routers	28
	Example: Using Chef for Junos OS to Configure Any Hierarchy Level	37

Part 4

Index

Index	45
-------------	----

List of Figures

Part 1	Overview	
Chapter 1	Introduction	3
	Figure 1: Major Components of a Chef for Junos OS Deployment	6

List of Tables

	About the Documentation	ix
	Table 1: Notice Icons	x
	Table 2: Text and Syntax Conventions	x
Part 3	Configuration	
Chapter 4	Example	19
	Table 3: VLANs Defined in the vlan_create Recipe	20
	Table 4: Access Interfaces Defined in the access_interface_create Recipe	20
	Table 5: LAGs Defined in the uplink_interface_create Recipe	20
	Table 6: VLANs Defined in the vlan_create Recipe	29
	Table 7: Interfaces Defined in the interface_create and l2interface_create Recipes	29
	Table 8: LAGs Defined in the lag_interface_create Recipe	29

About the Documentation

- Documentation and Release Notes on page ix
- Documentation Conventions on page ix
- Documentation Feedback on page xi
- Requesting Technical Support on page xii

Documentation and Release Notes

To obtain the most current version of all Juniper Networks® technical documentation, see the product documentation page on the Juniper Networks website at <http://www.juniper.net/techpubs/>.

If the information in the latest release notes differs from the information in the documentation, follow the product Release Notes.

Juniper Networks Books publishes books by Juniper Networks engineers and subject matter experts. These books go beyond the technical documentation to explore the nuances of network architecture, deployment, and administration. The current list can be viewed at <http://www.juniper.net/books>.

Documentation Conventions

Table 1 on page x defines notice icons used in this guide.

Table 1: Notice Icons







Icon	Meaning	Description
	Informational note	Indicates important features or instructions.
	Caution	Indicates a situation that might result in loss of data or hardware damage.
	Warning	Alerts you to the risk of personal injury or death.
	Laser warning	Alerts you to the risk of personal injury from a laser.
	Tip	Indicates helpful information.
	Best practice	Alerts you to a recommended use or implementation.

Table 2 on page x defines the text and syntax conventions used in this guide.

Table 2: Text and Syntax Conventions

Convention	Description	Examples
Bold text like this	Represents text that you type.	To enter configuration mode, type the configure command: user@host> configure
<code>Fixed-width text like this</code>	Represents output that appears on the terminal screen.	user@host> show chassis alarms No alarms currently active
<i>Italic text like this</i>	<ul style="list-style-type: none"> Introduces or emphasizes important new terms. Identifies guide names. Identifies RFC and Internet draft titles. 	<ul style="list-style-type: none"> A policy <i>term</i> is a named structure that defines match conditions and actions. <i>Junos OS CLI User Guide</i> RFC 1997, <i>BGP Communities Attribute</i>
<i>Italic text like this</i>	Represents variables (options for which you substitute a value) in commands or configuration statements.	Configure the machine's domain name: [edit] root@# set system domain-name <i>domain-name</i>

Table 2: Text and Syntax Conventions (*continued*)

Convention	Description	Examples
Text like this	Represents names of configuration statements, commands, files, and directories; configuration hierarchy levels; or labels on routing platform components.	<ul style="list-style-type: none">To configure a stub area, include the stub statement at the [edit protocols ospf area area-id] hierarchy level.The console port is labeled CONSOLE.
< > (angle brackets)	Encloses optional keywords or variables.	stub <default-metric <i>metric</i>>;
(pipe symbol)	Indicates a choice between the mutually exclusive keywords or variables on either side of the symbol. The set of choices is often enclosed in parentheses for clarity.	broadcast multicast (<i>string1</i> <i>string2</i> <i>string3</i>)
# (pound sign)	Indicates a comment specified on the same line as the configuration statement to which it applies.	rsvp { # Required for dynamic MPLS only
[] (square brackets)	Encloses a variable for which you can substitute one or more values.	community name members [<i>community-ids</i>]
Indentation and braces ({ })	Identifies a level in the configuration hierarchy.	<pre>[edit] routing-options { static { route default { nexthop <i>address</i>; retain; } } }</pre>
;(semicolon)	Identifies a leaf statement at a configuration hierarchy level.	
GUI Conventions		
Bold text like this	Represents graphical user interface (GUI) items you click or select.	<ul style="list-style-type: none">In the Logical Interfaces box, select All Interfaces.To cancel the configuration, click Cancel.
> (bold right angle bracket)	Separates levels in a hierarchy of menu selections.	In the configuration editor hierarchy, select Protocols>Ospf .

Documentation Feedback

We encourage you to provide feedback, comments, and suggestions so that we can improve the documentation. You can provide feedback by using either of the following methods:

- Online feedback rating system—On any page of the Juniper Networks TechLibrary site at <http://www.juniper.net/techpubs/index.html>, simply click the stars to rate the content, and use the pop-up form to provide us with information about your experience. Alternately, you can use the online feedback form at <http://www.juniper.net/techpubs/feedback/>.

- E-mail—Send your comments to techpubs-comments@juniper.net. Include the document or topic name, URL or page number, and software version (if applicable).

Requesting Technical Support

Technical product support is available through the Juniper Networks Technical Assistance Center (JTAC). If you are a customer with an active J-Care or Partner Support Service support contract, or are covered under warranty, and need post-sales technical support, you can access our tools and resources online or open a case with JTAC.

- JTAC policies—For a complete understanding of our JTAC procedures and policies, review the *JTAC User Guide* located at <http://www.juniper.net/us/en/local/pdf/resource-guides/7100059-en.pdf>.
- Product warranties—For product warranty information, visit <http://www.juniper.net/support/warranty/>.
- JTAC hours of operation—The JTAC centers have resources available 24 hours a day, 7 days a week, 365 days a year.

Self-Help Online Tools and Resources

For quick and easy problem resolution, Juniper Networks has designed an online self-service portal called the Customer Support Center (CSC) that provides you with the following features:

- Find CSC offerings: <http://www.juniper.net/customers/support/>
- Search for known bugs: <http://www2.juniper.net/kb/>
- Find product documentation: <http://www.juniper.net/techpubs/>
- Find solutions and answer questions using our Knowledge Base: <http://kb.juniper.net/>
- Download the latest versions of software and review release notes: <http://www.juniper.net/customers/csc/software/>
- Search technical bulletins for relevant hardware and software notifications: <http://kb.juniper.net/InfoCenter/>
- Join and participate in the Juniper Networks Community Forum: <http://www.juniper.net/company/communities/>
- Open a case online in the CSC Case Management tool: <http://www.juniper.net/cm/>

To verify service entitlement by product serial number, use our Serial Number Entitlement (SNE) Tool: <https://tools.juniper.net/SerialNumberEntitlementSearch/>

Opening a Case with JTAC

You can open a case with JTAC on the Web or by telephone.

- Use the Case Management tool in the CSC at <http://www.juniper.net/cm/>.
- Call 1-888-314-JTAC (1-888-314-5822 toll-free in the USA, Canada, and Mexico).

For international or direct-dial options in countries without toll-free numbers, see <http://www.juniper.net/support/requesting-support.html>.

PART 1

Overview

- [Introduction on page 3](#)

CHAPTER 1

Introduction

- [Chef for Junos OS Overview on page 3](#)

Chef for Junos OS Overview

Chef software automates the provisioning and management of compute, networking, and storage resources, whether these resources are on-site, in the cloud, or both. Chef software transforms infrastructure into code, enabling you to configure, deploy, and scale in real time, while reducing the risk of human error. See *Chef for Junos OS* at <https://docs.chef.io/junos.html>.

Using Chef, you write abstract definitions of your infrastructure in Ruby and manage the definitions like you manage source code. These abstract definitions are applied to the nodes in your infrastructure by the Chef clients running on those nodes. When you bring a new node online, the Chef client running on that node needs only to determine which definitions to apply.

Chef for Junos OS enables Chef support on selected Juniper Networks devices. You can use Chef for Junos OS to automate common switching network configurations, such as physical and logical Ethernet link properties and VLANs, on these devices. See the [Chef for Junos OS Release Notes](#) for information about which Juniper Network devices support Chef clients.

- [Understanding Cookbooks, Recipes, Resources, and Providers on page 3](#)
- [Chef for Junos OS Features on page 4](#)
- [Components of a Chef for Junos OS Deployment on page 5](#)
- [Deploying Chef for Junos OS Overview on page 7](#)
- [Running the Chef Client on page 7](#)

Understanding Cookbooks, Recipes, Resources, and Providers

Within the Chef framework, the abstract infrastructure definitions are contained in reusable cookbooks and recipes:

- *Cookbooks* are packages that contain the *recipes*, files, attribute definitions, and so on that describe a portion of your infrastructure and how to deploy, configure, and manage it. For example, the `apache2` cookbook maintained by Chef contains recipes for installing and configuring an Apache HTTP Server.
- *Recipes* are written in Ruby and describe the installation, configuration, and management of the infrastructure elements.
- *Resources* are the major building blocks of recipes. A resource is a platform-neutral representation of an element of the system and its desired state—for example, a service that should be started or a file that should be written.
- *Providers* are the underlying platform-specific implementations that bring resources to their desired states. For example, a resource might specify a particular software package to be installed, without describing how it is installed. The providers associated with the resource direct the Chef client how to perform the installation on specific platforms.

Chef for Junos OS Features

To support the Chef framework on Juniper Networks devices running Junos OS, Chef for Junos OS provides the following software components:

- [Providers for the `netdev` Cookbook Resources on page 4](#)
- [Native Chef Client on page 5](#)
- [Native Ohai on page 5](#)
- [Ruby Interpreter and `junos-ez-stdlib` on page 5](#)

Providers for the `netdev` Cookbook Resources

The `netdev` cookbook, developed and maintained by Chef, contains platform-neutral primitives for the following network resources:

- Physical interfaces—Physical Ethernet interface attributes, such as administrative state, description, speed, duplex mode, and MTU with the `netdev_interface` resource
- Layer 2 Ethernet switching services—Logical Ethernet switching interface attributes, such as description, VLAN membership, and port mode (access or trunk) with the `netdev_l2_interface` resource
- Link aggregation groups (LAGs)—LAG interface attributes, such as name, member links, LACP mode, and minimum up links required with the `netdev_lag` resource
- VLANs—VLAN attributes, such as name, ID, and description with the `netdev_vlan` resource
- Configuration at any hierarchy level—Custom configuration with the `netdev_group` resource



NOTE: Juniper Networks OCX1100 switches support only the `netdev_interface` physical interface resource.

Chef for Junos OS supports providers that are specific to Junos OS for the switching resources. These providers translate the configuration modeled by the resources into the NETCONF XML code required to implement the configuration on the device the Chef client is running on. Together, the netdev cookbook resources and Junos OS providers enable you to automate your configuration of Juniper Networks devices running Junos OS without having knowledge of specific CLI commands or XML code.

The netdev cookbook is available at the Chef supermarket website at <https://supermarket.getchef.com/cookbooks/netdev>. For more information about the netdev cookbook resources, see *Chef for Junos OS* at <https://docs.chef.io/junos.html>.

Native Chef Client

The Chef client (chef-client) is an agent that runs locally on every managed node in a Chef deployment and performs the configuration defined in recipes. Chef for Junos OS provides a Chef client that runs natively on supported Juniper Networks devices running Junos OS.

Native Ohai

Ohai is a tool that collects detailed data about a node, such as hardware properties, memory and processor usage, networking statistics, kernel data, and hostname. It provides this data to the Chef client at the start of every Chef client run. This data is also uploaded to the Chef server at the end of each Chef client run, making it available to searches.

Chef for Junos OS provides a version of Ohai that runs natively on supported Juniper Networks devices running Junos OS. This version includes a plug-in that extends Ohai to collect Junos OS and platform-specific attributes. For a description of Ohai options and an example of using Ohai, see the Chef website at https://docs.chef.io/ctl_ohai.html.

Ruby Interpreter and junos-ez-stdlib

Chef for Junos OS provides a version of the Ruby Interpreter that is compatible with the Chef client. It also provides junos-ez-stdlib, which contains libraries used by the netdev cookbook providers and by Ohai.

Components of a Chef for Junos OS Deployment

A Chef for Junos OS deployment consists of the following major components:

- Chef server—The server acts as a hub for configuration data. The server stores cookbooks and the node object metadata that describes each registered node the Chef client manages.

You can choose between two types of servers:

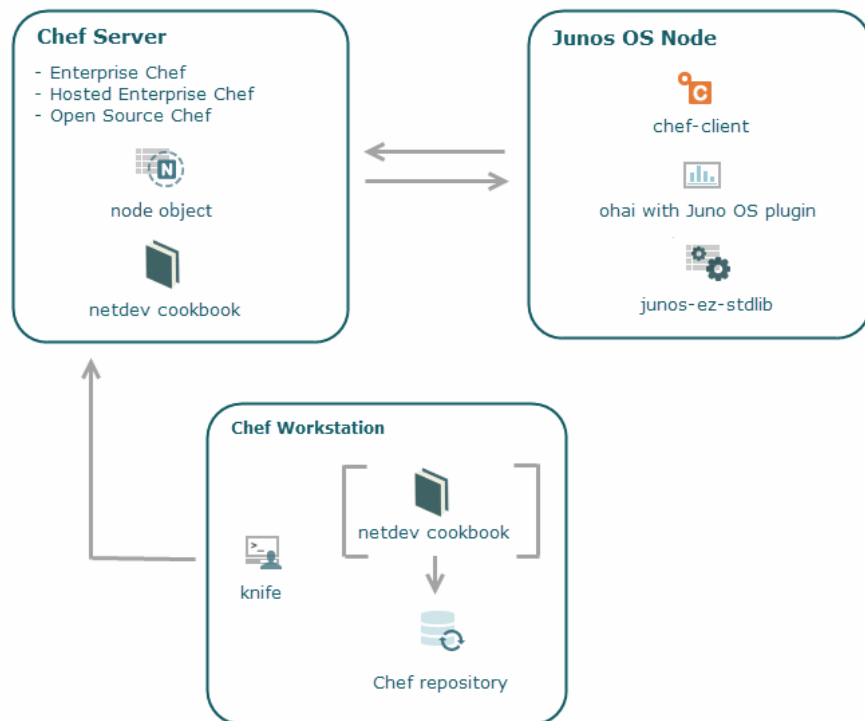
- Enterprise Chef—Highly scalable server that includes premium features and support. You can install the server behind a firewall, or you can use a cloud-based server hosted by Chef.
- Open source Chef—Open-source, free version of the server that is the basis for Enterprise Chef.

- **Workstations**—You perform most of your work on a workstation. You use the Chef CLI, which is called knife, to develop cookbooks and recipes, which are stored in a local Chef repository. From the workstation, you can synchronize the local repository with your version-control system, upload cookbooks to the Chef server, and perform operations on nodes.
- **Nodes**—A node is any device or virtual device that is configured for the Chef client to manage. To manage a node, the Chef client running on the node obtains the configuration details, such as recipes, templates, and file distributions, from the Chef server. The Chef client then does as much of the configuration work as possible on the node itself.

For a Juniper Networks device to be a Chef node, it must have the Chef client installed and configured on it. See the [Chef for Junos OS Release Notes](#) for information about Juniper Networks devices running Junos OS that support the Chef client.

Figure 1 on page 6 shows the major components of a Chef for Junos OS deployment. For more details about all the components that constitute a Chef deployment, see the Chef documentation at <https://docs.chef.io/>.

Figure 1: Major Components of a Chef for Junos OS Deployment



Deploying Chef for Junos OS Overview

The following major steps describe how you deploy Chef for Junos OS:

1. Set up the Chef server. For more information, see the Chef documentation at <https://docs.chef.io/>.
2. Set up the Chef workstation. The major steps for doing so are:
 - a. Install the Chef client and Ruby Interpreter on your workstation. You can install both at the same time by using the Chef omnibus installer.
 - b. Set up the Chef repository (chef-repo) and the version-control system.
 - c. Install authentication keys and verify that you can connect to the Chef server from your workstation.

For more information about setting up the Chef workstation, see the Chef documentation at <https://docs.chef.io/>.

- d. After you have set up the workstation, download the netdev cookbook to the chef-repo repository and extract the cookbook files.

knife cookbook site download netdev

tar -zxvf netdev-*n.n.n*.tar.gz -C cookbooks

3. If the Chef client is not already installed on the Junos OS nodes, install the client by using the Chef for Junos OS installation package as described in “[Installing or Removing the Chef Client on Juniper Networks Devices Running Junos OS](#)” on page 11.



NOTE: On Juniper Networks switches running Junos OS with Junos Automation Enhancements, you do not need to install the Chef client because the Chef client and related components are installed with the Junos OS software.

4. Configure the Chef client on the Junos OS nodes so that it can connect with the Chef server. For more information, see “[Configuring the Chef Client on Juniper Networks Devices Running Junos OS](#)” on page 17.

Running the Chef Client

To run the Chef client on a managed node, log in as the root user and enter one of the following commands from the UNIX-level shell:

- If the Juniper Networks version of the Chef client is 2.x (for example, Chef client version 11.10.4_2.0), enter:

```
%/opt/jet/chef/bin/ruby /opt/jet/chef/bin/chef-client -c /var/db/chef/client.rb
```

- If the Juniper Networks version of the Chef client is 1.x (for example, Chef client version 11.10.4_1.1), enter:

```
%/opt/sdk/chef/bin/ruby /opt/sdk/chef/bin/chef-client -c /var/db/chef/client.rb
```

These commands assume that your **client.rb** file resides in the **/var/db** directory. We recommend using this directory.

On a device with redundant Routing Engines, you must run the Chef client from the master Routing Engine.

When the Chef client runs, it obtains an exclusive configuration lock, which it releases after it commits all pending configuration changes. If you enable the reporting add-on on your Enterprise Chef server, the Chef client reports the results of the run back to the server. On successful Chef client runs, the Chef client sends a list of updated resources to the server; on failed Chef client runs, it sends a full exception stacktrace to the server.

The configuration of a resource on a managed node always reflects the resource state defined in the last recipe that was run that contains that resource. For example, if you run a recipe that defines a LAG resource as containing the member links ge-0/0/0 and ge-0/0/1 and then later run a recipe that defines the same LAG resource as containing the member links ge-0/0/2 and ge-0/0/3, the resulting configuration for the LAG on the managed node contains only the member links ge-0/0/2 and ge-0/0/3.

**Related
Documentation**

- [Installing or Removing the Chef Client on Juniper Networks Devices Running Junos OS on page 11](#)
- [Configuring the Chef Client on Juniper Networks Devices Running Junos OS on page 17](#)
- [Example: Using Chef for Junos OS to Configure Ethernet Switching on EX Series, OCX Series, and QFX Series Switches on page 19](#)
- [Example: Using Chef for Junos OS to Configure Ethernet Switching on MX Series Routers on page 28](#)

PART 2

Installation

- [Installing the Chef Client on page 11](#)

CHAPTER 2

Installing the Chef Client

- [Installing or Removing the Chef Client on Juniper Networks Devices Running Junos OS on page 11](#)

Installing or Removing the Chef Client on Juniper Networks Devices Running Junos OS

This topic describes how to install or remove the Chef client on Juniper Networks devices running Junos OS.



NOTE: On Juniper Networks switches running Junos OS with Junos Automation Enhancements, the Chef client is automatically installed when the Junos OS software is installed. Skip this installation procedure and go directly to configuring the Chef client as described in [“Configuring the Chef Client on Juniper Networks Devices Running Junos OS” on page 17](#).

This topic covers:

- [Installing or Upgrading the Chef Client on page 11](#)
- [Removing the Chef Client from the Juniper Networks Device on page 13](#)

Installing or Upgrading the Chef Client

The Chef client is part of an installation package that includes the Chef client, Ohai, the Ruby Interpreter, and junos-ez-stdlib. The following template describes the package format through the 15.1 Junos OS releases:

`chef-bundle-client-version_sdk-indicator.juniper-iteration.junos.platform.tgz`

and the following template describes the package format for 16.1R1 and later releases:

`chef-client-version_sdk-indicator.juniper-iteration_x86-32.tgz`

where:

- ***client-version*** is the version of the Chef client (for example, 11.10.4).
- ***sdk-indicator*** indicates the Junos OS SDK infrastructure used to create the package. A 1 indicates the Junos SDK; a 2 indicates the Junos Extension Toolkit (JET).

- **juniper-iteration** is the version of Juniper Networks version of the package.
- **platform** is the platform microprocessor architecture, either **powerpc** or **i386**.

You must use the installation package that matches the microprocessor architecture of your device. If you do not know the architecture used by your device, you can use the UNIX shell command **uname -a** to determine it.

To install or upgrade the Chef client on a Juniper Networks device:

1. Access the Chef for Junos OS download page at <http://www.juniper.net/support/downloads/?p=chefforjunos#sw>.

The *Chef for Junos OS Release Notes* are also available at the download site. Consult them for information about what package to install on your platform.

2. Download the Chef for Junos OS software package that is specific to your platform to the **/var/tmp/** directory on the device.



BEST PRACTICE: We recommend you install the software package from the **/var/tmp/** directory on your device to ensure the maximum amount of disk space and RAM for the installation.

3. In the Junos OS CLI, enter configuration mode.

```
user@host> configure
```

4. Configure the provider name, license type, and deployment scope associated with the application.

```
[edit]
user@host# set system extensions providers chef license-type juniper
deployment-scope commercial
user@host# commit and-quit
```

5. Install the software package by using the **request system software add** operational mode command.

```
user@host> request system software add /var/tmp/chef-package.tgz
```

6. Verify that the installation is successful by issuing the **show version** operational mode command.

If the installation is successful, the list of installed software includes the Chef, Ruby Interpreter, and junos-ez-stdlib packages. For example:

- If your installation package was built with the Junos Extension Toolkit, only one package is installed, JET app chef. This package includes all the required components, including the Ruby Interpreter and junos-ez-stdlib. To verify the installation:

```
user@host> show version | match chef
JET app chef [11.10.4_2.0]
```

- If your installation package was built with the Junos SDK, three packages are installed: the Chef, Ruby Interpreter, and junos-ez-stdlib packages. To verify the installation:

```
user@host> show version

fpc0:
-
Hostname: host
Model: ex4300-24p
Junos: 14.1X53-D10.2
JUNOS EX Software Suite [14.1X53-D10.2]
JUNOS FIPS mode utilities [14.1X53-D10.2]
JUNOS Online Documentation [14.1X53-D10.2]
JUNOS EX 4300 Software Suite [14.1X53-D10.2]
JUNOS Web Management Platform Package [14.1X53-D10.2]
JUNOS py-base-powerpc [14.1X53-D10.2]
Ruby Interpreter [11.10.4_1.junos.powerpc]
Chef [11.10.4_1.junos.powerpc]
junos-ez-stdlib [11.10.4_1.junos.powerpc]
```

After you install the Chef client, you must configure it as described in [“Configuring the Chef Client on Juniper Networks Devices Running Junos OS”](#) on page 17.

Removing the Chef Client from the Juniper Networks Device

To remove the Chef client from the Juniper Networks device, use the **request system software delete** CLI command to delete the installed packages. For example:

- To delete a Chef client package built by the Junos Extension Toolkit (JET), enter:

```
user@host> request system software delete chef
```

- To delete the Chef client and related packages built by the Junos SDK, enter:

```
user@host> request system software delete chef
user@host> request system software delete junos-ez-stdlib
user@host> request system software delete ruby
```

Related Documentation

- [Configuring the Chef Client on Juniper Networks Devices Running Junos OS](#) on page 17
- [Example: Using Chef for Junos OS to Configure Ethernet Switching on EX Series, OCX Series, and QFX Series Switches](#) on page 19
- [Example: Using Chef for Junos OS to Configure Ethernet Switching on MX Series Routers](#) on page 28
- [Chef for Junos OS Overview](#) on page 3

PART 3

Configuration

- [Client Configuration on page 17](#)
- [Example on page 19](#)

CHAPTER 3

Client Configuration

- [Configuring the Chef Client on Juniper Networks Devices Running Junos OS on page 17](#)

Configuring the Chef Client on Juniper Networks Devices Running Junos OS

To enable the Chef client to communicate with the Chef server, you must configure the Chef client after it is installed on the Juniper Networks device.



NOTE: You must set up the Chef workstation and the Chef server so that they can communicate before you perform this procedure.

To configure the Chef client:

1. On your Juniper Networks device that is running Junos OS, log in as the root user and create the `/var/db/chef` directory.

```
mkdir -p /var/db/chef
```

2. Copy your validation key into the `/var/db/chef` directory.

If you do not have your validation key, you can obtain it as follows:

- If you are using Open Source Chef, you can obtain your validation key from `/etc/chef` on your server. The key is named **chef-validator.pem**.
- If you are using Enterprise Chef (hosted or on-premise), you can obtain your validation key from the Enterprise Chef management console. The key is named **orgname-validator.pem**, where *orgname* is your organization name.

3. Create a **client.rb** file in `/var/db/chef` that contains the following statements:

```
current_dir = File.dirname(__FILE__)
log_level      :auto
log_location   STDOUT
chef_server_url "value"
validation_client_name "value"
node_name      "value"
validation_key  "#{current_dir}/value"
client_key      "#{current_dir}/client.pem"
file_cache_path "#{current_dir}/cache"
verbose_logging true
```

where:

- **chef_server_url** is the URL of your Chef server
- **validation_client_name** is **chef-validator** if you are using Open Source Chef and **orgname-validator** if you are using Enterprise Chef
- **node_name** is optional if the switch has a hostname configured
- **validation_key** is **chef-validator.pem** if you are using Open Source Chef and **orgname-validator.pem** if you are using Enterprise Chef

For more information about the settings in the **client.rb** file, see https://docs.chef.io/config_rb_client.html.

4. Run the Chef client.

- If the Juniper Networks version of the Chef client is 2.x (for example, Chef client version 11.10.4_2.0), enter:

```
%/opt/jet/chef/bin/ruby /opt/jet/chef/bin/chef-client -c /var/db/chef/client.rb
```

- If the Juniper Networks version of the Chef client is 1.x (for example, Chef client version 11.10.4_1.1), enter:

```
%/opt/sdk/chef/bin/ruby /opt/sdk/chef/bin/chef-client -c /var/db/chef/client.rb
```

Related Documentation

- [Installing or Removing the Chef Client on Juniper Networks Devices Running Junos OS on page 11](#)
- [Example: Using Chef for Junos OS to Configure Ethernet Switching on EX Series, OCX Series, and QFX Series Switches on page 19](#)
- [Example: Using Chef for Junos OS to Configure Ethernet Switching on MX Series Routers on page 28](#)
- [Chef for Junos OS Overview on page 3](#)

CHAPTER 4

Example

- [Example: Using Chef for Junos OS to Configure Ethernet Switching on EX Series, OCX Series, and QFX Series Switches on page 19](#)
- [Example: Using Chef for Junos OS to Configure Ethernet Switching on MX Series Routers on page 28](#)
- [Example: Using Chef for Junos OS to Configure Any Hierarchy Level on page 37](#)

Example: Using Chef for Junos OS to Configure Ethernet Switching on EX Series, OCX Series, and QFX Series Switches

This example shows how you can use resources in the netdev cookbook to write recipes that configure the switching interfaces on EX Series switches, OCX Series switches, and QFX Series switches running the Chef client. For more information about the light-weight resources in the netdev cookbook, see *Chef for Junos OS* at <https://docs.chef.io/junos.html>.

- [Requirements on page 19](#)
- [Overview on page 20](#)
- [Configuration on page 21](#)
- [Verification on page 27](#)

Requirements

This example uses the following hardware and software components:

- A properly set up and configured Chef workstation and Chef server
- A supported Junos OS release (as specified in the *Chef for Junos OS Release Notes*)
- A Juniper Networks switch that the Chef client manages



NOTE: This example uses netdev resources not supported on an OCX1100 switch. Only the `netdev_interface` resource is supported on an OCX1100 switch.

Before you begin, the number of aggregated Ethernet interfaces supported on the switch must have been already configured before you run the Chef client. To verify that a sufficient number of aggregated Ethernet interfaces has been configured, use the **show chassis**

aggregated-devices configuration mode CLI command. Use the **set chassis aggregated-devices ethernet device-count** command to set the number of supported aggregated Ethernet interfaces.

Overview

This example takes you through using Chef for Junos OS to configure the switching interfaces on an access switch.

In this example, you create a cookbook, called `netdev_access_switch`, that is based on the `netdev` cookbook. Within the cookbook, you create three recipes:

- **vlan_create** recipe—Defines `netdev_vlan` resources for the VLANs shown in [Table 3 on page 20](#).
- **access_interface_create** recipe—Defines `netdev_interface` and `netdev_l2_interface` resources for the access interfaces shown in [Table 4 on page 20](#).
- **uplink_interface_create** recipe—Defines `netdev_lag` and `netdev_l2_interface` resources for the link aggregation group (LAG) interfaces shown in [Table 5 on page 20](#).

Table 3: VLANs Defined in the `vlan_create` Recipe

Name	VLAN ID	Description
blue	100	the blue VLAN
green	200	the green VLAN
red	300	the red VLAN

Table 4: Access Interfaces Defined in the `access_interface_create` Recipe

Name	Port Mode	VLAN Membership	Description
et-0/0/4	Access	blue	Access interface
et-0/0/5	Access	green	Access interface
et-0/0/6	Access	red	Access interface

Table 5: LAGs Defined in the `uplink_interface_create` Recipe

Name	Member Interfaces	Minimum Links	LACP	Port Mode	VLAN Membership	Description
ae0	et-0/1/0 et-0/1/2	1	Active	Trunk	blue, green, red	Uplink interface
ae1	et-0/2/0 et-0/2/2	1	Active	Trunk	blue, green, red	Uplink interface

In your own implementation of Chef for Junos OS, you can structure recipes in any way that makes sense for deploying and managing your switching resources. The recipes used in this example are simply one way of doing so.

After you create the recipes, you upload the cookbook to the Chef server and add the recipes to the run list for the access switch. Finally, you run the Chef client on the access switch. The client then uses the Junos OS providers in the netdev cookbook to implement the configuration described in the recipes.

Configuration

Step-by-Step Procedure

To configure the access switch by using Chef for Junos OS:

1. From the **chef-repo** directory on the Chef workstation, download the netdev cookbook and extract the cookbook files to the cookbooks directory:


```
knife cookbook site download netdev
tar -zxvf netdev-n.n.n.tar.gz -C cookbooks
```
2. Copy the netdev cookbook to create a new cookbook, netdev_access_switch, in the cookbooks directory.
3. In an editor of your choice, write the **vlan_create** recipe for creating the blue, green, and red VLANs.

```
#
# Cookbook Name:: netdev_access_switch
# Recipe:: vlan_create
#
# Copyright 2013, YOUR_COMPANY_NAME
#
# All rights reserved - Do Not Redistribute
#

netdev_vlan "blue" do
  vlan_id 100
  description "the blue VLAN"
  action :create
end

netdev_vlan "green" do
  vlan_id 200
  description "the green VLAN"
  action :create
end

netdev_vlan "red" do
  vlan_id 300
  description "the red VLAN"
  action :create
end
```

4. Save the recipe in **cookbooks/netdev_access_switch/recipes/vlan_create.rb**.
5. In an editor of your choice, write the **access_interface_create** recipe, which configures the physical and Layer 2 properties of the access interfaces.

```
#
# Cookbook Name:: netdev_access_switch
# Recipe:: access_interface_create
#
# Copyright 2013, YOUR_COMPANY_NAME
#
# All rights reserved - Do Not Redistribute
#

# Physical interface creation using the following defaults:
# auto-negotiation on, MTU 1500, administratively up

netdev_interface "et-0/0/4" do
  description "access interface"
  action :create
end

netdev_interface "et-0/0/5" do
  description "access interface"
  action :create
end

netdev_interface "et-0/0/6" do
  description "access interface"
  action :create
end

# Logical interface creation, setting port mode to access (vlan_tagging false)
# and assigning interface to a VLAN

netdev_l2_interface "et-0/0/4" do
  description "belongs to blue VLAN"
  untagged_vlan "blue"
  vlan_tagging false
  action :create
end

netdev_l2_interface "et-0/0/5" do
  description "belongs to green VLAN"
  untagged_vlan "green"
  vlan_tagging false
  action :create
end

netdev_l2_interface "et-0/0/6" do
  description "belongs to red VLAN"
  untagged_vlan "red"
  vlan_tagging false
  action :create
end
```

6. Save the recipe in `cookbooks/netdev_access_switch/recipes/ access_interface_create.rb`.
7. In an editor of your choice, write the `uplink_interface_create` recipe, which configures the LAG trunk interfaces.

```
#
# Cookbook Name:: netdev-access-switch
# Recipe:: uplink_interface_create
#
# Copyright 2013, YOUR_COMPANY_NAME
#
# All rights reserved - Do Not Redistribute
#

netdev_l2_interface "et-0/1/0" do
  action :delete
end

netdev_l2_interface "et-0/1/2" do
  action :delete
end

netdev_l2_interface "et-0/2/0" do
  action :delete
end

netdev_l2_interface "et-0/2/2" do
  action :delete
end

# Create the LAGs

netdev_lag "ae0" do
  links [ "et-0/1/0", "et-0/2/0" ]
  minimum_links 1
  lacp "active"
  action :create
end

netdev_lag "ae1" do
  links [ "et-0/2/2", "et-0/1/2" ]
  minimum_links 1
  lacp "active"
  action :create
end

# Configure Layer 2 switching on the LAGs. Define the port mode as trunk
# (vlan_tagging true), with membership in the blue, green, and red VLANs.

netdev_l2_interface "ae0" do
  description "Uplink interface"
  tagged_vlans [ "blue", "green", "red" ]
  vlan_tagging true
  action :create
end

netdev_l2_interface "ae1" do
  description "Uplink interface"
  tagged_vlans [ "blue", "green", "red" ]
  vlan_tagging true
end
```

```
        action :create
    end
```

8. Save the recipe in `cookbooks/netdev_access_switch /recipes/uplink_interface_create.rb`.
9. Upload the `netdev_access_switch` cookbook to the Chef server.

```
$ knife cookbook upload netdev_access_switch
```

10. Edit the node object that represents the access switch.

```
$ knife node edit access_switch_node_name
```

Knife starts your editor and opens a JSON file that contains the node attributes.

11. Enter the recipes in the `run-list` attribute and then save the JSON file.

```
{
  "name": "access_switch_node_name",
  "chef_environment": "_default",
  "normal": {
  },
  "run_list": [
    "recipe[netdev_access_switch::vlan_create]",
    "recipe[netdev_access_switch::access_interface_create]",
    "recipe[netdev_access_switch::uplink_interface_create]"
  ]
}
```

The order in which you enter the recipes matters—for example, the Chef client runs the `vlans_create` recipe first because it is listed first.

12. If the number of aggregated Ethernet interfaces supported on the switch is not already configured, log in to the access switch, enter configuration mode, and configure the number of aggregated Ethernet interfaces supported.

```
root@access-switch-node# set chassis aggregated-devices ethernet device-count
2
root@access-switch-node# commit and-quit
```

13. On the access switch, log in as the root user.

14. From the UNIX-level shell, run the Chef client.

- If the Juniper Networks version of the Chef client is 2.x (for example, Chef client version 11.10.4_2.0), enter:

```
%/opt/jet/chef/bin/ruby /opt/jet/chef/bin/chef-client -c /var/db/chef/client.rb
```

- If the Juniper Networks version of the Chef client is 1.x (for example, Chef client version 11.10.4_1.1), enter:

```
%/opt/sdk/chef/bin/ruby /opt/sdk/chef/bin/chef-client -c
/var/db/chef/client.rb
```

The Chef client displays status messages during its run to indicate its progress in performing the configuration. For example:

```
[2014-02-24T15:17:53-08:00] INFO: Forking chef instance to converge...
Starting Chef Client, version 11.10.4
```

```

[2014-02-24T15:17:53-08:00] INFO: *** Chef 11.10.4 ***
[2014-02-24T15:17:53-08:00] INFO: Chef-client pid: 41108
[2014-02-24T15:17:54-08:00] INFO: Run List is [recipe[netdev::vlan_create_doc],
recipe[netdev::access_interface_create_doc],
recipe[netdev::uplink_interface_create_doc]]
[2014-02-24T15:17:54-08:00] INFO: Run List expands to [netdev::vlan_create_doc,
netdev::access_interface_create_doc, netdev::uplink_interface_create_doc]
[2014-02-24T15:17:54-08:00] INFO: Starting Chef Run for access-switch-node

.
.
.
2014-02-24T15:18:07-08:00] INFO: Chef Run complete in 12.618406061 seconds
Running handlers:
[2014-02-24T15:18:07-08:00] INFO: Running report handlers
[2014-02-24T15:18:09-08:00] INFO: Committed pending Junos candidate
configuration changes
[2014-02-24T15:18:09-08:00] INFO: Released exclusive Junos configuration lock
- JunosCommitTransactionHandler
Running handlers complete

```

Results

To check the results of the configuration:

1. On the access switch, enter the CLI.

```
% cli
```

2. Enter the following CLI operational mode command:

```

root@access-switch-node> show configuration | compare rollback 1
[edit]
+ interfaces {
+   et-0/0/4 {
+     description "access interface";
+     unit 0 {
+       description "belongs to blue VLAN";
+       family ethernet-switching {
+         interface-mode access;
+         vlan {
+           members blue;
+         }
+       }
+     }
+   }
+   et-0/0/5 {
+     description "access interface";
+     unit 0 {
+       description "belongs to green VLAN";
+       family ethernet-switching {
+         interface-mode access;
+         vlan {
+           members green;
+         }
+       }
+     }
+   }
+ }

```

```
+   }
+ }
+ et-0/0/6 {
+   description "access interface";
+   unit 0 {
+     description "belongs to red VLAN";
+     family ethernet-switching {
+       interface-mode access;
+       vlan {
+         members red;
+       }
+     }
+   }
+ }
+ et-0/1/0 {
+   ether-options {
+     802.3ad ae0;
+   }
+ }
+ et-0/1/2 {
+   ether-options {
+     802.3ad ae1;
+   }
+ }
+ et-0/2/0 {
+   ether-options {
+     802.3ad ae0;
+   }
+ }
+ et-0/2/2 {
+   ether-options {
+     802.3ad ae1;
+   }
+ }
+ ae0 {
+   aggregated-ether-options {
+     minimum-links 1;
+     lacp {
+       active;
+     }
+   }
+   unit 0 {
+     description "Uplink interface";
+     family ethernet-switching {
+       interface-mode trunk;
+       vlan {
+         members [ blue green red ];
+       }
+     }
+   }
+ }
+ ae1 {
+   aggregated-ether-options {
+     minimum-links 1;
+     lacp {
+       active;
```



```

+   }
+ }
+ unit 0 {
+   description "Uplink interface";
+   family ethernet-switching {
+     interface-mode trunk;
+     vlan {
+       members [ blue green red ];
+     }
+   }
+ }
+ }
+ }
+ }
+ vlans {
+   blue {
+     description "the blue VLAN";
+     vlan-id 100;
+   }
+   green {
+     description "the green VLAN";
+     vlan-id 200;
+   }
+   red {
+     description "the red VLAN";
+     vlan-id 300;
+   }
+ }
+ }

```

Verification

Verifying the Status of the VLANs

Purpose Verify the VLANs and VLAN memberships are correct.

Action Use the **show vlans** command to verify VLAN membership.

```
root@access-switch-node> show vlans
```

Routing instance	VLAN name	Tag	Interfaces
default-switch	blue	100	ae0.0* ae1.0* et-0/0/4.0*
default-switch	green	200	ae0.0* ae1.0* et-0/0/5.0*
default-switch	red	300	ae0.0* ae1.0* et-0/0/6.0*

Meaning The output shows that the VLANs have been created correctly and contain the correct member interfaces.

- Related Documentation**
- [Installing or Removing the Chef Client on Juniper Networks Devices Running Junos OS on page 11](#)
 - [Configuring the Chef Client on Juniper Networks Devices Running Junos OS on page 17](#)
 - [Chef for Junos OS Overview on page 3](#)

Example: Using Chef for Junos OS to Configure Ethernet Switching on MX Series Routers

This example shows how you can use resources in the netdev cookbook to write recipes that configure the switching interfaces on MX Series routers running the Chef client. For more information about the light-weight resources in the netdev cookbook, see *Chef for Junos OS* at <https://docs.chef.io/junos.html>.

- [Requirements on page 28](#)
- [Overview on page 28](#)
- [Configuration on page 30](#)
- [Verification on page 36](#)

Requirements

This example uses the following hardware and software components:

- A properly set up and configured Chef workstation and Chef server
- An MX Series router that the Chef client manages
- Junos OS Release 16.1 or later

Before you begin, the number of aggregated Ethernet interfaces supported on the router must already be configured before you run the Chef client.

- To verify that a sufficient number of aggregated Ethernet interfaces has been configured, use the **show chassis aggregated-devices** configuration mode CLI command. Use the **set chassis aggregated-devices ethernet device-count** command to set the number of supported aggregated Ethernet interfaces.
- If the number of aggregated Ethernet interfaces supported on the router is not already configured, log in to the router, enter configuration mode, and configure the number of aggregated Ethernet interfaces supported:

```
root@router-node# set chassis aggregated-devices ethernet device-count 2
root@router-node# commit and-quit
```

Overview

This example takes you through using Chef for Junos OS to configure the switching interfaces on an MX Series router.

In this example, you create a cookbook, called `netdev_router`, that is based on the `netdev` cookbook. Within the cookbook, you create four recipes:

- **vlan_create** recipe—Defines `netdev_vlan` resources for the VLANs shown in [Table 6 on page 29](#).
- **interface_create** recipe—Defines the `netdev_interface` resources for the interfaces shown in [Table 7 on page 29](#).
- **l2interface_create** recipe—Defines the `netdev_l2_interface` resources for the interfaces shown in [Table 7 on page 29](#).
- **lag_interface_create** recipe—Defines `netdev_lag` and `netdev_l2_interface` resources for the link aggregation group (LAG) interfaces shown in [Table 8 on page 29](#).

Table 6: VLANs Defined in the `vlan_create` Recipe

Name	VLAN ID	Description
blue	100	Chef-created blue VLAN
green	200	Chef-created green VLAN
red	300	Chef-created red VLAN

Table 7: Interfaces Defined in the `interface_create` and `l2interface_create` Recipes

Name	Port Mode	VLAN Membership	Description
ge-1/0/1	Access	blue	Chef-created interface
ge-1/0/2	Access	green	Chef-created interface
ge-1/0/3	Access	red	Chef-created interface

Table 8: LAGs Defined in the `lag_interface_create` Recipe

Name	Member Interfaces	Minimum Links	LACP	Port Mode	VLAN Membership	Description
ae0	ge-1/0/6 ge-1/0/7	1	Active	Trunk	blue, green, red	Chef-created LAG interface
ae1	ge-1/0/8 ge-1/0/9	1	Active	Trunk	blue, green, red	Chef-created LAG interface

In your own implementation of Chef for Junos OS, you can structure recipes in any way that makes sense for deploying and managing your switching resources. The recipes used in this example are simply one way of doing so.

After you create the recipes, you upload the cookbook to the Chef server and add the recipes to the run list for the managed router. Finally, you run the Chef client on the router. The client then uses the Junos OS providers in the netdev cookbook to implement the configuration described in the recipes.



NOTE: The number of aggregated Ethernet interfaces supported on the router must already be configured before you run the Chef client.

Configuration

Step-by-Step Procedure

To configure the router by using Chef for Junos OS:

1. From the **chef-repo** directory on the Chef workstation, download the netdev cookbook and extract the cookbook files to the **cookbooks** directory.

knife cookbook site download netdev

tar -zxvf netdev-n.n.n.tar.gz -C cookbooks
2. Copy the netdev cookbook to create a new cookbook, netdev_router, in the cookbooks directory.
3. In an editor of your choice, write the **vlan_create** recipe for creating the blue, green, and red VLANs.

```
#
# Cookbook Name:: netdev_router
# Recipe:: vlan_create
#

netdev_vlan "blue" do
  vlan_id 100
  description "Chef-created blue VLAN"
  action :create
end

netdev_vlan "green" do
  vlan_id 200
  description "Chef-created green VLAN"
  action :create
end

netdev_vlan "red" do
  vlan_id 300
  description "Chef-created red VLAN"
  action :create
end
```

4. Save the recipe in **cookbooks/netdev_router/recipes/vlan_create.rb**.
5. In an editor of your choice, write the **interface_create** recipe, which configures the physical properties of the interfaces.

```
#
# Cookbook Name:: netdev_router
```

```
# Recipe:: interface_create
#

# Physical interface creation using the following defaults:
# auto-negotiation on, MTU 1500, administratively up

netdev_interface "ge-1/0/1" do
  description "Chef-created interface"
  action :create
end

netdev_interface "ge-1/0/2" do
  description "Chef-created interface"
  action :create
end

netdev_interface "ge-1/0/3" do
  description "Chef-created interface"
  action :create
end
```

6. Save the recipe in `cookbooks/netdev_router/recipes/interface_create.rb`.
7. In an editor of your choice, write the `l2interface_create` recipe, which configures the Layer 2 properties of the interfaces.

```
#
# Cookbook Name:: netdev_router
# Recipe:: l2interface_create
#

# Logical interface creation, setting port mode to access (vlan_tagging false)
# and assigning interface to a VLAN

netdev_l2_interface "ge-1/0/1" do
  description "belongs to blue VLAN"
  untagged_vlan "blue"
  vlan_tagging false
  action :create
end

netdev_l2_interface "ge-1/0/2" do
  description "belongs to green VLAN"
  untagged_vlan "green"
  vlan_tagging false
  action :create
end

netdev_l2_interface "ge-1/0/3" do
  description "belongs to red VLAN"
  untagged_vlan "red"
  vlan_tagging false
  action :create
end
```

8. Save the recipe in `cookbooks/netdev_router/recipes/l2interface_create.rb`.

9. In an editor of your choice, write the **lag_interface_create** recipe, which configures the LAG trunk interfaces.

```
#
# Cookbook Name:: netdev-router
# Recipe:: lag_interface_create
#

netdev_l2_interface "ge-1/0/6" do
  action :delete
end

netdev_l2_interface "ge-1/0/7" do
  action :delete
end

netdev_l2_interface "ge-1/0/8" do
  action :delete
end

netdev_l2_interface "ge-1/0/9" do
  action :delete
end

# Create the LAGs

netdev_lag "ae0" do
  links [ "ge-1/0/6", "ge-1/0/7" ]
  minimum_links 1
  lacp "active"
  action :create
end

netdev_lag "ae1" do
  links [ "ge-1/0/8", "ge-1/0/9" ]
  minimum_links 1
  lacp "active"
  action :create
end

# Configure Layer 2 switching on the LAGs. Define the port mode as trunk
# (vlan_tagging true), with membership in the blue, green, and red VLANs.

netdev_l2_interface "ae0" do
  description "Chef-created LAG interface"
  tagged_vlans [ "blue", "green", "red" ]
  vlan_tagging true
  action :create
end

netdev_l2_interface "ae1" do
  description "Chef-created LAG interface"
  tagged_vlans [ "blue", "green", "red" ]
  vlan_tagging true
  action :create
end
```

10. Save the recipe in `cookbooks/netdev_router/recipes/lag_interface_create.rb`.

11. Upload the `netdev_router` cookbook to the Chef server.

```
$ knife cookbook upload netdev_router
```

12. Edit the node object that represents the router.

```
$ knife node edit router_node_name
```

Knife starts your editor and opens a JSON file that contains the node attributes.

13. Enter the recipes in the `run-list` attribute and then save the JSON file.

```
{
  "name": "router_node_name",
  "chef_environment": "_default",
  "normal": {
  },
  "run_list": [
    "recipe[netdev_router::interface_create]",
    "recipe[netdev_router::vlan_create]",
    "recipe[netdev_router::l2interface_create]",
    "recipe[netdev_router::lag_interface_create]"
  ]
}
```

The order in which you enter the recipes matters—for example, the Chef client runs the `interfaces_create` recipe first because it is listed first.

14. Log in as the root user.

15. From the UNIX-level shell, run the Chef client.

- If the Juniper Networks version of the Chef client is 2.x (for example, Chef client version 11.10.4_2.0), enter:

```
%/opt/jet/chef/bin/ruby /opt/jet/chef/bin/chef-client -c /var/db/chef/client.rb
```

- If the Juniper Networks version of the Chef client is 1.x (for example, Chef client version 11.10.4_1.1), enter:

```
%/opt/sdk/chef/bin/ruby /opt/sdk/chef/bin/chef-client -c
/var/db/chef/client.rb
```

The Chef client displays status messages during its run to indicate its progress in performing the configuration. For example:

```
[2015-08-21T18:07:27+05:30] INFO: Forking chef instance to converge...
Starting Chef Client, version 11.10.4
[2015-08-21T18:07:28+05:30] INFO: *** Chef 11.10.4 ***
[2015-08-21T18:07:28+05:30] INFO: Chef-client pid: 9351
[2015-08-21T18:07:32+05:30] INFO: Run List is [recipe[netdev::interface_create],
recipe[netdev::vlan_create], recipe[netdev::l2interface_create],
recipe[netdev::lag_interface_create]]
[2015-08-21T18:07:32+05:30] INFO: Run List expands to [netdev::interface_create,
netdev::vlan_create, netdev::l2interface_create, netdev::lag_interface_create]
[2015-08-21T18:07:32+05:30] INFO: Starting Chef Run for router-node
.
.
.
```

```
[2015-08-21T18:09:36+05:30] INFO: Chef Run complete in 123.446606904 seconds
```

```
Running handlers:
```

```
[2015-08-21T18:09:36+05:30] INFO: Running report handlers
```

```
[2015-08-21T18:09:54+05:30] INFO: Committed pending Junos candidate  
configuration changes
```

```
[2015-08-21T18:09:58+05:30] INFO: Released exclusive Junos configuration lock
```

```
- JunosCommitTransactionHandler
```

```
Running handlers complete
```

```
[2015-08-21T18:09:58+05:30] INFO: Report handlers complete
```

```
Chef Client finished, 13/17 resources updated in 150.983654211 seconds
```

Results

From operational mode, confirm your configuration by entering the **show configuration | compare rollback 1** command. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
root@router-node> show configuration | compare rollback 1
```

```
[edit]
```

```
+ interfaces {  
+   ge-1/0/1 {  
+     description "Chef-created interface";  
+     unit 0 {  
+       description "belongs to blue VLAN";  
+       family bridge {  
+         interface-mode access;  
+         vlan-id 100;  
+       }  
+     }  
+   }  
+   ge-1/0/2 {  
+     description "Chef-created interface";  
+     unit 0 {  
+       description "belongs to green VLAN";  
+       family bridge {  
+         interface-mode access;  
+         vlan-id 200;  
+       }  
+     }  
+   }  
+   ge-1/0/3 {  
+     description "Chef-created interface";  
+     unit 0 {  
+       description "belongs to red VLAN";  
+       family bridge {  
+         interface-mode access;  
+         vlan-id 300;  
+       }  
+     }  
+   }  
+   ge-1/0/6 {  
+     gigether-options {
```



```

+     802.3ad ae0;
+   }
+ }
+ ge-1/0/7 {
+   gigether-options {
+     802.3ad ae0;
+   }
+ }
+ ge-1/0/8 {
+   giether-options {
+     802.3ad ae1;
+   }
+ }
+ ge-1/0/9 {
+   gigether-options {
+     802.3ad ae1;
+   }
+ }
+ ae0 {
+   aggregated-ether-options {
+     minimum-links 1;
+     lacp {
+       active;
+     }
+   }
+   apply-macro "netdev_lag[:links]" {
+     ge-1/0/6;
+     ge-1/0/7;
+   }
+   unit 0 {
+     description "Chef-created LAG interface";
+     family bridge {
+       interface-mode trunk;
+       vlan-id-list [ 100 200 300 ];
+     }
+   }
+ }
+ ae1 {
+   aggregated-ether-options {
+     minimum-links 1;
+     lacp {
+       active;
+     }
+   }
+   apply-macro "netdev_lag[:links]" {
+     ge-1/0/8;
+     ge-1/0/9;
+   }
+   unit 0 {
+     description "Chef-created LAG interface";
+     family bridge {
+       interface-mode trunk;
+       vlan-id-list [ 100 200 300 ];
+     }
+   }
+ }

```

```

+ }
+ bridge-domains {
+   blue {
+     description "Chef-created blue VLAN";
+     domain-type bridge;
+     vlan-id 100;
+   }
+   green {
+     description "Chef-created green VLAN";
+     domain-type bridge;
+     vlan-id 200;
+   }
+   red {
+     description "Chef-created blue VLAN";
+     domain-type bridge;
+     vlan-id 300;
+   }
+ }

```



NOTE: The `apply-macro` statement under the `ae0` and `ae1` interface configuration is a normally hidden statement that is exposed when the configuration is generated by a Chef client.

Verification

Verifying the Status of the VLANs

Purpose Verify the VLANs and VLAN memberships are correct.

Action Use the `show bridge domain` command to verify VLAN membership.

```

root@mx-node> show bridge domain

```

Routing instance	Bridge domain	VLAN ID	Interfaces
default-switch	blue	100	ae0.0* ae1.0* ge-1/0/1.0*
default-switch	green	200	ae0.0* ae1.0* ge-1/0/2.0*
default-switch	red	300	ae0.0* ae1.0* ge-1/0/3.0*

Meaning The output shows that the VLANs have been created correctly and contain the correct member interfaces.

- Related Documentation**
- [Installing or Removing the Chef Client on Juniper Networks Devices Running Junos OS on page 11](#)
 - [Configuring the Chef Client on Juniper Networks Devices Running Junos OS on page 17](#)
 - [Chef for Junos OS Overview on page 3](#)

Example: Using Chef for Junos OS to Configure Any Hierarchy Level

This example shows how you can use the **netdev_group** resource in the netdev cookbook to write recipes that configure any hierarchy level on devices running Chef for Junos OS. For more information about the light-weight resources in the netdev cookbook, see *Chef for Junos OS* at <https://docs.chef.io/junos.html>.

- [Requirements on page 37](#)
- [Overview on page 37](#)
- [Configuration on page 38](#)
- [Verification on page 41](#)

Requirements

This example uses the following hardware and software components:

- A properly set up and configured Chef workstation and Chef server
- Junos OS Release 16.1
- A Juniper Networks device that the Chef client manages



NOTE: This example uses the **netdev_group** resource that is not supported on an OCX1100 switch. Only the **netdev_interface** resource is supported on an OCX1100 switch.

Before you begin, make sure that the local autonomous system number is already defined on the device.

Overview

The **netdev_group** resource specifies an Embedded Ruby (ERB) template file that defines a Junos OS configuration to be applied to the **groups** hierarchy level on the device. For information about Chef cookbook templates, see <https://docs.chef.io/templates.html>. When the client downloads the catalog, it adds the configuration data generated by the template under the **[edit groups]** hierarchy level and configures the **apply-groups** statement to include the group name. If the commit succeeds, the configuration inherits the statements in the configuration group. The configuration file is created in **/var/tmp/name**, where *name* is the name of a Junos OS group on the Chef client.

The **netdev_group** resource has the following actions:

- **:create**—Create a Junos OS group (default).

- **:delete**—Delete a Junos OS group.

The **netdev_group** resource has the following attributes:

- **name**—The name of the Junos OS group under which configuration is applied.
- **template_path**—The path of the template used to create the Junos OS configuration file in the format **template-file-name.config-format.erb**, where *template-file-name* is the name of the file and *config-format* is one of **xml**, **set**, or **text**. If *config-format* is not specified, **xml** is the default format.
- **variables**—(Optional) Variables input to the template file.

Configuration

This example creates a **netdev_group** resource named **bgp_create.rb** that configures statements for internal and external BGP peering. The **netdev_group** resource references the **bgp.xml.erb** template that generates the configuration data for the resource. The template is located in the **netdev/templates/junos** directory. The attributes that apply to the template are defined in **netdev/attributes/default.rb** under the variable name **bgp**.

The BGP variable definition contains the node-specific configuration values that the template uses to generate the configuration data for that group. The data is provided in a hash that uses the BGP group names as keys. Each key maps to another hash that contains the details for that group including the group type, and the IP addresses and AS number of the peers. When the template is referenced, it iterates over the hash and generates the Junos OS configuration data for the **groups** command.

- [Creating the netdev_group Resource on page 38](#)
- [Creating the ERB Template on page 39](#)
- [Creating the Attributes for the Template on page 40](#)
- [Configuring the Device by Using Chef for Junos OS on page 40](#)

Creating the netdev_group Resource

Step-by-Step Procedure

To create the **netdev_group** resource:

1. From the **chef-repo** directory on the Chef workstation, download the netdev cookbook and extract the cookbook files to the **cookbooks** directory.

```
knife cookbook site download netdev
```

```
tar -zxvf netdev-n.n.n.tar.gz -C cookbooks
```

2. Copy the netdev cookbook to create a new cookbook, **netdev_device**, in the **cookbooks** directory.
3. In an editor of your choice, write the **bgp_group** recipe for creating the BGP configuration in the **cookbooks/netdev_device/recipes/bgp_create.rb** file.

```
#  
# Cookbook Name:: netdev_device  
# Recipe:: bgp_create  
#
```

```

netdev_group 'bgp_create' do
  template_path 'bgp.xml.erb'
  action :create
  variables({
    :bgp => node[:netdev][:bgp]
  })
end

```

Creating the ERB Template

Step-by-Step Procedure

To create and stage the ERB template:

- Create a new template file named **bgp.xml.erb** in the **netdev/templates/junos** directory, and add the text and Ruby tags required to generate the desired configuration data, in Junos OS XML format, for the BGP resource.

```

<% @bgp.each do | name, hash | %>
  <protocols>
    <bgp>
      <group>
        <name><%=name%></name>
        <type><%= hash['type']%></type>
        <local-address><%= hash['local-address']%></local-address>
        <% hash['neighbor'].each do | neighbor | %>
          <neighbor>
            <name><%=neighbor%></name>
          </neighbor>
        <% end %>
        <peer-as><%= hash['peer-as']%></peer-as>
      </group>
    </bgp>
  </protocols>
<% end %>

```

Creating the Attributes for the Template

Step-by-Step Procedure

To create and save the attributes for the template:

- Add the BGP attributes to the end of the end of the **default.rb** file:

```
root@chef-client]# cat netdev/attributes/default.rb
default[:netdev][:bgp] = {
  'internal' => {
    'type' => 'internal',
    'neighbor' => [ '10.10.10.10', '10.10.10.11' ],
    'local-address' => '20.20.20.20',
    'peer-as' => '100'
  },
  'external' => {
    'type' => 'external',
    'neighbor' => [ '30.30.10.10', '30.30.10.11' ],
    'local-address' => '20.20.20.20',
    'peer-as' => '200'
  }
}
```

Configuring the Device by Using Chef for Junos OS

Step-by-Step Procedure

To configure the device by using Chef for Junos OS:

1. Upload the netdev_device cookbook to the Chef server.

```
$ knife cookbook upload netdev_device
```

2. Edit the node object that represents the device.

```
$ knife node edit device_node_name
```

Knife starts your editor and opens a JSON file that contains the node attributes.

3. Enter the recipe in the **run-list** attribute and then save the JSON file.

```
{
  "name": "device_node_name",
  "chef_environment": "_default",
  "normal": {
  },
  "run_list": [
    "recipe[netdev_group:bgp_create]",
  ]
}
```

The order in which you enter the recipes matters. The last configuration overrides any previous configuration.

4. Log in as the root user.

5. From the UNIX-level shell, run the Chef client.

- If the Juniper Networks version of the Chef client is 2.x (for example, Chef client version 11.10.4_2.0), enter:

```
%/opt/jet/chef/bin/ruby /opt/jet/chef/bin/chef-client -c /var/db/chef/client.rb
```

- If the Juniper Networks version of the Chef client is 1.x (for example, Chef client version 11.10.4_1.1), enter:

```
%/opt/sdk/chef/bin/ruby /opt/sdk/chef/bin/chef-client -c
/var/db/chef/client.rb
```

The Chef client displays status messages during its run to indicate its progress in performing the configuration.

Verification

To verify that the commit was successful and the configuration reflects the new BGP resource, perform these tasks:

- [Verifying the Commit on page 41](#)
- [Verifying the Configuration on page 41](#)

Verifying the Commit

Meaning The **JUNOS: OK: COMMIT success!** message and the commit log indicate that the Chef client successfully applied the configuration changes generated by the template.

Verifying the Configuration

Purpose Verify that the BGP configuration group is in the active configuration on the device and that the configuration group name is configured for the **apply-groups** statement.

Action From operational mode, enter the **show configuration groups bgp_group** and the **show configuration apply-groups** commands.

```
chef@chef-client> show configuration groups bgp_group
```

```
protocols {
  bgp {
    group internal {
      type internal;
      local-address 20.20.20.20;
      peer-as 100;
      neighbor 10.10.10.10;
      neighbor 10.10.10.11;
    }
    group external {
      type external;
      local-address 20.20.20.20;
      peer-as 200;
      neighbor 30.30.10.10;
      neighbor 30.30.10.11;
    }
  }
}
```

```
chef@chef-client> show configuration apply-groups
apply-groups [ global re0 re1 bgp_group ];
```

Meaning The output shows that the BGP configuration was successfully configured in the **groups** hierarchy and that **bgp_group** was added to the **apply-groups** hierarchy.

- Related Documentation**
- [Example: Using Chef for Junos OS to Configure Ethernet Switching on MX Series Routers on page 28](#)
 - [Example: Using Chef for Junos OS to Configure Ethernet Switching on EX Series, OCX Series, and QFX Series Switches on page 19](#)
 - *groups*

PART 4

Index

- [Index on page 45](#)

Index

Symbols

#, comments in configuration statements.....	xi
(), in syntax descriptions.....	xi
< >, in syntax descriptions.....	xi
[], in configuration statements.....	xi
{ }, in configuration statements.....	xi
(pipe), in syntax descriptions.....	xi

B

braces, in configuration statements.....	xi
brackets	
angle, in syntax descriptions.....	xi
square, in configuration statements.....	xi

C

Chef resources	
apply_group.....	37
comments, in configuration statements.....	xi
conventions	
text and syntax.....	x
curly braces, in configuration statements.....	xi
customer support.....	xii
contacting JTAC.....	xii

D

documentation	
comments on.....	xi

E

ERB templates.....	37
--------------------	----

F

font conventions.....	x
-----------------------	---

M

manuals	
comments on.....	xi

N

netdev_group	
configuring devices.....	37

P

parentheses, in syntax descriptions.....	xi
------------------------------------------	----

R

resources	
netdev_group.....	37

S

support, technical See technical support	
syntax conventions.....	x

T

technical support	
contacting JTAC.....	xii

