ICOS x86 Platform
Administrator’s Guide

ICOS x86 Platform Administrator’s Guide

Table of Contents

List of Tables

Chapter 1. About This Document

1.1. Purpose and Audience

This document describes how ICOS OS operates as a Linux service on x86-based switches, in contrast to how it operates as an embedded operating system on other CPU architectures. It is intended to help network administrators learn about features specific to the x86 ICOS OS platform and to decide whether the x86 platform or an embedded platform is a the best solution for their planned network deployment.

1.2. Document Conventions

The following conventions may be used in this document:

Convention	Description
Bold	User input and actions: for example, type **exit**, click **OK**, press **Alt+C**
Monospace	Code: *#include <iostream>* HTML: *<td rowspan = 3>* Command line commands and parameters: *wl [-l] <command>*
< >	Placeholders for required elements: enter your <username> or *wl <command>*
`[ ]`	Indicates *optional* command-line parameters: *wl [-l]* Indicates bit and byte ranges (inclusive): [0:3] or [7:0]

1.3. Technical Support

Netberg provides customer access to a wide range of information, including software updates through its customer support support@netbergtw.com [mailto:support@netbergtw.com]

In addition, Netberg provides other product support through its Downloads and Support site http://netbergtw.com/top-support/

Chapter 2. ICOS OS x86 Platform

ICOS OS software supports several new platforms based on the Intel x86 processor family. In response to the industry movement toward open and standards-based networking equipment, the x86 platforms take a different approach to how ICOS OS software runs on the switches and which features the software provides.

Traditionally, on PowerPC, MIPS, and ARM-based platforms, ICOS OS software owns the switch. These platforms are referred to as embedded platforms. On embedded platforms, network administrators have limited ability to add and run their own software, and are limited to using the supported ICOS OS tools for managing switch configuration and software.

On x86 platforms, however, the software is handled like an x86-based server. Network administrators can install pre-built software packages on the switches and manage them in a way similar to how x86 servers are managed. For example, an administrator may install Puppet or Chef to manage configuration and software packages on the switch. On x86 platforms, ICOS OS operates as one of the Linux applications and is installed and configured by tools such as Puppet.

As of the current release, this “ICOS OS-as-an-application” approach is supported only on the x86 CPU architecture, as an established Linux ecosystem exists only for the x86 processor family. For brevity, this document refers to Linux-based ICOS OS-as-an-application platforms as “x86” platforms and other CPU architectures as “embedded” platforms.

This section is divided into the following subsections.

- Section 2.1, “ICOS OS for x86 Features Overview” provides a list of the feature differences between the x86 and the embedded platforms.
- Section 2.2, “ICOS OS for x86 Network Deployment Considerations” discusses how the x86 switches are typically deployed in the network. The physical topology and management strategy for x86 platforms is different than embedded platforms.
- Section 2.3, “ICOS OS for x86 Detailed Feature Differences Description” offers an in-depth description of the feature differences between x86 and embedded platforms.

For more detailed information on deploying ICOS OS on x86 systems, see Chapter 3, *ICOS OS x86 Deployment*

2.1. ICOS OS for x86 Features Overview

ICOS OS for x86 and ICOS OS for embedded platforms support largely the same feature set. The differences mainly exist in features that administrators are expected to implement using utilities available in the standard Linux distribution.

This section lists the features that are different between the x86 platforms and the embedded platforms.

2.1.1. ICOS OS for x86 Feature Matrix

Table below summarizes the feature differences between ICOS OS on x86 platforms vs. embedded platforms. ICOS OS features not listed in this table work the same way on both platform types.

Table 2.1. ICOS Feature Matrix

Feature Overview	x86 Platform	Embedded Platform
Linux Distribution	Ubuntu 14.04 LTS with special packages.	Custom embedded Linux distribution.
Open Source Linux Packages	Binary-compatible packages can be downloaded from the Ubuntu mirror sites. Vendor provides a Debian package file for the customized kernel.	Not Supported.
Zero-Touch Provisioning	Loads a script specified via DHCP option 239. Executes commands in the script to install additional packages and provision the switch.	Able to automatically load a configuration file and upgrade the OS image.
ONIE Code Installation	ONIE is required for installing the initial image.	ONIE is optional.
Dual ICOS Images.	Not supported. The switch has one file system with ICOS and one file system with the ONIE image. If the ICOS file system is corrupted, the recovery must be done with ONIE.	Dual ICOS images are supported. Automatic switch-over between images is supported if one image is corrupted.
Code and Configuration Management with Puppet.	Puppet is preinstalled in the ONIE ICOS image. The user is responsible for configuring Puppet and for creating the Puppet manifests.	Supported on some embedded platforms. The embedded switches support loading Puppet and other Linux packages during the Zero-Touch Provisioning procedure. This feature should be used only in conjunction with ONIE.
File Management	Loading files, such as configuration, scripts, and code images is done using standard Linux utilities from the Linux shell or from tools such as Puppet. Any file transfer mode supported by Linux can be used to download files. The ICOS copy command is not supported.	Files are managed using the copy command from the ICOS command prompt. The copy command supports TFTP, FTP, SCP, and SFTP.
Configuration File Compression	Configuration files and scripts are stored uncompressed in the file system. This enables the user to view and edit the files.	To conserve NVRAM, the configuration and script files are compressed. The copy com-

Feature Overview	x86 Platform	Embedded Platform
		mand must be used to offload the uncompressed version of the file to an external file server.
User Login Management	The recommended approach is to use native Linux user authentication methods, and then use commands (ICOS-console or ICOS-cli) to manage ICOS from the Linux shell. ICOS authentication methods are also available and can be optionally enabled.	User authentication is controlled by ICOS. ICOS supports statically configured user IDs and passwords, RADIUS, and TACACS+.
Domain Name Server (DNS) Client	The x86 platforms use the Linux DNS client. Administrators must configure the Linux DNS client outside of ICOS. ICOS for x86 does not include the internal DNS client, but instead uses the libresolv library to resolve the DNS names.	The DNS client is embedded in ICOS and is configured only using the ICOS CLI commands.
Time of Day	The x86 platforms rely on the Linux NTP to manage the time zone and the time of day. The NTP is configured outside of ICOS. ICOS for x86 does not include the internal SNTP client.	ICOS uses an internal SNTP client and provides CLI commands for managing the SNTP client configuration and status.
Syslog	The x86 platforms send syslog events to the Linux System Logger. The recommended approach is to configure the Linux syslog client outside of ICOS. This allows all software applications running on the switch to generate the syslog events from the same client. The ICOS syslog client is also available and may be used instead of the Linux syslog client.	The ICOS supports only the internal syslog client. The Linux System Logger is not supported.
Ethernet Service Port	The x86 platforms provide read-only access within ICOS to view the service port IP address. The Ethernet service port IP address assignment is controlled by Linux.	ICOS provides CLI commands to enable DHCP or statically assign the IP address to the Ethernet service port.
Host name assignment	On the x86 platforms ICOS reads the host name from Linux and uses the name in the CLI prompt. The ICOS CLI command to set the host name is supported, but it only sets the name in Linux using *sethostname()* and does not affect the ICOS configuration file.	ICOS provides a CLI command to set the host name. The host name is saved in the ICOS configuration file.
Switch Database Management (SDM) Template – Used for configuring routing scaling factors	On x86 platforms, the SDM template can be set via an ICOS command and by writing the template number into the file */mnt/fastpath/sdm-templatetext.cfg*. The file mechanism enables the administrator to avoid an extra ICOS restart by setting the template in the ZTP script before ICOS starts.	The SDM template is set via the ICOS CLI commands and the switch is rebooted for the new template to take effect.

Feature Overview	x86 Platform	Embedded Platform
Interface Management via Linux Interfaces	The x86 platforms support the ability to monitor and manage physical ports, LAGs, and Routing interfaces via Linux tools such as ethtool and iproute2 collection. The same operations are available through the ICOS user interface as well.	Not supported. ICOS physical ports, LAGs, and routing interfaces are managed through the ICOS user interface.
Packet transmit/receive via Linux interfaces	The x86 platforms enable applications to send and receive packets on physical ports and routing interfaces via Linux interfaces. This feature enables tools like Wireshark or LLDPD to run on the switch.	Not Supported.
Route Table Management via Linux Commands	The x86 platforms enable third-party applications to add IPv4 and IPv6 routes using standard utilities, such as iproute2 or directly via the NETLINK socket. This feature enables third-party routing protocols, such as Quagga, to run on the switch. Not all routing features are supported via this mechanism. For example, the VRF feature is not supported.	Not Supported.

2.2. ICOS OS for x86 Network Deployment Considerations

This section discusses key concepts to consider before deploying the network using ICOS OS x86 platforms.

2.2.1. Physical Topology

Managing switches in a way similar to how Linux servers are managed offers significant advantages in some environments, by enabling network administrators to use the same tools, such as Chef or Puppet, to manage software packages and switch configuration, and to run any Linux software on the switches.

However, there are some disadvantages as well to the “Linux server” approach. The key issue is that the tools designed to manage a Linux server assume that it is always attached to the network. With switches, the ICOS OS software provides the network service, so until the ICOS OS software is started and configured, the switch does not have in-band connectivity to the rest of the network.

To enable the standard Linux management tools to work on switches, ICOS OS x86 switches must be connected to an out-of-band management network via the Ethernet service port. Some network administrators already deploy an independent out-of-band management network, but those who manage switches exclusively in-band would need to change their approach.

Once the ICOS OS software is installed and configured, the switch can be managed in-band. However subsequent code upgrades require an out-of-band management network to do a complete software reinstall or recover from issues with ICOS OS startup.

2.2.2. Service Port Default Gateway

One complication with using the service port for management is handling of the default gateway. When the switch is powered on, the DHCP client obtains the IP address and the default gateway from the DHCP server and initializes the service port with these addresses. The switch can reach any IP address via the service port using the default gateway. However, if a routing protocol is configured on the switch and it creates a default route toward the network, this default route takes precedence, making it difficult for applications running on the switch to access devices on the remote subnets attached to the service port.

Many applications support an explicit mechanism to bind to a specific interface. For example, the command *ping -I eth0 10.0.0.1* forces the ping command to use the default gateway on the service port. Unfortunately, some Linux utilities, such as the Puppet client, do not have the ability to bind to a specific IP address, so they cannot access devices on remote subnets via the service port while a default route is installed via a routing protocol.

To resolve this issue, the administrator should add explicit routes to the kernel routing table during switch provisioning for the subnets that need to be accessed by the switch via the service port.

2.2.3. ONIE

The Open Network Install Environment (ONIE) is the industry-standard mechanism for installing a Network OS on bare-metal switches. Information about ONIE is available on www.onie.org.

The x86 switches come from the factory with the ONIE software installed as the active boot image. After ONIE boots, it searches for an HTTP server containing an image file for the switch. Every switch model uses a unique image file name.

The network administrator is required to set up an HTTP server accessible from the management network and put the ICOS OS ONIE installer for the target platform on the HTTP server.

Once ONIE installs the switch image, it activates the image in the x86 boot code, and reboots the switch. The x86 switches use the GRUB boot code.

When ICOS OS is running, the administrator can boot back into ONIE by using the icos-mgmt application, or by selecting the ONIE partition during switch boot-up from the console at the GRUB prompt.

ONIE takes from 5 to 10 minutes to erase the ICOS OS partition and program the new ICOS OS image. This means that using ONIE in production disrupts traffic for a relatively long time when the top-of-rack (ToR) switch needs to be updated. The Spine switches in a typical CLOS topology have redundancy, so do not cause traffic forwarding issues as long as a sufficient number of Spine switches remain in service.

More typically, ICOS OS can be updated by simply loading a new ICOS OS Debian package, which takes the system down for a less than 1 minute. Other system components, such as the kernel, can also be updated with a Debian package, but require a system reboot to take effect.

The ONIE download is needed if the file system becomes corrupted or an ICOS OS update requires a new version of Ubuntu.

The network administrators may need to plan on moving workloads away from the servers attached to the ToR switch that is being upgraded with ONIE.

2.2.4. Zero-Touch Provisioning and Software Management

After downloading the ICOS OS image with ONIE, the network administrator needs to customize and configure the switch. The customization may include installing additional software packages on the switch.

The ICOS OS switch uses DHCP option 239 for specifying the provisioning script file. The file must be stored on an HTTP server, and DHCP option 239 specifies the URL of the provisioning script file.

ICOS OS executes the commands in the provisioning script using the Linux shell. The provisioning script may contain commands to install additional software and commands to configure the switch.

Optionally, the network administrator can load tools such as Puppet or Chef to manage the switch software and configuration. The Puppet agent is already preinstalled in the ICOS OS image, but Chef needs to be installed separately.

The Puppet/Chef tools are not required in order to run ICOS OS. The network administrator can simply download an ICOS OS configuration file *startup-config* and copy it to the */mnt/fastpath* directory.

After the provisioning script is done, ICOS OS modifies its configuration to not invoke the provisioning script on the next reboot, and reboots the switch. The ICOS OS service is then started automatically on every reboot.

2.2.5. Login User ID and Password Management

The recommended approach for managing user access to the x86 switches is to use the native Linux utilities to control the user IDs, passwords, and SSL/SSH security certificates.

For example, if the network uses LDAP to manage user access, the administrator can install and configure an LDAP client on the switch using the provisioning script and install security certificates for SSH. In this setup, when the user logs into the switch, the Linux LDAP client authenticates the user ID and password and logs the user into the Linux shell.

To manage ICOS OS, the user must have “sudo” privileges.

If the network administrator does not install and configure any user access, the default login name is **admin** and the default password is **admin**. The same password is used for the sudo password.

The ICOS OS user interface can be accessed using the *icos-console* and the *icos-cli* programs. The *icos-console* script emulates serial port access. This script is helpful if the user wants to see debug messages that were generated by the ICOS OS process as it started. These messages are queued to the serial console. Only one *icos-console* session can be used at a time.

The *icos-cli* program enables multiple users to manage the switch concurrently. The users can simply start new SSH sessions with the switch and then invoke the *icos-cli* program to access the ICOS OS command prompt.

The *icos-cli* and *icos-console* utilities do not require a user ID or password when accessing the ICOS OS prompt. Users are automatically placed into the privileged mode and can execute any ICOS OS command.

The **<Ctrl> + z** character combination exits the *icos-cli* and *icos-console* programs. Restarting the ICOS OS serviced also exits the *icos-cli* and *icos-console* programs.

The x86 platforms retain the ability to manage users in ICOS OS. For example, the network administrator can configure the ICOS OS SSH server and RADIUS/TACACS+ user authentication. This feature might be helpful if the network has a mix of embedded and x86 switches. In this scenario, the ICOS OS SSH server needs to run on a different IP port number from the Linux SSH server.

2.3. ICOS OS for x86 Detailed Feature Differences Description

The following sections provide additional details on some of the key features supported by the x86 platforms.

2.3.1. Linux Distribution

The x86 platforms use the Ubuntu 14.04 distribution. This is a long term support (LTS) package released in April 2014 and is expected to be supported by Ubuntu until April 2019.

The Ubuntu 14.04 is based on kernel version 3.16.0-29. Netberg does provide the Debian packages to support the platform.

The x86 platforms use the 64-bit Linux distribution and are binary compatible with the pre-built packages available for the Ubuntu 14.04. These packages can be installed on the x86 switch using the *apt-get* utility.

If the administrator needs to develop C/C++ code for the switch, then this code should be compiled on an Ubuntu 14.04 server. The administrator can even install the GCC compiler on the x86 switch and build the code natively on the switch.

The ONIE install package for ICOS OS is approximately 200 MB and consumes 2.5 GB of disk space once it is installed on the x86 platform. ICOS allocates all available space ICOS OS by default, so over 50 GB is available for additional packages.

Administrators can modify the ONIE install script to change the default partition size if needed. Our x86 switches are shipped with 64GB of disk space, so a smaller primary partition or a separate secondary partition can be created to accommodate additional user software.

After the initial ONIE installation, the administrator can update ICOS OS using the ICOS OS Debian package. The package is about 30MB, and only requires stopping and starting the ICOS OS service.

In contrast to the embedded platforms, where Linux kernel, root file system, and the ICOS OS application is included in the download image, the x86 platforms use separate Debian packages to update various parts of the system. If ICOS OS moves to a new Ubuntu distribution, such as 16.04, then a new ONIE installation is required.

2.3.2. Switch Database Management (SDM)

The SDM templates control the maximum number of IPv4/IPv6 routes, and whether the MPLS feature is enabled or disabled.

The SDM template is configured using the *config/ sdm prefer* command, and the ICOS OS is restarted in order for the new SDM template to take effect.

To reduce the number of times the ICOS OS service must be restarted, the x86 platforms provide a mechanism to set the SDM template by writing a template identifier to the */mnt/fastpath/sdm-template-text.cfg* file.

For example, to change the SDM template, use the command:

```
echo 9 > /mnt/fastpath/sdm-template-text.cfg.
```

The following SDM templates are supported:

1. “Dual Default”
2. “IPv4-Only Default”
3. “IPv4 Data Center”
4. “IPv4 Data Center Plus”
5. “IPv4 and IPv6 Data Center”
6. “IPv4, IPv6, and MPLS Data Center”
7. “IPv4, IPv6 and Data Center VPN”
8. “IPv4 Data Center VPN”

2.3.3. Interface Management via Linux Interfaces

ICOS OS enables the network administrators to monitor and manage physical ports, LAGs, and routing interfaces via standard Linux tools, such as *ip*, *ethtool*, *ifconfig*, and *ifenslave*.

The ICOS OS integration with Linux Networking allows the network administrators to use familiar Linux tools to monitor and manage the network.

Since ICOS OS provides more features than supported in Linux Networking, only a subset of ICOS OS interfaces and features can be monitored and managed via the standard Linux tools.

The following sections summarize the interface types and the commands supported on these interfaces.

When interface configuration is changed with the Linux command, the change is reflected in ICOS OS so, if the ICOS OS configuration is later saved, the change becomes permanent.

2.3.4. Physical Ports

The Linux device interfaces corresponding to physical ports are named fpti1_0_<n>. The <n> is the front panel port number. For example, the ICOS OS port 0/47 corresponds to the Linux interface fpti1_0_47.

The following commands can be used to display and configure physical ports:

- *ethtool -S <dev>* reports the counters for the port.
- *ethtool <dev>* reports port link status and mode.
- *ifconfig <dev>* down puts the port in administrative shutdown mode.

2.3.5. Link Aggregation Groups

The ICOS OS LAG interfaces are named in Linux using the “bond<n>” naming convention, where <n> is the LAG number. Therefore, ICOS OS “lag1” corresponds to the Linux “bond1” interface.

In addition to managing LAGs with the *ethtool* and the *ifconfig* commands, the network administrator can manipulate the LAG membership via the Linux *ifenslave* or *ip link* commands.

2.3.6. Routing Interfaces

ICOS OS has several types of routing interfaces. The following interfaces can be managed via Linux:

- Port-based routing interfaces. These interfaces are named rt1_0_<n>, where <n> is the port number.
- VLAN routing interfaces. These interfaces are named rt_v<n>, where <n> is the VLAN ID. For example, the interface for VLAN 100 is rt_v100.
- Loopback interfaces. These interfaces are named loopback_<n>, where <n> is the loopback interface number.

ICOS OS supports viewing status and changing the IP address on these routing interfaces using the *ip* command.

2.3.7. Unsupported Interfaces

ICOS OS does not allow Virtual Router (VRF) interfaces to be managed via Linux. Also, the tunneling interfaces cannot be managed via Linux.

2.3.8. Packet TX/RX via Linux Interfaces

ICOS OS enables the applications running as processes on the switch to send and receive traffic on the ICOS OS physical ports, LAGs, and routing interfaces.

This feature can be used to run network monitoring tools such as Wireshark and network protocols such as LLDPD.

By default, switch transit traffic is forwarded in the hardware, so monitoring tools can only see packets that are destined to the switch CPU and packets transmitted by the switch CPU. It is possible to redirect all received traffic on a port to the CPU or capture certain protocols by setting up an access list, but this may affect protocol operation.

2.3.9. Route Table Management via Linux

ICOS OS on the x86 platforms supports synchronization between the kernel routing table and the hardware. This feature enables the standard Linux tools, such as ip route, to add routes to the hardware. The feature also enables network administrators to run third-party protocols stacks, such as Quagga on the x86 switch.

The ICOS OS supports synchronizing IPv4 and IPv6 unicast routes, including ECMP. ICOS OS handles routes to ICOS OS IP routing interfaces only. For example, routes to the service port are not added to the hardware.

ICOS OS does not support source routing, so Linux routes specified with the source option are not added to the hardware.

ICOS OS supports synchronizing routes only from the default network space. In other words, even when ICOS OS is configured for multiple VRFs, the routes within the non-default VRF cannot be synchronized with the hardware using the standard Linux commands.

The kernel routes are treated as highest priority routes in ICOS OS. Therefore, if the same route is learned by another protocol in ICOS OS, it is withdrawn from the hardware in favor of the kernel route.

BGP supports kernel route redistribution. This feature enables the BGP protocol to export the kernel routes to the rest of the network.

Chapter 3. ICOS OS x86 Deployment

By default, we deploy ICOS OS in-house. However, Netberg may deliver the x86 switches with the ONIE code loader only. The network administrator is responsible for loading the Linux distribution on the switch, loading additional Linux packages, configuring the Linux services, and configuring and managing ICOS OS. The previous section provides an overview of what needs to be done to deploy an x86-based ICOS OS switch in the network.

The detailed description of ONIE and the detailed instructions on how to configure the various Linux services and tools such as User Authentication, NTP, and Puppet are outside the scope of this document. However, this section provide examples to help the administrators install and configure the Linux services on an x86 switch and it provides pointers to various documentation sources for further learning.

Once the Linux services are configured on the x86 switches, the ICOS OS software can be managed via the ICOS OS user interface. Additionally, the x86 platforms integrate with Linux Networking to allow interface monitoring and limited interface management via the standard Linux tools.

This section contains the following subsections:

3.1. Switch Setup and Configuration

This section describes how to take a white-box switch shipped from the switch manufacturer and deploy the switch in the network with ICOS OS.

The following topics are covered in this section:

- Section 3.1.1, “Working with ONIE” — Basic ONIE introduction.
- Section 3.1.2, “ICOS OS Installation” - Explains how to load ICOS OS image with ONIE.
- Section 3.1.2.1, “ICOS OS update” - Explains how to update existing ICOS OS.
- Section 3.1.2.4, “ICOS OS Image Management” - Explaind ONIE menu options.
- Section 3.1.2.5, “icos-mgmt” - Explains how to select the boot menu option that the switch selects after a reboot.
- Section 3.1.3, “Configuring Linux Services” — Explains how to set up the Linux OS on the switch with the minimal features required to run ICOS OS.
- Section 3.1.4, “Setting Up ICOS OS” — Explains how to set up ICOS OS configuration to start using the switch for data forwarding.

3.1.1. Working with ONIE

White box switches are shipped to users factory-programmed with a supported ONIE boot image. When the switch boots up the first time, it loads ONIE and initiates a discovery process that helps install an Networking Operating System image such as ICOS OS.

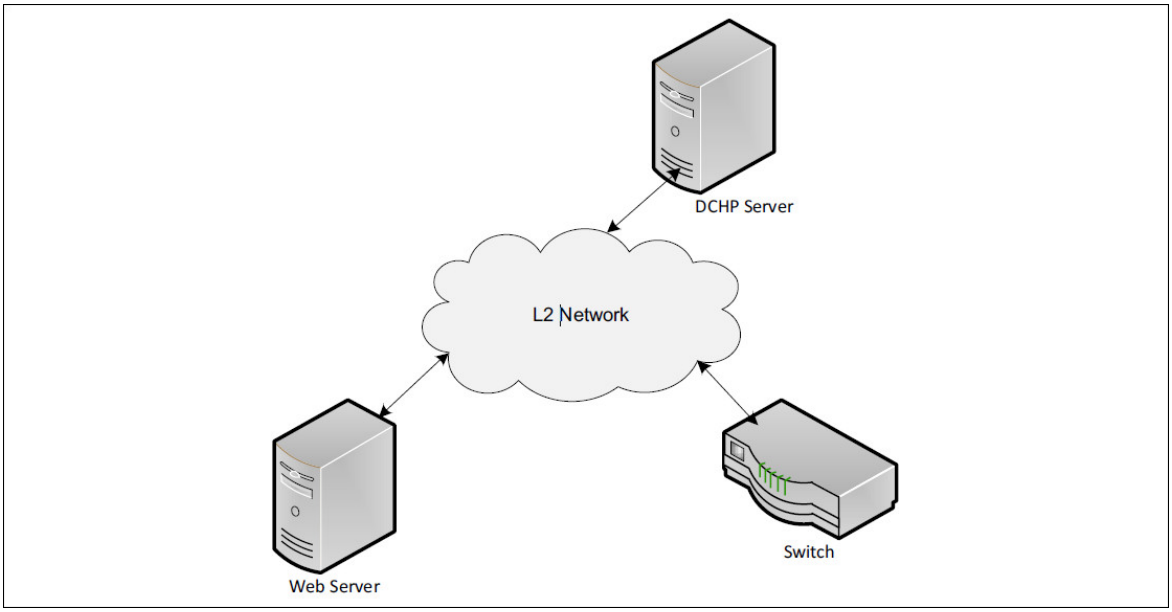
For more information about ONIE and its features, refer to the ONIE project site: https://github.com/opencomputeproject/onie/wiki.

The following sections describe one networking install method — an ICOS OS image is installed using DHCP options and a web server connected to the same network subnet as the switch service port.

3.1.2. ICOS OS Installation

In this example, a DHCP server, target switch, and web server are all located on the same layer-2 network (see the figure below). The DHCP server provides the following information to the target switch:

- The switch IP address, from a pool of IP addresses
- The URL of the ICOS OS installer image residing on the web server



The following is an example data from a *dhcpd.conf* file for an ISC-DHCP server.

```
subnet 10.12.1.0 netmask 255.255.255.0 {
range 10.12.1.10.20 10.12.1.200;
option default-url = "http://10.12.1.2/onie-installer-x86_64-nba720_
rangeley";
}
```

ONIE is assigned an IP address from specified range. Then, it attempts to download the ICOS OS installer image file (in this example, onie-installer-x86_64-nba720_rangeley) from the specified URL. After it is downloaded, ONIE extracts and installs the image on the switch. This image is an ICOS OS ONIE installer image for a Netberg Aurora 720 switch. After a successful installation, the switch reboots to display the ICOS OS Linux login prompt. If the installation fails, the switch boots into ONIE.

It is important not to interrupt the ONIE installation procedure and not to power-cycle the switch during this procedure. As part of the installation procedure, the Master Boot Record (MBR) of the switch storage device is modified to boot into the ICOS OS partition. These MBR updates take about 2–6 seconds and occur at the start and end of the installation procedure. Disruption of the installation procedure outside of these MBR updates has no effect on the switch and, on the next reboot, it will drop back into the ONIE boot loader where the installation can be attempted again. Disruption during the MBR updates can result in a corrupted GRUB bootloader configuration. If such an event occurs, the switch boots into a GRUB rescue mode.

The following command at the GRUB prompt can recover the switch to display the ONIE boot menu:

```
grub> configfile (hd0,gpt2)/grub/grub.cfg
```

To repair the corrupted MBR, select **ONIE: Rescue mode**, and wait for the ONIE prompt. At the ONIE prompt, the following commands can be used to install a fresh copy of the bootloader onto the MBR:

```
ONIE:/ # chattr -i /mnt/onie-boot/grub/i386-pc/core.img
ONIE:/ # grub-install --force --boot-directory=/mnt/onie-boot /dev/sda
ONIE:/ # grub-install --boot-directory=/mnt/onie-boot --recheck /dev/sda2
ONIE:/ # chattr +i /mnt/onie-boot/grub/i386-pc/core.img
```

After installation, the switch can reboot into ONIE and proceed with installing an ICOS OS image.

If the switch does not boot to ONIE, to an installed ICOS OS image, or to the GRUB rescue prompt, then refer to switch manufacturer documentation for steps to reinstall ONIE or recover the switch. Most switches have a USB port that can be used to recover the switch and some manufactures also provide a network boot option.

The ICOS OS OS can also be installed using the *onie-nos-install* command from the ONIE prompt. The following are examples of downloading an image from a web server or TFTP server. To learn more about other supported installation methods, refer to the ONIE documentation.

```
ONIE:/ # onie-nos-install http://10.12.1.2/onie-installer-x86_64-nba720_rangeley
```

Or:

```
ONIE:/ # onie-nos-install tftp://10.12.1.2/onie-installer-x86_64-nba720_rangeley
```

Like the ICOS OS OS image, the ONIE image on the switch can be upgraded using the *onie-self-update* command from the ONIE prompt.

```
ONIE:/ # onie-self-update http://10.12.1.2/onie-updater-x86_64-nba720_rangeley-r0:
```

Or:

```
ONIE:/ # onie-self-update tftp://10.12.1.2/onie-updater-x86_64-nba720_rangeley-r0
```

3.1.2.1. ICOS OS update

To upgrade the ICOS package:

1. From the Linux shell, download the image into the switch.
2. After the image is downloaded, install it into the system.
3. Delete the download file and reboot the system.

Example:

```
Connecting to 172.19.96.169:22... Connection established.
To escape to local shell, press 'Ctrl+Alt+]'.
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.16.0-29-generic x86_64)
* Documentation: https://help.ubuntu.com/
Last login: Tue May 31 08:43:05 2016 from 172.19.96.126 admin@localhost:~$
admin@localhost:~$ sudo -i
[sudo] password for admin:
root@localhost:~#
root@localhost:~# cd /mnt/download/
root@localhost:/mnt/download# tftp 172.19.96.126
tftp> bin
```

```
tftp> g nba720-2.0.6-rc.deb
tftp> quit
root@localhost:/mnt/download# dpkg –i nba720-2.0.6-rc.deb
(Reading database ... 52155 files and directories currently installed.)
Preparing to unpack nba720-2.0.6-rc.deb …
icos stop/waiting
Unpacking icos-fxn-x86 (1.2.0.5.0) over (1.2.0.6.0) ...
Setting up icos-fxn-x86 (1.2.0.6.0) ...
icos start/running, process 32199
Processing triggers for ureadahead (0.100.0-16) ...
Processing triggers for libc-bin (2.19-0ubuntu6) ...
root@localhost:/mnt/download# rm nba720-2.0.6-rc.deb
root@localhost:/mnt/download# reboot
```

3.1.2.2. ICOS OS Operating System — Login

ICOS OS uses a standard Linux distribution such as Ubuntu Server as its base operating system. ICOS OS is distributed as an application package that provides the network protocol stack to realize the capabilities of a white box switch.

ICOS OS uses a single ext4 formatted 59 GB partition. This partition is mounted as the root partition in Ubuntu. Any empty space left on the storage device can be partitioned by the user using gdisk or other partition utilities as required. The installation partition size is fixed and cannot be modified by the user without creating a new ICOS OS installer image.

After a successful ICOS OS image installation using ONIE, the switch reboots and shows the Linux login prompt. ICOS OS switch application starts as a Linux service and initializes the switch silicon to perform layer-2/layer-3 switching functionality.

3.1.2.3. ICOS OS Linux Login

The user name and password for the Linux login prompt is **admin/admin**.

The root account is disabled by default and sudo permissions are enabled for the admin user.

Use the *sudo* command to execute any commands with root permissions. The user can permanently switch to root mode using *sudo -s* or *sudo -i* commands. The root password is **admin**.

3.1.2.4. ICOS OS Image Management

After ICOS OS installation, the GRUB bootloader menu displays two menu options, ICOS OS and ONIE. The boot menu is configured to boot into ICOS OS by default. When this boot menu is displayed during the boot process, the user can select ONIE to perform image administration.

ONIE operates in the following modes:

- Install OS — This is the default ONIE mode which starts the ONIE discovery process to search and install an OS installer image.
- Rescue — This mode is provided to boot into ONIE to debug any issues encountered during ONIE installation or due to corruption of the ICOS OS OS image. The ONIE discovery process is not started by default in this mode.

- Uninstall OS — This mode is provided to erase the storage device and restore the switch to the factory default state.
- Update ONIE — This mode is provided for ONIE to discover and upgrade the existing ONIE image on the switch. Contact the switch manufacturer for the latest ONIE image.

3.1.2.5. icos-mgmt

ICOS OS provides the *icos-mgmt* Linux utility to automatically select the boot menu option that the switch selects after a reboot.

For example, use the *-u* option to set the boot menu option, as in the following example, to enable the Uninstall OS mode. On the next reboot, the switch will automatically uninstall the existing ICOS OS image and boot to ONIE install mode to reinstall a fresh ICOS OS image, as if it is booting for the first time.

```
bash-4.1$ sudo icos-mgmt -u
[sudo] password for admin:
!!!!!!!!!!!!!!!
! WARNING !
!!!!!!!!!!!!!!!
! You are about to remove existing ICOS operating system image.
! All system data will be lost and cannot be recovered.
! Are you sure? [y/N] y
Updating ONIE GRUB configuration to uninstall ICOS OS
Reboot for changes to take effect
```

The following are usage instructions for the *icos-mgmt* command.

Command	icos-mgmt
Description	ICOS OS management utility to perform configuration and image management.

Example:

```
icos-mgmt < -c | -z | -i | -u | -r | -R | -h | -v > [-f]
Command Line Options:
-c Clear saved startup configuration of ICOS service
-z Erase all configuration files to restore ICOS to factory
default configuration
-i Set default GRUB entry to boot ONIE in OS install mode
-r Set default GRUB entry to boot ONIE in rescue mode
-u Set default GRUB entry to boot ONIE in OS uninstall mode
-R Restore default GRUB entry to factory default
-v Display ICOS package information
-f Force mode to perform control command without prompting user for confirmation
-h Help. Prints this text.
```

ICOS OS service must be stopped to execute the command with the -z option.

3.1.3. Configuring Linux Services

3.1.3.1. Install Linux Applications

A Linux distribution requires a rapid, practical, and efficient way to install additional packages as per user requirements. This procedure manages package dependencies automatically and takes care of package configuration files while upgrading the switch. ICOS OS on Ubuntu Linux-based images uses APT, the Advanced Packaging Tool, which comes preinstalled on the switch.

APT maintains a list of packages that are available in the online Ubuntu repositories and enables the user to search and install the required packages.

The first command to execute is *apt-get* update. This is required by the switch to refresh its information about all the available packages in the online repositories. The user can modify file /etc/apt/sources.list to include any privately maintained repositories. The default setting is to download from Ubuntu official repositories for the corresponding Ubuntu release.

```
admin@localhost:~$ sudo apt-get update
Ign http://us.archive.ubuntu.com precise InRelease
Ign http://archive.ubuntu.com precise InRelease
Ign http://security.ubuntu.com precise-security InRelease
Hit http://us.archive.ubuntu.com precise Release.gpg
Hit http://us.archive.ubuntu.com precise Release
Hit http://archive.ubuntu.com precise Release.gpg
Get:1 http://security.ubuntu.com precise-security Release.gpg [198 B]
Hit http://us.archive.ubuntu.com precise/universe amd64 Packages
<snip>
Hit http://security.ubuntu.com precise-security/restricted Translation-en
Get:16 http://security.ubuntu.com precise-security/universe Translation
-en [64.7 kB]
Fetched 1558 kB in 2min 14s (11.6 kB/s)
Reading package lists... Done
admin@localhost:~$
```

Use the *apt-get install <package_name>* command to download and install requested package.

The following is an example of installing Net-SNMP agent software on the switch.

```
admin@localhost:~$ sudo apt-get install snmpd
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  snmpd
0 upgraded, 1 newly installed, 0 to remove and 50 not upgraded.
Need to get 76.0 kB of archives.
After this operation, 267 kB of additional disk space will be used.
Get:1 http://security.ubuntu.com/ubuntu/ precise-security/main snmpd
amd64 5.4.3~dfsg-2.4ubuntu1.2 [76.0 kB]
Fetched 76.0 kB in 1s (67.5 kB/s)
Preconfiguring packages ...
```

```
Selecting previously unselected package snmpd.
(Reading database ... 49238 files and directories currently installed.)
Unpacking snmpd (from .../snmpd_5.4.3~dfsg-2.4ubuntu1.2_amd64.deb) ...
Processing triggers for ureadahead ...
Processing triggers for man-db ...
Setting up snmpd (5.4.3~dfsg-2.4ubuntu1.2) ...
update-rc.d: warning: snmpd stop runlevel arguments (1) do not match
LSB Default-Stop values (0 1 6)
* Starting network management services:
admin@localhost:~$
```

The ICOS OS image comes with a variety of utilities that help the user manage the switch. Use the *dpkg -l* command to obtain the complete list of packages installed on the switch.

The user can opt to remove some of these packages. Use the *apt-get purge <package_name>* command to remove any installed packages. Note that some packages have interdependencies with others and cannot be removed unless all the dependent packages are removed. Be sure to follow instructions and warnings provided by APT commands to safely install or remove application packages. For example, various Ruby libraries are installed along with Puppet. These are also used by ICOS OS, so when Puppet is uninstalled, these packages should not be removed.

For example, Puppet, the popular configuration management utility, comes preinstalled with ICOS OS. If this utility is not preferred, or a different version of Puppet is needed, then this package can be removed.

```
admin@localhost:~$ sudo apt-get purge puppet
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer
required:
bind9-host libaugeas-ruby1.8 libaugeas0 libdns81 augeas-lenses libruby
libisccc80 ruby1.8 libshadow-ruby1.8 ruby liblwres80 libreadline5 facter
libbind9-80 geoip-database libgeoip1 libisccfg82 puppet-common libruby1.8
libisc83 dmidecode
Use 'apt-get autoremove' to remove them.
The following packages will be REMOVED:
puppet*
0 upgraded, 0 newly installed, 1 to remove and 50 not upgraded.
After this operation, 345 kB disk space will be freed.
Do you want to continue [Y/n]? y
(Reading database ... 49200 files and directories currently installed.)
Removing puppet ...
* Stopping puppet agent
...done.
Purging configuration files for puppet ...
Processing triggers for ureadahead ...
Processing triggers for man-db ...
admin@localhost:~$
```

More information on the capabilities of APT is available in the Ubuntu online help at: https://help.ubuntu.com/community/AptGet/Howto.

3.1.3.2. Essential Linux Services

This section describes some of the basic services used by the ICOS OS application.

User Accounts

ICOS OS creates a default user with name **admin** and password as **admin**.

To create additional Linux users, use the *useradd* utility. The following example creates a new user named john:

```
sudo useradd -s /bin/bash -m john
```

Use passwd to set password for the newly added user, john:

```
sudo passwd john
```

To make user john part of administrative user group, use following command:

```
sudo adduser john sudo
```

Now user john is able to login to the switch and has administrative rights to manage the switch. Note that the root user account is disabled by default. Use *sudo* as a prefix to perform actions that require root permissions.

Users created using the *useradd* command are local to the switch. In a large setup, it is recommended that a centralized login mechanism be used for efficient and effective user administration. Utilities such as LDAP or NIS can be used to allow users to log in on multiple computers using a single password. Refer to the documentation for these utilities for installation and configuration steps.

Host Name

Every Linux device connected to a network is identified by a unique host name. Use the hostname command to view and set the host name.

The default host name set in ICOS OS is **localhost**. To view the current host name, use the following command:

```
admin@localhost:~$ hostname
localhost
```

To change the host name to a different value, use the command *hostname <new-value>*:

```
admin@localhost:~$sudo hostname changeme
admin@localhost:~$ hostname
changeme
```

To ensure that a newly assigned host name is persistent across reboots, set the new value in the file */etc/hostname*.

The host name can be changed from ICOS OS and the new value is updated in the */etc/hostname* file automatically.

Syslog

Syslog is a standard message recording mechanism available in Linux. Log messages generated by various applications in Linux, the kernel, and the ICOS OS application can be viewed in syslog. The rsyslog application daemon is responsible for receiving these messages. All the messages can be viewed in the file */var/log/syslog*.

The rsyslog application can be configured to forward messages to a remote rsyslog server. It can also be configured to filter messages recorded based on the severity of the message. Refer to man pages of *rsyslog.conf* for more information. The configuration file for the rsyslog daemon is */etc/rsyslog.conf*.

For example, when added to */etc/rsyslog.conf*, the following string results in the forwarding of log messages to a remote log server with host name 'log-srv.broadcom.com'. Log messages of the configured switch are visible in */var/log/syslog* of host *log-srv.broadcom.com*.

```
*.* @log-srv.broadcom.com:514
```

Domain Name System (DNS)

In a typical IP-based network, each network device is assigned a numeric IP address. A Domain Name Server (DNS) is typically available in the network and is used to resolve the requested textual domain name to a numeric IP address.

The DNS IP address is configured on devices which need the domain name resolution. The configuration can be viewed in */etc/resolv.conf*. This is an auto-generated file and can be read and modified by multiple applications seeking domain name resolution.

To update configuration in this file, modify */etc/resolvconf/resolv.conf.d/tail* with the intended configuration and use the *resolvconf -u* command to update the */etc/resolv.conf* file.

In the following example, a new name server, 10.13.1.3, is added to the existing list of configured name servers in the */etc/resolv.conf* file.

```
admin@localhost:~$ cat /etc/resolv.conf
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by
resolvconf(8)
# DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
nameserver 10.27.138.20
nameserver 10.27.138.21
search rtp.broadcom.com
root@localhost:~# echo "nameserver 10.13.1.3" >> /etc/resolvconf/resolv.
conf.d/tail
root@localhost:~# resolvconf -u
root@localhost:~# cat /etc/resolv.conf
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by
resolvconf(8)
# DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
nameserver 10.27.138.20
nameserver 10.27.138.21
nameserver 10.27.64.21
search rtp.broadcom.com
nameserver 10.13.1.3
root@localhost:~#
```

Refer to the man pages of *resolv.conf* for more information.

Network Time Protocol Service (NTP)

The Network Time Protocol (NTP) is a networking protocol for clock synchronization between computer systems over packet-switched, variable-latency data networks. NTP is intended to synchronize all participating computers to within a few milliseconds of Coordinated Universal Time (UTC).

Even though most devices are equipped with a real-time clock, their clocks may not be set to the same value as other devices. This might cause a time-sensitive application to detect time variance and break its operation.

The NTP service is a Linux application running on a device which uses the Network Time Protocol to communicate with a remote standard time server and update its current time. Multiple network time servers can be configured to be polled in the configuration file */etc/ntp.conf*.

The following is an example configuration statement that can be added in */etc/ntp.conf* to include a new NTP server.

```
server 0.pool.ntp.org
```

Use the *ntpq* command to view the status of NTP updates received by the NTP daemon. The following is an example command to print a list of the peers known to the server as well as a summary of their state.

```
admin@localhost:~$ ntpq -p
remote           refid           st t when poll reach delay   offset jitter
==============================================================================
+4.53.160.75     64.113.32.5     2  u 1117 1024 336   139.182 0.223  3.894
*pool-test.ntp.o 216.218.192.202 2  u 382  1024 377   76.337 -1.390  1.500
-clock.trit.net  69.36.224.15    2  u 693  1024 373   88.362 -5.643  2.353
+helium.constant 18.26.4.105     2  u 42m  1024 374   146.278 0.297  3.832
+golem.canonical 193.79.237.14   2  u 578  1024 373   227.527 0.340  3.023
admin@localhost:~$
```

For more information, refer to man pages of ntpd and ntp.conf.

3.1.3.3. Zero-Touch Provisioning

One way to use the configuration tools and services described in Section 3.1.3.2, “Essential Linux Services” is to log into the system and make configuration changes or issue appropriate commands. In a large network topology containing hundreds of switches, it is not practical for a user to log in to each switch and configure Linux applications.

Configuration management utilities like Chef and Puppet are used to manage multiple devices using configuration templates. However, these configuration management utilities require some basic configuration on the device to kickstart their process. ICOS OS may not have all the configuration utilities preinstalled or preconfigured.

ICOS OS Zero-Touch Provisioning (ZTP) helps users to automatically perform initial switch configuration when the switch boots for the first time. To use ZTP, the user needs to place the switch in the network and boot the switch with the factory-default configuration. The ZTP process starts automatically over the management network when DHCP information is received by making use of DHCP client hooks.

The following sections describe this process in more detail.

DHCP Server Configuration During the DHCP process over the Service port, the Linux DHCP client requests the following information:

- Host name
- Name server information
- NTP server information
- Log server information
- Provisioning script URL (DHCP-Option 239)

The DHCP client processes the above information and configures: */etc/hostname* with provided host name; */etc/resolv.conf* with provided name server IP address; */etc/ntp.conf* with time server information; and */etc/rsyslog.conf* with remote rysyslog server information.

The DHCP-Option 239 string is used to specify the URL from where the executable provisioning script can be downloaded. ZTP uses the wget utility to download the URL. For ZTP to work, the administrator must configure the DHCP server to offer option 239 to clients. The following sample DHCP configuration provides all the requested information:

```
ddns-update-style none;
option domain-name "qalab.broadcom.com";
option domain-name-servers 192.168.1.2;
option log-servers 10.130.1.2, 10.130.1.150;
option ntp-servers 10.255.13.50, 10.255.13.51;
default-lease-time 7200;
max-lease-time 7200;
log-facility local7;
option provision-url code 239 = string;
subnet 192.168.1.0 netmask 255.255.255.0 {
range 192.168.1.10 192.168.1.100;
option routers 192.168.1.1;
option provision-url "http://192.168.1.20/provisioning_script.sh";
host icos_switch
{
option host-name "dc1-tor-1";
hardware ethernet 00:0c:29:93:78:72;
fixed-address 192.168.1.10;
}
}
```

Provisioning Script

ZTP uses DHCP option 239 to specify a provisioning script URL. The provisioning script is downloaded from the web server using this URL and is executed on the switch. The provisioning script can be used to execute Linux commands and modify Linux application configuration files.

The provisioning script executes only once, and the ZTP process is disabled for subsequent boot operations. To re-enable the ZTP process, modify the value of *AUTO_PROVISION_URL_MODE*

to **TRUE** in */etc/ztp/ztp.cfg*. Note that when *AUTO_PROVISION_URL_MODE=TRUE*, execution of downloaded provisioning script is the only action that is taken as part of ZTP daemon.

The provisioning script can be a simple Shell script. The script must be placed on a web server, and the URL of the script is used as the *provision-url* (DHCP option 239) in DHCP server configuration.

The following is a sample provisioning script that updates the Puppet agent configuration and helps the switch to communicate with the Puppet master to enable subsequent configuration management:

```
#!/bin/sh
# Sample post install script
#
cd /
echo "Provisioning Script Start"
# Updated puppet master address in puppet agent configuration file
sed -i "s/server = puppet-srv/server = openstctl.rtp.broadcom.com/g" \
/etc/puppet/puppet.conf
# Test puppet connection with puppet master
puppet agent --test --waitforcert 10 --verbose -trace
# Start puppet agent service and set it up to always start on bootup
service puppet start
update-rc.d puppet enable
echo "Provisioning Script End"
exit 0
```

Administrators must place a valid script on the web server, as the ZTP process marks that the provisioning is completed even if the script execution fails. Administrators need to monitor the script execution logs in the Syslog server and take appropriate action.

3.1.4. Setting Up ICOS OS

3.1.4.1. Command Line Interface

ICOS OS provides a comprehensive Industry Standard-Command Line Interface (IS-CLI) to allow the administrator to enable and configure the various features supported by the switch. To access the ICOS OS CLI, use the following Linux command:

```
icos-cli
```

This command provides users access to the switch CLI commands to view switch configuration and issue configuration commands. Users can start the *icos-cli* session only after the ICOS OS service has started and has initialized the CLI manager. With the *icos-cli* command the user can initiate a maximum of four parallel CLI sessions. Use logout or **<Ctrl> + z** escape sequence to end the *icos-cli* session.

The following are usage instructions for the *icos-cli* command.

Command	icos-cli
Description	This command opens a connection to ICOS OS and provides CLI access to the user. If ICOS OS service is not running, this command will wait for service to start.

Example:

```
icos-cli [-f<session-id>] [-s] [-h]
Command Line Options:
-f<Session-id>
Detect and close if particular session already open. The command option
also creates a new session.
-s Displays existing sessions.
-h Help. Prints this text.
```

Users can exit out of the command by entering ^z at any point of time.

icos-console

This is an always-on CLI session which the user can use to login and execute commands similar to *icos-cli*.

The standard output of the ICOS OS service is displayed on the *icos-console* session. The *icos-console* command is used when it is necessary to see the ICOS OS application console output that was generated before an *icos-console* session was started. Use the logout command or the **<Ctrl> + z** escape sequence to end the *icos-console* session.

At any given time, only one user can start an *icos-console* session. If a session is busy, a force option (-f) is provided to kill the existing session and start a new session.

The following are usage instructions for the *icos-console* command.

Command	icos-console
Description	This command opens pseudo terminal to ICOS OS and provides CLI access to the user. If ICOS OS service is not running, this command will wait for service to start.

Example:

```
icos-console [-r] [-h]
Command Line Options:
-r Retry to connect to pseudo terminal in the event ICOS restarts or
user tries to exit. User will have to press ^z twice to completely exit.
-f Detect and close if a session is already open
-h Help. Prints this text.
```

Users can exit out of this command by entering ^z at any time. NOTE: If Python is not installed on the system, this command launches a GNU screen session to access the pseudo terminal. Use GNU screen commands to exit.

3.1.4.2. Configuration Setup

ICOS OS allows saving the active configuration of a switch in a clear text file called *startup-config*. This configuration file can be found in */mnt/fastpath* directory. When the ICOS OS service starts, it checks for the presence of this file and, if found, reads the previously saved values and configures the switch accordingly.

Use the *write memory* ICOS OS CLI command to save or overwrite the existing startup-config file.

The user can also directly edit the startup-config file and provide a valid configuration to the ICOS OS service. Updated configuration is applied only after the ICOS OS service is restarted.

icos-cfg

The *icos-cfg* Linux command can be used to apply a clear text configuration file. It also enables reading the current active configuration of the switch and saving it to a file. Creation of a start-up-config file is also possible using this command.

Provide a valid file name after the command option to apply/generate/validate a configuration script. The following are usage instructions for the *icos-cfg* command.

Command	icos-cfg
Description	ICOS OS management utility to perform configuration and image management

Example:

```
icos-cfg [options]
Command Line Options:
-a Apply script: Apply CLI configuration script
-d Debug: Debug mode suppresses output, applicable for "apply", "validate"
and "generate"
-g Generate script: Generate running-config and writes to file
-s Save: Save running-config to startup-config
-t Timeout seconds: Wait for ICOS process in seconds. The default is 30
seconds.
-v Validate script: Validate CLI configuration script
-h Help: Display this message
```

3.1.4.3. Simple Network Management Protocol (SNMP) Agent

ICOS OS has a built-in SNMP agent to allow remote management using an SNMP management station. By default, the ICOS OS SNMP agent listens on standard UDP port 161 for SNMP requests from the management station. The SNMP agent in ICOS OS is provided by SNMP Research and can only manage the ICOS OS MIBs. The agent cannot manage other Linux processes that the user might want to run on the switch.

To manage other applications with SNMP, it is possible to run the Net-SNMP agent alongside the ICOS OS SNMP agent. Care should be taken to configure the two agents to use non-conflicting IP port numbers.

Use the ICOS OS CLI command *snmp-server port <1-65535>* to assign a non-default UDP port for ICOS OS SNMP agent.

To modify the IP port of Net-SNMP agent, modify */etc/snmpd.conf* and specify the appropriate value using *agentaddress udp:<1-65535>*. For more configuration options, refer to the man page of *snmpd.conf*.

Based on the IP port on which the SNMP request is received, the corresponding SNMP agent responds with the requested information.

3.1.4.4. Default Switch Configuration

When the ICOS OS service starts with no startup-config file, it boots up with a factory-defined configuration.

Use the *no shutdown* ICOS OS CLI command in Interface Configuration mode to operationally enable a switch port. The following is a sample configuration that enables ports 0/1 and 0/2.

```
config
interface 0/1
no shutdown
exit
interface 0/2
no shutdown
exit
exit
```

Use *show port* command to see current status of a switch port.

```
(localhost) #configure
(localhost) (Config)#interface 0/1
(localhost) (Interface 0/1)#no shutdown
(localhost) (Interface 0/1)#interface 0/2
(localhost) (Interface 0/2)#no shutdown
(localhost) (Interface 0/2)#exit
(localhost) (Config)#show port all
          Admin     Physical   Physical   Link   Link    LACP   Actor
Intf Type Mode      Mode       Status     Status Trap    Mode   Timeout
---- ---- --------- ---------- ---------- ------ ------- ------ --------
0/1       Enable    10G Full   10G Full   Up     Enable  Enable long
0/2       Enable    10G Full   10G Full   Up     Enable  Enable long
0/3       Disable   10G Full              Down   Enable  Enable long
0/4       Disable   10G Full              Down   Enable  Enable long
0/5       Disable   10G Full              Down   Enable  Enable long
0/6       Disable   10G Full              Down   Enable  Enable long
0/7       Disable   10G Full              Down   Enable  Enable long
<snip>
```

All of the ports can be enabled using the *no shutdown all* command

```
(localhost) #configure
(localhost) (Config)#no shutdown all
(localhost) (Config)#show port all
Admin Physical Physical Link Link LACP Actor
Intf Type Mode Mode Status Status Trap Mode Timeout
--------- ------ --------- ---------- ---------- ------ ------- ------ --------
0/1       Enable    10G Full   10G Full   Up     Enable  Enable long
0/2       Enable    10G Full   10G Full   Up     Enable  Enable long
0/3       Enable    10G Full              Down   Enable  Enable long
0/4       Enable    10G Full              Down   Enable  Enable long
0/5       Enable    10G Full              Down   Enable  Enable long
0/6       Enable    10G Full              Down   Enable  Enable long
```

```
0/7       Enable    10G Full              Down   Enable  Enable long
<snip>
```

After enabling switch ports, the user can assign IP addresses to them. In following example, interfaces 0/1 and 0/2 are assigned IP addresses and are enabled to route IP traffic between them.

```
(localhost) #configure
(localhost) (Config)#interface 0/1
(localhost) (Interface 0/1)# ip address 192.168.1.1 /24
(localhost) (Interface 0/1)# routing
(localhost) (Interface 0/1)# interface 0/2
(localhost) (Interface 0/2)# ip address 192.168.2.1 /24
(localhost) (Interface 0/2)# routing
(localhost) (Interface 0/2)# end
(localhost) #show ip interface brief
Interface  State IP Address      IP Mask         Method
---------- ----- --------------- --------------- -------
0/1        Up    192.168.1.1     255.255.255.0   Manual
0/2        Up    192.168.2.1     255.255.255.0   Manual
(localhost) #show ip route
Route Codes: R - RIP Derived, O - OSPF Derived, C - Connected, S - Static
B - BGP Derived, IA - OSPF Inter Area
E1 - OSPF External Type 1, E2 - OSPF External Type 2
N1 - OSPF NSSA External Type 1, N2 - OSPF NSSA External Type 2
S U - Unnumbered Peer, L - Leaked Route, K - Kernel
C 192.168.1.0/24 [0/0] directly connected, 0/1
C 192.168.2.0/24 [0/0] directly connected, 0/2
```

The routing interfaces enabled in ICOS OS are also visible in the Linux kernel stack.

The *ip route show* Linux command displays all the routes in Linux kernel stack, including the default route.

```
admin@localhost:~$ ip route show
default via 10.27.20.1 dev eth0
10.27.20.0/22 dev eth0 proto kernel scope link src 10.27.21.53
169.254.0.0/16 dev eth0 scope link metric 1002
192.168.1.0/24 dev rt1_0_1 proto kernel scope link src 192.168.1.1
192.168.2.0/24 dev rt1_0_2 proto kernel scope link src 192.168.2.1
250.0.0.0/30 dev rfc5549if proto kernel scope link src 250.0.0.1
```

3.1.4.5. Management Access

Packets of various protocol applications like DNS/NTP and management applications like syslog/puppet/ssh/ telnet depend on the default route in the kernel IP stack to reach their intended destination IP address. In most network topologies, these services use the out-of-band network (service port) so that the switch is available for administration and provisioning at all times. For this reason, a default route pointing to service port eth0 can be seen in the output of *ip route show* Linux command.

ICOS OS routing protocols can also add routing interfaces into the kernel IP stack as a result of protocol operation. The kernel IP stack gives priority to an ICOS OS-added default route that points toward in-band interfaces (switch ports), as the metric associated with an in-band interface

(10 Gbps or more) is typically better compared to the out-of-band interface (1 Gbps or less). This can result in a situation where packets of protocol applications such as DNS and syslog are incorrectly routed to the in-band network and, thus, do not reach their intended destination on the out-of-band interface network.

To overcome this problem, in network topologies where there is a possibility of a default route being added by ICOS OS, explicit static routes are added to resolve the destination IP address of management servers, such as a Puppet master or syslog server, to the service port. This can be achieved by modifying */etc/network/interface* file to add explicit routes under the “iface eth0” section. The *post-up route add* commands are executed after the service port and eth0 links are up.

```
post-up ip route add <syslog-server-ip> via <eth0-ip>
post-up ip route add <puppet-master-ip> via <eth0-ip>
```

3.2. ICOS OS Linux Integration

The features described in this section make it easier to manage ICOS OS via Linux utilities. These features are not required to be used for ICOS OS to operate properly. ICOS OS can be managed completely via the ICOS OS command-line interface or via SNMP.

The following topics are covered in this section:

- Section 3.2.1, “Interface Management via Linux” provides examples of how to monitor and configure physical ports, routing interfaces, and LAGs using standard Linux utilities such as ethtool, ip, ifconfig, and ifenslave ().
- Section 3.2.2, “Saving Configuration via the Linux File” describes the interface configuration parameters that can be saved in the /etc/network/interfaces file instead of the ICOS OS configuration file.
- Section 3.2.3, “Route Table Management” describes how ICOS OS pushes routes from the kernel into the hardware. This feature enables third-party routing protocols to be used in conjunction with ICOS OS.

3.2.1. Interface Management via Linux

3.2.1.1. Physical Interface Management

The physical interfaces are named fpti<unit>_<slot>_<port>.

The following table shows physical interfaces representation in Linux and their corresponding interfaces in ICOS OS. This example shows first three interfaces, but it is applicable to all physical interfaces supported by the switch.

Linux	ICOS OS
fpti1_0_1	0/1
fpti1_0_2	0/2
fpti1_0_3	0/3

Standard Linux interface administration commands can be used on these interfaces, as listed in the table below.

Port configuration values include:

- Speed
- Duplex mode
- Auto negotiation mode
- Advertised speed, duplex mode, and flow control mode
- Supported speeds, auto negotiation capability, flow control mode, and media type

- Port media type, flow control mode, port speed, auto negation mode, and capabilities

Table 3.1. Physical Interface Management — Linux Command Examples

Action	**Linux Interface Command Example**
Configuring interface speed	ethtool -s fpti1_0_1 speed 10000
Configuring interface duplex	ethtool -s fpti1_0_1 duplex half
Configuring interface autoneg	ethtool -s fpti1_0_1 autoneg off
Fetching interface statistics	ethtool -S fpti1_0_1 See Chapter 4, *Ethtool Samples* for list of statistics in the output.
Fetching interface configuration	ethtool fpti1_0_1 Settings for fpti1_0_1: Supported ports: [FIBRE] Supported link modes: 1000baseT/Full Supported pause frame use: Symmetric Receive-only Supports auto-negotiation: Yes Advertised link modes: Not reported Advertised pause frame use: No Advertised auto-negotiation: No Speed: 10000Mb/s Duplex: Full Port: FIBRE PHYAD: 0 Transceiver: external Auto-negotiation: off Current message level: 0xfffffffa1 (-95) drv ifup tx_err tx_queued intr tx_done rx_status pktdata hw wol 0xffff8000 Link detected: no
Configuring admin state	ifconfig fpti1_0_1 down ifconfig fpti1_0_1 up ip link set fpti1_0_1 up

Action	Linux Interface Command Example
	ip link set fpti1_0_1 down
Configuring MTU	ifconfig fpti1_0_1 mtu 1200 ip link set fpti1_0_1 mtu 1200
Fetching admin state and MTU	ifconfig fpti1_0_1 ip addr show dev fpti1_0_1
Fetching link state	ethtool fpti1_0_1

3.2.1.2. Port-based Routing Interfaces Management

The port-based routing interfaces are named rt<unit>_<slot>_<port>. and typical Linux interface administration commands can be used on these interfaces, as listed in the table below.

Table 3.2. Port-based Routing Interfaces Management — Linux Command Examples

Action	Linux Interface Command Example
Configure routing state	ifconfig rt1_0_1 up ifconfig rt1_0_1 down ip link set rt1_0_1 up ip link set rt1_0_1 down
Configure MTU	ifconfig rt1_0_1 mtu 1200 ip link set rt1_0_1 mtu 1200
Set/unset primary IPv4 address	ip addr add 3.3.3.3/24 dev rt1_0_2 ip addr del 3.3.3.3/24 dev rt1_0_2
Set/unset secondary IPv4 address	ip addr add 3.3.3.3/24 dev rt1_0_2 ip addr del 3.3.3.3/24 dev rt1_0_2
Add IPv6 address	ip addr add 2001:DB8::/126 dev rt1_0_2 ip addr del 2001:DB8::/126 dev rt1_0_2
Fetch routing state, MTU and primary address	ifconfig rt1_0_1 ip addr show dev rt1_0_1
Fetch interface statistics	ethtool -S rt1_0_1 See Chapter 4, *Ethtool Samples* for a list of statistics in the output.
Fetch link state	ethtool rt1_0_1 See Chapter 4, *Ethtool Samples* for a list of statistics in the output.

3.2.1.3. VLAN Routing Interfaces Management

The VLAN routing interfaces are named rt_v<vlan-id> and typical Linux interface administration commands can be used on these interfaces, as listed in the table below.

Table 3.3. VLAN Routing Interfaces Management — Linux Command Examples

Action	Linux Interface Command Example
Disable routing state	ifconfig rt_v10 down ip link set rt_v10 down
Configuring MTU	ifconfig rt_v10 mtu 1200 ip link set rt_v10 mtu 1200
Set/Unset primary IPv4 address	ip addr add 3.3.3.3/24 dev rt_v10 ip addr del 3.3.3.3/24 dev rt_v10
Set/Unset secondary IPv4 address	ip addr add 3.3.3.3/24 dev rt_v10 ip addr del 3.3.3.3/24 dev rt_v10
Add IPv6 address	ip addr add 2001:DB8::/126 dev rt_v10 ip addr del 2001:DB8::/126 dev rt_v10
Fetch MTU and primary address	ifconfig rt_v10 ip addr show dev rt1_v10
Fetch link state	ethtool rt_v10 See Chapter 4, *Ethtool Samples* for list of statistics in the output.

3.2.1.4. Bonding Interfaces Management

The bonding interfaces are named bond<bond-id>. The following table shows bonding interface representations in Linux and their corresponding interfaces in ICOS OS. This example shows first three port-channel interfaces, but it is applicable to all port-channel interfaces supported by the switch.

Linux	ICOS OS Logical Interface	ICOS OS Port-Channel Name
bond1	3/1	ch1
bond2	3/2	ch2
bond3	3/3	ch3

Standard Linux interface administration commands can be used on these interfaces, as listed the table below.

Table 3.4. Bonding Interfaces Management — Linux Command Examples

Action	Linux Interface Command Example
Configuring admin state	ifconfig bond1 down ifconfig bond1 up

Action	Linux Interface Command Example
	ip link set bond1 up ip link set bond1 down
Add ports	ifenslave bond1 fpti1_0_1 ip link set fpti1_0_1 master bond1 echo +fpti1_0_1 > /sys/class/net/bond1/bonding/slaves Interface links can go down and come back up when using the *ifenslave* command.
Delete ports	ifenslave -d bond1 fpti1_0_1 ip link set fpti1_0_1 nomaster echo -fpti1_0_1 > /sys/class/net/bond1/bonding/slaves Interface links can go down and come back up when using the *ifenslave* command.
Show ports	cat /sys/class/net/bond1/bonding/slaves cat /proc/net/bonding/bond1
Show mode/min_links	cat /sys/class/net/bond1/bonding/mode cat /sys/class/net/bond1/bonding/min_links
Set bond mode	echo 0 > /sys/class/net/bond1/bonding/mode echo 4 > /sys/class/net/bond1/bonding/mode
Set min_links	echo 2 > /sys/class/net/bond1/bonding/ min_links
Fetching link state	ethtool bond1 Refer to Chapter 4, *Ethtool Samples* for link state output.
Fetching interface statistics	ethtool -S bond1 See Chapter 4, *Ethtool Samples* for list of statistics in the output.

3.2.2. Saving Configuration via the Linux File

When Ubuntu Linux boots, a standard configuration program “ifupdown” reads the interface configuration from the */etc/network/interfaces* file and configures the interfaces. There are four types of hooks (pre-up, up, down, and post-down) that can be configured in */etc/network/interfaces*.

Note: To bring the changes into effect, networking service must be restarted using one of the following commands:

- service networking stop; service networking start
- /etc/init.d/networking restart

3.2.2.1. Configuring Physical Interface Properties

```
auto fpti1_0_1
iface fpti1_0_1 inet static
       address 0.0.0.0
       mtu 1540
       pre-up /sbin/ethtool -s fpti1_0_1 speed 10 duplex half
       pre-down /sbin/ethtool -s fpti1_0_1 speed 10 duplex half
```

3.2.2.2. Configuring a Port-based Routing Interface

```
auto rt1_0_2
iface rt1_0_2 inet static
     mtu 200
      address 4.2.2.2
      netmask 255.255.255.0
      up ip addr add 3.3.3.3/24 dev rt1_0_2
     up ip addr add 2001::1/64 dev rt1_0_2
```

3.2.2.3. Configuring a VLAN Routing Interface

```
auto rt_v20
iface rt_v20 inet static
     mtu 200
      address 4.2.2.2
      netmask 255.255.255.0
      up ip addr add 3.3.3.3/24 dev rt_v20
     up ip addr add 2001::1/64 dev rt_v20
```

3.2.2.4. Configuring Physical Interface Properties

```
auto fpti1_0_1
iface fpti1_0_1 inet static
        address 0.0.0.0
        mtu 1540
        pre-up /sbin/ethtool -s fpti1_0_1 speed 10 duplex half
        pre-down /sbin/ethtool -s fpti1_0_1 speed 10 duplex half
```

3.2.2.5. Configuring Routes

```
auto rt_v20
iface rt_v20 inet static
     mtu 200
      address 4.2.2.2
      netmask 255.255.255.0
      up route add -net 5.5.5.0 netmask 255.255.255.0 gw 4.2.2.3
     up ip route add 6.6.6.0/24 via 4.2.2.3
```

3.2.2.6. Configuring a LAG Interface

```
auto bond1
```

```
iface bond1 inet static
        address 0.0.0.0
        mtu 2000
        up ip link set fpti1_0_1 master bond1
      up echo 4 > /sys/class/net/bond1/bonding/mode
      up echo 2 > /sys/class/net/bond1/bonding/ min_links
```

3.2.3. Route Table Management

3.2.4. Install IPv4 Route

Adding IPv4 Routes:

```
root@host:/home/admin# ip route add 9.9.9.0/24 via 2.0.0.3
root@host:/home/admin# route add -net 4.4.4.0 netmask 255.255.255.0 gw 7.0.0.5
root@host:/home/admin#
root@host:/home/admin# route -n
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
0.0.0.0 10.130.84.129 0.0.0.0 UG 100 0 0 eth0
10.130.84.128 0.0.0.0 255.255.255.128 U 0 0 0 eth0
7.0.0.0 0.0.0.0 255.255.255.0 U 0 0 0 rt1_0_11
2.0.0.0 0.0.0.0 255.255.255.0 U 0 0 0 rt1_0_18
4.4.4.0 7.0.0.5 255.255.255.0 UG 0 0 0 rt1_0_11
9.9.9.0 2.0.0.3 255.255.255.0 UG 0 0 0 rt1_0_18
root@host:/home/admin#
root@host:/home/admin#icos-cli
(host) #
(host) #show ip route
Route Codes: R - RIP Derived, O - OSPF Derived, C - Connected, S - Static
       B - BGP Derived, IA - OSPF Inter Area
       E1 - OSPF External Type 1, E2 - OSPF External Type 2
       N1 - OSPF NSSA External Type 1, N2 - OSPF NSSA External Type 2
       S U - Unnumbered Peer, L - Leaked Route, K - Kernel
C 7.0.0.0/24 [0/1] directly connected, 0/11
C 2.0.0.0/24 [0/1] directly connected, 0/18
K 9.9.9.0/24 [1/6493] via 2.0.0.3, 00d:00h:1m, 0/18
K 4.4.4.0/24 [1/6493] via 7.0.0.5, 00d:00h:1m, 0/11
(host) #
(host) #
```

Deleting IPv4 Routes:

```
root@host:/home/admin# ip route del 9.9.9.0/24 via 2.0.0.3
root@host:/home/admin# route del -net 4.4.4.0 netmask 255.255.255.0 gw 7.0.0.5
root@host:/home/admin#
root@host:/home/admin# route -n
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
0.0.0.0 10.130.84.129 0.0.0.0 UG 100 0 0 eth0
10.130.84.128 0.0.0.0 255.255.255.128 U 0 0 0 eth0
```

```
7.0.0.0 0.0.0.0 255.255.255.0 U 0 0 0 rt1_0_11
2.0.0.0 0.0.0.0 255.255.255.0 U 0 0 0 rt1_0_18
root@host:/home/admin#
```

3.2.4.1. Install IPv6 Route

Adding IPv6 Routes:

```
root@host:/home/admin#
root@host:/home/admin# ip -6 route add 3001:33:3::/64 via 2009:1::12
root@host:/home/admin# route add -6 5001:55:5::/64 gw 2044:1::14
root@host:/home/admin#
root@host:/home/admin# route -6 -n
Kernel IPv6 routing table
Destination     Next Hop     Flag Met Ref Use If
2009:1::/64     ::           U    256 0   6   rt1_0_11
2044:1::/64     ::           U    256 0   7   rt1_0_18
3001:33:3::/64  2009:1::12   UG   1   0   0   rt1_0_11
5001:55:5::/64  2044:1::14   UG   1   0   0   rt1_0_18
root@host:/home/admin#
root@host:/home/admin#
root@host:/home/admin#icos-cli
(host) #
(host) #show ipv6 route
IPv6 Routing Table - 4 entries
Codes: C - connected, S - static, 6To4 - 6to4 Route, B - BGP Derived
O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF Ext 1, OE2 - OSPF Ext 2
ON1 - OSPF NSSA Ext Type 1, ON2 - OSPF NSSA Ext Type 2, K - kernel
C 2009:1::/64 [0/0]
via ::, 0/11
C 2044:1::/64 [0/0]
via ::, 0/18
K 3001:33:3::/64 [1/0]
via 2009:1::12, 00h:00m:25s, 0/11
K 5001:55:5::/64 [1/0]
via 2044:1::14, 00h:00m:35s, 0/18
(host) #
(host) #
```

Deleting IPv6 Routes:

```
root@host:/home/admin#
root@host:/home/admin# ip -6 route del 3001:33:3::/64 via 2009:1::12
root@host:/home/admin# route del -6 5001:55:5::/64 gw 2044:1::14
root@host:/home/admin#
root@host:/home/admin# route -6 -n
Kernel IPv6 routing table
Destination Next Hop Flag Met Ref Use If
2009:1::/64 :: U 256 0 6 rt1_0_11
2044:1::/64 :: U 256 0 7 rt1_0_18
root@host:/home/admin#
root@host:/home/admin#
```

3.2.4.2. Install IPv4 ECMP Route

Adding IPv4 ECMP Routes:

```
root@host:/home/admin#
root@host:/home/admin# route add -net 4.4.4.0 netmask 255.255.255.0 gw 7.0.0.5
root@host:/home/admin# route add -net 4.4.4.0 netmask 255.255.255.0 gw 27.0.0.5
root@host:/home/admin# route add -net 4.4.4.0 netmask 255.255.255.0 gw 27.0.0.2
root@host:/home/admin#
root@host:/home/admin# route -n
Kernel IP routing table
Destination    Gateway        Genmask          Flags  Metric  Ref  Use  Iface
0.0.0.0        10.130.84.129  0.0.0.0          UG     100     0    0    eth0
10.130.84.128  0.0.0.0        255.255.255.128  U      0       0    0    eth0
7.0.0.0        0.0.0.0        255.255.255.0    U      0       0    0    rt1_0_11
27.0.0.0       0.0.0.0        255.255.255.0    U      0       0    0    rt1_0_13
4.4.4.0        7.0.0.5        255.255.255.0    UG     0       0    0    rt1_0_11
4.4.4.0        27.0.0.5       255.255.255.0    UG     0       0    0    rt1_0_13
4.4.4.0        27.0.0.2       255.255.255.0    UG     0       0    0    rt1_0_13
root@host:/home/admin#
root@host:/home/admin#icos-cli
(host) #
(host) #show ip route
Route Codes: R - RIP Derived, O - OSPF Derived, C - Connected, S - Static
B - BGP Derived, IA - OSPF Inter Area
E1 - OSPF External Type 1, E2 - OSPF External Type 2
N1 - OSPF NSSA External Type 1, N2 - OSPF NSSA External Type 2
S U - Unnumbered Peer, L - Leaked Route, K - Kernel
C 7.0.0.0/24 [0/1] directly connected, 0/11
C 27.0.0.0/24 [0/1] directly connected, 0/13
K 4.4.4.0/24 [1/6493] via 7.0.0.5, 02d:20h:19m, 0/11
                     via 27.0.0.2, 02d:20h:19m, 0/13
                     via 27.0.0.5, 02d:20h:19m, 0/13
(host) #
(host) #
```

Deleting IPv4 ECMP Routes:

```
root@host:/home/admin#
root@host:/home/admin# route del -net 4.4.4.0 netmask 255.255.255.0 gw 7.0.0.5
root@host:/home/admin# route del -net 4.4.4.0 netmask 255.255.255.0 gw 27.0.0.5
root@host:/home/admin# route del -net 4.4.4.0 netmask 255.255.255.0 gw 27.0.0.2
root@host:/home/admin#
root@host:/home/admin# route -n
Kernel IP routing table
Destination    Gateway        Genmask          Flags  Metric  Ref  Use  Iface
0.0.0.0        10.130.84.129  0.0.0.0          UG     100     0    0    eth0
10.130.84.128  0.0.0.0        255.255.255.128  U      0       0    0    eth0
7.0.0.0        0.0.0.0        255.255.255.0    U      0       0    0    rt1_0_11
27.0.0.0       0.0.0.0        255.255.255.0    U      0       0    0    rt1_0_13
root@host:/home/admin#
root@host:/home/admin#
```

3.2.4.3. Install IPv6 ECMP Route

Adding IPv6 ECMP Routes:

```
root@host:/home/admin#
root@host:/home/admin# route add -6 3001:33:3::/64 gw 2009:1::13
root@host:/home/admin# route add -6 3001:33:3::/64 gw 2044:1::14
root@host:/home/admin# route add -6 3001:33:3::/64 gw 2044:1::15
root@host:/home/admin#
root@host:/home/admin# route -6 -n
Kernel IPv6 routing table
Destination      Next Hop       Flag Met Ref Use If
2009:1::/64      ::             U    256 0   6   rt1_0_11
2044:1::/64      ::             U    256 0   7   rt1_0_13
3001:33:3::/64   2009:1::13     UG   1   0   0   rt1_0_11
3001:33:3::/64   2044:1::14     UG   1   0   0   rt1_0_13
3001:33:3::/64   2044:1::15     UG   1   0   0   rt1_0_13
root@host:/home/admin#
root@host:/home/admin#
root@host:/home/admin#icos-cli
(host) #
(host) #show ipv6 route
IPv6 Routing Table - 3 entries
Codes: C - connected, S - static, 6To4 - 6to4 Route, B - BGP Derived
O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF Ext 1, OE2 - OSPF Ext 2
ON1 - OSPF NSSA Ext Type 1, ON2 - OSPF NSSA Ext Type 2, K - kernel
C 2009:1::/64 [0/0]
   via ::, 0/11
C 2044:1::/64 [0/0]
   via ::, 0/13
K 3001:33:3::/64 [1/0]
   via 2009:1::13, 00h:07m:25s, 0/11
   via 2044:1::14, 00h:07m:25s, 0/13
   via 2044:1::15, 00h:07m:25s, 0/13
(host) #
(host) #
```

Deleting IPv6 ECMP Routes:

```
root@host:/home/admin#
root@host:/home/admin# route add -6 3001:33:3::/64 gw 2009:1::13
root@host:/home/admin# route add -6 3001:33:3::/64 gw 2044:1::14
root@host:/home/admin# route add -6 3001:33:3::/64 gw 2044:1::15
root@host:/home/admin#
root@host:/home/admin# route -6 -n
Kernel IPv6 routing table
Destination      Next Hop       Flag Met Ref Use If
2009:1::/64      ::             U    256 0   6   rt1_0_11
2044:1::/64      ::             U    256 0   7   rt1_0_13
root@host:/home/admin#
root@host:/home/admin#
```

3.2.4.4. Redistribution of Kernel Routes with BGP (ICOS OS BGP)

As an example, assume there are two routers connected back-to-back (Router1----Router2) and BGP is enabled in ICOS OS for both. The following example shows how redistribution of kernel routes (routes configured/added to kernel) to the peer router can be achieved via BGP.

Redistribution of IPv4 Kernel Routes via BGP:

In the originating node, Router-1, add a route to the kernel as follows:

```
root@host:/home/admin # route add -net 6.5.5.0 netmask 255.255.255.0 gw 2.2.2.9
root@host:/home/admin # route -n
Kernel IP routing table
Destination     Gateway       Genmask          Flags Metric Ref Use Iface
0.0.0.0         10.130.187.1  0.0.0.0          UG    16     0   0   eth0
2.2.2.0         0.0.0.0       255.255.255.0    U     0      0   0   rt1_0_9
6.5.5.0         2.2.2.10      255.255.255.0    UG    0      0   0   rt1_0_9
10.130.187.0    0.0.0.0       255.255.255.128  U     0      0   0   eth0
1 27.0.0.0      0.0.0.0       255.0.0.0        U     0      0   0   lo
root@host:/home/admin # exit
```

Now, check the *icos-cli* and to see the kernel route added to the hardware, as follows:

```
(Routing) (Config-router)#show ip route
Route Codes: R - RIP Derived, O - OSPF Derived, C - Connected, S - Static
     B - BGP Derived, IA - OSPF Inter Area
     E1 - OSPF External Type 1, E2 - OSPF External Type 2
     N1 - OSPF NSSA External Type 1, N2 - OSPF NSSA External Type 2
     S U - Unnumbered Peer, L - Leaked Route, K - Kernel
C 2.2.2.0/24 [0/0] directly connected, 0/9
K 6.5.5.0/24 [1/15] via 2.2.2.9, 00h:21m:30s, 0/9
```

Go to Router BGP Config mode and enable the redistribution of kernel routes:

```
(Routing) #config
(Routing) (Config)#router bgp 20
(Routing) (config-router)#redistribute kernel
(Routing) (config-router)#
```

Now, go to the peer, Router-2, and verify that this kernel route appears as a BGP route.

```
(Routing) #show ip route
Route Codes: R - RIP Derived, O - OSPF Derived, C - Connected, S - Static
      B - BGP Derived, IA - OSPF Inter Area
      E1 - OSPF External Type 1, E2 - OSPF External Type 2
      N1 - OSPF NSSA External Type 1, N2 - OSPF NSSA External Type 2
      S U - Unnumbered Peer, L - Leaked Route, K - Kernel
C 2.2.2.0/24 [0/0] directly connected, 0/9
B 6.5.5.0/24 [20/0] via 2.2.2.10, 00h:00m:07s, 0/9
```

Redistribution of IPv6 Kernel Routes via BGP

In the originating node, Router-1, add a route to the kernel as follows:

```
# ip -6 route add 3005::/64 via 2003::3
# ip -6 route show
2003::/64 dev rt1_0_9 proto kernel metric 256
3005::/64 via 2003::3 dev rt1_0_9 metric 1024
fe80::/64 dev eth0 proto kernel metric 256
fe80::/64 dev dtl0 proto kernel metric 256
fe80::/64 dev rt1_0_9 proto kernel metric 256
#
```

Now, check the *icos-cli* to verify that the kernel route is added to the hardware, as follows:

```
(Routing) (config-router-af)#show ipv6 route
IPv6 Routing Table - 2 entries
Codes: C - connected, S - static, 6To4 - 6to4 Route, B - BGP Derived
    O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF Ext 1, OE2 - OSPF Ext 2
    ON1 - OSPF NSSA Ext Type 1, ON2 - OSPF NSSA Ext Type 2, K - kernel
C 2003::/64 [0/0]
   via ::, 0/9
K 3005::/64 [1/0]
   via 2003::3, 00h:00m:13s, 0/9
```

Go to the Router BGP Config mode and enable the redistribution of kernel routes:

```
(Routing) (Config)#router bgp 20
(Routing) (Config-router)#address-family ipv6
(Routing) (config-router-af)#redistribute kernel
(Routing) (config-router-af)#
```

Now, go to the peer, Router-2, and verify that this kernel route appears as a BGP route:

```
(Routing) (config-router-af)#show ipv6 route
IPv6 Routing Table - 2 entries
Codes: C - connected, S - static, 6To4 - 6to4 Route, B - BGP Derived
      O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF Ext 1, OE2 - OSPF Ext 2
      ON1 - OSPF NSSA Ext Type 1, ON2 - OSPF NSSA Ext Type 2, K - kernel
C 2003::/64 [0/0]
   via ::, 0/9
B 3005::/64 [20/0]
   via fe80::210:18ff:fe99:f762, 00h:00m:03s, 0/9
```

3.2.4.5. How to Install the Quagga Package

1. Download the Quagga package from (http://download.savannah.gnu.org/releases/quagga/) to any location on the x86 platform.

2. Untar it using the *tar zxvf <package-name>* command. It will create a folder.

3. Go to that folder and enter the following commands:

   a. export CROSS_COMPILE_VAR=/usr/bin

      /usr/bin is the path where the complier resides in this system. It may vary. You can check the path where the compiler resides by using the *which gcc* command.

If gcc is not available, then install it using the *apt-get install build-essential* command.

b. export QUAGGA_INSTALL_DIR=/root/quagga_compiled

/root/quagga_compiled is the path where all the compiled binaries of the Quagga package will be stored.

Before executing this command, a folder must be created (in this example *quagga_compiled* is created in the root directory with read and write permissions).

c. export ac_cv_func_malloc_0_nonnull=yes

d. export ac_cv_func_realloc_0_nonnull=yes

e. ./configure --enable-netlink --prefix=/root/quagga_compiled --enable-user=root --enablegroup=root --enable-vty-group=root

f. Make

g. Make install

4. Now, exit from the current folder and navigate to the compiled folder:

   cd /root/quagga_compiled

5. root@localhost [mailto:root@localhost]:/home/admin/quagga_com# ls

   etc include lib sbin share----------These all are compiled files.

6. Navigate to the /sbin folder and start the desired service. For example, to start BGP:

   root@localhost [mailto:root@localhost]:/home/admin/quagga_com/sbin# ./bgpd &[3] 2162

   root@localhost [mailto:root@localhost]:/home/admin/quagga_com/sbin# 2014/10/22 18:36:32 BGP: BGPd 0.99.18 starting: vty@2605 [mailto:vty@2605], bgp@<all>:179

   To check whether or note this service is running, use the *ps -ef* command.

7. To login into the Quagga CLI:

telnet 127.0.0.1 2601 ----------to start Zebra

telnet 127.0.0.1 2602----------to start ripd

The following are the fixed port numbers for different services: 2601(zebra)/2602 (ripd)/2604(ospfd)/2605(bgpd).

While entering into Quagga CLI, it will prompt for password. Check the corresponding *.conf* file in the /etc folder.

In the */etc* folder, some files may have names like “bgpd.conf.sample”. These need to be copied or renamed to “bgpd.conf” (i.e., remove ‘sample’ at the end).

3.2.4.6. Configuring IPv4 Routes in Quagga

The following procedure configures IPv4 routes in Quagga:

1. From Quagga (route configuration in Quagga):

```
(localhost) #telnet 127.0.0.1 2601
Trying 127.0.0.1...
Connected to 127.0.0.1
Hello, this is Quagga (version 0.99.18).
Copyright 1996-2005 Kunihiro Ishiguro, et al.
User Access Verification
Password:
Password:
Router>
Router> en
Password:
Password:
Router#
Router# config t
Router(config)# ip route 7.3.6.0 255.255.255.0 1.2.3.5
  Dec 18 15:32:39 localhost-1 VR_AGENT[procLOG]: rto_netlink.c(1121) 651
%% Successfully added
kernel route ip 7.3.6.0 mask 255.255.255.0 via 1.2.3.5 on IntIf 25.
Router(config)# exit
Router# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
I - ISIS, B - BGP, > - selected route, * - FIB route
K>* 0.0.0.0/0 via 10.130.171.129, eth0
C>* 1.2.3.0/24 is directly connected, rt1_0_25
S>* 7.3.6.0/24 [1/0] via 1.2.3.5, rt1_0_25
C>* 10.130.171.128/25 is directly connected, eth0
C>* 127.0.0.0/8 is directly connected, lo
C>* 250.0.0.0/30 is directly connected, rfc5549if
```

2. In kernel (Quagga configured route added to kernel):

```
root@localhost:/home/admin# ip route
default via 10.130.171.129 dev eth0 metric 100
1.2.3.0/24 dev rt1_0_25 proto kernel scope link src 1.2.3.4
7.3.6.0/24 via 1.2.3.5 dev rt1_0_25 proto zebra
10.130.171.128/25 dev eth0 proto kernel scope link src 10.130.171.168
250.0.0.0/30 dev rfc5549if proto kernel scope link src 250.0.0.1
```

3. In *icos-cli* (route gets propagated from kernel to ICOS OS, and then to hardware):

```
(host) #show ip route
Route Codes: R - RIP Derived, O - OSPF Derived, C - Connected, S - Static
B - BGP Derived, IA - OSPF Inter Area
E1 - OSPF External Type 1, E2 - OSPF External Type 2
N1 - OSPF NSSA External Type 1, N2 - OSPF NSSA External Type 2
S U - Unnumbered Peer, L - Leaked Route, K - Kernel
```

```
C 1.2.3.0/24 [0/0] directly connected, 0/25
K 7.3.6.0/24 [1/56] via 1.2.3.5, 00h:02m:11s, 0/25
```

3.2.4.7. Configuring IPv6 Routes in Quagga

The following procedure configures IPv6 routes in Quagga:

1. From Quagga (route configuration in Quagga):

```
(localhost) #telnet 127.0.0.1 2601
Trying 127.0.0.1...
Connected to 127.0.0.1
Hello, this is Quagga (version 0.99.18).
Copyright 1996-2005 Kunihiro Ishiguro, et al.
User Access Verification
Password:
Password:
Router>
Router> en
Password:
Password:
Router# show ipv6 route
Codes: K - kernel route, C - connected, S - static, R - RIPng, O - OSPFv3,
I - ISIS, B - BGP, * - FIB route.
C>* ::1/128 is directly connected, lo
C>* 2001::/64 is directly connected, rt1_0_25
C * fe80::/64 is directly connected, rt1_0_25
C * fe80::/64 is directly connected, dtl0
C>* fe80::/64 is directly connected, eth0
Router(config)# ipv6 route 5555::/64 2001::2
  Dec 18 15:41:08 localhost-1 VR_AGENT[procLOG]: rto_netlink.c(1180) 656
%% Successfully added
kernel IPv6 route 5555::/64 via 2001::2 on IntIf 25.
exit
Router# show ipv6 route
Codes: K - kernel route, C - connected, S - static, R - RIPng, O - OSPFv3,
I - ISIS, B - BGP, * - FIB route.
C>* ::1/128 is directly connected, lo
C>* 2001::/64 is directly connected, rt1_0_25
S>* 5555::/64 [1/0] via 2001::2, rt1_0_25
C * fe80::/64 is directly connected, rt1_0_25
C * fe80::/64 is directly connected, dtl0
C>* fe80::/64 is directly connected, eth0
```

2. From kernel (Quagga configured route added to kernel):

```
root@localhost:/home/admin# route -6
Kernel IPv6 routing table
Destination                    Next Hop     Flag Met  Ref Use  If
2001::/64                      ::           U    256  0   1    rt1_0_25
2001::/64                      ::           UAe  256  0   0    dtl0
5555::/64                      2001::2      UG   1024 0   0    rt1_0_25
```

```
fe80::/64                       ::          U    256  0   0    eth0
fe80::/64                       ::          U    256  0   0    rt1_0_25
fe80::/64                       ::          U    256  0   0    dtl0
::/0                            ::          !n   -1   1   2465 lo
::1/128                         ::          Un   0    1   10   lo
2001::/128                      ::          Un   0    1   0    lo
2001::1/128                     ::          Un   0    1   6    lo
fe80::/128                      ::          Un   0    1   0    lo
fe80::2e60:cff:fe04:e105/128    ::          Un   0    1   1577 lo
fe80::2e60:cff:fe04:e106/128    ::          Un   0    1   0    lo
fe80::2e60:cff:fe04:e108/128    ::          Un   0    1   0    lo
ff00::/8                        ::          U    256  0   0    eth0
ff00::/8                        ::          U    256  0   0    rt1_0_25
ff00::/8                        ::          U    256  0   0    dtl0
::/0                            ::          !n   -1   1   2465 lo
```

3. From ICOS OS (route gets propagated from kernel to ICOS OS, and then to hardware):

```
(localhost) #show ipv6 route
IPv6 Routing Table - 2 entries
Codes: C - connected, S - static, 6To4 - 6to4 Route, B - BGP Derived
     O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF Ext 1, OE2 - OSPF Ext 2
     ON1 - OSPF NSSA Ext Type 1, ON2 - OSPF NSSA Ext Type 2, K - kernel
C 2001::/64 [0/0]
   via ::, 0/25
K 5555::/64 [1/0]
   via 2001::2, 00h:00m:14s, 0/25
```

3.3. Automated Software and Configuration Management

One important reason that network administrators are interested in x86 platforms is that the software on these platforms can be managed using tools available for managing the Linux x86 servers. Tools such as Puppet and Chef can manage software packages as well as switch configuration. The following sections provide examples of how to manage switches using these tools.

3.3.1. Periodic Synchronization of Configuration

3.3.1.1. Non-disruptive Configuration

The automated software, such as Chef or Puppet, can be configured for periodic synchronization of ICOS OS configuration. The Chef/Puppet configuration file defines the interval at which the tool fetches the ICOS OS configuration file. When the software pulls the configuration for the first time, the configuration file is applied and then saved to a defined target location. On subsequent fetches, before updating the configuration, the newly downloaded configuration file is compared against the file in the target location. If the files are same, no action is taken. If the files are different, the new configuration file is applied and is saved in the target location for future comparison.

3.3.1.2. Disruptive Configuration

If the user wants the configuration to be nonvolatile, the user can configure the automated software to save the configuration as *startup-config*. To achieve this behavior, the following configurations should be made to the manifest file.

- The configuration file is marked as ‘disruptive’.
- The target is mentioned as ‘/mnt/fastpath/startup-config’

In this case, the behavior is similar to periodic configuration. The new configuration file is honored only if it is different than the current configuration. In this case, the configuration is not applied using *icos-cfg*, but saved to the ‘startup-config’, and the ICOS OS application is restarted.

3.3.2. Managing the Switch with Puppet

Puppet agent (Version 3.4.3 and may be updated with the future ICOS release) software is preinstalled as part of ONIE. The Puppet agent on the switch needs to contact the Puppet master to download the manifest intended for the switch. The Puppet agent needs to know the Puppet master’s IP address or name in the puppet configuration file to contact Puppet master. There is no standard DHCP option to request that the DHCP server get the Puppet master’s IP address or name. The puppet configuration file can be updated manually, or a provisioning script can be used to update Puppet configuration file. Refer to Section 3.3.1.1, “Non-disruptive Configuration”, for the provisioning script to modify the puppet configuration file and trigger the Puppet agent to start.

3.3.2.1. Update Puppet Agent Configuration Using Provisioning Script

```
#!/bin/sh
# Sample post install script
```

```
#
cd /
echo "Provisioning Script Start"
# Create and make changes to puppet agent config
sed -i "s/server = puppet-srv/server = puppet.rtp.broadcom.com/g"
/etc/puppet/puppet.conf
# Contact puppet
puppet agent --test --waitforcert 10 --verbose --trace
update-rc.d puppet enable
service puppet start
echo "Provisioning Script End"
exit 0
```

3.3.2.2. Removing Puppet Agent

Users can install their choice of configuration management utilities using a provisioning script. It is recommended that the preinstalled Puppet package be uninstalled/removed. The following sample provisioning script can be used to remove the Puppet agent on the switch.

```
#!/bin/sh
# Sample post install script
# Uninstall puppet package
dpkg --purge puppet
dpkg --purge puppet-common
echo "Provisioning Script End"
exit 0
```

3.3.2.3. Adding Additional Packages

In addition using the provisioning script to update configuration files in Linux, it can be extended to install additional Debian packages:

```
#!/bin/sh
# Sample post install script
#
cd /
echo "Provisioning Script Start"
# Install apache server
apt-get update
apt-get -y install apache2
echo "Provisioning Script End"
exit 0
```

3.3.2.4. Update ICOS OS Configuration

Create the manifest for the switch on the Puppet master and ensure that the certificate is signed for the agent on the server. A sample manifest to configure the switch is shown in the following example. Be sure that netdev modules are copied to the *modules* folder on the Puppet master.

```
node 'icosx86.rtp.broadcom.com' {
  netdev_device { $hostname: }
  # Create an interface 0/31 with the specified interface properties
  netdev_interface { "0/31":
```

```
    description => "Puppet modified interface 0/31",
    admin => up
  }
  # Create an interface 0/32 with the specified interface properties
  netdev_interface { "0/32":
    description => "Puppet modified interface 0/32",
    admin => down
  }
  # Create list of VLANs 10, 20 and 30
  $vlans_new = {
    'White' => { vlan_id => 10, description => "This is a White vlan" },
    'Pink' => { vlan_id => 20, description => "This is a Pink vlan" },
    'Blue' => { vlan_id => 30, description => "This is a Blue vlan" }
  }
  create_resources( netdev_vlan, $vlans_new )
  # Create list of access ports 0/4, 0/5 and 0/6
  $access_ports = [
    '0/4',
    '0/5',
    '0/6'
  ]
  netdev_l2_interface { $access_ports:
    untagged_vlan => 'Blue',
    description => "This is a Puppet created access port"
  }
  # Create list of uplink/trunk ports 0/7, 0/8 and 0/9
  $uplink_ports = [
    '0/7',
    '0/8',
    '0/9'
  ]
  netdev_l2_interface { $uplink_ports:
    tagged_vlans => [ 'White', 'Pink' ],
    description => "This is a Puppet created uplink port"
  }
  # Create a LAG interface 3/1 and add the interfaces 0/11 and 0/12 and
  0/13 to LAG 3/1
  netdev_lag { "3/1":
    links => [ '0/11', '0/12', '0/13' ],
    minimum_links => 3,
    lacp => active
  }
  # Create VLANs 100, 200
  $vlans = {
    'Yellow' => { vlan_id => 100, description => "This is a Yellow vlan" },
    'Green' => { vlan_id => 200, description => "This is a Green vlan" }
  }
  create_resources( netdev_vlan, $vlans )
}
```

3.3.2.5. Update ICOS OS Configuration Using icos-cfg

Apply Text-based Configuration

The following Puppet manifest illustrates applying the configuration on the switch.

```
node 'icosx86.rtp.broadcom.com' {
  # Ensure that the service is running.
  #
  service {'icos':
     ensure => running,
  }
  $path_var = "/usr/bin:/usr/sbin:/bin:/sbin"
  # Sample text configuration
  #
  # vlan database
  # vlan 111
  # exit
  #
  # Location of the sample script on the switch
  #
  $sample_scr = "/broadcom/sample"
  file {$sample_scr:
     ensure => present,
     owner => 'root',
     group => 'root',
     mode => '0600',
     source => 'puppet:///modules/icos/sample.scr',
  }
  # Apply sample configuration using icos-cfg
  #
  exec { 'apply_sample_script':
     command => "icos-cfg -a $sample_scr -t 120 -d",
     path => $path_var,
     subscribe => File[$sample_scr],
     logoutput => true,
     returns => 0,
     refreshonly => true,
  }
}
```

Periodic Synchronization of Configuration

The following Puppet manifest illustrates handling periodic configuration in disruptive and non-disruptive modes:

```
node 'tr3demo.hyd.broadcom.com' {
  #
  #
  $non_disruptive_mode = "true"
  # Ensure that the service is running.
  service { 'icos':
     ensure => running,
```

```
}
$path_var = "/usr/bin:/usr/sbin:/bin:/sbin"
$startup_file = "/broadcom/startup-config"
file {$startup_file:
   ensure => present,
   owner => 'root',
   group => 'root',
   mode => '0600',
   source => 'puppet:///modules/icos/startup-config',
}
if $non_disruptive_mode == "true" {
   file { '/usr/sbin/config-create.sh':
      ensure => present,
      owner => 'root',
      group => 'root',
      mode => '0755',
      source => 'puppet:///modules/icos/scripts/config-create.sh',
   }
   file { '/usr/sbin/config-apply.sh':
      ensure => present,
      owner => 'root',
      group => 'root',
      mode => '0755',
      source => 'puppet:///modules/icos/scripts/config-apply.sh',
   }
   $delta_file = "/broadcom/delta-config"
   file { '/usr/sbin/delta-config.sh':
      ensure => present,
      owner => 'root',
      group => 'root',
      mode => '0755',
      source => 'puppet:///modules/icos/scripts/delta-config.sh',
   }
   exec { 'disruptive-startup-config-create':
      command => "config-create.sh $startup_file",
      path => $path_var,
      subscribe => File[$startup_file],
      creates => "/mnt/fastpath/startup-config",
      logoutput => true,
      returns => 0,
   }
   exec { 'non-disruptive-startup-config-update':
      command => "config-apply.sh $startup_file",
      path => $path_var,
      subscribe => File[ $startup_file],
      logoutput => true,
      returns => 0,
      refreshonly => true,
   }
   #Change script
   file {$delta_file:
```

```
        ensure => present,
        mode => 600,
        owner => "root",
        group => "root",
        source => 'puppet:///modules/icos/delta-config.scr',
     }
     exec { 'non-disruptive-delta-config':
        command => "delta-config.sh $delta_file",
        path => $path_var,
        subscribe => File[$delta_file],
        logoutput => true,
        returns => 0,
        refreshonly => true,
     }
  } else {
     exec { "disruptive-startup-config":
        command => "cp -f $startup_file /mnt/fastpath/startup-config;
                initctl restart icos",
        path => $path_var,
        subscribe => File[$startup_file],
        onlyif => "icos-cfg -v $startup_file -t 120",
        logoutput => true,
        returns => 0,
        refreshonly => true
     }
  }
}
```

3.3.3. Managing the Switch with Chef

As mentioned in Section 3.3.1, “Periodic Synchronization of Configuration”, Puppet agent software (Version 3.4.3 and may be updated with the future ICOS release) is preinstalled as part of ONIE. Users need to install the Chef Client on the switch, however, to manage the switch using Chef. It is recommended that the Puppet agent be uninstalled before installing the Chef Client. Refer to Section 3.3.2.2, “Removing Puppet Agent” for instructions on removing the preinstalled Puppet Agent from the switch.

3.3.3.1. Install Chef Client Using Provisioning Script

The provisioning script can be used to install the Chef Client on the switch. The following is a sample provisioning script for installing the Chef Client (this script works for Ubuntu 14.04):

```
#!/bin/sh
# Sample post install script
cd /
echo "Provisioning Script Start"
#add the Opscode repository to the server's apt sources
echo "deb http://apt.opscode.com/ precise-0.10 main" | sudo tee
/etc/apt/sources.list.d/opscode.list
#add the GPG key and update the apt index
mkdir -p /etc/apt/trusted.gpg.d
```

```
gpg --keyserver keys.gnupg.net --recv-keys 83EF826A
gpg --export packages@opscode.com | sudo tee /etc/apt/trusted.
gpg.d/opscode-keyring.gpg > /dev/null
# Install chef client
apt-get update
apt-get --yes --force-yes install opscode-keyring
apt-get --yes --force-yes install chef
echo "Provisioning Script End"
exit 0
```

3.3.3.2. Update Chef Client Configuration Using Provisioning Script

The Chef client on the switch needs to contact a Chef Server to download recipes intended for the switch. To contact Chef server, the Chef client needs to have in its configuration file the server's IP address or name. There is no standard DHCP option to request a DHCP server to provide a Chef server's IP address or name. The Chef configuration file can be updated manually, or a provisioning script can be used to update chef configuration file.

The following sample provisioning script modifies the Chef configuration file and triggers starts the Chef client.

```
#!/bin/sh
# Sample post install script
#
cd /
echo "Provisioning Script Start"
# Create and make changes to chef client config
sed -i 's|'.*chef_server_url.*'|chef_server_url "chef.rtp.broadcom.com
"|g' /etc/chef/client.rb
# change the node name (need to match with hostname)
sed -i 's/.*node_name.*/node_name "icosx86"/g' /etc/chef/client.rb
# Contact chef server
chef-client
service chef-client start
echo "Provisioning End"
exit 0
```

Be sure to copy client certificate to the switch. Optionally, you can utilize the provisioning script to download the certificate and place it in the /etc/chef/' folder.

3.3.3.3. Removing Chef Client

Users can install their choice of configuration management utilities using a provisioning script. To install their choice of configuration management utilities, it is recommended that the Chef Client be uninstalled/removed.

The following script can be used to uninstall the Chef client:

```
#!/bin/sh
```

```
# Uninstall Chef Client package
dpkg --purge chef
echo "Provisioning Script End"
exit 0
```

3.3.3.4. Adding Additional Packages

Refer to section Section 3.3.2.3, “Adding Additional Packages” for the sample provisioning script to install additional packages on the switch.

3.3.3.5. Update ICOS OS Configuration

Create a node and runlist intended for the switch on the Chef server and ensure that the validation key is copied to the switch. The following example shows a sample data bag to configure the switch and ensure that netdev modules are copied to the Chef server.

```
{
  "id": "icosx86",
  "netdev_l2_interface":
  {
     "0/4":
     {
        "name": "interface 4",
        "description": "chef modified: interface description",
        "untagged_vlan": "30",
        "tagged_vlans": ["10", "20"],
        "vlan_tagging": "enable"
     },
     "0/1":
     {
        "name": "interface 1",
        "description": "chef modified: interface description",
        "untagged_vlan": "30",
        "tagged_vlans": ["10", "20"],
        "vlan_tagging": "enable"
     },
     "0/7":
     {
        "name": "interface 7",
        "description": "chef modified: interface description",
        "untagged_vlan": "20",
        "tagged_vlans": ["10", "20", "30"],
        "vlan_tagging": "enable"
     }
  },
  "netdev_interface":
  {
     "0/1":
     {
        "name": "0/1",
        "admin": "up",
        "description": "Interface ONE description"
```

```
    }
  },
  "netdev_vlan":
  {
     "VLAN 10":
     {
        "vlan_id": 10,
        "description": "chef modified vlan 10 description"
     },
     "VLAN 20":
     {
        "vlan_id": 20,
        "description": "chef modified vlan 20 description"
     },
     "VLAN 30":
     {
        "vlan_id": 30,
        "description": "chef modified vlan 30 description"
     },
     "VLAN 100":
     {
        "vlan_id": 100,
        "description": "chef modified vlan 100 description"
     },
  },
 }
}
```

3.3.3.6. Update ICOS OS Configuration Using icos-cfg

Apply Text-based Configuration

The following recipe illustrates applying the configuration on the switch.

```
#
# Cookbook Name:: ICOS
# Recipe:: default
#
# Copyright 2015, Broadcom
#
# All rights reserved - Do Not Redistribute
#
execute "applyCfg" do
 command "icos-cfg -a /broadcom/delta-config -t 120 -d"
 action :nothing
end
cookbook_file '/broadcom/delta-config' do
 source 'delta-config.scr'
 notifies :run, "execute[applyCfg]", :immediately
end
```

Periodic Synchronization of Configuration

The following recipe illustrates handling periodic configuration in disruptive and non-disruptive modes:

```
#
# Cookbook Name:: ICOS
# Recipe:: default
#
# Copyright 2015, Broadcom
#
# All rights reserved - Do Not Redistribute
#
non_disruptive_mode = true
execute "disruptive-startup-config-create" do
  command "config-create.sh /broadcom/startup-config"
  action :nothing
end
execute "non-disruptive-startup-config-update" do
  command "config-apply.sh /broadcom/startup-config"
  action :nothing
end
execute "non-disruptive-delta-config" do
  command "delta-config.sh /broadcom/delta-config"
  action :nothing
end
execute "disruptive-startup-config" do
  command "cp -f /broadcom/startup-config /mnt/fastpath/startup-config;
  initctl restart ICOS"
  only_if "icos-cfg -v /broadcom/startup-config -t 120"
  action :nothing
end
if non_disruptive_mode
  cookbook_file '/usr/sbin/config-create.sh' do
     source 'config-create.sh'
     mode '0755'
  end
  cookbook_file '/usr/sbin/config-apply.sh' do
     source 'config-apply.sh'
     mode '0755'
  end
  cookbook_file '/usr/sbin/delta-config.sh' do
     source 'delta-config.sh'
     mode '0755'
  end
  cookbook_file '/broadcom/delta-config' do
     source 'delta-config'
     notifies :run, "execute[non-disruptive-delta-config]", :immediately
  end
  cookbook_file '/broadcom/startup-config' do
     source 'startup-config'
     notifies :run, "execute[disruptive-startup-config-create]", :immediately
     notifies :run, "execute[non-disruptive-startup-config-update]", :immediately
  end
```

```
else
  cookbook_file '/broadcom/startup-config' do
     source 'startup-config'
     notifies :run, "execute[disruptive-startup-config]", :immediately
  end
end
```

Chapter 4. Ethtool Samples

4.1. Ethtool Physical Interface Statistics Sample

```
root@localhost:~# ethtool -S fpti1_0_1

NIC statistics:

rx_bytes: 0
rx_64_byte_packets: 0
rx_65_to_127_byte_packets: 0
rx_128_to_255_byte_packets: 0
rx_256_to_511_byte_packets: 0
rx_512_to_1023_byte_packets: 0
rx_1024_to_1518_byte_packets: 0
rx_1519_to_1530_byte_packets: 0
rx_good_oversized_packets: 0
rx_error_oversized_packets: 0
rx_good_undersized_packets: 0
rx_error_undersized_packets: 0
rx_ucast_packets: 0
rx_mcast_packets: 0
rx_bcast_packets: 0
rx_align_error_packets: 0
rx_fcs_error_packets: 0
rx_overrun_packets: 0
rx_too_long_packets: 0
rx_local_traffic_discards: 0
rx_pause_frames: 0
rx_unaccept_frame_type_discards: 0
rx_ingress_filter_discards: 0
rx_vlan_viable_discards: 0
rx_mcast_tree_viable_discards: 0
rx_rsvd_address_discards: 0
rx_bcast_storm_recovery_discards: 0
rx_cfi_discards: 0
rx_upstream_threshold_discards: 0
rx_l3_in: 0
rx_l3_forwarded: 0
rx_l3_frag_discard: 0
rx_l3_in_hdr_errors: 0
rx_l3_addr_errors: 0
rx_l3_disc_routed: 0
rx_l3_mac_mcast_discards: 0
rx_l3_arp_to_cpu: 0
rx_l3_ip_to_cpu: 0
tx_bytes: 0
tx_64_byte_packets: 0
tx_65_to_127_byte_packets: 0
tx_128_to_255_byte_packets: 0
tx_256_to_511_byte_packets: 0
```

```
tx_512_to_1023_byte_packets: 0
tx_1024_to_1518_byte_packets: 0
tx_1519_to_1530_byte_packets: 0
tx_ucast_packets: 0
tx_mcast_packets: 0
tx_bcast_packets: 0
tx_fcs_errors: 0
tx_oversized: 0
tx_underrun_errors: 0
tx_one_collision: 0
tx_multiple_collision: 0
tx_excessive_collision: 0
tx_pause_frames: 0
tx_port_membership_discards: 0
tx_vlan_viable_discards: 0
tx_late_collisions: 0
tx_rx_64_byte_packets: 0
tx_rx_65_to_127_byte_packets: 0
tx_rx_128_to_255_byte_packets: 0
tx_rx_256_to_511_byte_packets: 0
tx_rx_512_to_1023_byte_packets: 0
tx_rx_1024_to_1518_byte_packets: 0
tx_rx_1519_to_1522_byte_packets: 0
tx_rx_1523_to_2047_byte_packets: 0
tx_rx_2048_to_4095_byte_packets: 0
tx_rx_4096_to_9216_byte_packets: 0
ether_stats_drop_events: 0
snmpIfOutDiscards: 0
snmpIfInDiscards: 0
rx_ipv6_in: 0
rx_ipv6_in_hdr_errors: 0
rx_ipv6_in_addr_errors: 0
rx_ipv6_in_discards: 0
rx_ipv6_out_fwd: 0
rx_ipv6_out_discards: 0
rx_ipv6_in_mcast: 0
rx_ipv6_out_mcast: 0
link_down_counter: 0
ether_link_flaps: 0
```

```
root@localhost:~# ethtool fpti1_0_1
Settings for fpti1_0_1:
       Supported ports: [ FIBRE ]
       Supported link modes:   40000baseLR4/Full
       Supported pause frame use: Symmetric Receive-only
       Supports auto-negotiation: No
       Advertised link modes:  Not reported
       Advertised pause frame use: No
       Advertised auto-negotiation: No
       Speed: 100000Mb/s
       Duplex: Full
       Port: FIBRE
```

```
        PHYAD: 0
        Transceiver: external
        Auto-negotiation: off
        Current message level: 0xffffffa1 (-95)
                               drv ifup tx_err tx_queued intr tx_done

rx_status pktdata hw wol 0xffff8000
```

4.2. Ethtool Routing Interfaces Statistics Sample

```
root@localhost:~# ethtool -S rt1_0_1

no stats available

root@localhost:~# ethtool rt1_0_1

Settings for rt1_0_1:
       Link detected: no
```

4.3. Ethtool VLAN Routing Interfaces Management Sample

```
root@localhost:~# ethtool rt_v10
Settings for rt_v10:
Cannot get device settings: No such device
Cannot get wake-on-lan settings: No such device
Cannot get message level: No such device
Cannot get link status: No such device
No data available
```

4.4. Ethtool Bonding Interfaces Sample

```
root@localhost:~# ethtool bond1
Settings for bond1:
       Supported ports: [ ]
       Supported link modes:   Not reported
       Supported pause frame use: No
       Supports auto-negotiation: No
       Advertised link modes:  Not reported
       Advertised pause frame use: No
       Advertised auto-negotiation: No
       Speed: Unknown!
       Duplex: Unknown! (255)
       Port: Other
       PHYAD: 0
       Transceiver: internal
       Auto-negotiation: off
       Link detected: no
```

```
root@localhost:~# ethtool -S bond1
NIC statistics:
    rx_bytes: 0
    rx_64_byte_packets: 0
    rx_65_to_127_byte_packets: 0
    rx_128_to_255_byte_packets: 0
    rx_256_to_511_byte_packets: 0
    rx_512_to_1023_byte_packets: 0
    rx_1024_to_1518_byte_packets: 0
    rx_1519_to_1530_byte_packets: 0
    rx_good_oversized_packets: 0
    rx_error_oversized_packets: 0
    rx_good_undersized_packets: 0
    rx_error_undersized_packets: 0
    rx_ucast_packets: 0
    rx_mcast_packets: 0
    rx_bcast_packets: 0
    rx_align_error_packets: 0
    rx_fcs_error_packets: 0
    rx_overrun_packets: 0
    rx_too_long_packets: 0
    rx_jabber_frames: 0
    rx_local_traffic_discards: 0
    rx_pause_frames: 0
    rx_unaccept_frame_type_discards: 0
    rx_ingress_filter_discards: 0
    rx_vlan_viable_discards: 0
    rx_mcast_tree_viable_discards: 0
    rx_rsvd_address_discards: 0
    rx_bcast_storm_recovery_discards: 0
    rx_cfi_discards: 0
    rx_upstream_threshold_discards: 0
```

```
    rx_l3_in: 0
    rx_l3_forwarded: 0
    rx_l3_frag_discard: 0
    rx_l3_in_hdr_errors: 0
    rx_l3_addr_errors: 0
    rx_l3_disc_routed: 0
    rx_l3_mac_mcast_discards: 0
    rx_l3_arp_to_cpu: 0
    rx_l3_ip_to_cpu: 0
    tx_bytes: 0
    tx_64_byte_packets: 0
    tx_65_to_127_byte_packets: 0
    tx_128_to_255_byte_packets: 0
    tx_256_to_511_byte_packets: 0
    tx_512_to_1023_byte_packets: 0
    tx_1024_to_1518_byte_packets: 0
    tx_1519_to_1530_byte_packets: 0
    tx_ucast_packets: 0
    tx_mcast_packets: 0
    tx_bcast_packets: 0
    tx_fcs_errors: 0
    tx_oversized: 0
    tx_underrun_errors: 0
    tx_one_collision: 0
    tx_multiple_collision: 0
    tx_excessive_collision: 0
    tx_pause_frames: 0
    tx_port_membership_discards: 0
    tx_vlan_viable_discards: 0
    tx_late_collisions: 0
    tx_rx_64_byte_packets: 0
    tx_rx_65_to_127_byte_packets: 0
    tx_rx_128_to_255_byte_packets: 0
    tx_rx_256_to_511_byte_packets: 0
    tx_rx_512_to_1023_byte_packets: 0
    tx_rx_1024_to_1518_byte_packets: 0
    tx_rx_1519_to_1522_byte_packets: 0
    tx_rx_1523_to_2047_byte_packets: 0
    tx_rx_2048_to_4095_byte_packets: 0
    tx_rx_4096_to_9216_byte_packets: 0
    ether_stats_drop_events: 0
    snmpIfOutDiscards: 0
    snmpIfInDiscards: 0
    rx_ipv6_in: 0
    rx_ipv6_in_hdr_errors: 0
    rx_ipv6_in_addr_errors: 0
    rx_ipv6_in_discards: 0
    rx_ipv6_out_fwd: 0
    rx_ipv6_out_discards: 0
    rx_ipv6_in_mcast: 0
    rx_ipv6_out_mcast: 0
```

```
    link_down_counter: 0
    ether_link_flaps: 0
```

4.5. Ethtool Virtual Routing Interfaces Sample

```
root@localhost:~# ethtool -S rt_vrf_0_stk
NIC statistics:
    peer_ifindex: 8
```

```
root@localhost:~# ethtool rt_vrf_0_stk
Settings for rt_vrf_0_stk:
       Supported ports: [ ]
       Supported link modes:   Not reported
       Supported pause frame use: No
       Supports auto-negotiation: No
       Advertised link modes:  Not reported
       Advertised pause frame use: No
       Advertised auto-negotiation: No
       Speed: 10000Mb/s
       Duplex: Full
       Port: Twisted Pair
       PHYAD: 0
       Transceiver: internal
       Auto-negotiation: off
       MDI-X: Unknown
       Link detected: yes
```