CONFIGURATION GUIDE

Extreme SLX-OS Management Configuration Guide, 18s.1.01

Supporting the ExtremeSwitching SLX 9140 and SLX 9240 Switches

9035259
November 2018

Notice: This document is for informational purposes only and does not set forth any warranty, expressed or implied, concerning any equipment, equipment feature, or service offered or to be offered by Brocade. Brocade reserves the right to make changes to this document at any time, without notice, and assumes no responsibility for its use. This informational document describes features that may not be currently available. Contact a Brocade sales office for information on feature and product availability. Export of technical data contained in this document may require an export license from the United States government.

The authors and Brocade Communications Systems, Inc. assume no liability or responsibility to any person or entity with respect to the accuracy of this document or any loss, cost, liability, or damages arising from the information contained herein or the computer programs that accompany it.

The product described by this document may contain open source software covered by the GNU General Public License or other open source license agreements. To find out which open source software is included in Brocade products, view the licensing terms applicable to the open source software, and obtain a copy of the programming source code, please visit http://www.brocade.com/support/oscd.

Contents

Preface

Document conventions

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

Notes, cautions, and warnings

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

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

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

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

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

Text formatting conventions

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

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

Format	Description
	Identifies command syntax examples.

Command syntax conventions

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

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

Extreme resources

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

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

Document feedback

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

You can provide feedback in two ways:

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

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

Contacting Extreme Technical Support

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

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

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

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

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

About This Document

What's new in this document

The following table summarizes information added to this guide for the SLX-OS 18s.1.01 software release.

TABLE 1 Summary of enhancements in SLX-OS release 18s.1.01

Feature	Description	Described in
Dynamic breakout mode	Configures 40-Gbe or 100-Gbe ports as four 10-Gbe or four 25-Gbe ports, without the need to reboot the switch.	Configuring breakout mode on page 31

Supported hardware and software

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

Although many different software and hardware configurations are tested and supported by Extreme Networks, Inc. for this SLX-OS release, documenting all possible configurations and scenarios is beyond the scope of this document.

The following hardware platforms are supported by this release:

- ExtremeSwitching SLX 9140
- ExtremeSwitching SLX 9240

NOTE

Some of the commands in this document use a slot/port designation. Because the SLX 9140 and the SLX 9240 do not contain line cards, the slot designation must always be "0" (for example, 0/1 for port 1).

Configuration Fundamentals

Configuration files

Extreme devices support three types of configuration files, default, startup, and running configuration files.

The startup configuration resides in the /var/config/vcs/scripts directory. Though the default configuration files are physically present in this directory, they are linked to the directory from their actual location.

When you boot up a device for the first time, the running configuration is identical to the startup configuration. As you configure the device, the changes are written to the running configuration. To save the changes, you must save the currently effective configuration (the running configuration) as the startup configuration. When the device reboots, the configuration changes become effective.

Default configuration files

Default configuration files are part of the firmware package for the device and are automatically applied to the startup configuration under the following conditions:

- When the device boots up for the first time and no customized configuration is available.
- When you restore the default configuration.

You cannot remove, rename, or change the default configuration. The default configuration file names are as follows:

- defaultconfig.standalone
- defaultconfig.cluster

Startup configuration files

The startup configuration is persistent. It is applied when the system reboots.

- When the device boots up for the first time, it uses the default configuration as the startup configuration, depending on the mode.
- When you make configuration changes to the running configuration and save the changes to the startup configuration with the **copy** command, the running configuration becomes the startup configuration.

The startup configuration file name is startup-config.

Running configuration files

The configuration currently effective on the device is referred to as the running configuration. Any configuration change you make while the device is online is made to the running configuration.

- The running configuration is nonpersistent.
- To save configuration changes, you must copy the running configuration to the startup configuration. If you are not sure about the changes, you can copy the changes to a file, and apply the changes later.

The running configuration file name is running-config.

Displaying configurations

The following examples illustrate how to display the default, startup, and running configurations, respectively.

Displaying the default configuration

To display the default configuration, enter the **show file** command with the default configuration filenames in privileged EXEC mode.

```
device# show file defaultconfig.standalone
device# show file defaultconfig.cluster
```

Displaying the startup configuration

To display the contents of the startup configuration, enter the **show startup-config** command in privileged EXEC mode.

```
device# show startup-config
```

Displaying the running configuration

To display the contents of the running configuration, enter the **show running-config** command in the privileged EXEC mode.

```
device# show running-config
```

Saving configuration changes

Configuration changes are nonpersistent and are lost on reboot unless you save them permanently. You have two options for saving configuration changes:

- Copy the running configuration to the startup configuration. The changes become effective upon reboot.
- Copy the running configuration to a file, and apply it at some later date.

NOTE
Always make a backup copy of your running configuration before you upgrade or downgrade the firmware.

Saving the running configuration

To save the configuration changes you made, copy the running configuration to the startup configuration. The next time the device reboots, it uses the startup configuration and the changes you made earlier become effective.

Enter the **copy running-config startup-config** command in privileged EXEC mode.

```
device# copy running-config startup-config
This operation will modify your startup configuration. Do you want to continue? [Y/N]: y
```

Saving the running configuration to a file

If you want to save the changes you made to the configuration, but you do not want the changes to take effect when the device reboots, you can save the running configuration to a file. You can apply the changes at some later time.

Enter the **copy running-config** command in privileged EXEC mode. Specify the file name as the file URL.

```
device# copy running-config flash://myconfig
```

Verify the transaction by listing the directory contents.

```
device# dir
total 32
drwxr-xr-x 2  251 1011   4096 Jun  5 07:08 .
drwxr-xr-x 3  251 1011   4096 Mar 11 00:00 ..
-rw-r--r-- 1 root sys     410 Jun  3 00:56 defaultconfig.standalone
-rw-r--r-- 1 root sys     695 Jun  3 00:56 defaultconfig.cluster
-rw-r--r-- 1 root root 183118 Jun  7 12:19 myconfig
-rw-r--r-- 1 root root 185650 Jun  5 09:38 startup-config
```

Applying previously saved configuration changes

When you are ready to apply the configuration changes you previously saved to a file, copy the file (*myconfig* in the example) to the startup configuration. The changes take effect after the device reboots.

Enter the **copy** command in privileged EXEC mode. Specify the file name as the file URL followed by the **startup-config** keyword.

```
device# copy flash://myconfig startup-config
This operation will modify your startup configuration. Do you want to continue? [Y/N]: y
```

Backing up configurations

Always keep a backup copy of your configuration files, so you can restore the configuration in the event the configuration is lost or you make unintentional changes.

The following recommendations apply:

- Upload the configuration backup copies to an external host or to an attached Extreme-branded USB device.
- Avoid copying configuration files from one device to another. Instead restore the device configuration files from the backup copy.

Copying a configuration file to an external host

You can copy the startup-config or running-config file to a remote server through FTP, SCP, TFTP, or SFTP.

In the following example, the startup configuration is copied to a file on a remote server by means of FTP.

```
device# copy startup-config ftp://admin:******@10.34.98.133//archive/startup-config_device24-08_20101010
```

Backing up the startup configuration to a USB device

When you make a backup copy of a configuration file on an attached USB device, the destination file is the file URL on the USB device. You do not need to specify the target directory. The file is automatically recognized as a configuration file and stored in the default configuration directory.

1. Enable the USB device.

   ```
   device# usb on
   USB storage enabled
   ```

2. Enter the **copy startup-config** command with the destination file name.

   ```
   device# copy startup-config usb://startup-config_slx-08_20160510
   ```

Configuration restoration

Restoring a configuration involves overwriting a given configuration file on the device by downloading an archived backup copy from an external host or from an attached USB device.

All interfaces remain online. The following parameters are unaffected:

- Interface management IP address
- Software feature licenses installed on the device
- Virtual IP address

Restoring the default configuration

This restoration procedure resets the configuration to the factory defaults. The default configuration files are always present on the device and can be restored with the **copy** command.

To restore the default configuration, perform the following procedure in privileged EXEC mode.

1. Enter the **copy default-config startup-config** command to overwrite the running configuration with the default configuration.

   ```
   device# copy default-config startup-config
   ```

2. Confirm that you want to make the change by entering Y when prompted.

   ```
   This operation will modify your startup configuration. Do you want to continue? [Y/N]: y
   ```

3. Reboot the device.

   ```
   device# reload system
   ```

Managing flash files

The Extreme device provides a set of tools for removing, renaming, and displaying files you create in the device flash memory. You can use the display commands with any file, including the system configuration files. The **rename** and **delete** commands only apply to copies of configuration files you create in the flash memory. You cannot rename or delete any of the system configuration files.

Listing the contents of the flash memory

To list the contents of the flash memory, enter the **dir** command in privileged EXEC mode.

```
device# dir
total 572
drwxr-xr-x 2  251 1011   4096 Jun  5 07:08 .
drwxr-xr-x 3  251 1011   4096 Mar 11 00:00 ..
-rw-r--r-- 1 root sys     410 Jun  3 00:56 defaultconfig.standalone
-rw-r--r-- 1 root sys     695 Jun  3 00:56 defaultconfig.cluster
-rw-r--r-- 1 root root 185650 Jun  5 09:38 startup-config
```

Deleting a file from the flash memory

To delete a file from the flash memory, enter the **delete** command with the file name in privileged EXEC mode.

```
device# delete myconfig
```

NOTE

You cannot delete a system configuration file in flash memory.

Renaming a flash memory file

To rename a file in the flash memory, enter the **rename** command with the source and destination file names in privileged EXEC mode.

```
device# rename myconfig myconfig_20101010
```

NOTE

You cannot rename a system configuration file in flash memory.

Viewing the contents of a file in the flash memory

To investigate the contents of a file in the flash memory, enter the **show file** command with the file name in privileged EXEC mode.

```
device# show file defaultconfig.cluster
vlan dot1q tag native
!
cee-map default
remap fabric-priority priority 0
remap lossless-priority priority 0
priority-group-table 15.0 pfc off
priority-group-table 1 weight 40 pfc on
priority-group-table 2 weight 60 pfc off
priority-table 2 2 2 1 2 2 2 15.0
!
!
port-profile default
vlan-profile
  switchport
  switchport mode trunk
  switchport trunk allowed vlan all
!
protocol lldp
!
!
logging auditlog class CONFIGURATION
logging auditlog class FIRMWARE
logging auditlog class SECURITY
!
end
```

NOTE
To display the contents of the running configuration, use the **show running-config** command. To display the contents of the startup configuration, use the **show startup-config** command.

Rebooting the device

CAUTION
All reboot operations are disruptive, and the commands prompt for confirmation before executing. When you reboot a device, all traffic to and from it stops. All ports on that device remain inactive until the device comes back online.

NOTE
During the boot process system initialization, configuration data (default or user-defined) are applied to the device through configuration replay.

- The **reload system** command performs a cold reboot that powers off and restarts the entire chassis. All session connections must be restarted. If the power-on self-test (POST) is enabled, POST is executed when the system comes back up.

```
    device# reload system
```

Session connection

You can connect to your device through a console session on the serial port, or through a Telnet or Secure Shell (SSH) connection to the management port or the inband port belonging to either the mgmt-vrf or default-vrf Virtual Routing and Forwarding (VRF). You can use any account login present in the local device database or on a configured authentication, authorization, and accounting (AAA) server for authentication. For initial setup procedures, use the pre-configured administrative account that is part of the default device configuration.

The device must be physically connected to the network. If the device network interface is not configured or the device has been disconnected from the network, use a console session on the serial port.

Refer to the appropriate hardware guide for information on connecting through the serial port and establishing an Ethernet connection for a console session.

Console access

A console daemon runs on the host and opens a console connection. The user can switch the console connection among host, SLXVM, and TPVM by pressing the following key sequences, respectively.

NOTE
By default, the console is connected to SLXVM.

ctrl + y + 1 - host

ctrl + y + 2 - SLXVM

ctrl + y + 3 - TPVM

Telnet

Telnet allows access to management functions on a remote networking device. Unlike SSH, Telnet does not provide a secure, encrypted connection to the device.

Telnet support is available in privileged EXEC mode on all Extreme platforms. The device supports a maximum of 32 CLI sessions. Both IPv4 and IPv6 addresses are supported.

The Telnet service is enabled by default on the device. When the Telnet server is disabled, remote access to the device is restricted. Existing inbound Telnet connections are terminated and access to the device by additional inbound connections is not allowed until the Telnet server is re-enabled. If you have admin privileges, you can disable and re-enable inbound Telnet connections from global configuration mode.

> NOTE
> Outgoing Telnet connections from the device to any remote device are not affected by disabling or enabling the Telnet server in the device.

> NOTE
> When using Telnet, the root ID is blocked and you cannot login as root.

The following features are not supported with Telnet:

- Displaying Telnet sessions
- Terminating hung Telnet sessions

> NOTE
> Telnet/SSH from standby to outside is not allowed.

Connecting to a Extreme device with Telnet

You can use the Telnet service to connect to the Extreme device.

A Telnet session allows you to access a device remotely using port 23. However, it is not secure. If you need a secure connection, use SSH.

1. Establish a Telnet session to the Extreme device from a remote device.

   ```
   client# telnet 10.17.37.157
   ```

   The example establishes a Telnet session to the device with the IP address of 10.17.37.157.

   If the device is active and the Telnet service is enabled on it, a display similar to the following appears.

   ```
   Trying 10.17.37.157...
   Connected to 10.17.37.157.
   Escape character is '^]'.
   ```

2. Once you have established the Telnet connection, you can log in normally.

   ```
   device login: admin
   Password:
   SECURITY WARNING: The default password for at least
   one default account (root, admin and user) have not been changed.

   Welcome to the Brocade SLX-OS Software
   admin connected from 10.252.24.5 using telnet on device
   device#
   ```

   > NOTE
   > The default admin login name is admin. The default user name is user. The default password for both admin and user accounts is password.

   Extreme recommends that you change the default account password when you log in for the first time. For more information on changing the default password, refer to the *Extreme SLX-OS Security Configuration Guide*.

Connecting to a remote device with Telnet

You can connect to a remote server from the device using a Telnet connection.

A Telnet session is not secure. If you need a secure connection, use SSH.

To connect to a remote server with Telnet, perform the following steps:

1. Establish a Telnet session connection to the remote device.

   ```
   device# telnet 10.20.51.68 vrf mgmt-vrf
   ```

   The example establishes a Telnet session to a device with the IP address of 10.20.51.68.

   You can override the default port by using the **port-number** *port* option. However, the device must be listening on this port for the connection to succeed.

   If the device is active and the Telnet service is enabled on it, a display similar to the following appears.

   ```
   /fabos/cliexec/vrfcmd 0 /usr/bin/telnet 10.20.51.68 23
   Trying 10.20.51.68...
   Connected to 10.20.51.68.
   Escape character is '^]'.
   ...
   device login:
   ```

2. Once you have established the Telnet connection, you can log in normally.

Shutting down and re-enabling the Telnet service

The Telnet service is enabled by default. Shutting down the Telnet service forcibly disconnects all Telnet sessions running on a device.

To shut down and then re-enable the Telnet service, perform the following steps.

1. In privileged EXEC mode, enter global configuration mode.

   ```
   device# configure terminal
   ```

2. To shut down the Telnet service on the device, enter **telnet server shutdown**.

   ```
   device(config)# telnet server shutdown
   ```

   All Telnet sessions including any currently active sessions are immediately terminated, and cannot be re-established until the service is re-enabled.

3. To re-enable the Telnet service on the device, enter **no telnet server shutdown**.

   ```
   device(config)# no telnet server shutdown
   ```

SSH

Secure Shell (SSH) allows secure access to management functions on a remote networking device. Unlike Telnet, which offers no security, SSH provides a secure, encrypted connection to the device.

SSH support is available in privileged EXEC mode on all Extreme platforms. The device supports a maximum of 32 CLI sessions. Both IPv4 and IPv6 addresses are supported.

The SSH service is enabled by default on the device. When the SSH server is disabled, remote access to the device is restricted. Existing inbound SSH connections are terminated and access to the device by additional inbound connections are not allowed until the SSH server is re-enabled. If you have admin privileges, you can disable and re-enable inbound SSH connections from global configuration mode.

> NOTE
> Outgoing SSH connections from the device to any remote device are not affected by disabling or enabling the SSH server in the device.

> NOTE
> When using SSH, the root ID is blocked and you cannot login as root.

> NOTE
> Telnet/SSH from standby to outside is not allowed.

Feature support for SSH

SSHv2 is the supported version of SSH, but not all features typically available with SSHv2 are supported on the Extreme devices.

The following encryption algorithms are supported:

- **3des-cbc** Triple-DES (default)
- **aes256-cbc** : AES in Cipher Block Chaining (CBC) mode with 256-bit key
- **aes192-cbc** : AES in CBC mode with 192-bit key
- **aes128-cbc** : AES in CBC mode with 128-bit key
- **aes256-gcm** : AES in Galios/Counter Mode (GCM) mode with 256-bit key
- **aes192-gcm** : AES in GCM mode with 192-bit key
- **aes128-gcm** : AES in GCM mode with 128-bit key
- **aes256-ctr** : AES in Counter Mode (CTR) mode with 256-bit key
- **aes192-ctr** : AES in CTR mode with 192-bit key
- **aes128-ctr** : AES in CTR mode with 128-bit key

The following Hash-based Message Authentication Code (HMAC) message authentication algorithms are supported:

- **hmac-md5** : MD5 encryption algorithm with 128-bit key (default).
- **hmac-sha1** : SHA1 encryption algorithm with 160-bit key.
- **hmac-sha2-256** : SHA2 encryption algorithm with 256-bit key.
- **hmac-sha2-512** : SHA2 encryption algorithm with 512-bit key.

The following host keys are supported:

- ssh-dsa
- ssh-rsa
- ECDSA

The following key exchange algorithms are supported:

- diffie-hellman-group-exchange-sha256
- diffie-hellman-group-exchange-sha1
- diffie-hellman-group14-sha1
- diffie-hellman-group1-sha1

SSH user authentication is performed with passwords stored on the device or on an external authentication, authorization, and accounting (AAA) server.

The following features are not supported with SSH:

- Displaying SSH sessions
- Deleting stale SSH keys

Connecting to a Extreme device with SSH

You can use SSH to connect to the Extreme device.

An SSH session allows you to access a device remotely using port 22.

1. Establish an SSH session connection to the Extreme device.

   ```
   client# ssh admin@10.17.37.157
   ```

   The example establishes an SSH session to the device with the IP address of 10.17.37.157.

2. Enter yes if prompted.

   ```
   The authenticity of host '10.17.37.157 (10.17.37.157)' can't be established.
   RSA key fingerprint is 9f:83:62:cd:55:6c:b9:e8:1d:79:ab:b4:04:f4:f6:2a.
   Are you sure you want to continue connecting (yes/no)? yes
   Warning: Permanently added '10.17.37.157' (RSA) to the list of known hosts.
   admin@10.17.37.157's password:


   SECURITY WARNING: The default password for at least
   one default account (root, admin and user) have not been changed.

   Welcome to the Brocade SLX-OS Software
   admin connected from 10.70.4.113 using ssh on device
   device#
   ```

   > NOTE
   > The default admin login name is admin. The default user name is user. The default password for both admin and user accounts is password.

   Extreme recommends that you change the default account password when you log in for the first time. For more information on changing the default password, refer to the *Extreme SLX-OS Security Configuration Guide*.

Connecting to a remote server with SSH

You can connect to a remote server from the device using the SSH (Secure Socket Handling) protocol to permit a secure (encrypted) connection.

To connect to a remote server with SSH, perform the following steps:

1. Establish an SSH connection with the login name and IP address for the remote server.

   ```
   device# ssh 10.20.51.68 -l admin vrf mgmt-vrf
   ```

   You can use the **-m** and **-c** options to override the default encryption and hash algorithms

2. Enter **yes** if prompted.

   ```
   The authenticity of host '10.20.51.68 (10.20.51.68)' can't be established.
   RSA key fingerprint is ea:32:38:f7:76:b7:7d:23:dd:a7:25:99:e7:50:87:d0.
   Are you sure you want to continue connecting (yes/no)? yes

   Warning: Permanently added '10.20.51.68' (RSA) to the list of known hosts.
   admin@10.20.51.68's password: ********

   SECURITY WARNING: The default password for at least
   one default account (root, admin and user) have not been changed.
   Welcome to the Brocade SLX-OS Software
   admin connected from 10.20.51.66 using ssh on C60_68F
   ```

Managing SSH public keys

You can import SSH public keys to establish an authenticated login for a device. You can also delete the key from the device to prevent it from being used for an authenticated login.

To manage the SSH keys, perform the following steps:

1. In privileged EXEC mode, import an SSH public key to the device.

   ```
   device# certutil import sshkey user admin host 10.70.4.106 directory /users/home40/bmeenaks/.ssh
   file id_rsa.pub login fvt
   ```

   This example imports the SSH public key for the admin user from the remote 10.70.4.106 host using the directory and file information to the key.

2. Enter the password for the user.

   ```
   Password: ***********
   ```

   When the SSH key is imported, the following message appears.

   ```
   device# 2016/01/14-10:28:58, [SEC-3050], 75,, INFO, SLX9140, Event: sshutil, Status: success, Info:
   Imported SSH public key from 10.70.4.106 for user 'admin'.
   ```

3. Delete an SSH public key from the device prevents it from being used.

   ```
   device# no certutil sshkey user admin
   ```

   This example deletes the SSH key for the admin user.

Shutting down and re-enabling the SSH service

The SSH service is enabled by default. Shutting down the SSH service forcibly disconnects all SSH sessions running on a device.

To shut down and then re-enable the SSH service, perform the following steps.

1. In privileged EXEC mode, enter global configuration mode.

   ```
   device# configure terminal
   ```

2. Shut down the SSH service on the device.

   ```
   device(config)# ssh server shutdown
   ```

   All SSH sessions are immediately terminated, and cannot be re-established until the service is re-enabled.

3. To re-enable the SSH service on the device, enter **no ssh server shutdown**.

   ```
   device(config)# no ssh server shutdown
   ```

Configuring the terminal session parameters

You can set the terminal parameters for the current session. You can also set password attributes for the length of the session and login attempts.

To set the parameters, perform the following steps:

1. In privileged EXEC mode, set the display length.

   ```
   device# terminal length 30
   ```

   This example sets the lines to be displayed on the terminal session at 30 lines.
2. Set the timeout length.

   ```
   device# terminal timeout 3600
   ```

   This example sets the timeout of 3600 seconds (60 minutes) for the terminal session.
3. Access global configuration mode.

   ```
   device# configure terminal
   ```

4. Configure the maximum login attempts to establish a session.

   ```
   device(config)# password-attributes max-retry 4
   ```

   This example sets the maximum login attempts of four to establish a session.
5. Set the maximum number of minutes after which the user account is unlocked.

   ```
   device(config)# password-attributes admin-lockout max-lockout-duration 5
   ```

   This example specifies that the user account be unlocked after 5 minutes.

The following configuration is the example of the previous steps.

```
device# terminal length 30
device# terminal timeout 3600
device# configure terminal
device(config)# password-attributes max-retry 4
device(config)# password-attributes admin-lockout max-lockout-duration 5
```

Configuring a login banner

The Extreme device can be configured to display a greeting message on user terminals as a banner when they enter the Privileged EXEC CLI level or access the device through Telnet.

Complete the following steps to set and display a banner.

1. In privileged EXEC mode, access global configuration mode.

   ```
   device# configure terminal
   Entering configuration mode terminal
   ```

2. Configure the login banner.

   ```
   device(config)# banner login "Please do not disturb the setup on this device"
   ```

   This example configure a text message on a single line by enclosing the text in double quotation marks (" ").

   The banner can be up to 2048 characters long. To create a multi-line banner, enter the **banner login** command followed by the **Esc-m** keys. Enter **Ctrl-D** to terminate the input.

   You can use the **no banner login** command to remove the banner.
3. Verify the configured banner.

   ```
   device(config)# do show running-config banner
   ```

   The configured banner is displayed.

   ```
   banner login "Please do not disturb the setup on this device"
   ```

The following example is the configuration of the previous steps.

```
device# configure terminal
Entering configuration mode terminal
device(config)# banner login "Please do not disturb the setup on this device"
```

Ethernet management interfaces

The Ethernet network interface provides management access, including direct access to the device CLI. You must configure at least one IP address using a serial connection to the CLI before you can manage the system with other management interfaces. You can either configure static IP addresses, or you can use a Dynamic Host Configuration Protocol (DHCP) client to acquire IP addresses automatically. For IPv6 addresses, both static IPv6 and stateless IPv6 autoconfiguration are supported.

NOTE
DHCP Relay Agent Information Option 82 is supported for both Layer 2 and Layer 3. For details, see the "VLANs" chapter in the *Extreme SLX-OS Layer 2 Switching Configuration Guide* and the "DHCPv4" chapter in the *Extreme SLX-OS Layer 3 Routing Configuration Guide*, respectively.

ATTENTION
Setting static IPv4 addresses and using DHCP are mutually exclusive. If DHCP is enabled, remove the DHCP client before you configure a static IPv4 address. However, this does not apply to IPv6 addresses.

Management interface setup is supported on SLXVM, TPVM, and Host. SLXVM and TPVM use eth0 as the management interface and connects to the outside network through the host physical interface, which makes it appear as a normal host to the rest of the network. SSH/Telnet access to SLXVM and TPVM are provided through the eth0 interface's IP address configured on SLXVM and TPVM respectively. On Host, Linux Bridge (br0) is configured to receive management traffic and the Host physical interface eth0 is added into this bridge. For more information on management interface IP configuration on TPVM, refer to the *Assigning a static IP address on the TPVM Linux OS* section.

Configuring a static IPv4 address on management interface from SLXVM

You can configure IPv4 static Ethernet network interface addresses in environments where the DHCP service is not available.

Before you configure a static address, connect to the device through the serial console.

To configure static Ethernet network interface addresses, perform the following steps:

1. In privileged EXEC mode, enter global configuration mode.

   ```
   device# configure terminal
   ```

2. Access the interface management mode for the management interface.

   ```
   device(config)# interface management 0
   ```

3. Disable DHCP.

   ```
   device(config-management-0)# no ip address dhcp
   ```

4. Configure the IP address for the management interface.

   ```
   device(config-Management-0)# ip address 10.24.85.81/20
   device(config-Management-0)# exit
   ```

5. Configure an IPv4 management VRF with a default gateway address.

   ```
   device(config)# vrf mgmt-vrf
   device(config-vrf-mgmt-vrf)# address-family ipv4 unicast
   device(vrf-ipv4-unicast)# ip route 0.0.0.0/0 <default-gateway-ipv4-addr>
   device(vrf-ipv4-unicast)# exit
   device(config-vrf-mgmt-vrf)# exit
   ```

The following example is the configuration of the previous steps.

```
device# configure terminal
device(config)# interface management 0
device(config-Management-0)# no ip address dhcp
device(config-Management-0)# ip address 10.24.85.81/20
device(config-Management-0)# exit
device(config)# vrf mgmt-vrf
device(config-vrf-mgmt-vrf)# address-family ipv4 unicast
device(vrf-ipv4-unicast)# ip route 0.0.0.0/0 <default-gateway-ipv4-addr>
device(vrf-ipv4-unicast)# exit
device(config-vrf-mgmt-vrf)# exit
```

Configuring a static IPv6 address on management interface from SLXVM

You can configure IPv6 static Ethernet network interface addresses in environments where the DHCP service is not available.

Before you configure a static address, connect to the device through the serial console.

To configure static Ethernet network interface addresses, perform the following steps:

1. In privileged EXEC mode, enter global configuration mode.

   ```
   device# configure terminal
   ```

2. Access the interface management mode for the management interface.

   ```
   device(config)# interface management 0
   ```

3. Configure an IPv6 address for the management interface.

   ```
   device(config-management-0)# ipv6 address fd00:60:69bc:832:e61f:13ff:fe67:4b94/64
   device(config-management-0)# exit
   ```

4. Configure an IPv6 management VRF with a default gateway address.

   ```
   device(config)# vrf mgmt-vrf
   device(config-vrf-mgmt-vrf)# address-family ipv6 unicast
   device(vrf-ipv6-unicast)# ipv6 route 0:0:0:0:0:0:0:0/0 <default-gateway-ipv6-addr>
   device(vrf-ipv6-unicast)# exit
   device(config-vrf-mgmt-vrf)# exit
   ```

The following example is the configuration of the previous steps.

```
device# configure terminal
device(config)# interface management 0
device(config-Management-0)# ipv6 address fd00:60:69bc:832:e61f:13ff:fe67:4b94/64
device(config-management-0)# exit
device(config)# vrf mgmt-vrf
device(config-vrf-mgmt-vrf)# address-family ipv6 unicast
device(vrf-ipv6-unicast)# ipv6 route 0:0:0:0:0:0:0:0/0 <default-gateway-ipv6-addr>
device(vrf-ipv6-unicast)# exit
device(config-vrf-mgmt-vrf)# exit
```

Configuring static address and gateway on Host

You can configure IPv4 and IPv6 static network interface addresses using either the host Linux shell commands or executing commands from SLXVM.

You can configure a Host IP address and a default gateway to the Host br0 interface using the **ifconfig** command and **route** command. The address configured using these commands are not persistent across reboot.

```
host# ifconfig br0 10.24.84.123 netmask 255.255.240.0
host# route add default gw 10.24.80.1
```

To configure host IP address from SLXVM, perform the following steps.

1. In privileged EXEC mode, configure the host IP address. You can configure either IPv4 address or IPv6 address.
   - Configure host IPv4 address.

     ```
     device# host ipv4 address 192.168.1.111/1
     ```

   - Configure host IPv6 address.

     ```
     device# host ipv6 address 2002::100/64
     ```

2. Configure the gateway for the Host. You can configure either IPV4 gateway or IPv6 gateway.
   - Configure IPv4 gateway.

     ```
     device# host ipv4 gateway 192.168.1.1
     ```

   - Configure IPv6 gateway.

     ```
     device# host ipv6 gateway 2002::1
     ```

3. Verify the configuration.

   ```
   device# show host ip-address
    Host IPv4 address: 192.168.1.111
    Host IPv4 Gateway: 192.168.1.1
    Host IPv6 address: 2002::100/64
    Host IPv6 Gateway: 2002::1
   ```

Displaying a management interface

You can display the information about a management interface on the device. If an IP address has not been assigned to the network interface, you must connect to the CLI using a console session on the serial port. Otherwise, connect to the device through Telnet or SSH.

```
device# show interface Management 0
interface Management 0
 line-speed actual "1000baseT, Duplex: Full"
 line-speed configured Auto
 oper-status up
 ip address "static 10.20.161.66/20"
 ipv6 ipv6-address [ "static 2620:100:0:f814:10:20:161:66/64 preferred" ]
```

Port management

The Extreme device allows the port management of the Extreme SLX 9140 and SLX 9240, and features for interface Ethernet ports.

- Extreme SLX 9140 port management includes the following:
  - Supports 54 ports in total. The first 48 ports support 10G and 25G speed (default is10G). Breakout is not supported.
  - The last 6 ports support 40G and 100G (default 100G) and breakout is supported.
  - Forward Error Correction (FEC) is supported for 25G and 100G speed.
- Extreme SLX 9240 port management includes the following:
  - Supports 32 ports in total. All ports support 40G and 100G (default 100G) and breakout is supported.
  - FEC is supported for 100G speed.

  NOTE
  The breakout configuration changes require the reload of the switch.

You can configure speed on all the interfaces and breakout on 40G/100G interfaces. Speed change configuration allows you to configure the ports in different supported speeds. The breakout mode support allows you to break out each 40G/100G port into four 10G ports.

Forward error correction support

Forward error correction (FEC) provides a data transmission error control method by including redundant data (error-correcting code) to ensure error-free transmission on a specified port or port range. FEC uses unused bits within the signaling protocol to generate an Error Correcting Code (ECC) and correct bits as needed. This correction eliminates the need to request a retransmission of the frame, an action that creates disruption and a high latency spike in the data flow. While the use of FEC adds a minimal level of latency to the ASIC processing time, it offers the advantage of ensuring highly deterministic transit times across the network for consistency of performance.

In SLX-OS 17s.1.00 release, FEC is enabled by default on 25G and 100G links (in non-breakout configuration). This feature provides a user configurable option to disable or enable FEC or set a specific mode (RS-FEC, FC-FEC or auto-negotiation) using the **fec mode** command from the interface configuration mode. The **fec mode** command must be executed on both sides of the link. Both sides need to be FEC enabled or FEC disabled together. If one side is FEC enabled and other side is FEC disabled, the link will not come up. FEC mode must be the same at both ends. You can also view the current FEC status using the **show interface** command.

Configuring breakout mode

You can configure any 40G/100G port on the Extreme SLX 9140 and SLX 9240 as four 10G or four 25G ports dynamically, without the need to reboot the switch.

Before performing the following procedure, you can view the ports in the switch and their status, use the **show ip interface brief** command:

```
Interface
Protocol                IP-Address     Vrf                Status
==================     ==========     ==============   ====================    ========
Ethernet 0/1             unassigned    default-vrf        administratively down  down
Ethernet 0/2             unassigned    default-vrf        administratively down  down
:
Ethernet 0/31:1          unassigned    default-vrf        administratively down  down
Ethernet 0/31:2          unassigned    default-vrf        administratively down  down
Ethernet 0/31:3          unassigned    default-vrf        administratively down  down
Ethernet 0/31:4          unassigned    default-vrf        administratively down  down
Ethernet 0/32            unassigned    default-vrf        administratively down  down
```

Perform the following steps to configure breakout mode on 40G/100G port:

1. Access global configuration mode.

   ```
   device# configure terminal
   ```

2. Access hardware configuration mode.

   ```
   device(config)# hardware
   ```

3. Access the connector configuration mode for the port you wish to break out.

   ```
   device(config-hardware)# connector 0/49
   ```

4. Configure the port into four 10G ports.

   ```
   device(config-connector-0/49)# breakout mode 4x10g
   %Warning: Sfp Breakout is a disruptive command.
   device(config-port-group-0/49)# Ctrl-z
   device# copy running-config startup-config
   This operation will modify your startup configuration.
   Do you want to continue? [y/n]: y
   device# reload system
   ```

   NOTE
   Dynamic breakout is supported; the user does not need to reboot the switch to execute the breakout.

   This is a disruptive command. It is not necessary to reload the switch by means of the **reload system** command. Following a breakout, the old port configuration is lost and the new port is configured with the default configuration.

5. Verify the breakout configuration.

```
device(config-connector-0/49)# do show ip interface brief | include
49
Ethernet 0/49          unassigned           default-vrf             administratively down
down

SLX(config-connector-0/49)# breakout mode 4x25g
SLX(config-connector-0/49)# do show ip interface brief | include
49
Ethernet 0/49:1        unassigned           default-vrf             administratively down     down
Ethernet 0/49:2        unassigned           default-vrf             administratively down     down
Ethernet 0/49:3        unassigned           default-vrf             administratively down     down
Ethernet 0/49:4        unassigned           default-vrf             administratively down     down
SLX(config-connector-0/49)# no breakout
SLX(config-connector-0/49)# do show ip interface brief | include
49
Ethernet 0/49          unassigned           default-vrf             administratively down     down
SLX(config-connector-0/49)#
```

6. Access privileged EXEC mode.

```
device(config-connector-0/49)# Ctrl-z
```

7. Save the running configuration.

```
device# copy running-config startup-config
This operation will modify your startup configuration.
Do you want to continue? [y/n]: y
```

8. Verify the configuration.

Configuring port speed

You can configure the speed for Extreme SLX 9140 and SLX 9240 ports.

The inserted optic must match the configured speed.

Perform the following steps to change the port speed.

1. From privileged EXEC mode, access global configuration mode.

```
device# configure terminal
```

2. Access the interface configuration mode for the port that you want to configure.

```
device(config)# interface Ethernet 0/1
```

3. Change the port speed of interface, if required.

```
device(config-if-eth-0/1# speed 10000
```

   In this example, the port speed of interface 0/1 is changed to 10G.

4. Verify the configuration.

```
device(config-if-eth-2/1)# do show running-config interface ethernet 0/1
interface Ethernet 0/1
 ....
 speed 10000
 no shutdown
!
```

The following example shows the steps in the previous configuration.

```
device# configure terminal
device(config)# interface Ethernet 0/1
device(config-if-eth-0/1)# speed 10000
```

Interface Ethernet ports

All Extreme device ports are pre-configured with default values that allow the device to be fully operational at initial startup without any additional configuration. In some configuration scenarios, changes to the port parameters may be necessary to adjust to attached devices or other network requirements.

Displaying device interfaces

The device supports Ethernet, loopback, management, and virtual Ethernet interfaces (VEs).

Enter the **show running-config interface** command to display the interfaces and their status.

The following example displays the Ethernet interfaces on the device. They are identified by the slot number, and port number, separated by forward slashes (/).

```
device# show running-config interface ethernet
interface Ethernet 0/1
 no shutdown
!
interface Ethernet 0/2
 channel-group 101 mode active type standard
 lacp timeout long
 no shutdown
!
interface Ethernet 0/3
 channel-group 101 mode active type standard
 lacp timeout short
 no shutdown
!
interface Ethernet 0/4
 shutdown
!
interface Ethernet 0/5
 shutdown
!
interface Ethernet 0/6
 shutdown
!
interface Ethernet 0/8
 channel-group 143 mode active type standard
 lacp timeout short
 no shutdown
!
interface Ethernet 0/9
 shutdown
!
```

Chassis and host names

A device can be identified by its IP address or by its host name and chassis name. You can customize the host name and chassis name. You can also change the default chassis IPv4 or IPv6 address.

Customizing chassis and host names

Extreme recommends that you customize the chassis name for each device. Some system logs identify the device by its chassis name; if you assign a meaningful chassis name, logs are more useful. You can also configure the host name.

To customize the chassis name and host name, perform the following steps:

1. In privileged EXEC mode, enter global configuration mode.

   ```
   device# configure terminal
   ```

2. Configure the chassis name.

   ```
   device(config)# switch-attributes chassis-name SLX-market1
   ```

   A chassis name can be from 1 through 30 characters long, must begin with a letter, and can contain letters, numbers, and underscore characters.

3. Configure the host name.

   ```
   device(config)# switch-attributes host-name SLX-mrkt
   SLX-mrkt(config)#
   ```

   This example changes the host name to SLX-mrkt and it is displayed in the prompt.

   A host name can be from 1 through 30 characters long. It must begin with a letter, and can contain letters, numbers, and underscore characters. The default host name is SLX.

4. Exit global configuration mode.

   ```
   SLX-mrkt(config)# exit
   ```

5. Verify the configuration.

   ```
   SLX-mrkt# show running-config switch-attributes
   switch-attributes chassis-name SLX-market1
   switch-attributes host-name SLX-mrkt
   !
   ```

The following configuration is an example of the previous steps.

```
device# configure terminal
device(config)# switch-attributes chassis-name SLX-market1
device(config)# switch-attributes host-name SLX-mrkt
SLX-mrkt(config)#
```

System clock

The operation of the device does not depend on the date and time and the Extreme device with an incorrect date and time value functions properly. However, since logging, error detection, and troubleshooting use the date and time, you should set the clock correctly.

> NOTE
> You can set the system clock if there are no NTP servers configured. Otherwise, an active NTP server, if configured, automatically updates and overrides the system clock.

Setting the clock

The Extreme device allows you to manually set the system clock. The time counter setting is retained across power cycles.

To set the clock, perform the following step:

1. In privileged EXEC mode, set the date and time.

   ```
   device# clock set 2016-07-07T12:30:45
   ```

   This example sets the time and date to 12:30:45 on July 07, 2016.
2. Enter global configuration mode.

   ```
   device# configure terminal
   ```

3. Change the time zone.

   ```
   device(config)# clock timezone America/Bogata
   ```

   This example changes the time zone to the region of America and the city of Bogata.

The following configuration is an example of the previous steps.

```
device# clock set 2016-07-07T12:30:45
device# configure terminal
device(config)# clock timezone America/Bogata
```

Management VRFs

Virtual Routing and Forwarding (VRF) is a technology that controls information flow within a network, isolating the traffic by partitioning the network into different logical VRF domains.

All management services on the Extreme device are VRF aware. The management services can select a particular VRF to reach a remote server based on a VRF. The VRFs are management (mgmt-vrf), default (default-vrf), and user defined (user-vrf).

By default, the device creates a VRF for management named mgmt-vrf and, all manageability services are accessible through this VRF. Multiple instances of IP services can be instantiated in multiple VRFs. For example, SSH can be in more than one VRF. IP services can have up to five VRF instances.

VRF reachability

The Extreme device supports the VRF reachability service. Reachability determines which VRF contains the routing information needed to reach the application servers. For example, when you configure an SSH server, you can configure the VRF information for the VRF context to resolved the SSH server route.

VRF reachability indicates the details of the VRF for servicing requests from the clients. It also indicates the clients specifying the VRF for reaching a source to ensure that the management packets are serviced or routed in a server VRF domain.

These two types of reachability services are also referred to as device-initiated and server-based services.

VRF reachability for device-initiated services

The following table lists the device-initiated services and associated commands that VRF reachability supports.

TABLE 3 Device-initiated services and associated commands that VRF reachability supports

Service	VRF-related command	Additional information
Firmware download	**firmware download** [**default-config**] **ftp** | **scp** | **sftp** [**use-vrf** *vrf-name*]...	Uses TCP IPv4or IPv6. By default, mgmt-vrf is used and a user-defined VRF is optional.
Logging server	**logging syslog-server** { **ipv4** | **ipv6** *address* } [**use-vrf** *vrf-name*]	Uses TCP UDP IPv4 or IPv6.
NTP	**ntp server** *ip-address* [**use-vrf** *vrf-name*]	Uses NTP UDP IPv4.
RADIUS	**radius-server host** *host-name* [**use-vrf** *vrf-name*]	Uses UDP IPv4 or IPv6
sFlow	[**no**] **sflow collector** *ipv4/ipv6 address port-number* [**use-vrf** *vrf-name*] [**no**] **sflow source-interface** *interface-type interface-number*	Uses sFlow UDP IPv4 or IPv6. In the case of the **sflow source-interface** command, the VRF of the specified interface is used. NOTE Any given interface can belong to only one VRF at any given time.
SNMP notification	**snmp-server host** *ip-address* [**use-vrf** *vrf-name*] **snmp-server v3host** *ip-address* [**use-vrf** *vrf-name*]	Uses SNMP UDP IPv4 or IPv6. By default, mgmt-vrf is used to send the SNMP notifications.
Support save	**copy support** { **ftp** | **scp** } [**use-vrf** *vrf-name*] ...	Uses TCP IPv4 or IPv6. By default, mgmt-vrf is used and a user-defined VRF is optional.
TACACS+	**tacacs-server host** *host-name* [**use-vrf** *vrf-name*]	Uses TCP IPv4 or IPv6

All these implementations use forward referencing of the VRF name in the **use-vrf** option, unless noted. At runtime when making the socket connection, the VRF ID by name must be resolved. If it does not resolve, it will result in a connection error.

VRF reachability for server-based services

The server services running on the Extreme device must listen to the requests in all the VRFs or a specified VRF and send the response back to the client in the same VRF where the request arrived. Thus, the services can come through any in-band interface bound to any VRF.

Each server-based service can have a maximum of six VRF instances including mgmt-vrf. The following table lists the server services and associated commands that VRF reachability supports.

> NOTE
> The SNMP server listens on all VRFs and sends the response back on the same VRF where the request arrived.
> HTTP and HTTPS are mutually exclusive on the Extreme device and both will not be enabled in different VRFs.

TABLE 4 Server-based services and associated commands that VRF reachability supports

Service	VRF-related command	Additional information
HTTP	[**no**] **http server** [**use-vrf** *vrf-name*] [**shutdown**]	By default, the HTTP service is associated and started on mgmt-vrf and default-vrf. A user-defined VRF is optional.
SSH	[**no**] **ssh server** [**use-vrf** *vrf-name*] **shutdown**	By default, the SSH service is associated and started on mgmt-vrf and default-vrf. A user-defined VRF is optional.
Telnet	[**no**] **telnet server** [**use-vrf** *vrf-name*] **shutdown**	By default, the Telnet service is associated and started on mgmt-vrf and default-vrf. A user-defined VRF is optional.

Telnet and SSH limitations and considerations

Telnet and SSH limitations and considerations are as follows:

- By default, SSH and Telnet services are associated and started on mgmt-vrf and default-vrf.
- You cannot remove mgmt-vrf from the SSH and Telnet services.
- Telnet and SSH server can be enabled on a maximum number of 6 VRFs.
- When you use the **ssh** or **telnet server shutdown** command without a user-defined VRF, the service is shut down on mgmt-vrf only.
- SSH and Telnet Services started on VRF context is applicable for both IPv4 and IPv6 addresses.
- A maximum of 32 user logins are allowed in the device. These sessions are a cumulative count of login sessions through SSH and Telnet across all the configured VRFs.

Zero Touch Provisioning

Zero Touch Provisioning (ZTP) is an automated process utilizing the DHCP process for firmware download and setting up the device configuration.

The ZTP process eliminates the need to login manually to the console to bring up the device with the correct firmware and required configuration. When the device is in the factory default configuration, ZTP can start automatically upon device bootup.

This process reduces the time taken for firmware download, device configuration . All switches download the same firmware and configuration script from the ZTP configuration file.

The following configuration considerations apply to Zero Touch Provisioning:

- ZTP is not supported for customers who do not use DHCP.
- ZTP supports only DHCPv4.
- The DHCP server must be configured with options 66 and 67 to set the ZTP configuration file.
- ZTP is triggered on a new device in the factory default state on an existing device using the **write erase** command.
- ZTP supports both in-band ports and management interface in the management VRF.
- After ZTP completes, all the in-band ports return to the default VRF state.
- To establish network connectivity, ZTP will retry indefinitely to establish a network connection among in-band ports and management interfaces until the firmware download completes, downloading all firmware packages before the device reboots.
- The interface is selected when it passes the sanity test. The order of selection is based on the response order of get option 66 or 67 during the DHCP server detection process.
- All the interfaces will be scanned in parallel to detect DHCP option 66 or 67.
- If ZTP is enabled and there is no DHCP server configured with option 66/67 for ZTP, the device will indefinitely try to discover a DHCP server. You will need to disable ZTP using the **dhcp ztp cancel** command and reboot the device before applying any configuration.
- Network connectivity over the management interface will have higher priority over in-band ports.
- ZTP is only supported in standalone mode.
- Customer configuration is supported with the Python script.
- The ZTP configuration file supports both a common setting and/or device specific settings.
- The ZTP progress is displayed on the serial console and saved in a log file.
- The DHCP client ID of the device must be set up on the device specific ZTP configuration file.

- The RASlog is disabled during the early stages of the ZTP process.
- Breakout ports are not supported because a device reboot is required.
- Only the default speeds (10G or 100G) on inband port are supported for the ZTP process.

Routing for Zero Touch Provisioning

The DHCP and FTP server may not be reachable by all the nodes in the IP Fabric. A route must be configured on the first level node with a connection to DHCP and FTP servers. ZTP must first be run on the first level using the Python script to enable iphelp to forward the traffic to the servers. The zero touch provisioning process can then run on the next level nodes. Eventually the farthest nodes can connect to the servers for ZTP.

Using zero touch provisioning

Follow these steps for zero touch provisioning in the standalone mode.

1. Establish a network connection with cable on one-to-multiple in-band ports or a management interface.
2. Power up the device in the factory default configuration or run the **write erase** command from the SLX CLI.
3. On device boot up, the Zero Touch Provisioning process performs the following actions:

   a) Disables the RASlog to the serial output.

   b) Enables in-band ports in the management VRF.

   c) Detects a DHCP server with option 66 or 67 configured over a management interface and all in-band port interfaces.

   d) Selects one interface not used before to assign the DHCP IP address.

   e) Downloads the ZTP configuration file from the FTP server.

   f) Validates the ZTP configuration file.

   g) If required ZTP performs a firmware download or reboots the device. Firmware download reboots the device automatically.

   In the current state the ZTP process will return to sub-step b. in the following situations:

   - If there is a failure in any of the above-mentioned sub-steps from b. to g.
   - If the device reboots from the CLI.
   - If the device crashes.

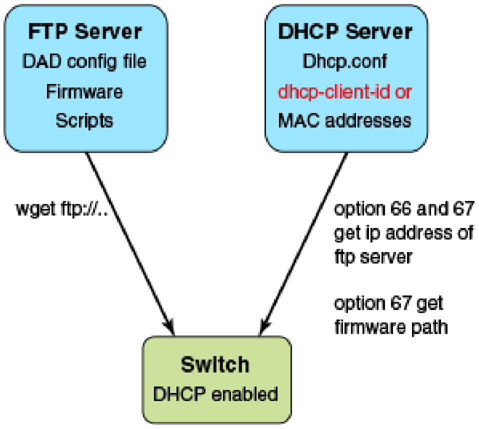
   On device bootup, the continuation of the ZTP process is indicated on the console. Wait for firmware commit to complete. If the firmware commit fails, the ZTP process aborts. If the script is enabled, the script is launched automatically.
4. Enable the RAS log.

   For more information and log outputs for canceling DHCP ZTP, refer to the *SLX-OS Command Reference Guide*.

ZTP configuration

To manage devices, the DHCP server and the FTP server must be set up to provide the environment.

FIGURE 1 ZTP configuration



DHCP server

DHCP Server version 4.2.4 is installed on Ubuntu 14.04 (Trusty) for development. The dhcpd.conf file must have option 66 (TFTP Server Name) and option 67(Filename) set for ZTP. Option 66 is used for the FTP server IP address or host name. Option 67 is used for the ZTP configuration file path.

When the device starts the DHCP process, it sends the DHCP client ID to the DHCP server to get the IP address and option 66/67. The device then downloads the ZTP configuration file from the FTP server. To set up a different ZTP configuration file for different devices, the DHCP Client ID can be used in the dhcpd.conf. Whenever dhcpd.conf is changed, the dhcpd server must be restarted.

FTP server

vsFTP server version 3.0.2 is installed on Ubuntu 14.04 (Trusty) for development. The FTP server stores the ZTP configuration file, firmware, switch configuration file or Python script. The location of these configuration files under the FTP server base directory is flexible.

ZTP configuration script

The ZTP process can run the script to set up the device configuration automatically. For now, only the Python script is supported. The script takes no parameters.

The script can automate any command line including NOS CLI and Linux commands, such as configuration download command, “copy ftp://.... Running-config”.

ZTP configuration file

The ZTP configuration file has two configuration sections; common and device specific. The common section is shared by all the switches in the IP Fabric. The settings in switch-specific section can be used for a single switch or a group of switches with the DHCP client ID. If the host_client_id string matches the starting substring of the DHCP client ID of the switch, the switch-specific section is used by the switch.

Python script example

The following example shows a sample of the Python script:

```
#!/usr/local/python/3.3.2/bin/python3
import os
import sys, getopt

def main(argv):
      log.write("apply config\n")
# change login banner
      CLI(“conf ; banner login DAD ; end”)
# config download
      CLI(“copy scp://root:brocade123@192.169.0.2/castorT.startup.cfg running-config”)
if __name__ == "__main__":
   main(sys.argv[1:])
```

FTP server configuration file

The following example shows a sample of the FTP server configuration file.

```
local_enable=YES
write_enable=YES
local_umask=022
dirmessage_enable=YES
xferlog_enable=YES
connect_from_port_20=YES
xferlog_std_format=YES
listen=NO
listen_ipv6=YES

pam_service_name=vsftpd
userlist_enable=NO
tcp_wrappers=YES

#dad settings
anonymous_enable=YES
no_anon_password=YES
anon_root=/var/ftp
delay_failed_login=30
max_clients=100
anon_max_rate=8388608
```

DHCP server configuration file

The following example shows a sample of the DHCP server configuration file dhcp.conf

```
#ddns-update-style standard;
ddns-update-style interim;
ddns-ttl 600;
ignore client-updates; # Overwrite client configured FQHNs
ddns-domainname "infralab.com.";
ddns-rev-domainname "in-addr.arpa.";

option ntp-servers 192.168.0.2;
option domain-name-servers 192.168.0.2;
option domain-name "infralab.com";
option domain-search "infralab.com";

default-lease-time 600;
max-lease-time 7200;

authoritative;

log-facility local7;

key "rndc-key" {
    algorithm hmac-md5;
    secret "dtBgNTAoqZmwV5c4SueybjOvhe6OIqgac1uQrzGBv5O4X4nIEBEEGWRf0lCnbFhuIJXGExNBjDdNSqgBMeNI8w==";
};

subnet 192.168.0.0 netmask 255.255.255.0 {
  range 192.168.0.100 192.168.0.200;
  option subnet-mask 255.255.255.0;
  option broadcast-address 192.168.0.255;
  zone 0.168.192.in-addr.arpa. {
        primary 192.168.0.2;
        key "rndc-key";
  }
  zone infralab.com. {
        primary 192.168.0.2;
        key "rndc-key";
  }
}
# cluster switches
group{
  option bootfile-name "/config/unified-cfg.min";
  option tftp-server-name "192.168.0.2";
  option routers 192.168.0.2;

  # sw0
  host sw0 {
    option dhcp-client-identifier = "BROCADE##SLX9240##EXG3342L00V";
    hardware ethernet 52:54:00:0E:95:8B;
    fixed-address 192.168.0.90;         # fixed ip address
```

ZTP configuration file

The following has three sections - common, switch specific 1 and switch specific 2.

```
version=3
date=04/29/2016
supported_nos=17s.1.00 17r.1.00 17r.1.01 17s.1.0017

  common_begin
  vcsmode=SA
  scriptcfgflag=2 #0-config file only, 1-script only, 2 both
  script=/script/ztp.py
  fwdir=/fw/slxos17s.1.00a_bld02
  common_end

  # model SXL9140 hosts
  host_client_id=BROCADE##SLX9140
  script=/script/Frreddomlic.py
  startup=/config/freedomlic.cfg
  host_end

  # model SXL9240 hosts
  host_client_id=BROCADE##SLX9240
  script=/script/dad2_new.py
  startup=/config/cedarlic.cfg
  host_end

  # freedom
  host_client_id=BROCADE##SLX9140##EXH3327M014
  startup=/config/freedom_ospf.cfg
  script=/script/FreedomZTP.py
  host_end

  # cedar
  host_client_id=BROCADE##SLX9240##EXG3326M00P
  startup=/config/cedar_ospf.cfg
  script=/script/dad1_new.py
  host_end
```

ZTP configuration file definitions

The following table contains the ZTP configuration file definitions.

TABLE 5 ZTP configuration file definitions

Variable description	Description
version	Only Version 3+ is supported.
date	The last modified date.
supported_nos	The release firmware version supporting the ZTP configuration file.
host_client_id, host_end	Host_client_id marks the beginning of the section host_end marks the end. User could set up the switch specific section with full dhcp client id or its prefix. Ex. *host_client_id=BROCADE##SLX9140##EXH3319M01J* NOTE EXH3319M01J is the device serial number. *script=/script/dad1new.py* *host_end*
common_begin, common_end	The setting in the section will be shared by all switches.
vcsmode=SA	Only standalone mode is supported.
vcstimeout	If omitted, the default is 60 minutes. The timeout to wait for ZTP to complete configuration file download or Python script. If the configuration download process or Python script has issues, the zero touch provisioning process will stop the download after timeout and claim that ZTP is complete. You will need to increase the timeout if configuration download or Python script takes a long time to complete.
fwdir	Firmware path in the FTP server. For example Fwdir=/fw/ slxoss17s. 1.00_bld34. If base directory of the FTP server is /var/ftp, then the absolute path of firmware in ftp server is located at /var/ftp/fw/slxoss17s. 1.00_bld34.
startup	The path to the switch configuration file in the FTP server. If omitted, the switch will take the default configuration. The value can be "default" or user configuration file.
scriptcfgflag	The default is 0, when not specified. The meaning of the value is: 0 - only use startup, **script** is ignored 1 - only use script, **startup** is ignored 2 - uses both script and config file.
script	The switch configuration Python script file.

Example of Zero Touch Provisioning in a two node topology

The following section describes Zero Touch Provisioning IP routing in a two node topology.

In figure 1 switch 1 Eth 0/8 has direct connection to the DHCP or FTP server. Switch 1 acts as a router for switch 2 to reach the DHCP or FTP server. A default route on switch 1 is configured on the server for traffic sent from the DHCP server to reach switch 2 (see default route below). External access to DHCP sever is on Eth 0. There are two configurations for switch 1 - one is set up on Eth 0/8 by the DHCP server for ZTP to establish a connection to the DHCP server to download the ZTP configuration file, the other is set up by the Python script to configure the switch 1 as router Eth 0/8 and Eth 0/3 for switch 2. DHCP relay is configured on Eth 0/3 in switch 1 for DHCP requests from switch 2. Switch 1 Eth 0/8 and Eth 0/3 have to be in different subnets.

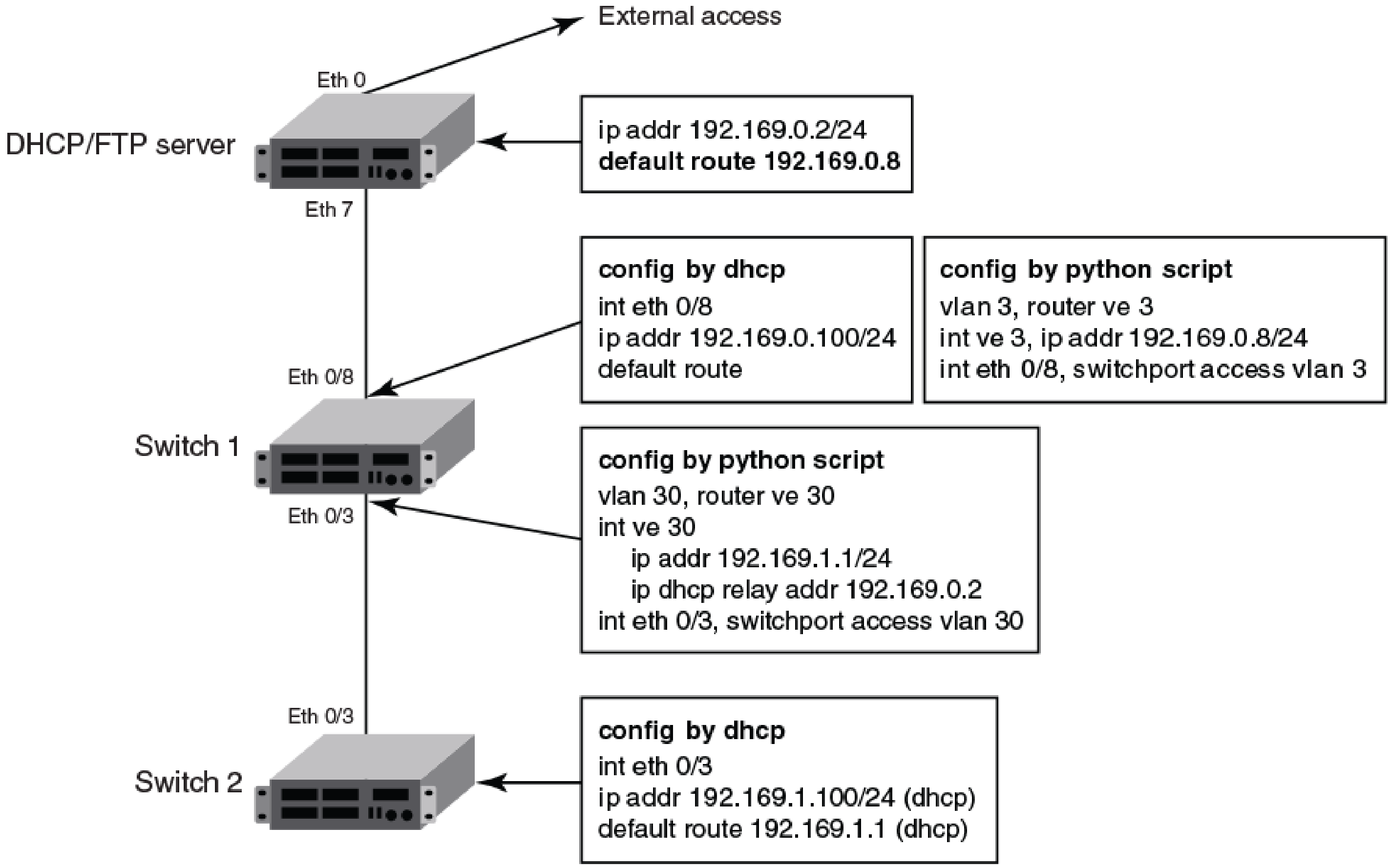
NOTE

The Python script for switch 1 should be manually tested to verify the routing set up before starting ZTP for switch 2.

DHCP server configuration will have two subnet address pools based on the DHCP client id. "level_1" for Switch 1 and "level_2" for switch 2.

```
class "level_1" {
  match if option dhcp-client-identifier = "BROCADE##SLX9140##EXH3319M01J"; <EXH3319M01J is the device 
serial number>
}
class "level_2" {
  match if option dhcp-client-identifier = "BROCADE##SLX9140##EXH3314M00L"; <EXH3314M00L is the device 
serial number>
}
subnet 192.169.0.0 netmask 255.255.255.0 {
  pool {
        allow members of "level_1";
        range 192.169.0.100 192.169.0.200;
  }
  option bootfile-name "/config/ztp.cfg";
  option tftp-server-name "192.169.0.2";
  option routers 192.169.0.2;
  option subnet-mask 255.255.255.0;
  option broadcast-address 192.169.0.255;
}
subnet 192.169.1.0 netmask 255.255.255.0 {
  pool {
        allow members of "level_2";
        range 192.169.1.100 192.169.1.200;
  }
  option bootfile-name "/config/ztp.cfg";
  option tftp-server-name "192.169.0.2";
  option routers 192.169.1.1; ip address as routers in level 2 subnet
  option subnet-mask 255.255.255.0;
  option broadcast-address 192.169.1.255;
}
```

FIGURE 2 ZTP two node topology



Configuration file example

```
version=3
date=04/29/2016
supported_nos=17s.1.00 17r.1.00

  common_begin
  vcsmode=SA
  fwdir=/bld/Nightly_nos_fusion_davinci_dev_160822_0600/dist
  scriptcfgflag=2 #0-config file only, 1-script only, 2 both
  script=/script/ztp.py
  common_end

  # model SXL9140 hosts
  host_client_id=BROCADE##SLX9140  for Switch 2
  startup=/config/startup.cfg
  host_end

  # switch 1 as router node
  host_client_id=BROCADE##SLX9140##EXH3319M01J
  startup=/config/freedom1_ospf.cfg
  host_end
```

Configuration flow

a. Enable ZTP on the inband port only by creating file /etc/fabos/ztp.enable with “echo \-I > /etc/fabos/ztp.enable”

b. Run the **write erase** from the CLI on both switch 1 and switch 2 simultaneously.

c. Switch 1 behaves as a single node ZTP.

d. Switch 2 is held up with detecting the DHCP server with option 66 or 67 until ZTP on switch 1 succeeds, so that the static route is configured successfully. If switch 1 fails, switch 2 will wait indefinitely.

SLX-OS and Linux Shell Interoperability

Overview

The SLX-OS supports interoperability between the SLX-OS CLI and the SLXVM Linux shell.

As an SLX-OS user with admin permissions, you can perform the following tasks:

- Running permitted Linux commands and scripts from the SLX-OS CLI
- Accessing the SLXVM Linux shell, and:
  - Running permitted Linux commands and scripts. However, if you have access to the root password, you can then escalate your permissions, by using the **su root** Linux command.
  - Running SLX-OS configuration and show commands.
  - Running scripts that contain multiple SLX-OS commands.

Limitations

- By default, only the Bash shell is supported. With Linux root permissions, you can install a different shell, such as the C shell or KornShell. However, shell-activity logging is supported only for the Bash shell.
- If you open multiple Bash sessions, the Linux shell timeout is applicable only on the current Bash session.
- If you run Linux commands as part of the script or through a file, the device logs the script or file execution. It does not log the commands.
- If you use the **cli_run** command to execute SLX-OS CLI **show** commands from the shell, pagination is not supported, and commands that require user input are also not supported.
- At the SLX-OS CLI, a window resizing issue occurs when you execute Linux commands such as **top** using the **oscmd** command. Extreme recommends that you execute these commands from the Linux shell.
- Although as an SLX-OS admin, you have permissions to run the following commands from the Linux shell, you do not have permissions to run them—from the SLX-OS CLI—appended to the **oscmd** command.
  - **bash**
  - **script**
  - **vi**
  - **vim**
- Do not modify SLX-OS user accounts from the Linux shell. For information on modifying user accounts, refer to the *Extreme SLX-OS Security Configuration Guide for SLX 9140 and SLX 9240*.

Executing Linux shell commands from SLX-OS

You can execute a Linux command from the SLX-OS CLI, appended to **oscmd**.

In the following example, the Linux **ps -ef** command lists the process status.

```
device#  oscmd ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 Jul24 ?        00:00:04 /sbin/init
root         2     0  0 Jul24 ?        00:00:00 [kthreadd]
root         3     2  0 Jul24 ?        00:00:00 [migration/0]
root         4     2  0 Jul24 ?        00:00:03 [ksoftirqd/0]
root         5     2  0 Jul24 ?        00:00:00 [migration/1]
root         6     2  0 Jul24 ?        00:00:03 [ksoftirqd/1]
root         7     2  0 Jul24 ?        00:00:00 [migration/2]
root         8     2  0 Jul24 ?        00:00:02 [ksoftirqd/2]
root         9     2  0 Jul24 ?        00:00:00 [migration/3]
root        10     2  0 Jul24 ?        00:00:02 [ksoftirqd/3]
root        11     2  0 Jul24 ?        00:00:00 [migration/4]
root        12     2  0 Jul24 ?        00:00:02 [ksoftirqd/4]
root        13     2  0 Jul24 ?        00:00:00 [migration/5]
root        14     2  0 Jul24 ?        00:00:03 [ksoftirqd/5]
root        27     2  0 Jul24 ?        00:00:00 [cpuset]
root        28     2  0 Jul24 ?        00:00:01 [khelper]
root        31     2  0 Jul24 ?        00:00:00 [netns]
root        34     2  0 Jul24 ?        00:00:00 [async/mgr]
root       270     2  0 Jul24 ?        00:00:00 [sync_supers]
root       272     2  0 Jul24 ?        00:00:00 [bdi-default]

...

root      8kblockd/6]182     1  0 Jul24 ?        00:00:00 /usr/sbin/inetd
root      8237     1  0 Jul24 ?        00:00:00 /usr/sbin/sshd
admin    27536 27535  0 04:19 pts/4    00:00:00 ps -ef
```

Executing scripts from SLX-OS

From the SLX-OS CLI, you can execute scripts that you copied to flash memory or created in the SLXVM Linux shell.

Downloading a script to the SLX-OS device

After writing and testing a user-defined script file, copy it from an accessible network location to the flash memory of the SLX-OS device.

```
device# copy ftp://MyUserID:MyPassword@10.10.10.10//<copy_script.sh> flash://copy_script.sh
```

After copying the script to the device, verify that the script file is displayed with the list of files in the flash memory of the device.

```
device# dir
total 24
drwxr-xr-x   2 root     sys          4096 Oct 26 15:22 .
drwxr-xr-x   3 root     root         4096 Oct  1  1970 ..
-rw-r--r--   1 root     root         1051 Oct 24 16:09 copy_script.sh
-rw-r--r--   1 root     root          207 Oct 24 16:09 create_vlans.py
-rw-r--r--   1 root     sys           557 Oct 26 10:37 defaultconfig.novcs
-rw-r--r--   1 root     sys           778 Oct 26 10:37 defaultconfig.vcs

1922789376 bytes total (828317696  bytes free)
```

If the copied script does not have executable permissions, you need to assign executable permissions from the SLXVM Linux shell. Note that you need root access for this action, as described in "Escalating Linux permissions to root."

```
[root@SLX]# cd /var/config/vcs/scripts/
[root@SLX]# chmod 755 copy_script.sh
```

```
[root@SLX]# ls -lart copy_script.sh
-rwxr-xr-x 1 root root 1051 Oct 24 16:09 copy_script.sh
```

You can also display the contents of a script file.

```
device# show file copy_script.sh
```

Creating scripts in the Linux shell

You can create scripts by using the Linux shell **vi** editor, as in the following example.

```
device# start-shell
Entering Linux shell for the user: admUser
[admUser@SLX]# vi create_script.sh
```

After you write the script, make sure that it exists in the `/fabos/users/admin` directory and is executable under Linux.

```
[admUser@SLX]#pwd
/fabos/users/admin
```

Running scripts from the SLX-OS CLI

You can run scripts directly from SLX-OS CLI. Enter **oscmd** followed by the name of the script.

```
device# oscmd my_script
```

Accessing the Linux shell from SLX-OS

With admin-level permissions, you can access the SLXVM Linux shell from the SLX-OS CLI, by using the **start-shell** command.

> NOTE
> Inside the SLXVM Linux shell, you can execute commands that do not require root permissions. To escalate your permissions, refer to "Escalating Linux permissions to root".

1. To access the SLXVM Linux shell, enter the **start-shell** command.

   ```
   device# start-shell
   Entering Linux shell for the user: admUser
   [admUser@SLX]#
   ```

2. Enter Linux commands and run scripts as needed. You can also run SLX-OS commands from the Linux shell.
3. To exit the shell and return to the SLX-OS CLI, enter **exit**.

   ```
   [admUser@SLX]# exit
   ```

   Upon exiting, the following message appears and you return to the SLX-OS CLI prompt.

   ```
   exit
   Exited from Linux shell
   device#
   ```

Executing SLX-OS commands from the Linux shell

From the SLXVM Linux shell, you can execute a single SLX-OS command or a file that contains a series of such commands.

1. To execute an SLX-OS command, enter the Linux **cli_run -c** command.

   ```
   [admUser@SLX]# cli_run -c "show ip interface brief" | grep Port-channel > /tmp/interface
   ```

   In the previous example, the output of **show ip interface brief** is redirected to the `/tmp/interface` file.
2. Display the contents of the file to verify the redirection.

   ```
   [admUser@SLX]# cat /tmp/interface
   Port-channel 1        unassigned        administratively down     down
   Port-channel 2        unassigned        administratively down     down
   ```
3. To execute a file containing multiple SLX-OS commands, enter the Linux **cli_run -f** command.

   ```
   [admUser@SLX]# cli_run -f slxcli_cmd_file > newfile
   ```

   In this example, `slxcli_cmd_file` contains the following commands:

   ```
   [admUser@SLX]# cat slxcli_cmd_file
        show ssh server status
        conf t
        router bgp
        local-as 23
        capability as4-enable
   ```

   NOTE
   Make sure that each command is on a new line.
4. Display the contents of the target file to verify that it contains the redirected output.

   ```
   [admUser@SLX]# cat newfile
   Welcome to the Brocade SLX-OS Software
   admin connected from 127.0.0.1 using console on SLX
   SLX# show ssh server status | nomore
   VRF-Name: mgmt-vrf      Status: Enabled
   VRF-Name: default-vrf   Status: Enabled
   device# conf t
   Entering configuration mode terminal
   Current configuration users:
   admin console (cli from 10.70.4.183) on since 2017-01-31 05:49:59 terminal mode
   device(config)# router bgp
   device(config-bgp-router)# local-as 23
   device(config-bgp-router)# capability as4-enable
   device(config-bgp-router)#
   ```

Escalating Linux permissions to root

In the SLXVM Linux shell, you can escalate your default admin access to root access (password protected).

CAUTION
A user with SLXVM Linux-shell root permissions can—unintentionally or maliciously—execute commands that can render the SLX-OS inoperable.

1. From the SLX-OS CLI prompt, enter **start-shell** to access the SLXVM Linux shell.

   ```
   device# start-shell
   Entering Linux shell for the user: admUser
   [admUser@SLX]#
   ```

   You can now execute commands that do not require root permissions.

2. To escalate your permissions, enter the Linux **su root** command.

   ```
   [admUser@SLX]# su root
   Password:
   ```

3. Enter the root password.

   ```
   Password:
   ```

   After successful login, the following warning is displayed:

   ```
   We trust you have received the usual lecture from the local System
   Administrator. It usually boils down to these three things:

       #1) Respect the privacy of others.
       #2) Think before you type.
       #3) With great power comes great responsibility.
   [root@SLX]#
   ```

4. Enter Linux commands and run scripts as needed.

   You can also run SLX-OS commands from the Linux shell.

5. To exit root level and return to the default SLXVM Linux shell, enter **exit**.

   ```
   [root@SLX]#  exit
   exit
   [admUser@SLX]#
   ```

6. To exit the default SLXVM Linux shell and return to the SLX-OS CLI, enter **exit**.

   ```
   [admUser@SLX]#  exit
   exit
   Exited from Linux shell
   device#
   ```

Saving and appending show command output to a file

For output of a **show** command saved or appended to a file, the **oscmd** command enables you to display the file.

1. Save the **show** command output to a file.

   ```
   device# show ssh server status | save status
   ```

   In this example, the **show ssh server status** output is saved to the status file.

2. Display the contents of the file with the Linux **cat** command.

   ```
   device# oscmd cat status
   VRF-Name: mgmt-vrf      Status: Enabled
   VRF-Name: default-vrf   Status: Enabled
   ```

3. Append the **show** output to an existing file.

   ```
   device# show interface ethernet 0/1 | last 5 | append status
   ```

   In this example, the **show interface ethernet 0/1** output is appended to the status file.

4. Display the contents of the file with the Linux **cat** command.

   ```
   device# oscmd cat status
   VRF-Name: mgmt-vrf      Status: Enabled
   VRF-Name: default-vrf   Status: Enabled
   Rate info:
       Input 0.000000 Mbits/sec, 0 packets/sec, 0.00% of line-rate
       Output 0.000000 Mbits/sec, 0 packets/sec, 0.00% of line-rate
   Time since last interface status change: 22:24:39
   ```

Logs of Linux shell activities

By default, SLX-OS logs users entering the SLXVM Linux shell, commands executed in that shell, and users exiting from the Linux shell back to the SLX-OS CLI.

Linux shell user entry and exit logs

SLX-OS uses RASLOG to log entries when the user enters and exits the Linux shell. If you configure a remote Syslog server, the same logs can also be seen on that server.

From privileged EXEC mode, use the **show logging raslog** command to display the RASLOG entries.

- When a user enters the Linux shell, the **show logging raslog** command displays an SH-1001 message.

  ```
  device# show logging raslog

  2016/06/25-06:42:54, [SH-1001], 1547, M1 | Active, INFO, SLX,  SLXVM Linux shell login information:
  User [admUser]. Login Time : Sat Jun 25 06:42:54 2016
  ```

- When a user exits the Linux shell, the **show logging raslog** command displays an SH-1002 message.

  ```
  device# show logging raslog

  2016/06/25-06:43:59, [SH-1002], 1548, M1 | Active, INFO, SLX, Event: exit, Status: success, Info:
  User [admUser] successfully exited from SLXVM Linux shell. Exit Time: Sat Jun 25 06:43:59 2016
  ```

  NOTE
  An SH-1003 message indicates failure to log in to the Linux shell.

Linux shell command execution logs

Command activities at the Linux shell are logged locally in the `/var/log/shell_activity.log` file and remotely on a Syslog server.

When a user executes a command at the Linux shell, the `shell_activity.log` file includes SH-1005 messages:

```
[admUser@SLX]# tail -f /var/log/shell_activity.log

Jun 25 06:43:28 10.24.12.113 [log value="SLXVM Linux shell activity log"] [timestamp value="Sat Jun 25
06:43:28 GMT 2016"] [msgid value="SH-1005"] [severity value="INFO"] [user value="admUser" desc="username"]:
pwd
Jun 25 06:43:29 10.24.12.113 [log value="SLXVM Linux shell activity log"] [timestamp value="Sat Jun 25
06:43:29 GMT 2016"] [msgid value="SH-1005"] [severity value="INFO"] [user value="admUser" desc="username"]:
ls
```

NOTE
The `/var/log/shell_activity.log` file is rotated every thirty minutes if it goes over 2 MB in size. The old version of the file is compressed; a maximum of four rotated files can exist at the same time.

Configuring remote logging of Linux shell activities

By default, SLX-OS logs Linux shell commands both locally (in `/var/log/shell_activity.log`) and remotely, on the Syslog server.

From SLX-OS CLI, you can perform the following tasks to control the logging of commands executed at the Linux shell to a remote Syslog server. These tasks do not affect the local logging.

NOTE
Changes of the **log-shell stop** and **log-shell start** commands are applicable only on new Linux shell sessions.

1. To disable remote logging, enter **log-shell stop**.

   ```
   device# log-shell stop
   ```

   Local logging of user activities continues.
2. To restart remote logging, enter **log-shell start**.

   ```
   device# log-shell start
   ```

3. To check the remote logging status, enter **log-shell status**.

   ```
   device# log-shell status
   ```

   When remote logging is enabled, the following message is displayed.

   ```
   Linux shell activity logging : Enabled
   ```

Guest OS for TPVM

TPVM, or Third-Party Virtual Machine, is a general server that resides on Extreme SLX-OS devices. The guest OS that accesses it is different from SLX-OS.

TPVM enables users to run applications such as Docker Container, syslog server, SNMP server, and RESTful applications, among others. TPVM runs as a separate, independent virtual machine (VM), sharing the host CPU, RAM, hard disk drive, and management resources with SLX-OS.

Extreme devices are shipped with the TPVM firmware, but it is not installed by default.

TPVM installation and management

TPVM installation and a variety of management details are described.

Installation

The TPVM firmware is shipped with the Extreme device and it is stored in the SLXVM filesystem. The user executes the **tpvm install** command to install it. During the subsequent firmware download, the TPVM firmware in the SLXVM filesystem is updated to be the same as the SLXVM firmware version. However, the user must run the **tpvm install** command again to re-install it. Otherwise, the currently running TPVM image is not changed.

> ATTENTION
> The installation is disruptive, and any data saved on the TPVM partition is erased. You must save any data manually before executing the **tpvm install** command.

Initially, the TPVM firmware stored in the SLXVM /tftpboot directory matches the version created by the netinstall process. However, in subsequent firmware downloads the TPVM firmware can be from different distributions.

After the installation, you can start and stop the image by means of the **tpvm start** and **tpvm stop** commands, respectively. To start the TPVM image automatically in subsequent reboots, use the **tvpm auto-boot enable** command. (TVPM may not come up if there are any issues with booting SLXVM.)

Once the TPVM image is running, you can download user-specific applications by copying them to the TPVM partition and starting them manually.

To uninstall the TPVM image and release its resources, use the **tpvm uninstall** command.

> ATTENTION
> When TPVM is re-installed, any user applications are deleted.

Resources and default XML configuration

TPVM runs the Linux 3.19 64-bit kernel. It has 4 GB of RAM, with a second SSD of 64 GB dedicated to it, and it shares one of the four CPU cores with SLXVM. Note that the host runs a different version of Linux (3.14 64-bit kernel). XML is used as the file format for storing the total configuration, including domain, network, storage, and other elements, and is used by QEMU/Libvirt to instantiate TPVM. That file cannot be edited by the user.

Resource usage

From the Linux host's perspective, TPVM appears as a process. All commands to check TPVM resources and control TPVM are executed from the host and have administrative (root) restrictions.

To check TPVM resource utilization, use the following commands:

- **ps aux | grep TPVM**
- **top –p** *pid*
- **cat /proc/** *pid* **/***

Console access

A console daemon runs on the host and opens a console connection to TPVM. The user can switch the console connection among host, SLXVM, and TPVM by pressing the following key sequences, respectively.

- Host: **Ctrl + y + 1**
- SLXVM: **Ctrl + y + 2**
- TPVM: **Ctrl + y + 3**

NOTE
By default, the console is connected to SLXVM.

TPVM can be accessed through the eth0 (management) interface. The eth0 interface connects to the outside network through the host physical interface, which makes it appear as a normal host to the rest of the network. SSH or Telnet access to TPVM is provided through the IP address of the eth0 interface configured on TPVM.

IP and MAC address management

The assignment of a TPVM IP address to a management interface uses DHCP by default. However, the user can also assign a static address and a default gateway to the TPVM eth0 interface by using the **ifconfig** command. See Assigning a static IP address on the TPVM Linux OS on page 72.

Communication between TPVM and SLXVM

An HTTP RESTful interface is provided for accessing the running configuration, interface statistics, interface states, and all system-related information. TPVM is prepackaged with a RESTful client to support, for example, cURL (command line RESTful access utility), to extract information from SLXVM. cURL uses HTTP methods (such as GET, PUT) to extract and modify configuration information so long as the requesting user is authenticated correctly.

The following example shows a simple request format from TPVM:

curl -s -u admin:*password* **http://***ip-addr***/rest/config/running/ . . .**

In addition to cURL, Advanced Rest Client, a Chrome-based RESTful client application, is prepackaged inside TPVM and is accessed through a browser interface.

Both HTTP and HTTPS secure access are enabled.

NFS mount support

TVPM supports the NFS mount of an external drive to support the storage of captured data.

Packages support and applications

No configuration is needed on TPVM to support RESTful access with cURL and Google-chrome.

In addition, TPVM comes with Tcpdump, Tshark, and Wireshark prepackaged to support packet capture.

Users or administrators can use the **apt-get** command with options to upgrade, update, purge, or remove (to downgrade to an older version. In addition, applications can be downloaded to provide a development environment that allows users or administrators to build their own applications, development tools (gdb, glibc (e.g. ANSI-C and POSIX), and gcc) for C/C++ . Similarly, python development tools can also be downloaded.

Containers

The following container binaries have been tested with TPVM:

- Docker container: docker-1.13.0
- Linux container: LXC 1.0

The above binaries do not come prepacked with TPVM. Use the **wget** or **apt-get** commands to install, upgrade, or downgrade Docker and Linux container binaries or packages in TPVM.

Frequently used commands

Command	Description
auto-boot	Enables or disables start of TPVM at next boot.
disk	Supports TPVM disk operations.
install	Installs TPVM.
password	Updates the root password.
show	Displays TPVM status, TPVM IP address and TPVM disk usage.
start	Starts TPVM.
stop	Stops TPVM.
uninstall	Uninstalls TPVM.

Using the tpvm command

The **tpvm** command is available at the SLX-OS CLI on a device, but it does not require root privileges.

The **tpvm** command, in privileged EXEC mode, allows you to manage TPVM with a variety of subcommands that do the following:

- Install, start, stop, and uninstall TPVM

- Specify the default behavior when SLX-OS boots
- Display TPVM status

The following example installs TPVM if it is not already installed.

```
device# tpvm install
Installation starts. To check the status, run 'tpvm status'
```

The following example uninstalls TPVM if it is installed.

```
device# tpvm uninstall
uninstallation succeeds
```

The following example starts TPVM if it is not running.

```
device# tpvm start
start succeeds
```

The following example stops TPVM if it is running.

```
device# tpvm stop
stop succeeds
```

The following example starts TPVM at the next boot of SLX-OS (without the need for the **start** keyword).

```
device# tpvm auto-boot enable
auto-boot enable succeeds
```

The following example prevents TPVM from starting at the next boot of SLX-OS.

```
device# tpvm auto-boot disable
auto-boot disable succeeds
```

> **NOTE**
> In this case, the **tpvm start** command is required to enable TPVM.

The following example displays TPVM status.

```
device# show tpvm status
TPVM is running, and AutoStart is disabled on this host.
```

The following examples add a disk to TPVM.

```
device# tpvm disk add name auto size 10g
disk add succeeds

device# tpvm disk add name vdd size 512m
disk add succeeds
```

The following examples remove a disk from TPVM.

```
device# tpvm disk remove name auto
'umount' is needed before this disk is removed. Continue? [y/n]: y
disk remove succeeds

device# tpvm disk remove name vdc
'umount' is needed before this disk is removed. Continue? [y/n]: y
disk remove succeeds
```

The following examples display disk information.

```
device# show tpvm disk
Value for 'name' : all
disk: vdb
Capacity: 10.00 GiB
Allocation: 196.00 KiB

total:
Capacity: 100.00 GiB
Allocation: 10.00 GiB
Available: 90.00 GiB

device# show tpvm disk name vdb
disk: vdb
Capacity: 10.00 GiB
Allocation: 196.00 KiB

total:
Capacity: 100.00 GiB
Allocation: 10.00 GiB
Available: 90.00 GiB
```

The following example displays the IP address.

```
device# show tpvm ip-address
IPv4:
eth0 10.24.7.149
IPv6:
eth0 fe80::629c:9fff:fe01:fe43
eth1 fe80::7:d0ff:fe02:100
```

NOTE
The **show ip-address** subcommand requires the qemu-guest-agent package on TPVM. If that package is removed, this subcommand fails.

The following example changes the root password for TPVM.

```
device# tpvm password
root password: ****
re-enter root password: ****
password succeeds
```

Insight Interface

The insight interface port provides data plane to mirror the traffic towards Third-Party Virtual Machine (TPVM) applications.

Third-Party Virtual Machine (TPVM), is a VM instance that resides on Extreme SLX-OS devices at management board. The mirrored traffic is analyzed using any data packet capturing tool such as tcpdump.

The following table details the support for the insight interface on Extreme SLXS platforms.

TABLE 6 Insight interface support on Extreme SLX 9140 platform

Extreme SLXS device	Support for insight interface
SLX 9140	Online card ports: 54 (48x10/25G + 6x40/100G)

Insight interface supported features

Insight interface supports the following features:

- Mirror/SPAN/ACL—Ability to apply any port-based or policy-based rule
- Bidirectional mode—Ability to connect to or from any TPVM from any switch front-panel port

Only one of the above can be configured at a time.

Insight interface datapath

The user enables the insight Interface by means of the **insight enable** command, which creates an internal high-speed data path between the switch and TPVM, and is available as an endpoint (port) at both ends. This port appears as "eth1" and is accessible as a port-channel on the switch when the insight interface is enabled.

In addition, a set of rules is created to mirror traffic to the insight Interface and is available at the TPVM insight Interface endpoint.

In the bidirectional-mode use case, both insight interface endpoints can be assigned Layer 3 addresses, to provide bidirectional reachability to TPVM from any front-panel port on the switch.

Enabling the insight interface from the switch endpoint

Do the following to configure insight interface on a port-channel.

1. Enter global configuration mode.

   ```
   device# configure terminal
   ```

2. In global configuration mode, specify a port-channel.

   ```
   device(config)# interface port channel 20
   ```

3. To enable the insight interface, enter the **insight enable** command.

   ```
   device(config-port-channel-20)# insight enable
   ```

4. In interface subtype configuration mode, enable the interface by means of the **no shutdown** command.

   ```
   device(config-port-channel-20)# no shutdown
   ```

5. Use any of the following commands, as in the respective examples below:
   - **show ip interface brief**
   - **show interface port channel**
   - **show port-channel summary**
   - **show port-channel detail**
   - **show running-config interface port-channel**
   - **shutdown/no shutdown**

```
device# show ip interface brief
Flags:  I - Insight Enabled
Interface             IP-Address        Vrf                   Status                  Protocol
==================    ==========        ==================    ====================    ========
Port-channel 20(I)    unassigned                              up                      up
Ethernet 0/1          unassigned        default-vrf           administratively down   down
Ethernet 0/2          unassigned        default-vrf           administratively down   down
...

device# show interface port-channel 20
Port-channel 20 is up, line protocol is up
Insight mode is enabled
Hardware is AGGREGATE, address is 609c.9f5a.a358
    Current address is 609c.9f5a.a358
Interface index (ifindex) is 671088660
Minimum number of links to bring Port-channel up is 1
MTU 1548 bytes
LineSpeed Actual     : 1000 Mbit
Allowed Member Speed : 1000 Mbit
…

device# show port-channel summary
Flags:  D - Down                P - Up in port-channel (members)
        U - Up (port-channel)   * - Primary link in port-channel
        S - Switched            I - Insight Enabled
        M - Not in use. Min-links not met
===== =============== ========== ===============
Group Port-channel    Protocol   Member ports
===== =============== ========== ===============
20    Po 20   (UI)    None

device# show port-channel detail
 Static Aggregator: Po 20
 Aggregator type: Standard
 Insight mode is enabled
 Number of Ports: 1

device# show running-config interface Port-channel
interface Port-channel 20
 insight enable
 no shutdown
```

Configuring mirroring for data analytics

This example illustrates both port-based mirroring and flow-based SPAN mirroring with ACLs.

TPVM must be installed and running, and its management port (eth0) must have a Layer 3 IP address (either static or for DHCP) in order to access the TPVM shell. This is required to configure any step related to the insight interface endpoint (port) eth1 interface, or even to execute the **tcpdump** command on eth1. To access the TPVM console shell, enter **Ctrl + y + 3** on the console terminal.

Configuring port-based mirroring

Follow these steps to configure port-based mirroring for data analytics.

For prerequisites, see Configuring mirroring for data analytics on page 61.

This can be used to analyze one or more pipeline flows. No Layer 2 or Layer 3 configuration is required at either end.

1. Prepare the system to receive packets by entering the following example SSH commands at the management eth0 port.

   ```
   $ ssh root@<tpvm_mgmt_eth0_ipv4_addr>
   password:
   $ ifconfig eth1 promisc
   $ tcpdump -I eth1
   ```

   Once mirroring starts, packets are dumped on the console in promiscuous mode.
2. In global configuration mode at the switch, specify a port-channel interface.

   ```
   device(config)# interface port-channel 20
   ```

3. Ensure that insight interface is enabled, by means of the **insight enable** command.

   ```
   device(config-Port-channel-20)# insight enable
   ```

4. Enable the interface, by means of the **no shutdown** command.

   ```
   device(config-Port-channel-20)# no shutdown
   ```

5. Create a monitor session, by means of the global **monitor session** command.

   ```
   device(config)# monitor session 1
   ```

Configuring flow-based SPAN mirroring

Follow these steps to configure flow-based SPAN mirroring with inbound ACLs.

For prerequisites, see Configuring mirroring for data analytics on page 61.

1. Enter global configuration mode.

   ```
   device# configure terminal
   ```

2. Create a monitor session.

   ```
   device(config)# monitor session 4
   device(config-session-4)# destination port-channel 20
   ```

3. In global configuration mode, create an extended ACL.

   ```
   device(config)# mac access-list extended a1
   device(conf-macl-ext)# permit any host MAC_address
   ```

4. Use this ACL in a class-map.

   ```
   device(config)# class c1
   device(config-classmap)# match access-group a1
   ```

5. Use the class-map in a policy-map.

   ```
   device(config)# policy-map p1
   device(config-policymap)# class c1
   device(config-policymap-class)# span session 1
   ```

6. Apply the policy-map to the source port.

   ```
   device(config)# interface port-channel 20
   device(config-Port-channel-20)# service-policy in p1
   ```

You must configure an appropriate hardware TCAM profile to support ACLs. Refer to Configuring TCAM profiles to support ACLs on page 63.

Configuring TCAM profiles to support ACLs

Follow these steps to configure an example TCAM profile.

To support ACL-based rules, the correct hardware TCAM profiles are required.

> ATTENTION
> The system must be rebooted following this configuration.

1. Enter global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **hardware** command to enable hardware configuration mode.

   ```
   device(config)# hardware
   ```

3. Enter the **profile tcam** command for the SLX 9140 to specify an ACL profile.

   ```
   device(config-hardware)# profile tcam l2-acl-l3-acl
   ```

   This optimizes resources for ingress, egress L2 ACLs and ingress IPv4 and IPv6 ACLs.

4. Enter the **reload system** command to reboot the device.

   ```
   device# reload system
   ```

Configuring bidirectional TPVM reachability

Complete the following steps to configure bidirectional TPVM reachability.

For prerequisites, see Configuring mirroring for data analytics on page 61.

This enables reachability to and from TPVM from any front-panel port on the switch. Note the following:

- The insight interface and both endpoints (switch and TPVM) are assigned a Layer 3 address in the same subnet (for example, 10.0.0.x/24).
- The insight interface switch endpoint's Layer 3 address is assigned as a virtual Ethernet (VE) interface to a port-channel.

- The insight interface TPVM endpoint's Layer 3 address is assigned statically. For DHCP-based assignment, refer to Configuring support for DHCP on page 65.

1. At the TPVM end, prepare the system to receive packets by entering the following SSH commands at the management eth0 port, to specify a subnet and route. The following are typical Ubuntu/Linux commands to configure the TPVM endpoint (port) eth1 with a Layer 3 address/mask and route entry.

   a) Configure an IPv4 address and mask on eth1.

   ```
   $ sudo ifconfig eth1 10.0.0.0.100 netmask 255.255.255.0
   ```

   b) Configure a route entry for TPVM.

   ```
   $ sudo route add default gw 10.0.0.1 eth1
   ```

2. Specify a VLAN.

   ```
   device(config)# vlan 100
   ```

3. In VLAN configuration mode, enter the **router-interface ve** command and specify a VE interface.

   ```
   device(config-vlan-100)# router-interface ve 100
   ```

   Ths attaches (binds) the router interface to a VLAN, creating a Layer 3 interface.

4. Exit VLAN configuration mode.

   ```
   device(config-vlan-100)# exit
   ```

5. In global configuration mode, specify the VE interface.

   ```
   device(config)# interface ve 100
   ```

6. In interface subtype configuration mode, specify an IP address and mask.

   ```
   device(config-if-Ve-100)# ip address 10.0.0.1/24
   ```

7. Exit interface subtype configuration mode.

   ```
   device(config-if-Ve-100)# exit
   ```

8. In global configuration mode, specify a port-channel interface.

   ```
   device(config)# interface port-channel 20
   ```

9. Enable the interface, by means of the **no shutdown** command.

   ```
   device(config-Port-channel-20)# no shutdown
   ```

10. Ensure that insight interface is enabled, by means of the **insight enable** command.

    ```
    device(config-Port-channel-20)# insight enable
    ```

11. Enter the **switchport** command to put the interface in Layer 2 mode.

    ```
    device(config-Port-channel-20)# switchport
    ```

12. Enter the **switchport mode access** command to configure the interface as access.

    ```
    device(config-Port-channel-20)# switchport mode access
    ```

13. Enter the **switchport access vlan** command and specify a VLAN.

    ```
    device(config-Port-channel-20)# switchport access vlan 100
    ```

14. Enable the interface.

    ```
    device(config-if-Ve-100)# no shutdown
    ```

Configuring support for DHCP

Follow these steps to configure support for DHCP by assigning a Layer 3 address to the TPVM insight interface endpoint (port) eth1.

Insight interface connectivity is like a point-to-point Ethernet connection between two machines. The SLX 9140 does not support any internal DHCP Server. Consequently, a TVPM DHCP request for Layer 3 address assignment cannot be served. To make DHCP Server reachable on another front panel port on the switch (for example, Ethernet 0/38), VLAN flooding provides a solution.

The following task assumes that the an appropriate port-channel is enabled for insight interface, a VLAN is established, and a port-channel is mapped to the VLAN.

1. In global configuration mode, specify an Ethernet interface.

    ```
    device(config)# interface ethernet 0/38
    ```

2. Enter the **switchport** command to put the interface in Layer 2 mode.

    ```
    device(conf-if-eth-0/38)# switchport
    ```

3. Enter the **switchport mode access** command to configure the interface as access.

    ```
    device(conf-if-eth-0/38)# switchport mode access
    ```

4. Enter the **switchport access vlan** command and specify a VLAN.

    ```
    device(conf-if-eth-0/38)# switchport access vlan 100
    ```

5. Enable the interface.

    ```
    device(conf-if-eth-0/38)# no shutdown
    ```

Configuring rate limiting for the insight interface

Complete the following steps to configure egress traffic shaping and ingress traffic policing for TPVM.

Rate limiting from the switch endpoint (or port, the insight-enabled port-channel) on the insight interface is recommended. This can be enabled from the TPVM insight interface endpoint eth1 also, by means of typical Linux QoS/terminal commands; however, this will unnecessarily load the host CPU.

Egress traffic from the switch towards TPVM can be rate limited to keep it even below the physical link bandwidth of the Insight interface. This may also help reduce the TPVM CPU load, although packets may be dropped in such egress traffic. Similarly (for example, in bidirectional mode), if required, the ingress traffic from TPVM toward the switch may be policed to a lower rate.

1. Complete the following substeps to configure egress traffic shaping.

   a) Enter global configuration mode.

   ```
   device# configure terminal
   ```

   b) Enter the **policy-map** command to create an outbound policy map, in this example "shaper_out".

   ```
   device(config)# policy-map shaper_out
   ```

   c) In policymap configuration mode, enter the **class** command and specify the "default" class.

   ```
   device(config-policymap)# class default
   ```

   d) In policymap class configuration mode, enter the **shape** command and specify a rate in Kbps.

   ```
   device(config-policymap-class)# speed 50000
   ```

   e) Exit to global configuration mode.

   ```
   device(config-policymap-class)# exit
   ```

   f) In global configuration mode, specify a port-channel interface.

   ```
   device(config)# interface port-channel 20
   ```

   g) In interface subtype configuration mode, enter the **service-policy** command and specify the "shaper_out" policy map for outbound traffic.

   ```
   device(config-Port-channel-20)# service-policy out shaper_out
   ```

2. Complete the following steps to configure ingress policing.
   a) Enter global configuration mode.

      ```
      device# configure terminal
      ```

   b) Enter the **policy-map** command to create an inbound policy map, in this example "policy_in".

      ```
      device(config)# policy-map policy_in
      ```

   c) In policymap configuration mode, enter the **class** command and specify the "default" class.

      ```
      device(config-policymap)# class default
      ```

   d) In policymap class configuration mode, enter the **police cir** command and specify parameters.

      ```
      device(config-policymap-class)# police cir 800000 cbs 500000 eir 100000 ebs 100000
      ```

   e) Exit to global configuration mode.

      ```
      device(config-policymap-class)# exit
      ```

   f) In global configuration mode, specify a port-channel interface.

      ```
      device(config)# interface port-channel 20
      ```

   g) In interface subtype configuration mode, enter the **service-policy** command and specify the "policy_in" policy map for inbound traffic.

      ```
      device(config-Port-channel-20)# service-policy in policy_in
      ```

Supported third-party applications, packages, and hardware

Note the following basic support and limitations for TPVM.

Third-party applications

- Packet capture applications
- RESTful support to access SLX-OS

Third-party packages

- Packages installed by default on the TPVM
- RESTful application: Chrome browser – GUI RESTful access
- RESTful application: cURL – Command line RESTful access
- Tcpdump: Command line packet-capture utility
- Tshark: Command line packet-capture utility
- Wireshark: GUI packet-capture utility
- Datadog or Splunk – External analytics software service

NOTE
Extreme SLX-OS provides support for built-in applications (third-party packages shipped with the SLX-OS) that are listed in the *Extreme SLX-OS Management Configuration Guide*.

- Extreme is committed to providing limited support for the interoperability of these applications with Extreme application interfaces.
- Extreme does not provide support for the application configuration, functionality, or deficiencies.
- Extreme does not provide any support for applications not listed in the *Extreme SLX-OS Management Configuration Guide*.

Hardware

The TPVM has 4 GB of RAM and 240GB of solid-state disk (SSD) memory, which limits the amount of data captured through packet capture applications. To overcome this limitation, Extreme provides support for the Network File System (NFS) mount of an external drive.

Docker containers

This section addresses the installation and management of Docker containers.

Installation

Complete the following steps to install the latest version of Docker under TPVM.

1. Install and export the missing Gnu Privacy Guard (GPG) key.

```
gpg --keyserver hkp://keyserver.ubuntu.com:80 --recv 1397BC53640DB551
gpg --export --armor 1397BC53640DB551 | apt-key add -
```

2. Add `[arch=amd64]` before `http://d1.google.com/linux/chrome/deb/ stable main` In the `/etc/apt/sources.list.d/google-chrome.list` file.
3. Update the repository: **apt-get -y update**
4. Install the CA certificates: **apt-get -y install apt-transport-https ca-certificates**
5. Add the new GPG key for Docker: **apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80 --recv-keys 58118E89F3A912897C070ADBF76221572C52609D**
6. Create or update the `/etc/apt/sources.list.d/docker.list` file to contain the following string: `deb https://apt.dockerproject.org/repo ubuntu-trusty main`
7. Update the Advance Packaging Tool (APT) package index: **apt-get -y update**
8. Purge the old repository: **apt-get -y purge lxc-docker**
9. Verify that APT is pulling the Docker engine from the proper repository: **apt-cache policy docker-engine**
10. Install the Docker engine: **apt-get -y install docker-engine**
11. Start Docker service: **service docker start**
12. Verify the Docker installation by running the "hello-world" image: **docker run hello-world**

Docker Linux binaries can also be obtained from the following URL by means of the **wget** command:

- Docker script: https://get.docker.com/

After downloading the binaries, you extract the archive by using the **tar -xvzf docker-latest.tgz** command, which puts the binaries in a directory named /docker in the current location.

Depending upon the Docker engine version, you may have to set "execute" permission on the Docker daemon, by using the **chmod +x docker** command.

Docker requires the binaries to be installed in your host's $PATH. For example, you can move these binaries to /usr/bin.

Starting Docker

Start Docker by using the**service docker start &** command.

The docker daemon always runs as the root user, and binds to a UNIX socket instead of to a TCP port. By default, that UNIX socket is owned by the user "root", and therefore is accessible by means of the **sudo** or **root** commands.

If you (or your Docker installer) create a UNIX group called "docker" and add users to it, then the docker daemon makes the ownership of the UNIX socket read/writable by the docker group when the daemon starts. The docker daemon must always run as the root user, but if you run the docker client as a user in the docker group, then you do not need to add **sudo** to all the client commands.

Upgrading Docker

To upgrade your manual installation of Docker, first kill the docker daemon by using the **killall docker** command.

Running and monitoring Docker containers

You can start, stop, and monitor Docker containers by using the **docker** command. The following table lists frequently used commands.

TABLE 7 Frequently used docker commands

Command	Description
docker help	Lists supported Docker commands
docker --version	Displays the Docker version
docker create *image*	Creates a new container
docker run -i -t ubuntu /bin/bash	Instantiates a Docker container with bash shell and console connection
docker ps -a	List all Docker containers
docker attach *container-id*	Attach to a running Docker container
docker start *container-id*	Start/restart a particular Docker container
docker stop *container-id*	Stop a particular Docker container
docker rm $(docker ps -a -q	Delete all Docker containers
docker rmi $(docker images -q	Delete all Docker images

Linux containers

This section addresses the installation of Linux and creating and managing containers.

Installation

LXC 1.0 was tested with TPVM. The lxc package can be installed as root by means of the **apt-get install lxc** command.

Your system will then have all the LXC commands, all LXC templates, and also the python3 binding should you want to script LXC.

Creating containers

You can create privileged or unprivileged containers. (Only privileged containers were tested for this release.)

Privileged containers are containers created by root and running as root. They can be created as follows: **sudo lxc-create -t download -n** *my-container*

This creates a new privileged container "my-container" on TPVM, using an image based on the download template. The download template contains a list of distributions, versions, and architectures to choose from. Good example templates would be "ubuntu" and "trusty".

Running and monitoring containers

Once the container is created, start it by using the **lxc-start -n** *my-container* **-d** command.

You can then confirm its status by using either of the following commands:

- **lxc-info -n** *my-container*
- **lxc-ls -f**

You can access the *my-container* console by using the**lxc-console -n** *my-container* command.

You get a shell inside the container by using the **lxc-attach -n** *my-container* command.

Once done, you can stop the container by using the **lxc-stop** command, and remove it by using the **lxc-destroy** command:

- **lxc-stop -n** *my-container*
- **lxc-destroy -n** *my-container*

To confirm connectivity, attach to one of the containers and check network access by pinging a server accessible from the host:

- **lxc-attach -n lxc1**
- **ping external-server**

Utilities installation and management

cURL

cURL is a command-line RESTful access utility. The following table lists useful installation and management commands.

TABLE 8 cURL commands

Command	Description
sudo apt-get update	Updates any dependency packages
sudo apt-get install curl	Installs the latest cURL package, or specifies a previous version number
sudo apt-get upgrade curl	Upgrades to the latest cURL package

Google-chrome

Google-chrome is a graphical user interface RESTful access utility. The following table lists useful installation and management commands

TABLE 9 Google-chrome commands

Command	Description
sudo apt-get update	Updates any dependency packages
sudo apt-get install google-chrome-stable	Installs the latest Google-chrome package, or specifies a previous version number
sudo apt-get upgrade google-chrome-stable	Upgrades to the latest Google-chrome package

ifconfig and route

The **Ifconfig** and **route** utility commands are available by default in the Ubuntu/Debian package, and can be used without any additional package installation. The following table lists useful command options.

TABLE 10 ifconfig and route command options

Command	Description
ifconfig eth0 *Net-IP-Addr* **netmask** *<Net-IP-Addr-Mask>*	Checks interface status and statistics
route add -net *Net-IP-Addr* **netmask** *Net-IP-Addr-Mask* **gw** *Gw-IP*	Adds a route
route add default gw *GW-IP*	Adds a default gateway

Ethtool

The Ethtool utility is used to get device information. The following table lists useful command options.

TABLE 11 Ethtool commands

Command	Description
sudo apt-get update	Updates any dependency packages
sudo apt-get install ethtool	Installs the latest Ethtool package, or specifies a previous version number
sudo apt-get upgrade ethtool	Upgrades to the latest Ethtool package

Tcpdump

Tcpdump is a command line utility that is used for packet capture by means of libpcap. The following table lists useful command options.

TABLE 12 Tcpdump commands

Command	Description
sudo apt-get update	Updates any dependency packages
sudo apt-get install tcpdump	Installs the latest Tcpdump package, or specifies a previous version number
sudo apt-get upgrade tcpdump	Upgrades to the latest Tcpdump package

Tshark

Tshark is a command line utility from the Wireshark community that is used for packet capture by means of libpcap. The following table lists useful command options.

TABLE 13 Tshark commands

Command	Description
sudo apt-get update	Updates any dependency packages
sudo apt-get install tshark	Installs the latest Tshark package, or specifies a previous version number
sudo apt-get upgrade tshark	Upgrades to the latest Tshark package

Wireshark

Wireshark is a GUI-based packet capture utility that is used for packet capture by means of libpcap. The following table lists useful command options.

TABLE 14 Wireshark commands

Command	Description
sudo apt-get update	Updates any dependency packages
sudo apt-get install wireshark	Installs the latest Wireshark package, or specifies a previous version number
sudo apt-get upgrade wireshark	Upgrades to the latest Wireshark package

Assigning a static IP address on the TPVM Linux OS

To use a static IP address, you add the static method for the interface in the file /etc/network/interfaces.

1. Obtain your current address, network mask, and broadcast address.

   ```
   device# ifconfig
   ...
   eth0 Link encap:Ethernet HWaddr 00:0a:21:ff:45:2a
    inet addr:10.10.10.0 Bcast:10.10.10.255 Mask:255.255.255.0
   ...
   ```

   This example displays the first Ethernet interface identified as eth0.

2. Obtain your gateway and network address.

   ```
   device# route -n
   Kernel IP routing table
    Destination  Gateway      Genmask        Flags  Metric  Ref  Use  Iface
    0.0.0.0      10.10.10.2   0.0.0.0        UG     100     0    0    eth0
    172.16.77.0  0.0.0.0      255.255.255.0  U      0       0    0    eth0
   ```

   Use flags **u** and **g** for the route gateway. The other IP address is the network IP address.

3. Open the interfaces file.

   ```
   device# sudo nano /etc/network/interfaces
   ```

   Nano is the GNU version of the Pico text editor. Use the editor of your choice.

4. Find the DHCP settings in the /interfaces file. They will appear as text similar to the following example.

```
...
auto eth0
iface eth0 inet dhcp
...
```

5. Replace the settings shown in step 4 with the following settings.

```
...
auto eth0
iface eth0 inet static
address 10.0.0.100
netmask 255.255.255.0
gateway 10.0.0.0
...
```

   This example configures the first Ethernet interface, identified as eth0.

6. Save the file and exit to the command prompt.
7. Make sure that your name server IP address is your gateway IP address.

```
device# sudo nano /etc/resolv.conf
```

8. Restart the networking components.

```
device# sudo /etc/init.d/networking restart
```

9. If you want this as a permanent change, remove the DHCP client so it can no longer assign dynamic IP addresses.

```
device# sudo apt-get remove dhcp-client
```

10. Verify connectivity.

```
device# ping www.brocade.com
```

11. Manually enable the interface.

```
device# sudo ifup eth0
```

Network Time Protocol (NTP)

Network Time Protocol overview

Network Time Protocol (NTP) maintains uniform time across all devices in a network. The NTP commands support the configuration of an external time server to maintain synchronization among all local clocks in a network.

To keep the time in your network current, it is recommended that each device have its time synchronized with at least one external NTP server.

Date and time settings

Extreme devices maintain the current date and time inside a battery-backed real-time clock (RTC) circuit. Date and time are used for logging events. Device operation does not depend on the date and time; a device with incorrect date and time settings can function correctly. However, because the date and time are used for logging, error detection, and troubleshooting, you should set them correctly.

Time zone settings

The time zone settings have the following characteristics:

- The setting automatically adjusts for Daylight Savings Time.
- Changing the time zone on a device updates the local time zone setup and is reflected in local time calculations.
- By default, all devices are in the Greenwich Mean Time (GMT) time zone (0,0).
- System services that have already started will reflect the time zone changes only after the next reboot.
- Time zone settings are not affected by NTP server synchronization.

NTP server

An NTP server provides the correct network time on your device using the Network Time Protocol (NTP). NTP can be used to synchronize the time on devices across a network.

An NTP time server is used to obtain the correct time from an external time source and adjust the local time in each connected device. When NTP server functionality is enabled, the NTP server starts listening on the NTP port for client requests and responds with the reference time. Up to five server addresses in IPv4 or IPv6 format can be configured. When you configure multiple NTP server addresses, the first obtainable address is set as the active NTP server. If there are no reachable time servers, then the local device time is the default time until a new active time server is configured.

The NTP server is stateless and will not maintain any NTP client information. Network time synchronization is guaranteed only when a common external time server is used by all devices.

NTP server authentication

You can create an authentication key for the purpose of authenticating an external Network Time Protocol (NTP) server.

The time kept on a device is a critical resource, so it is highly recommended to use the encrypted authentication mechanism. An authentication key is configured with a key identifier and secret key strings. The key is shared by the client (device) and an external NTP server by associating the key to an NTP server. NTP supports a symmetric key scheme. The scheme uses an MD5 keyed hash algorithm.

Configuring NTP

After setting the date and time on a device, the local time on a device can be synchronized with an Network Time Protocol (NTP) server.

The date and time are set in privileged EXEC mode and only have to be configured once per device because the value is written to nonvolatile memory. After the basic time information is set up, an NTP server is configured to allow the local time to be synchronized across the network.

1. Set the date and time for the device.

   ```
   device# clock set 2016-08-06T12:15:00
   ```

2. Access global configuration mode.

   ```
   device# configure terminal
   ```

3. Set the time zone for the device.

   ```
   device(config)# clock timezone America/Los_Angeles
   ```

4. Return to privileged EXEC mode.

   ```
   device(config)# exit
   ```

5. Display the local date, time, and time zone for the device.

   ```
   device# show clock
   2017-02-09 12:15:00 America/Los_Angeles
   ```

6. Enter global configuration mode.

   ```
   device# configure terminal
   ```

7. Synchronize the local time with an external source accessible from a user-specified VRF named myvrf.

   ```
   device(config)# ntp server 192.168.10.1 use-vrf myvrf
   ```

8. Exit to global configuration mode.

   ```
   device(config-server-192.168.10.1/myvrf)# exit
   ```

9. Exit to privileged EXEC mode.

   ```
   device(config)# exit
   ```

10. Display the active NTP server IP address.

    ```
    device# show ntp status
    active ntp server 192.168.10.1
    ```

In the following example, the date, time and time zone are set on a device and verified. The local device is configured to synchronize the local time with an external NTP server at a specific IP address, accessible from a user-specified VRF named myvrf.

```
device# clock set 2017-02-09 12:15:00
device# configure terminal
device(config)# clock timezone America/Los_Angeles
device(config)# exit
device# show clock
2017-02-09 12:15:00 America/Los_Angeles
device# configure terminal
device(config)# ntp server 192.168.10.1 use-vrf myvrf
device(config-server-19.168.10.1/myvrf)# exit
device(config)# exit
device# show ntp status
active ntp server 192.168.10.1
```

Authenticating an NTP server

An authentication key can be created for the purpose of authenticating an external Network Time Protocol (NTP) server.

This task demonstrates how to create an authentication key and associate the key to an NTP server.

1. Enter global configuration mode.

   ```
   device# configure terminal
   ```

2. Create an authentication key ID and key string.

   ```
   device(config)# ntp authentication-key 33 md5 check
   ```

   Up to five NTP authentication keys can be configured and each key ID must be unique.
3. Synchronize the local time with an external source, an NTP server, accessible by the management VRF.

   ```
   device(config)# ntp server 192.168.10.1
   ```

4. Associate the key to the NTP server.

   ```
   device(config-server-192.168.10.1/mgmt-vrf)# key 33
   ```

5. Exit to global configuration mode.

   ```
   device(config-server-192.168.10.1/mgmt-vrf)# exit
   ```

6. Exit to privileged EXEC mode.

   ```
   device(config)# exit
   ```

In the following example, an authentication key with an ID of 33 is created and the local time on the device is synchronized with an external NTP server at the IP address of 192.168.10.1.

```
device# configure terminal
device(config)# ntp authentication-key 33 md5 check
device(config)# ntp server 192.168.10.1
device(config-server-192.168.10.1/mgmt-vrf)# key 33
device(config-server-192.168.10.1/mgmt-vrf)# exit
device(config)# exit
```

Displaying the active NTP server

Information about the currently active NTP server can be displayed. When an NTP server has been configured, the server IP address is displayed. If an NTP server is not configured or the server is unreachable, the output displays LOCL (for local device time).

Only the local NTP server information is displayed.

NTP server status when an NTP server is not configured

The following example shows the local device NTP status when an NTP server is not configured:

```
device# show ntp status
active ntp server is LOCL
```

NTP server status when an NTP server is configured

The following example shows the status of a configured NTP server:

```
device# show ntp status
active ntp server is 192.168.10.1
```

Precision Time Protocol (PTP)

PTP overview

IEEE-1588 Precision Time Protocol (PTP) is a protocol designed to synchronize real-time clocks in the nodes of a distributed system that communicate on a local area network. PTP provides submicrosecond accuracy to distributed control applications such as system automation, telemetry, navigation, telecommunications, and consumer electronics. The original version is IEEE 1588-2002. PTP Version 2 (PTPv2) is IEEE 1588-2088, with improvements in accuracy, precision, and robustness. PTPv2 is not backward compatible with the original version.

A PTP system is a distributed, networked system consisting of a combination of PTP and non-PTP devices. PTP devices are ordinary clocks, boundary clocks, end-to-end transparent clocks, peer-to-peer transparent clocks, and management nodes. Non-PTP devices are bridges, routers, and other infrastructure devices.

The PTP protocol is a distributed protocol that specifies how the real-time clocks in the system synchronize with each other. These clocks are organized into a *master/slave* synchronization hierarchy, where a *grandmaster clock* at the top of the hierarchy determines the reference time for the entire system. The synchronization is achieved by exchanging PTP timing messages, with the slaves using the timing information to adjust their clocks to the time of their master in the hierarchy.

The protocol executes within a logical scope called a *domain*. A given physical network and individual devices connected to the network can be associated with multiple domains. The time established within one domain by the protocol is independent of the time in other domains.

PTP use cases

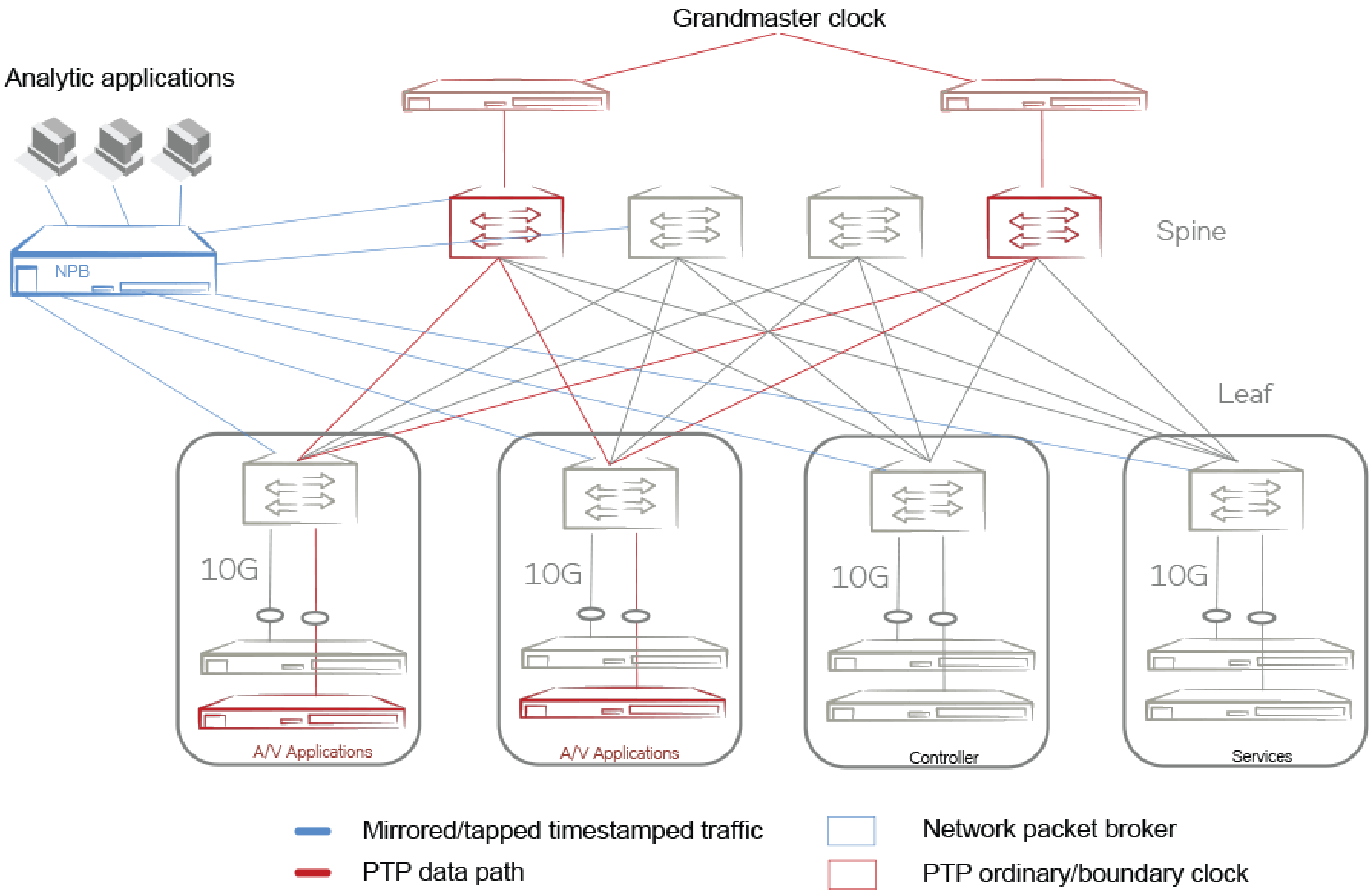
PTP in n IP Clos topology has two use cases: telemetry/analytics and audio/video applications. Both applications require a fabric to distribute synchronized and precise timing to switches and server farms in data center in a scalable fashion.

To support telemetry, a switch node requires a synchronized time source with which to timestamp the transit traffic. The timestamped traffic is then mirrored to an application that analyzes and logs the network's congestion and responsiveness on the basis of the embedded timestamp. The resulting data are then used to optimize the fabric's performance and assist network resource planning.

To support audio/video deployment in a data center, the fabric distributes synchronized timing to applications running at server farms attached to top-of-rack (TOR) devices. This allows applications to timestamp their video, audio, and subtitle streams and provides time synchronization for editing tools.

The following illustrates a PTP deployment in an IP Clos spine/leaf topology.

FIGURE 3 PTP deployment in an IP Clos topology



Example applications can include analytics such as forensics, intrusion detection systems (IDS) and network security, and application performance management (APM).

PTP clock types

IEEE 1588 defines three clock device types: ordinary, boundary, and transparent.

Ordinary clock: A clock that has a single PTP port in a domain and maintains the timescale used in the domain. It may serve as a source of time (as a master clock) or may synchronize to another clock (as a slave clock).

Boundary clock: A clock that has multiple PTP ports in a domain and maintains the timescale used in the domain. All ports share the same local clock. Each port behaves like a port in an ordinary clock and has an independent port-state machine. There is one slave port and all other ports are master ports. The slave port receives the best clock information from an upstream clock device. The master port propagates the best clock information to all downstream nodes in the distribution hierarchy. A boundary clock port terminates all ingress synchronization messages and does not forward them to other ports.

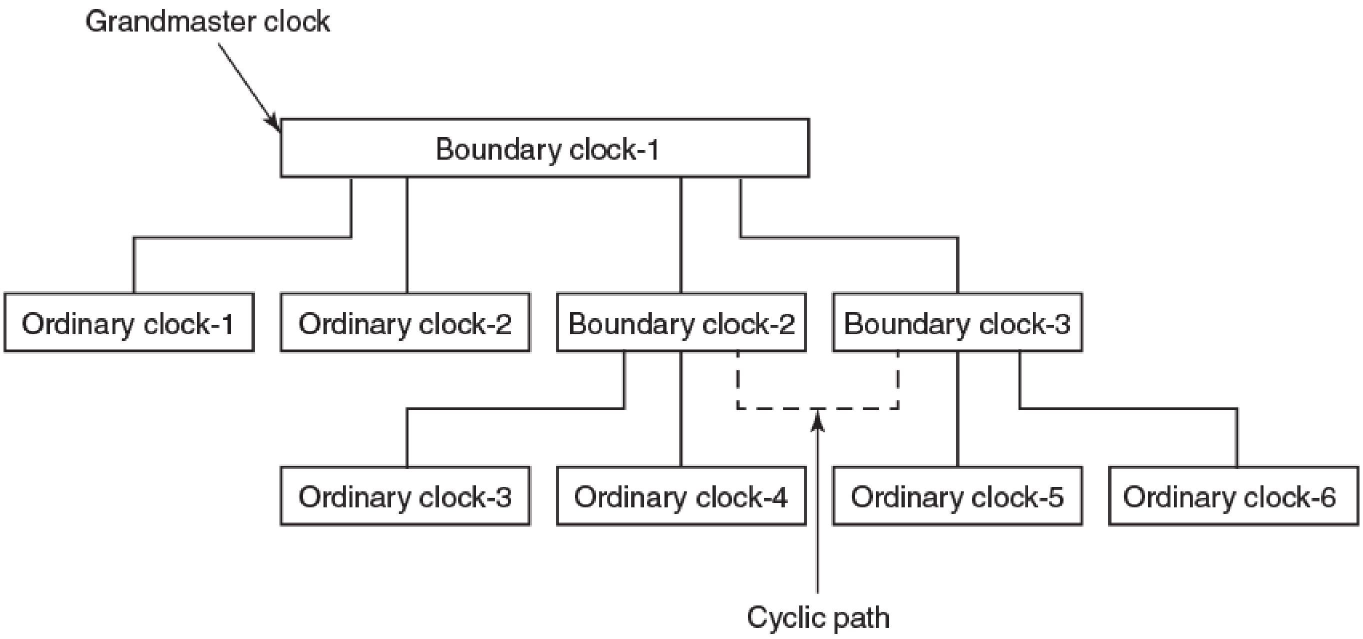
Transparent clock: A clock that measures the time taken for a PTP event message to transit itself and provides this information to clocks receiving this PTP event message. This clock type does not need to synchronize to the grandmaster clock. There are two types of transparent clock:

- An end-to-end transparent clock, which uses an end-to-end delay mechanism to measure the transit delay.

- A peer-to-peer clock, which uses a peer-to-peer delay mechanism to measure the transit delay.

The IEEE 1588 reference architecture is illustrated below.

FIGURE 4 IEEE 1588 Reference Architecture



PTP forwards frames on PTP-enabled interfaces. The protocol has a built-in loop-avoidance algorithm to detect loops in the underlying network. The resulting PTP clock distribution topology is loop free.

PTP message types

There are two PTP message types: General and Event.

Event messages are used to obtain delay measurements between a master and a slave node. A Sync message is used for measuring the master-to-slave delay. A Delay Request message is used for measuring the slave-to-master delay. In a one-step clock operation, the transmit timestamp of the Sync and Delay Request packet is in the packet itself. This requires hardware support to insert the timestamp as the packet egresses the node.

General messages are used for building the PTP topology, service negotiation, delay measurement, and management. In a two-step clock operation, Follow Up and Delay Response packets carry the timestamp in the payload used for offset and path delay measurement.

The following PTP messages are supported on Extreme SLX devices.

TABLE 15 PTP messages supported on Extreme SLX devices

Message type	Message class	Purpose
Sync	Event	Delay measurement
Delay Request	Event	End-to-end delay measurement
Follow Up	General	End-to-end delay measurement
Delay Response	General	End-to-end delay measurement

TABLE 15 PTP messages supported on Extreme SLX devices (continued)

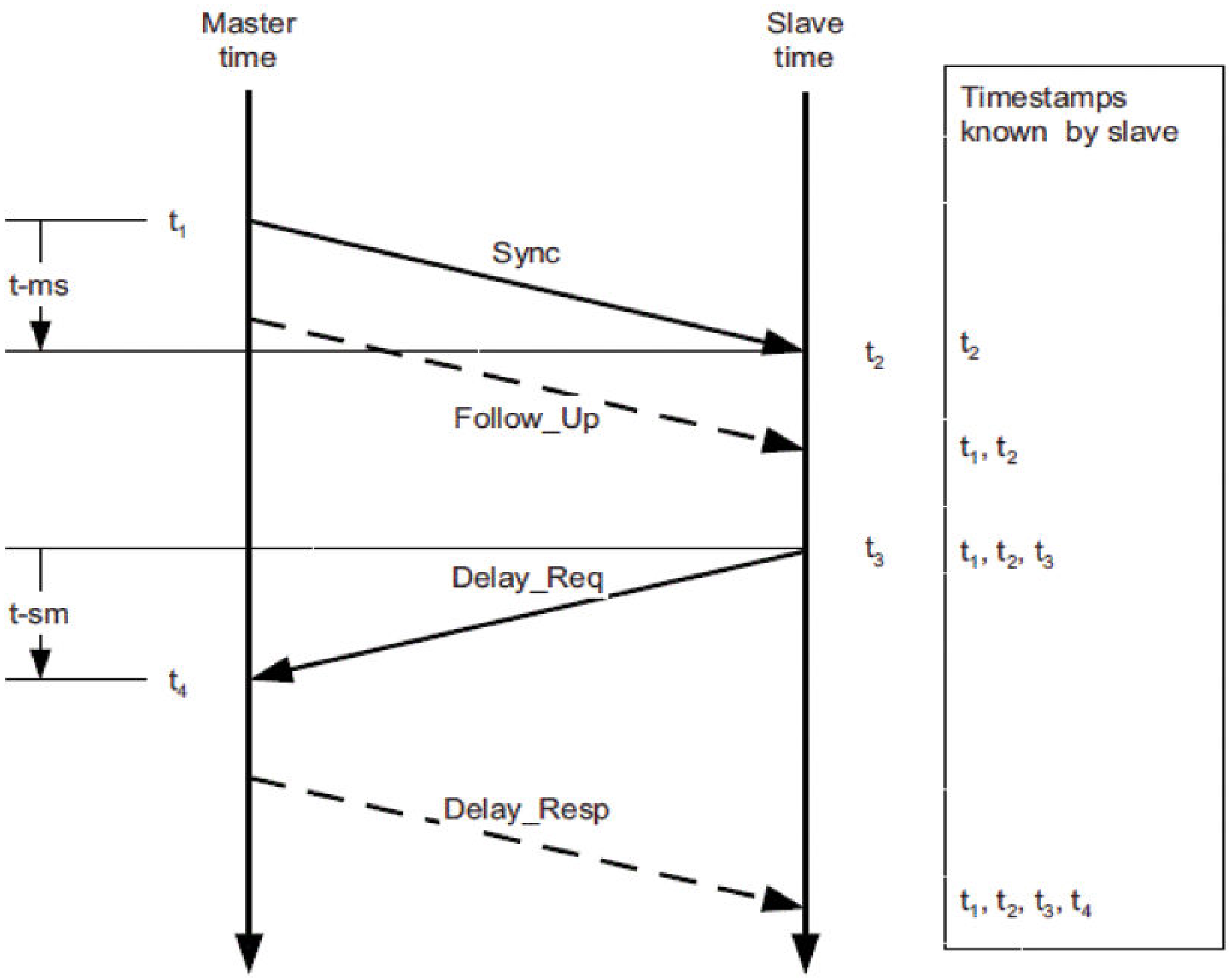
Message type	Message class	Purpose
Announce	General	Topology

PTP clock offset/delay calculation

Once the master/slave relationship is decided on a interface, the master and slave port exchange timestamps used for clock offset/delay calculation in a two-step clock operation, as follows.

The following illustrates the flow of PTP timestamp messages in a two-step synchronization message exchange.

FIGURE 5 PTP timestamp messages in a two-step synchronization message exchange



Master to slave delay = t-ms = $t_2 - t_1$

Slave to master delay = t-sm = $t_4 - t_3$

Clock Offset = (t-ms – t-sm)/ 2

Link Delay = (t-ms + t-sm/ 2

The master clock periodically sends out Sync message to a slave clock (received at t_2). The actual Sync transmit timestamp (t_1) is conveyed in the subsequent Follow Up message. The slave clock periodically sends a Delay Request message (t_3) to the master clock. The master clock replies with the actual time (t_4) at which it receives this message in the subsequent Delay Response message. All timestamping is done at the physical layer so as to eliminate the inclusion of protocol stack or queuing delay in the reported timestamp.

The Sync and Follow_Up packet calculates the master/slave delay (t-ms).

The Delay_Req and Delay_Resp calculates the slave/master delay (t-sm).

The rate of sending Sync and Delay_Req need not be the same. The lave clock calculates the clock offset and link delay after it has collected all four timestamps. It adjusts its local clock by the resulting offset in order to synchronize with the master clock.

PTPv2 encapsulation parameters

Version 2 of PTP, PTPv2 (IEEE 1588-2008), uses different encapsulation parameters than are used for Version 1. PTPv2 is not backward compatible with the original version.

The following table lists the destination MAC addresses, UDP/IPv4 parameters, and frame priorities that are used in PTPv2.

TABLE 16 PTPv2 destination MAC addresses, UDP/IPv4 parameters, and frame priorities

Destination MAC address	UDP/IPv4 parameters	Frame priority
01:00:5e:00:01:81. IP/UDP 802.3 (except for peer delay measurement)	UDP port 319 for Event message	802.1p priority 7
01:80:C2:00:00:0E (for peer delay measurement)	UDP port 320 for General message	DSCP traffic class selector 7
	224.0.1.129 for all except peer delay measurement	
	224.0.0.107 for peer delay measurement	

PTP interface types

PTP can be enabled on an unknown port, a switch port, or a router port. The underlying interface can be a physical interface or a port-channel (Multi-Chassis Trunking, or MCT) interface.

On an unknown port or a router port, a PTP frame is sent untagged. On a switch port, a PTP frame is sent by default on the native VLAN (tagged or untagged) or on a specific VLAN configured by the user. Frame forwarding on a switch port follows the STP state of the PTP VLAN.

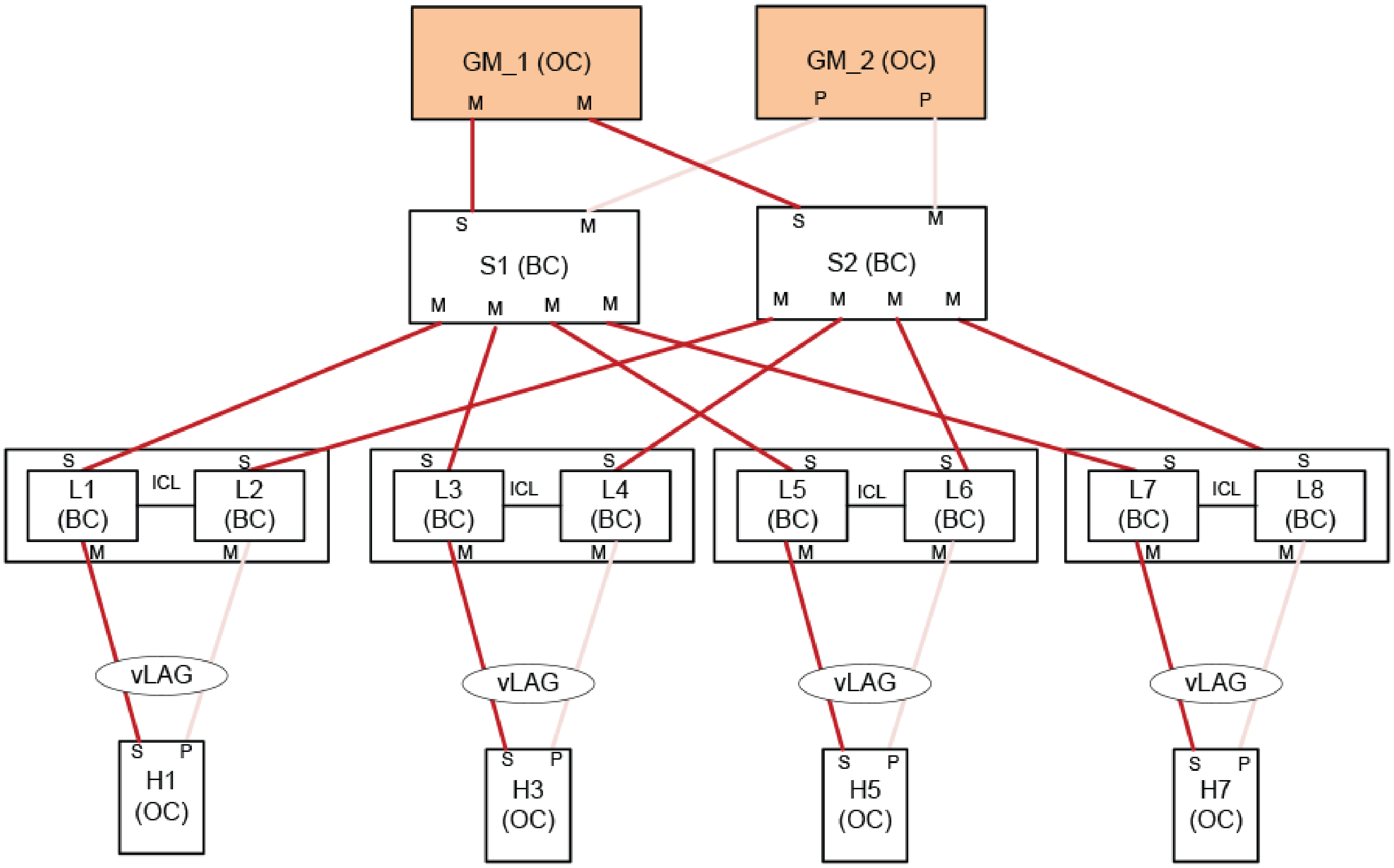
A boundary clock (BC) always terminates ingress PTP frames that are received and never forwards them to other PTP ports.

PTP IP Clos topology

PTP is supported in an IP Clos spine and leaf topology.

The following illustrates an IP Clos topology with PTP enabled on all spine (S) and leaf (L) nodes supporting connectivity to hosts (H)] . Spine and leaf nodes are connected through virtual Ethernet (VE) or router ports.

FIGURE 6 PTP IP Clos topology

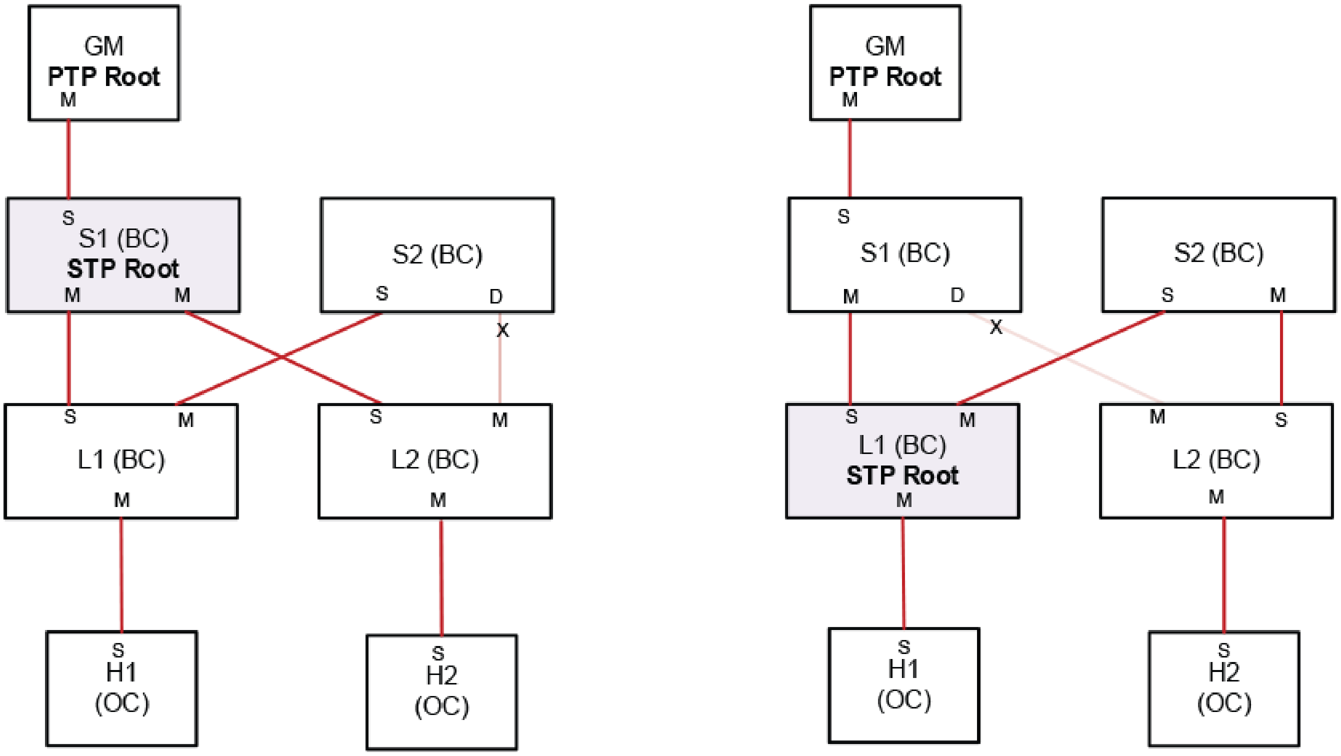


Each fabric node is a boundary clock (BC). At a leaf node, PTP is enabled on at least one of the uplinks. Additional uplinks to different spine nodes can be enabled to provide redundancy. Each server attached to the fabric is an ordinary clock (OC). PTP over MCT is supported. Enabling PTP over an MCT inter-chassis link (ICL) is optional if the grandmaster (GM) clock has connectivity to each MCT member node.

The GM clock connects to the fabric over a a spine BC node, and is at the top of the clock distribution hierarchy. The GM timing packets traverse spine/leaf tiers to reach each BC/OC in the fabric. The tiers between the GM and a PTP node should not exceed in order to meet the clock accuracy. Dual GM clocks can be used to provide redundancy

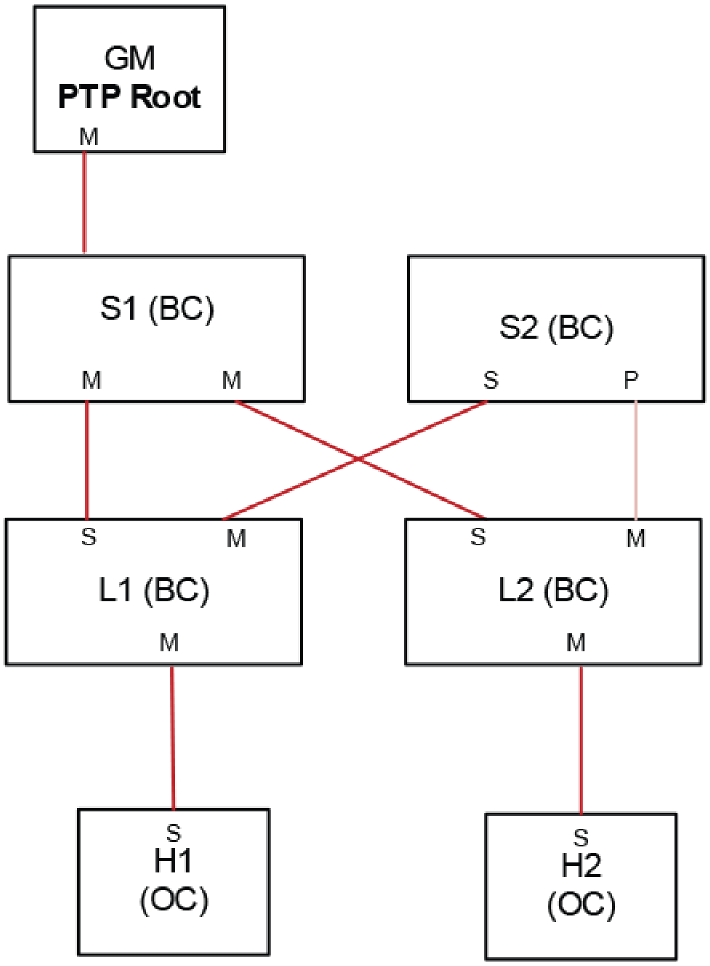
In the following figure, an active PTP session connects ports in master and slave state and Spanning Tree Protocol (STP) is enabled. Assuming that all uplinks are router ports, the highlighted links represent the PTP active topology determined by the PTP protocol. "X" represents an STP-blocked port. The greyed-out links are in standby.

FIGURE 7 Active PTP session connecting ports in master and slave states with STP



In the following figure, a standby session connects ports in master and passive states. Here STP is not enabled.

FIGURE 8 Standby PTP session connecting ports in master and passive states, without STP



PTP on MCT

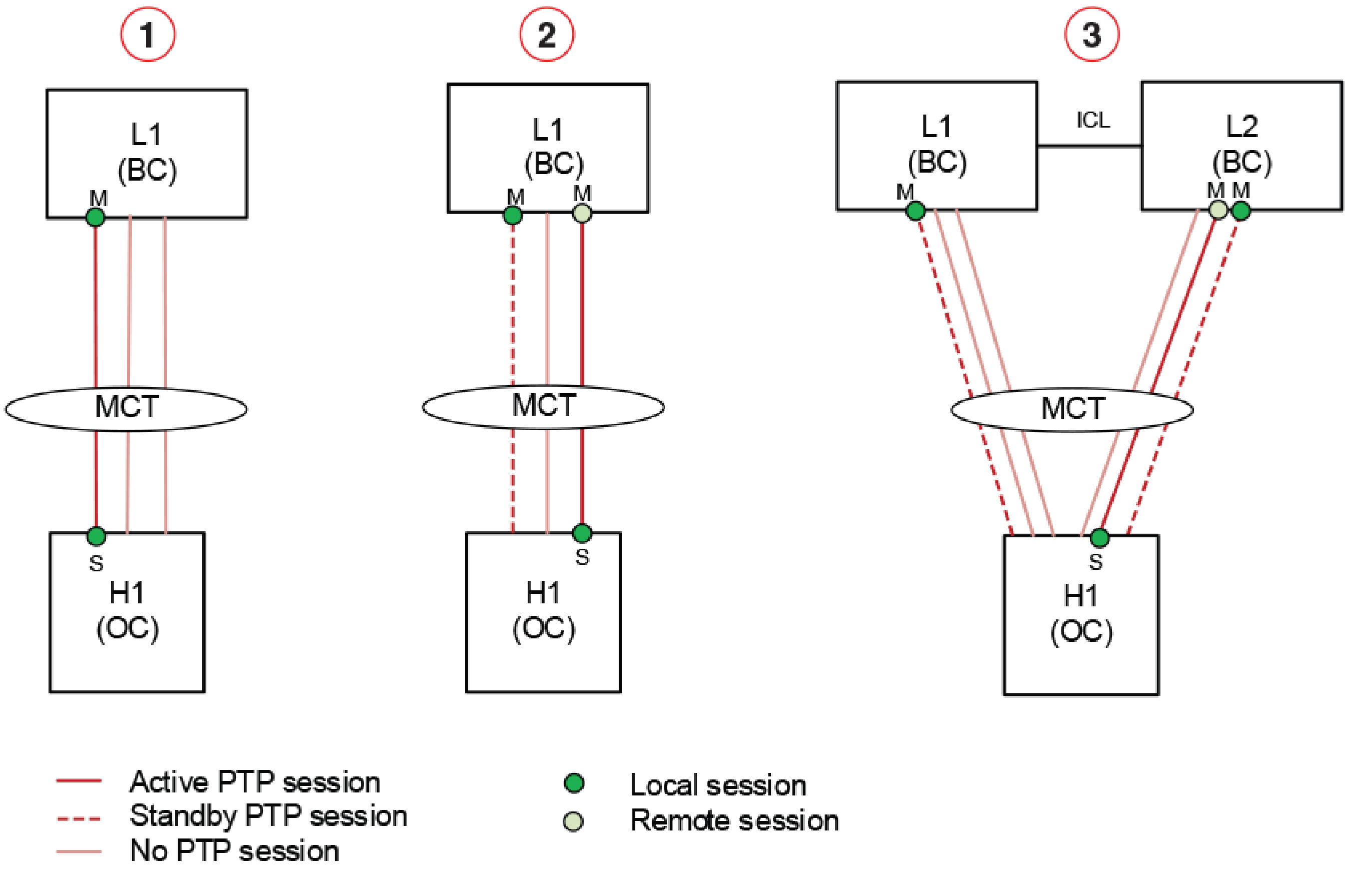
When PTP is enabled on a physical port, a PTP session is created to measure the link delay on the port. When PTP is enabled on a Extreme Multi-Chassis Trunk (MCT), it is applied at the MCT level and not on a specific member port. Each MCT member port is capable of supporting a PTP session. A maximum of two PTP sessions is created between a local and remote MCT node.

PTP session with hypervisor

When a Extreme SLX device connects to a hypervisor over an MCT, it does not assume on which member link the hypervisor starts its PTP session. If the hypervisor initiates the PTP session on a link that is different from the one the SLX device does, the SLX can establish a PTP session on that link as well. In this scenario, the SLX creates two PTP sessions, one for each member link. The active session is always on the link that the hypervisor uses to initiate the session. The other session is in standby.

Assuming that the hypervisor is capable of running only a single session at a time, the following figure illustrates three connectivity scenarios.

FIGURE 9 PTP sessions with hypervisor on same link, different link, and across nodes

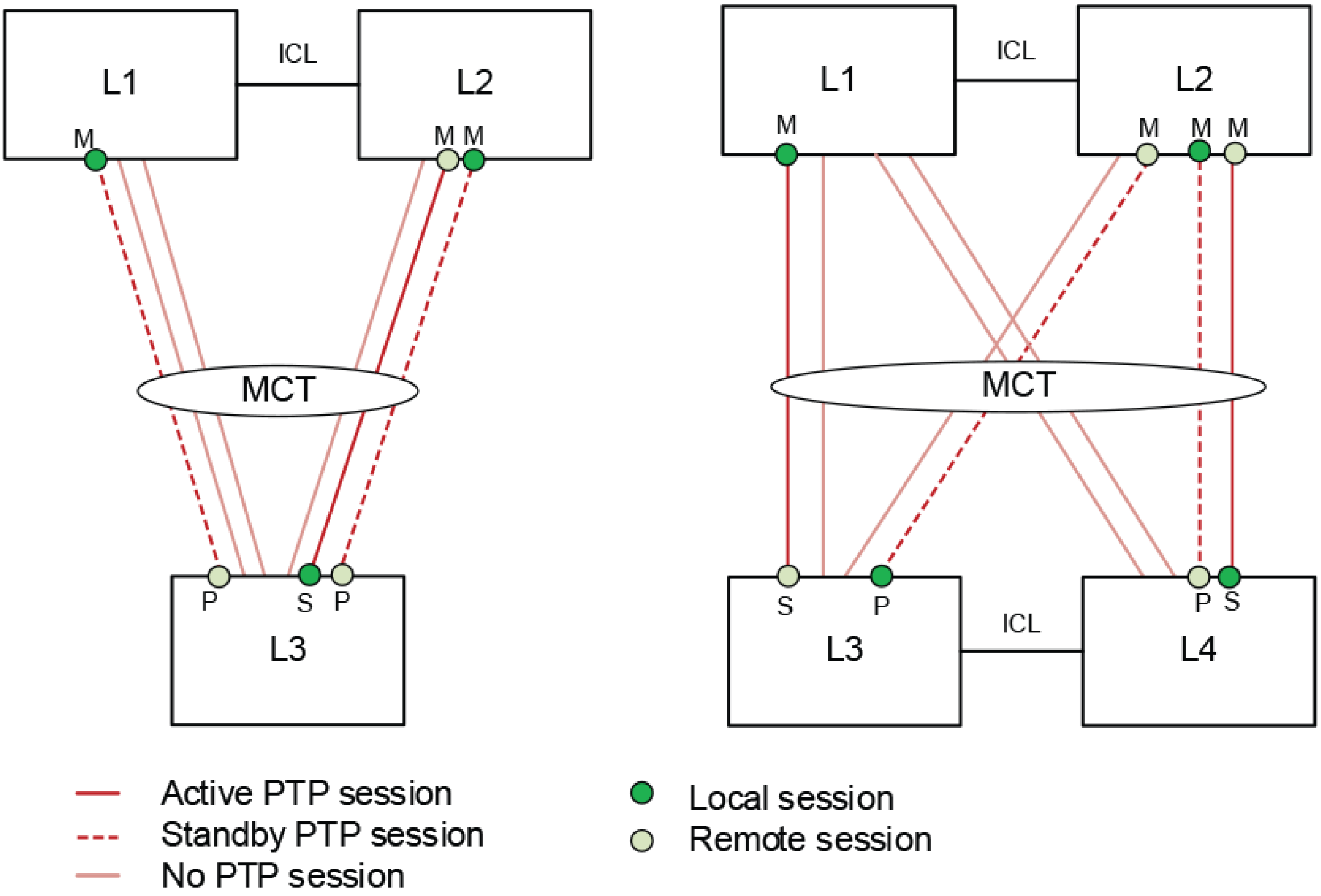


1. SLX and server initiates PTP session on the same link
2. SLX and server initiates PTP session on different links
3. Server initiates PTP session on a member link of MCT spanning two nodes

PTP session between SLX devices

SLX devices may be connected to each other over an MCT. The SLX does not assume on which link the remote SLX initiates the PTP session. If local and remote SLX devices start a PTP session on different links, each SLX accepts the remote session in addition to maintaining the locally initiated session. A maximum of two PTP sessions is created between a local and remote MCT node. The PTP BMC protocol determines which session becomes active, and which sessions go on standby. This is illustrated in the following figure.

FIGURE 10 PTP sessions between local and remote SLX devices



MCT nodes are connected by an ICL to form a logical switch. Enabling PTP on an ICL is optional If the GM has connectivity to each member node without using an ICL.

PTP clock profile

An SLX device functions as a boundary clock, and is not intended to serve as a grandmaster (GM) clock that provides stable timing to an end system. As a PTP clock, the SLX uses the following parameters to select the best master clock (BMC).

The following table lists clock property parameters in the profile that are used to select the BMC. The user can configure clock properties Priority1 and Priority2 to influence that selection. All other parameters are static and cannot be changed. The BMC selection compare the clock property parameters against what is announced by its peer to determine which node should be the master clock.

TABLE 17 Clock property parameters in the SLX profile that are used to select the BMC

Clock property	Description	SLX default value
Priority1	User-configurable priority 1	255 (lowest priority)
Priority2	User-configurable priority 2	255 (lowest priority)
Clock class	TAI (International Atomic Time) traceable for UTC offset and time scale	248 (undefined clock class)

TABLE 17 Clock property parameters in the SLX profile that are used to select the BMC (continued)

Clock property	Description	SLX default value
Clock accuracy	Time accuracy resolution	254 (unknown accuracy)
OffsetScaleLogVariance	Frequency stability variance	0xFFFF (clock variance not avvailable)

The following table lists the time property parameters in the profile.

TABLE 18 Time property parameters in the SLX profile that are used to select the BMC

Time property	Description	SLX default values
PTP Timescale	Origin of timescale	0 (arbitrary)
Time source	Internal oscillator	0xA0 (internal oscillator)

PTP clock synchronization

IEEE PTP 1588 defines the protocol to generate the timestamps needed for time synchronization, but it does not specify the mechanism a system uses to synchronize its clock to the grandmaster (GM) clock with a timestamp. This implementation is system specific.

The following table lists the clock synchronization states used by Extreme SLX devices.

TABLE 19 Extreme SLX PTP clock synchronization states

PTP clock state	Description
No Reference	No foreign master clock detected
Frequency Acquiring	Acquiring master clock frequency
Frequency Acquired	Locked to master clock frequency
Time Sync	Time is synchronized to MC, locked to MC in frequency and phase
Holdover	Loss of stable input frequency reference; clock is running at last established frequency

Only a slave clock needs to perform a local-clock time sync. A clock that assumes a GM role does not.

The SLX algorithm performs a time sync by aligning the local clock frequency and phase to the master clock (MC). The algorithm executes through multiple synchronization states before a time sync is completed. When the state reaches Time Sync, synchronization with the with MC is complete. For a GM clock, no time sync is required; the clock state is always at No Reference.

The clock state transitions are captured in the RASLog at the information level. The current state is shown in the output of the **show ptp clock** command.

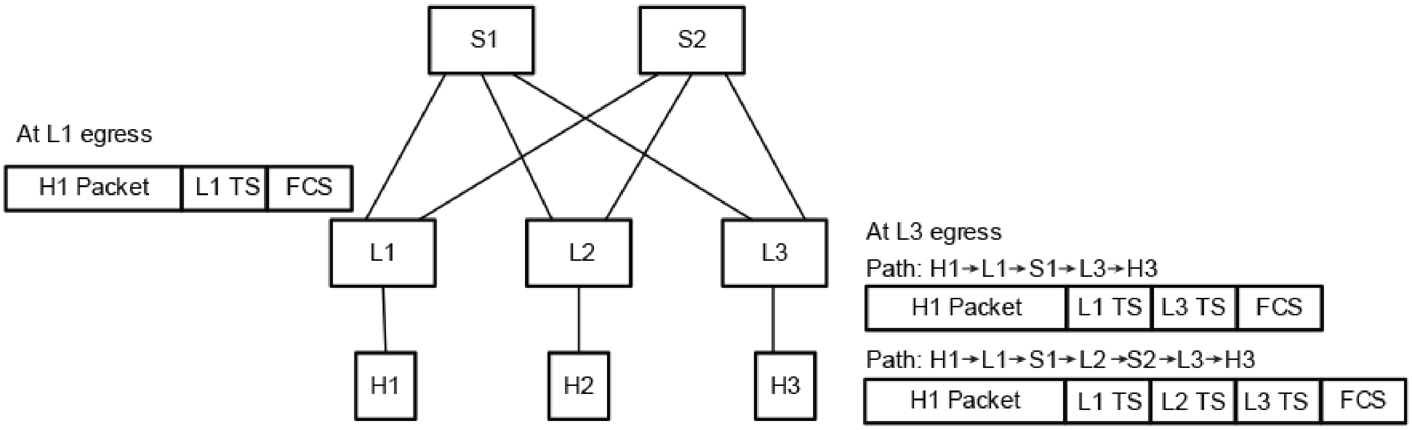
Packet timestamping

Network administrative or analytic applications continuously monitor network conditions that affect network health. When network anomalies such as congestion or security threats are detected, the applications apply correction actions to ensure that user applications are not affected. The ability to timestamp a packet arrival time enables these application to detect abnormal conditions.

To detect network congestion, an application could observe the latency of a packet as it traverses the fabric. With an SLX device, the time a packet ingress a switch could be appended to the payload when that packet is forwarded to the egress interface. The ingress timestamp collected at two adjacent switches reflects the sum of delays incurred by the ingress switch and the link between the two switches. Subtracting the link delay from this value gives the switch delay.

In the following figure, all leaf switches (L) have an egress timestamp enabled at all interfaces. An application that sends a packet from host H1 towards host H3 through different spine (S) nodes collect the timestamp as indicated at host H3.

FIGURE 11 Timestamps through leaf and spine switches



Timestamp source

The timestamp appended to the payload is in an eight-byte nanosecond time format. If PTP is enabled on the switch, the time source is what is specified in PTP. If PTP is not enabled, then the timestamp is the local system POSIX epoch time.

> NOTE
> The accuracy of switch delay measurements from a timestamp is highly dependent on the precision of clock synchronization among the nodes along the traffic path.

MTU considerations

Every switch interface can be configured to append timestamps to the payload. Each timestamp appended to the payload increases the original payload size by eight bytes. As a packet traverses multiple switches in the fabric, if the resulting packet length exceeds the MTU configured for the interface, the packet is discarded.

Timestamp limitations

Timestamping commands are available under interface subtype configuration mode. At the ingress port, the user can indicate whether an ingress packet carries a timestamp by executing the **system packet-timestamp ingress valid** command. It is important that the user specify a timestamp is valid only if all packets that ingress the interface carry a timestamp in the payload. If the packet does not carry the timestamp, the last eight bytes of the packet are modified when the packet is forwarded out the egress interface. At the egress interface, the user can specify whether a timestamp should be appended to the payload or an existing timestamp should be replaced or removed. Hardware performs these actions according to the configuration on the ingress interface, not on whether the timestamp actually exists in the payload. If the timestamp does not exist, hardware overwrites or removes the last eight bytes of the payload and modifies the original packet. The following table illustrates system behavior when the ingress timestamp is enabled as **valid** or not, with or without a timestamp in the payload.

TABLE 20 System behavior when the ingress timestamp is enabled as valid, with or without a timestamp in the payload

Ingress timestamp valid?	Egress timestamp action			
	None	Add	Replace	Remove
No	No action	Append	Append	No action
Yes; timestamp in payload	No action	No action	Replace	Remove
Yes; timestamp not in payload	No action	Replace last 8 bytes of payload	Replace last 8 bytes of payload	Remove last 8 bytes of payload

PTP firmware upgrade/downgrade

Because this a new feature, there are no issues in upgrading from an earlier release that does not support PTP. However, when downgrading to a release that does not support PTP, all PTP configurations must be removed before the downgrade is permitted.

Configuring PTP

Configuring PTP globally on a switch

This task configures PTP on a switch, specifying a domain, nondefault clock priority values, and a source iP address.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Enter the **protocol ptp** command to enter global PTP configuration mode.

```
device(config)# protocol ptp
```

3. (Optional) Enter the **domain** command to specify a nondefault PTP clock domain.

```
device(config-ptp)# domain 1
```

4. (Optional) Specify nondefault Priority 1 and Priority 2 clock values.

```
device(config-ptp)# priority1 200
device(config-ptp)# priority2 200
```

5. Enable PTP.

```
device(config-ptp)# enable
```

Configuring PTP on a basic port-channel

This task configures PTP on basic (Layer 2) port-channel interface and configures nondefault Announce and Sync message intervals.

1. Enter global configuration mode.

```
device# configure terminal
```

2. Enter the **interface port-channel** command to enter interface subtype configuration mode for a port-channel.

```
device(config)# interface port-channel 10
```

3. Enter the **protocol ptp** command to configure PTP on the interface.

   ```
   device(config-Port-channel-10)# protocol ptp
   ```

4. Enter the **enable** command to enable PTP on the interface.

   ```
   device(config-Port-channel-10-ptp)# enable
   ```

5. (Optional) Enter the **announce-interval** command to change the Announce interval from the default.

   ```
   device(config-Port-channel-10-ptp)# announce-interval 3
   ```

6. (Optional) Enter the **sync-interval** command to change the Sync interval from the default.

   ```
   device(config-Port-channel-10-ptp)# sync-interval 0
   ```

7. (Optional) Enter the **announce-timeout** command to change the Announce timeout from the default.

   ```
   device(config-Port-channel-10-ptp)# announce-timeout 4
   ```

8. Optional) Enter the **delay-request-min-interval** command to change the Delay Request interval from the default.

   ```
   device(config-Port-channel-10-ptp)# delay-request-min-interval -3
   ```

Configuring PTP on a router port for Layer 3 protocols

PTP support for Layer 3 protocols is provided on physical interfaces. This task enables PTP over an Ethernet Layer 3 interface.

1. Enter global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter interface subtype configuration mode for a port-channel.

   ```
   device(config)# interface ethernet 0/1
   ```

3. Configure an IPv4 address and mask.

   ```
   device(conf-if-eth-0/1)# ip address 20.1.1.1/24
   ```

4. Enter PTP configuration mode on the port-channel.

   ```
   device(conf-if-eth-0/1)# protocol ptp
   ```

5. Enable PTP.

   ```
   device(conf-if-eth-0/1-ptp)# enable
   ```

Configuring PTP on a trunk port

This task enables PTP on a port-channel trunk port, with optional PTP frame forwarding on a non-native VLAN.

By default, PTP frames are forwarded on native VLAN 1.

1. Enter global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter interface subtype configuration mode for a port-channel.

   ```
   device(config)# interface port-channel 12
   ```

3. Enable Layer 2 mode.

   ```
   device(config-Port-channel-11)# switchport
   ```

4. Set the mode as trunk.

   ```
   device(config-Port-channel-11)# switchport mode trunk
   ```

5. (Optional) In interface subtype configuration mode for the port-channel, specify a VLAN or range of VLANs.

   ```
   device(config-Port-channel-11)# switchport trunk allow vlan add 100-200
   ```

   This example instantiates a range, to support virtual Ethernet configuration as well.

6. Enter PTP configuration mode on the port-channel interface.

   ```
   device(config-Port-channel-11)# protocol ptp
   ```

7. Enable PTP.

   ```
   device(config-Port-channel-11-ptp)# enable
   ```

8. Enter the **ptp-vlan** command to specify the nondefault VLAN for PTP packets.

   ```
   device(config-Port-channel-11-ptp)# ptp-vlan 101
   ```

Configuring PTP on a VE switch port

This task configures PTP on a virtual Ethernet (VE) in switchport mode and specifies a nondefault VLAN.

PTP must be enabled on every switch port in the VLAN that is associated with the VE interface.

1. Enter global configuration mode.

   ```
   device# configure terminal
   ```

2. Assign a VE interface and enter interface subtype configuration mode.

   ```
   device(config)# interface ve 200
   ```

3. Assign an IPv4 address and mask, and exit to global configuration mode.

   ```
   device(config-Ve-200)# ip address 10.1.1.1/31
   device(config-Ve-200)# exit
   ```

4. In global configuration mode, specify a VLAN.

   ```
   device(config)# vlan 101
   ```

5. In interface subtype configuration mode, assign a VE interface, and exit to global configuration mode.

   ```
   device(config-vlan-101)# interface ve 200
   device(config-vlan-101)# exit
   ```

6. In global configuration mode, specify an Ethernet interface.

   ```
   device(config)# interface ethernet 0/1
   device(config-vlan-101)# exit
   ```

7. Enable Layer 2 mode.

   ```
   device(conf-if-eth-0/1)# switchport
   ```

8. Set the mode as *trunk*.

   ```
   device(conf-if-eth-0/1)# switchport mode trunk
   ```

9. In interface subtype configuration mode, specify a VLAN or range of VLANs.

   ```
   device(conf-if-eth-0/1)# switchport trunk allow vlan add 100-200
   ```

10. Enter PTP configuration mode on the Ethernet interface.

    ```
    device(conf-if-eth-0/1)# protocol ptp
    ```

11. Enable PTP and exit to interface subtype configuration mode.

    ```
    device(conf-if-eth-0/1-ptp)# enable
    ```

12. Enter the **ptp-vlan** command to specify the nondefault VLAN for PTP packets.

    ```
    device(conf-if-eth-0/1-ptp)# ptp-vlan 101
    ```

Managing ingress and egress timestamps

You can notify the SLX device as to whether ingressing packets on specific interfaces contain appended timestamps by using the **system packet-timestamp valid** command. You can also configure how these timestamps are processed when the packet is forwarded on the egress interface, by means of the **system packet-timestamp egress** command. The use of these commands is illustrated in the following examples.

To specify that timestamp exists in the payload:

```
device# configure terminal
device(config)# interface port-channel 1
device(config-Port-channel-1)# system packet-timestamp ingress valid
```

> **ATTENTION**
> The presence of the timestamp in the ingress payload is effectively indicated by this command. Hardware does not verify whether or not the timestamp is actually in the payload. If the use of this command specifies that the timestamp be present but the timestamp does not actually exist, then the hardware overwrites or removes the last eight bytes of payload data.

To specify that no timestamp exists in the payload:

```
device# configure terminal
device(config)# interface port-channel 1
device(config-Port-channel-1)# no system packet-timestamp ingress
```

To specify that the timestamp for when the packet ingresses the switch is appended to the end of the payload on an egress port-channel interface:

```
device# configure terminal
device(config)# interface port-channel 1
device(config-Port-channel-1)# no system packet-timestamp egress add
```

To specify that the timestamp for when the packet ingresses the switch is removed from the end of the payload on an egress port-channel interface:

```
device# configure terminal
device(config)# interface port-channel 1
device(config-Port-channel-1)# system packet-timestamp egress remove
```

To specify that the timestamp for when the packet ingresses the switch is replaced by the timestamp for when the packet egresses the port-channel interface:

```
device# configure terminal
device(config)# interface port-channel 1
device(config-Port-channel-1)# system packet-timestamp egress replace
```

To disable the processing of packets on an egress port-channel interface:

```
device# configure terminal
device(config)# interface port-channel 1
device(config-Port-channel-1)# no system packet-timestamp egress
```

Clearing PTP statistics on an interface

You can clear PTP statistics (counters) on all PTP interfaces or on a specified interface, as in the examples below.

To clear PTP statistics on all interfaces:

```
device# clear ptp counter interface
```

To clear PTP statistics on a specified interface:

```
device# clear ptp counter interface ethernet 0/1
```

PTP show commands

This section lists and describes PTP show commands

TABLE 21 PTP show commands

Command	Description
show ptp brief	Displays the status of each physical port that has a running PTP session, and for port channels the status of each member port that has a running PTP session.
show ptp clock	Displays the status of a local PTP clock.
show ptp clock foreign-masters record	Displays the status of foreign master clocks known to a PTP clock. For each foreign master clock, the output displays the clock identity, basic clock properties, and whether the clock is being used as a grandmaster clock.
show ptp corrections	Displays the recent history of PTP local clock offset adjustments.
show ptp parent	Displays properties of the PTP parent clock port.
show ptp port interface	Displays the properties and statistics of a PTP interface.
show ptp time-property	Displays the properties of the PTP clock.

SNMP

SNMP overview

Simple Network Management Protocol (SNMP) is a set of application layer protocols for managing complex networks. Devices within a network use SNMP to send messages, called protocol data units (PDUs), to different parts of a network.

Network management using SNMP requires three components:

- SNMP manager—Typically, network management systems (NMS) that manage networks by monitoring the network parameters, and optionally, setting parameters in managed devices. The SNMP manager communicates to the devices within a network using the SNMP protocol.
- SNMP agent—Software that resides in the managed devices in the network, and collects and stores data from these devices. Each device hosts an SNMP agent. The agent receive requests from the SNMP manager and responds with the requested data. In addition, the agent can asynchronously alert the SNMP manager about events by using special PDUs called traps.

  Multiple instances of the same MIB module can support a single SNMP agent by mapping a specific key called a context name to a virtual routing and forwarding (VRF) instance created within the Extreme device.
- Management Information Base (MIB)—Hierarchical database where SNMP agents in the managed devices store the data about these devices. The MIB is is structured on the standard specified in the RFC 2578 [Structure of Management Information Version 2 (SMIv2)].

  An SNMP manager can issue read or write operations to retrieve and use the MIB objects to manage and monitor devices on the network. However, the MIB structure determines the scope of management access allowed by a device.

The SNMP server on the Extreme device supports SNMP version 1 (SNMPv1), SNMP version 2 (SNMPv2), and SNMP version 3 (SNMPv3).

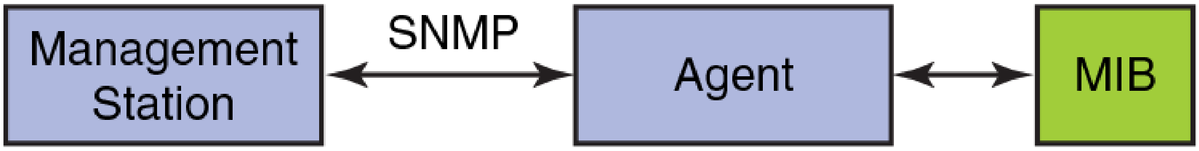
- SNMPv1 and SNMPv2 use community strings associated to SNMP groups. The group maps the user to MIB objects called SNMP views. The views restrict the access of the MIB OIDs .
- SNMPv3 provides additional security through authenticated users associated with groups to restrict the access of MIBs for SNMP requests through SNMP views.

Also, the device supports the configuration of trap hosts as a trap recipient to receive filtered traps based on their severity level, and optionally receive SNMP communication through a VRF.

Basic SNMP operation

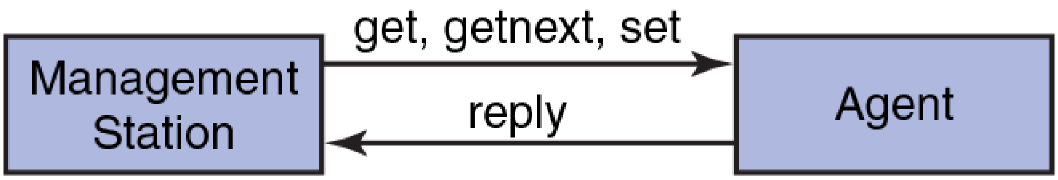
Every Extreme device carries an *agent* and management information base (MIB), as shown in the next figure. The agent accesses information about a device and makes it available to an SNMP network management station.

FIGURE 12 SNMP structure



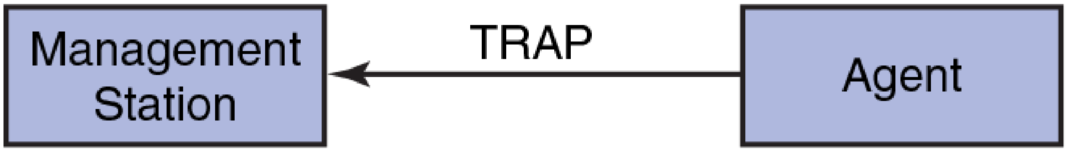
When active, the management station can "get" information or "set" information when it queries an agent. SNMP commands, such as **get**, **set**, **getnext**, and **getbulk**, are sent from the management station, and the agent replies once the value is obtained or modified as shown in the next figure. Agents use variables to report such data as the number of bytes and packets in and out of the device, or the number of broadcast messages sent and received. These variables are also known as managed objects. All managed objects are contained in a MIB.

FIGURE 13 SNMP query



The management station can also receive *traps*, unsolicited messages from the device agent if an unusual event occurs as shown in the next figure.

FIGURE 14 SNMP trap



The agent can receive queries from one or more management stations and can send traps to up to six management stations.

SNMP community strings

SNMP versions 1 and 2 use community strings to restrict SNMP access.

The community string can be associated with an SNMP group to restrict the access of MIBs for SNMPv1 and SNMPv2c requests. You can configure a total of 256 read-only and read-write community strings on the device.

The software automatically encrypts SNMP community strings. Users with read-only access or who do not have access to management functions in the CLI cannot display the strings. For users with read-write access, the strings are encrypted in the CLI.

By default, you cannot perform any SNMP Get or Set operations until you configure a read-only or read-write community string.

SNMP groups

SNMP groups map the SNMP user for SNMPv3 and the community for the SNMPv1 and SNMPv2 to SNMP views.

You can configure each group with any or all of the following views:

- Read view with read-only access
- Write view with read-write access
- Notify view to filter notifications to be encrypted and sent to target hosts

SNMP users that are mapped to a group with SNMP views use its views for access control.

SNMP users

SNMP version 3 (RFC 2570 through 2575) introduces a User-Based Security model (RFC 2574) for authentication and privacy services. This model provides a user that is associated with security information for authentication of its generated SNMP messages.

SNMP version 3 also supports View-Based Access Control Mechanism (RFC 2575) to control access at the PDU level. It defines mechanisms for determining whether to allow access to a managed object in a local MIB by a remote principal. You can create and associate SNMPv3 users with configured SNMP groups to use the group views for access control.

SNMP views

SNMP views are named groups of MIB objects that you can associate with groups to limit access by community strings and users for viewing and modifying the SNMP statistics and system configuration. With SNMP views, you can create or remove the access to a MIB object for inclusion or exclusion from viewing from user access.

SNMP views reference MIB objects using object names. It represents the hierarchical location of the object in the MIB tree. You associate the views with each group to restrict or allow access to the OIDs. You can create a maximum of 10 views on the device.

SNMP server hosts

On the Extreme device, the SNMP server host serves as a trap receiver to ensure that all SNMP traps sent by the device go to the same SNMP trap receiver or set of receivers, typically one or more host devices on the network.

For an SNMPv3 trap, you associate a SNMPv3 host with the SNMP users. When you specify the host, you also specify a community string for SNMPv1 and SNMPv2. The Extreme device sends all the SNMP traps to the specified hosts and includes the specified community string. Then, administrators can filter for traps from a Extreme device based on IP address or community string.

Multiple SNMP server context to VRF mapping

A single SNMP agent can support multiple instances of the same MIB module by the mapping of the context name to a virtual routing and forwarding (VRF) instance created within the device.

You map each VRF with a specific context name. The context name identifies the VRF and fetches the MIB details of the mapped VRF from the underlying modules. For example, the OSPF-MIB returns the queried OSPF-MIB object values pertaining to the default VRF (default-vrf).

For SNMPv1 and SNMPv2, the mapping of the context is with the community. This mapping is in addition to mapping of the context with the VRF. The SNMP agent supports 256 contexts to support context-to-VRF mapping.

For SNMPv3, you only need to map the context with the VRF. The SNMPv3 request PDU itself provisions for the context. Only one context is allowed for each VRF instance.

Configuring SNMPv2

SNMPv1 and SNMPv2 use community strings to restrict SNMP access. When you associate it with an SNMP group, you can restrict the access of MIBs for SNMPv1 and SNMPv2c requests.

To configure SNMPv2, perform the following steps.

1. In privileged EXEC mode, enter global configuration mode.

   ```
   device# configure terminal
   ```

2. Configure the location and contact information for the SNMP server.

   ```
   device(config)# snmp-server location "Building 3 Room 214" contact "Operator 12345"
   ```

   This example changes the default location from ExtremeTcsHyd to "Building 3 Room 214" and default contact information from ExtremeCommunicationSystem to "Operator 12345".

   The double quotes allows you to enter the string with spaces.

3. Add an SNMP view to restrict or allow access to the MIB OIDs.

   ```
   device(config)# snmp-server view view2 1.3.6.1 included
   ```

   This example adds the SNMP view2 view entry with included permission to allow access for the MIB 1.3.6.1 object ID ('internet').

4. Add an SNMP group.

   ```
   device(config)# snmp-server group admin v2c write view2 notify view2
   ```

   This example adds the admin group for SNMPv2 and maps the read-write access and notify views to view2.

5. Add an SNMP community string and associate it with a group.

   ```
   device(config)# snmp-server community comm1 group admin
   ```

   This example adds the comm1 community string and associates it with the admin group to access the MIBs for SNMPv2c requests.

6. Configure the SNMP trap host associated with community string.

   ```
   device(config)# snmp-server host 10.32.147.6 comm1 version 2c severity-level Warning
   ```

   This example configures 10.32.147.6 as a trap recipient with SNMPv2c on the default target port 162 and associates the comm1 community string.

7. Enable the traps.

   ```
   device(config)# snmp-server enable trap
   ```

8. Access privileged EXEC mode.

   ```
   device(config)# exit
   ```

9. Verify the configuration.

   ```
   device# show running-config snmp-server
   snmp-server contact  "Operator 12345"
   snmp-server enable trap
   snmp-server location  "Building 3 Room 214"
   snmp-server community comm1 group admin
   snmp-server group admin v2c write view2 notify view2
   snmp-server host 10.32.147.6 comm1 version 2c
    severity-level Warning
   !
   ```

The following example shows the previous steps to configure SNMPv2.

```
device# configure terminal
device(config)# snmp-server location "Building 3 Room 214" contact "Operator 12345"
device(config)# snmp-server view view2 1.3.6.1 included
device(config)# snmp-server group admin v2c write view2 notify view2
device(config)# snmp-server community comm1 group admin
device(config)# snmp-server host 10.32.147.6 comm1 version 2c severity-level Warning
device(config)# snmp-server enable trap
```

Configuring SNMPv3

SNMPv3 uses SNMP users to restrict SNMP access. When you map an SNMP user to an SNMP group, you can restrict the access of MIBs for SNMP requests through an SNMP view.

To configure SNMPv3, perform the following steps.

1. In privileged EXEC mode, enter global configuration mode.

   ```
   device# configure terminal
   ```

2. Configure the contact information for the SNMP server.

   ```
   device(config)# snmp-server contact "Network Management group - Contact # 123-123-1234"
   ```

   This example changes the default contact information from Field Support to "Network Management group - Contact # 123-123-1234".

   The double quotes allows you to enter the string with spaces.

3. Configure the location information for the SNMP server.

   ```
   device(config)# snmp-server location "South Room, Rack-11"
   ```

   This example changes the default location from End User Premise to "South Room, Rack-11".

   The double quotes allows you to enter the string with spaces.

4. Add an SNMP view to restrict or allow access to the MIB OIDs.

   ```
   device(config)# snmp-server view view2 1.3.6.1 included
   ```

   This example adds the SNMP view2 view entry with included permission to allow access for the MIB 1.3.6.1 object ID ('internet').

5. Add an SNMP group.

   ```
   device(config)# snmp-server group group1 v3 write view2 notify view2
   ```

   This example adds the group1 group for SNMPv3 and maps the read-write access and notify views to view2.

6. Add an SNMP user and associate it with a group.

   ```
   device(config)# snmp-server user user2 groupname group1 auth md5 auth-password private123 priv DES
   priv-password public123
   ```

   This example adds the user2 user and associates it with the group1 group to access of MIBs for SNMPv3 requests. For SNMPv3 users, the passwords for **auth-password** and **priv-password** keywords are encrypted while storing to the persistent memory or displaying it back to the user. You can configure either with a plain-text password or an encrypted password. In both cases, the **show running-config** command displays the passwords as encrypted.

7. Configure the SNMPv3 trap host associated with an SNMP user.

   ```
   device(config)# snmp-server v3host 10.26.3.166 user2 severity-level Info udp-port
   4425
   ```

   This example configures 10.26.3.166 as an SNMPv3 trap recipient host on the target port 4425 and associates the user2 user.

   The global SNMPv3 host can be associated with global SNMPv3 users only. You cannot create an SNMPv3 host in a global configuration by associating it with local SNMPv3 users.

8. Enable the traps.

   ```
   device(config)# snmp-server enable trap
   ```

9. Access privileged EXEC mode.

   ```
   device(config)# exit
   ```

10. Verify the configuration.

    ```
    device# show running-config snmp-server
    snmp-server contact "Network Management group - Contact # 123-123-1234"
    snmp-server enable trap
    snmp-server location "South Room, Rack-11"
    snmp-server group group1 v3 auth write view2 notify view2
    snmp-server user user2 groupname group1 md5 auth-password private123 priv DES priv-password public123
    snmp-server v3host 10.26.3.166 user2
     severity-level Info
     udp-port 4425
    !
    snmp-server view view2 1.3.6.1 included
    ```

The following example shows the previous steps to configure SNMPv3.

```
device# configure terminal
device(config)# snmp-server contact  "Network Management group - Contact # 123-123-1234"
device(config)# snmp-server location "South Room, Rack-11"
device(config)# snmp-server view view2 1.3.6.1 included
device(config)# snmp-server group group1 v3 auth write view2 notify view2
device(config)# snmp-server user user2 groupname group1 md5 auth-password private123 priv DES priv-password
public123
device(config)# snmp-server v3host 10.26.3.166 user2 severity-level Info udp-port 4425
device(config)# snmp-server enable trap
```

Configuring an SNMP server context to a VRF

A single SNMP agent can support multiple instances of the same MIB module by mapping the context name to a virtual routing and forwarding (VRF) instance created within the device. The SNMP context name is used to identify the VRF and fetch the MIB details of the mapped VRF from the underlying modules.

To configure an SNMP server context to a VRF for SNMPv1 or SNMPv2, perform the following steps.

NOTE
For SNMPv3, use the **snmp-server context** command only. The SNMPv3 request PDU itself has the provision for the context name as input.

1. In privileged EXEC mode, enter global configuration mode.

   ```
   device# configure terminal
   ```

2. Map the context with the community.

   ```
   device(config)# snmp-server community public groupname admin
   ```

3. Create a context and map it with a VRF.

   ```
   device(config)# snmp-server context mycontext vrf myvrf
   ```

4. Map the community to the context.

   ```
   device(config)# snmp-server mib community-map public context mycontext
   ```

5. Verify the configuration.

   ```
   device# show running-config snmp-server
   ...
   snmp-server community public groupname admin
   snmp-server context mycontext vrf myvrf
   ...
   snmp-server mib community-map public context mycontext
   ```

The following example shows the previous steps for the configuration.

```
device# configure terminal
device(config)# snmp-server community public groupname admin
device(config)# snmp-server context mycontext vrf myvrf
device(config)# snmp-server mib community-map public context mycontext
```

The following example shows steps to configure SNMP server context to a management VRF.

```
device# configure terminal
device(config)# snmp-server community community1 groupname admin
device(config)# snmp-server context context1 vrf mgmt-vrf
device(config)# snmp-server mib community-map community1 context context1
```

For information on supported MIBs, refer to *Extreme SLX-OS MIB Reference Guide*.

LLDP

LLDP overview

The IEEE 802.1AB Link Layer Discovery Protocol (LLDP) enhances the ability of network management tools to discover and maintain accurate network topologies and simplify LAN troubleshooting in multi-vendor environments. To efficiently and effectively operate the various devices in a LAN you must ensure the correct and valid configuration of the protocols and applications that are enabled on these devices. With Layer 2 networks expanding dramatically, it is difficult for a network administrator to statically monitor and configure each device in the network.

Using LLDP, network devices such as routers and switches advertise information about themselves to other network devices and store the information they discover. Details such as device configuration, device capabilities, and device identification are advertised. LLDP defines the following:

- A common set of advertisement messages.
- A protocol for transmitting the advertisements.
- A method for storing the information contained in received advertisements.

NOTE
LLDP runs over the data-link layer which allows two devices running different network layer protocols to learn about each other.

LLDP information is transmitted periodically and stored for a finite period. Every time a device receives an LLDP advertisement frame, it stores the information and initializes a timer. If the timer reaches the time to live (TTL) value, the LLDP device deletes the stored information ensuring that only valid and current LLDP information is stored in network devices and is available to network management systems.

Layer 2 topology mapping

The LLDP protocol lets network management systems accurately discover and model Layer 2 network topologies.

As LLDP devices transmit and receive advertisements, the devices store information they discover about their neighbors. Advertisement data such as a neighbor's management address, device type, and port identification is useful in determining what neighboring devices are in the network.

NOTE
The Extreme LLDP implementation supports up to two neighbors.

The higher level management tools, such as the Network Advisor, can query the LLDP information to draw Layer 2 physical topologies. The management tools can continue to query a neighboring device through the device's management address provided in the LLDP information exchange. As this process is repeated, the complete Layer 2 topology is mapped.

In LLDP the link discovery is achieved through the exchange of link-level information between two link partners. The link-level information is refreshed periodically to reflect any dynamic changes in link-level parameters. The basic format for exchanging information in LLDP is in the form of a type, length, value (TLV) field.

LLDP keeps a database for both local and remote configurations. The LLDP standard currently supports three categories of TLVs. The Extreme LLDP implementation adds a proprietary Extreme extension TLV set. The four TLV sets are described as follows:

- Basic management TLV set — This set provides information to map the Layer 2 topology and includes the following TLVs:
  - Chassis ID TLV — Provides the ID for the switch or router where the port resides. This is a mandatory TLV.
  - Port ID TLV—Provides a unique identifiable information of the port. The Port ID could be one of the following: MAC address, Network address, Interface name of the port. On the SLX-OS, the interface name of the port is provided. This is a mandatory TLV.
  - Port description TLV — Provides a description of the port in an alphanumeric format. If the LAN device supports RFC-2863, the port description TLV value equals the "ifDescr" object. This is an optional TLV.
  - System name TLV — Provides the system-assigned name in an alphanumeric format. If the LAN device supports RFC-3418, the system name TLV value equals the "sysName" object. This is an optional TLV.
  - System description TLV — Provides a description of the network entity in an alphanumeric format. This includes system name, hardware version, operating system, and supported networking software. If the LAN device supports RFC-3418, the value equals the "sysDescr" object. This is an optional TLV.
  - System capabilities TLV — Indicates the primary functions of the device and whether these functions are enabled in the device. The capabilities are indicated by two octets. The first octet indicates Other, Repeater, Bridge, WLAN AP, Router, Telephone, DOCSIS cable device, and Station, respectively. The second octet is reserved. This is an optional TLV.
  - Management address TLV — Indicates the addresses of the local switch. Remote switches can use this address to obtain information related to the local switch. This is an optional TLV.
- IEEE 802.1 organizational TLV set — This set provides information to detect mismatched settings between local and remote devices. A trap or event can be reported once a mismatch is detected. This is an optional TLV. This set includes the following TLVs:
  - Port VLANID TLV — Indicates the port VLAN ID (PVID) that is associated with an untagged or priority tagged data frame received on the VLAN port.
  - PPVLAN ID TLV — Indicates the port- and protocol-based VLAN ID (PPVID) that is associated with an untagged or priority tagged data frame received on the VLAN port. The TLV supports a "flags" field that indicates whether the port is capable of supporting port- and protocol-based VLANs (PPVLANs) and whether one or more PPVLANs are enabled. The number of PPVLAN ID TLVs in a Link Layer Discovery Protocol Data Unit (LLDPDU) corresponds to the number of the PPVLANs enabled on the port.
  - VLAN name TLV — Indicates the assigned name of any VLAN on the device. If the LAN device supports RFC-2674, the value equals the "dot1QVLANStaticName" object. The number of VLAN name TLVs in an LLDPDU corresponds to the number of VLANs enabled on the port.
  - Protocol identity TLV — Indicates the set of protocols that are accessible at the device's port. The protocol identity field in the TLV contains a number of octets after the Layer 2 address that can enable the receiving device to recognize the protocol. For example, a device that wishes to advertise the spanning tree protocol includes at least eight octets: 802.3 length (two octets), LLC addresses (two octets), 802.3 control (one octet), protocol ID (two octets), and the protocol version (one octet).
- IEEE 802.3 organizational TLV set — This is an optional TLV set. This set includes the following TLVs:
  - MAC/PHY configuration/status TLV — Indicates duplex and bit rate capabilities and the current duplex and bit rate settings of the local interface. It also indicates whether the current settings were configured through auto-negotiation or through manual configuration.
  - Power through media dependent interface (MDI) TLV — Indicates the power capabilities of the LAN device.
  - Link aggregation TLV — Indicates whether the link (associated with the port on which the LLDPDU is transmitted) can be aggregated. It also indicates whether the link is currently aggregated and provides the aggregated port identifier if the link is aggregated.
  - Maximum Ethernet frame size TLV — Indicates the maximum frame size capability of the device's MAC and PHY implementation.

DCBX

Storage traffic requires a lossless communication which is provided by DCB. The Data Center Bridging (DCB) Capability Exchange Protocol (DCBX) is used to exchange DCB-related parameters with neighbors to achieve more efficient scheduling and a priority-based flow control for link traffic.

DCBX uses LLDP to exchange parameters between two link peers; DCBX is built on the LLDP infrastructure for the exchange of information. DCBX-exchanged parameters are packaged into organizationally specific TLVs. The DCBX protocol requires an acknowledgment from the other side of the link, therefore LLDP is turned on in both transmit and receive directions. DCBX requires version number checking for both control TLVs and feature TLVs.

DCBX interacts with other protocols and features as follows:

- *LLDP* — LLDP is run in parallel with other Layer 2 protocols such as RSTP and LACP. DCBX is built on the LLDP infrastructure to communicate capabilities supported between link partners. The DCBX protocol and feature TLVs are treated as a superset of the LLDP standard.
- *QoS management* — DCBX capabilities exchanged with a link partner are passed down to the QoS management entity to set up the Extreme device to control the scheduling and priority-based flow control in the hardware.

The DCBX QoS standard is subdivided into two features sets:

- Enhanced Transmission Selection
- Priority Flow Control

Enhanced Transmission Selection

In a converged network, different traffic types affect the network bandwidth differently. The purpose of Enhanced Transmission Selection (ETS) is to allocate bandwidth based on the different priority settings of the converged traffic.

For example, Inter-process communications (IPC) traffic can use as much bandwidth as needed and there is no bandwidth check; LAN and SAN traffic share the remaining bandwidth. The following table displays three traffic groups: IPC, LAN, and SAN. ETS allocates the bandwidth based on traffic type and also assigns a priority to the three traffic types as follows: Priority 7 traffic is mapped to priority group 0 which does not get a bandwidth check, priority 2 and priority 3 are mapped to priority group 1, priorities 6, 5, 4, 1 and 0 are mapped to priority group 2.

The priority settings shown in the following table are translated to priority groups in the Extreme VDX hardware.

TABLE 22 ETS priority grouping of IPC, LAN, and SAN traffic

Priority	Priority group	Bandwidth check
7	0	No
6	2	Yes
5	2	Yes
4	2	Yes
3	1	Yes
2	2	Yes
1	2	Yes
0	2	Yes

Priority Flow Control

With Priority Flow Control (PFC), it is important to provide lossless frame delivery for certain traffic classes while maintaining existing LAN behavior for other traffic classes on the converged link. This differs from the traditional 802.3 PAUSE type of flow control where the pause affects all traffic on an interface.

PFC is defined by a one-byte bitmap. Each bit position stands for a user priority. If a bit is set, the flow control is enabled in both directions (Rx and Tx).

LLDP configuration guidelines and restrictions

Follow these LLDP configuration guidelines and restrictions when configuring LLDP:

- The Extreme implementation of LLDP supports standard LLDP information.
- Mandatory TLVs are always advertised.
- The exchange of LLDP link-level parameters is transparent to the other Layer 2 protocols. The LLDP link-level parameters are reported by LLDP to other interested protocols.

Configuring and managing LLDP

The following sections discuss working with the Link Layer Discovery Protocol (LLDP) on Extreme devices.

Understanding the default LLDP

The following table lists the default LLDP configuration. Consider this when making changes to the defaults.

TABLE 23 Default LLDP configuration

Parameter	Default setting
LLDP global state	Enabled
LLDP receive	Enabled
LLDP transmit	Enabled
Transmission frequency of LLDP updates	30 seconds
Hold time for receiving devices before discarding	120 seconds
DCBX-related TLVs to be advertised	dcbx-tlv

Disabling LLDP globally

LLDP is enabled globally by default. You can disable LLDP globally without changing any other aspect of the LLDP configuration.

To globally disable LLDP, perform the following steps:

1. From privileged EXEC mode, access global configuration mode.

   ```
   device# configure terminal
   ```

2. Access LLDP configuration mode.

   ```
   device(config)# protocol lldp
   ```

3. Disable LLDP globally.

   ```
   device(config-lldp)# disable
   ```

The following configuration is an example of the previous steps to disable LLDP.

```
device# configure terminal
device(config)# protocol lldp
device(config-lldp)# disable
```

If required, re-enable LLDP.

```
device(config-lldp)# no disable
```

Configuring LLDP global parameters

When LLDP is enabled, the default values of its parameters are set. You can change the configuration of these parameters in LLDP configuration mode.

Specifying a system name for the Extreme device hardware

The global system name for LLDP is useful for differentiating between devices. By default, the host name from the chassis/entity management information base is used. By specifying a descriptive system name, you may find it easier to distinguish the device with LLDP. The following example changes the system name to Brocade_Alpha.

```
device(conf-lldp)# system-name Brocade_Alpha
```

Specifying an LLDP system description

The default system description depends on the device type.

Extreme recommends that you use the operating system version for the description or use the description from the chassis/entity management information base (MIB).

Do not use special characters, such as #$!@, as part of the system name and description. The following example specifies the IT_1.6.2_LLDP_01 system description.

```
device(conf-lldp)# system-description IT_1.6.2_LLDP_01
```

Specifying a user description for LLDP

A user description for LLDP is for network administrative purposes and is not seen by neighboring devices. The following example specifies the LLDP-installed-jan-25 description.

```
device(conf-lldp)# description Brocade-LLDP-installed-jan-25
```

Enabling and disabling the receiving and transmitting of LLDP frames

By default both transmit and receive for LLDP frames is enabled.

- The following example enables only receiving of LLDP frames.

  ```
  device(conf-lldp)# mode rx
  ```

- The following example enables only transmitting of LLDP frames.

  ```
  device(conf-lldp)# mode tx
  ```

Configuring the transmit frequency of LLDP frames

The default transmit frequency of LLDP frames is 30 seconds. You can change the frequency from 4 to 180 seconds. The following example changes the frequency to 45 seconds.

```
device(conf-lldp)# hello 45
```

Configuring the hold time for receiving devices

By default, four consecutive LLDP hello packets can be missed before removing the neighbor information. You can configure from 1 to 10 consecutive LLDP hello packets that can be missed before removing the neighbor information. The following example configures the 6 consecutive LLDP hello packets.

```
device(conf-lldp)# multiplier 6
```

Advertising the optional LLDP TLVs

By default, the port description and system name are advertised

You can advertise the rest of the optional LLDP TLVs. The following example advertises the management address, capabilities, name and description of the device, and user-configured port.

```
device(conf-lldp)# advertise optional-tlv management-address port-description system-capabilities system-
name system-description
```

Configuring the advertisement of LLDP DCBX-related TLVs

The following TLV is advertised by default:

- dcbx-tlv

Configuring the advertisement of LLDP organizationally-specific TLVs

By default, dcbx-tlv is advertised.

You have the option of advertising dot1.tlv and dot3.tlv. The following example advertise dot1.tlv.

> NOTE
> Extreme does not recommend advertising dot1.tlv and dot3.tlv LLDPs if your network contains Converged Network Adapters (CNAs) from non-Extreme vendors. Functionality problems can occur.

```
device(conf-lldp)# advertise dot1-tlv
```

Configuring LLDP profiles

SLX 9240 supports 128 active profiles and SLX 9140 supports 72 active profiles. When you configure a profile, its default parameters are from the global LLDP configuration.

1. In privileged EXEC mode, access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter LLDP configuration mode.

   ```
   device(config)# protocol lldp
   ```

3. Configure the profile name.

   ```
   device(conf-lldp)# profile UK_LLDP_IT
   ```

4. Specify a description for the profile.

   ```
   device(conf-lldp-profile-UK_LLDP_IT)#description standard_profile_by_Jane
   ```

5. Configure the transmission frequency of LLDP updates.

   ```
   device(conf-lldp-profile-UK_LLDP_IT)# hello 10
   ```

6. Configure the hold time for receiving devices.

   ```
   device(conf-lldp-profile-UK_LLDP_IT)# multiplier 2
   ```

7. Advertise the optional LLDP TLVs.

   ```
   device(conf-lldp)# advertise optional-tlv system-name
   ```

8. Advertise the LLDP organizationally-specific TLVs.

   ```
   device(conf-lldp-profile-UK_LLDP_IT)# advertise advertise dot1-tlv
   ```

   NOTE
   Extreme does not recommend advertising dot1.tlv and dot3.tlv LLDPs if your network contains CNAs from non-Extreme vendors. Functionality problems can occur.

9. Return to privileged EXEC mode.

   ```
   device(conf-lldp-profile-UK_LLDP_IT)# end
   ```

10. Verify the configuration.

   ```
   device# show running-config protocol lldp profile
    profile UK_LLDP_IT
     hello 10
     multiplier 2
     advertise dot1-tlv
     advertise option-tlv system-name
     description standard_profile_by_Jane
   ```

The following configuration is an example of the previous steps.

```
device# configure terminal
device(config)# protocol lldp
device(conf-lldp)# profile UK_LLDP_IT
device(conf-lldp-profile-UK_LLDP_IT)# description standard_profile_by_Jane
device(conf-lldp-profile-UK_LLDP_IT)# hello 10
device(conf-lldp-profile-UK_LLDP_IT)# multiplier 2
device(conf-lldp-profile-UK_LLDP_IT)# advertise option-tlv system-name
device(conf-lldp-profile-UK_LLDP_IT)# advertise advertise dot1-tlv
```

Configuring iSCSI priority

The iSCSI TLV is used only to advertise the iSCSI traffic configuration parameters to the attached CEE enabled servers and targets. No verification or enforcement of the usage of the advertised parameters by the iSCSI server or target is done by the switch. The iSCSI priority setting is used to configure the priority to be advertised in the DCBx iSCSI TLV.

To configure the iSCSI priority, perform the following steps from privileged EXEC mode.

1. Enter the **configure terminal** command to access global configuration mode.
2. Enter LLDP configuration mode.

   ```
   device(config)# protocol lldp
   ```

3. Configure the iSCSI priority.

   ```
   device(conf-lldp)# iscsi-priority 4
   ```

   NOTE
   The default iscsi-priority is 4 and does not display unless you change the iscsi-priority to a different value.

4. Advertise the TLV.

   ```
   device(conf-lldp)# advertise dcbx-iscsi-app-tlv
   ```

Configuring the iSCSI profile

You can configure an iSCSI profile to be applied to individual interfaces. However, the priority bit must be set manually for each interface.

To configure iSCSI profiles, perform the following steps from privileged EXEC mode.

1. Configure the CEE map, if it has not already been created.

   ```
   device(config)# cee-map default
   device(conf-ceemap)# priority-group-table 1 weight 50 pfc
   device(conf-ceemap)# priority-group-table 2 weight 30 pfc on
   device(conf-ceemap)# priority-group-table 3 weight 20 pfc on
   device(conf-ceemap)# priority-table 1 1 1 1 2 3 1 1
   ```

   The **priority-table** command syntax is as follows:

   ```
   priority-table PGID0 PGID1 PGID2 PGID3 PGID4 PGID5 PGID6 PGID7
   ```

   For all PGID values, the PGID value range is 0 through 7 for the DWRR Priority Group, and 15.0 through 15.7 for the Strict Priority Group. The PGID value and the CoS value are equivalent, so that specifying PGID0 sets the Priority Group ID for all packets with CoS = 0, specifying PGID1 sets the Priority Group ID for all packets with CoS = 1, all the way through specifying PGID7, which sets the Priority Group ID for all packets with CoS = 7.

   Priority-Table in CEE map configuration requires that PGID 15.0 is dedicated for CoS7. Because of this restriction, make sure that PGID15.0 is configured only as the last parameter for Priority-Table configuration.

   An explanation of syntax "priority-table 1 2 2 2 2 2 2 15.0" is as follows:

   This shows the definition of a CEE Map with Priority to Priority Group mapping of CoS=1, CoS=2, CoS=3, CoS=4, CoS=5, and CoS=6 to a DWRR Priority Group ID of 2, and CoS=0 to a Priority Group ID of 1, and CoS=7 to a Strict Priority Group.

   This is one way to provision the CEE Priority to Priority Group Table, which maps each of the eight ingress CoS into a Priority Group.

2. Enter LLDP configuration mode.

   ```
   device(conf-ceemap)# protocol lldp
   ```

3. Create an LLDP profile for iSCSI.

   ```
   device(conf-lldp)# profile iscsi_config
   ```

4. Advertise the iSCSI TLV.

   ```
   device(conf-lldp-profile-iscsi_config)# advertise dcbx-iscsi-app-tlv
   ```

5. Enter configuration mode for the specific interface.

   ```
   device (conf-lldp-profile-iscsi_config)# interface ethernet 0/1
   ```

6. Apply the CEE provisioning map to the interface.

   ```
   device(conf-if-eth-0/1)# cee default
   ```

7. Apply the LLDP profile you created for iSCSI.

   ```
   device(conf-if-eth-0/1)# lldp profile iscsi_config
   ```

8. Set the iSCSI priority bits for the interface.

   ```
   device(conf-if-eth-0/1)# lldp iscsi-priority 4
   ```

9. Repeat steps 5 through 8 for additional interfaces.

Configuring an LLDP profile to an interface

You can assign only one LLDP profile to an interface. If you do not use the **lldp profile** option at the interface level, the interface uses the global LLDP configuration.

To configure LLDP interface-level command options, perform the following steps.

1. In privileged EXEC mode, access global configuration mode.

   ```
   device# configure terminal
   ```

2. Access the interface configuration mode.

   ```
   device(config)# interface Ethernet 0/8
   ```

3. Apply an LLDP profile to the interface.

   ```
   device(conf-if-eth-0/8)# lldp profile network_standard
   ```

4. Return to privileged EXEC mode.

   ```
   device(conf-if-eth-0/8)# end
   ```

5. Save the running-config file to the startup-config file.

   ```
   device# copy running-config startup-config
   ```

The following configuration is an example of the previous steps.

```
device# configure terminal
device(config)# interface Ethernet 0/8
device(conf-if-eth-0/8)# lldp profile network_standard
```

Displaying LLDP information

The **show lldp** command allows you to display the following information:

- LLDP status
- LLDP neighbor information
- LLDP statistics

Displaying LLDP status

To display the global LLDP status, use the **show lldp** command.

```
device# show lldp
LLDP Global Information
  system-name: SLX
  system-description: Brocade BR-SLX9140 Switch
  description: Brocade-LLDP
  State:                   Enabled
  Mode:                    Receive/Transmit
  Advertise Transmitted:   30 seconds
  Hold time for advertise: 120 seconds
  Tx Delay Timer:          1 seconds
  Transmit TLVs:           Chassis ID               Port ID
                           TTL                      Port Description
                           System Name
```

To display LLDP status for an Ethernet interface, use the **show lldp interface ethernet** command.

```
device# show lldp interface ethernet 0/18
LLDP information for Eth 0/18
  State:                   Enabled
  Mode:                    Receive/Transmit
  Advertise Transmitted:   30 seconds
  Hold time for advertise: 120 seconds
  Tx Delay Timer:          1 seconds
  Transmit TLVs:           Chassis ID               Port ID
                           TTL                      Port Description
                           System Name
```

Displaying LLDP neighbor information

To display the LLDP neighbor information, use the **show lldp neighbors** command. This command allows you to display the information for all Ethernet interfaces, a specific interface, or detailed neighbor information.

The following example displays the LLDP neighbor information for all interfaces.

```
device# show lldp neighbors
Local Port  Dead Interval  Remaining Life  Remote Port ID   Remote Port Descr Chassis ID        Tx  Rx
System Name
Eth 0/18      120            102             Ethernet 0/25   Eth 0/25         768e.f807.6000    653 652  R6
Eth 0/21      120            108             Ethernet 0/21   Eth 0/21         768e.f807.6000    653 652  R6
Eth 0/40      120            110             Ethernet 0/50   Eth 0/50         768e.f807.6000    653 650  R6
Eth 0/43      120            102             Ethernet 0/51   Eth 0/51         768e.f807.6000    653 652  R6
Eth 0/50      120            102             Ethernet 0/23   Eth 0/23         768e.f807.6000    653 611  R6
```

The following example displays the LLDP neighbor information for Ethernet interface 0/18.

```
device# show lldp neighbors interface ethernet 0/18
Local Port  Dead Interval  Remaining Life  Remote Port ID   Remote Port Descr Chassis ID       Tx  Rx
System Name
Eth 0/18      120            115             Ethernet 0/25    Eth 0/25        768e.f807.6000   655 654  R6
```

The following example displays the detailed LLDP neighbor information for Ethernet interface 0/18.

```
device# show lldp neighbors interface ethernet 0/18 detail
Neighbors for Interface Eth 0/18

MANDATORY TLVs
===============
Local Interface: Eth 0/18  (Local Interface MAC: 768e.f805.5816)
Remote Interface: Ethernet 0/25 (Remote Interface MAC: 768e.f807.610d)
Dead Interval: 120 secs
Remaining Life : 118 secs
Chassis ID: 768e.f807.6000
LLDP PDU Transmitted: 656  Received: 655

OPTIONAL TLVs
==============
Port Interface Description: Eth 0/25
System Name: R6
```

Displaying LLDP statistics

To display the LLDP statistics for all interfaces or a specific interface, use the **show lldp statistics** command.

The following example displays the statistics for Ethernet interface 0/18.

```
device# show lldp statistics interface ethernet 0/18
LLDP Interface statistics for Eth 0/18
Frames transmitted: 659
Frames Aged out:    0
Frames Discarded:   0
Frames with Error:  0
Frames Recieved:    657
TLVs discarded:     0
TLVs unrecognized:  0
```

If you do not include the **interface ethernet** option, the command displays the statistics for all interfaces.

Clearing LLDP-related information

You can clear LLDP neighbor and statistic information for all interfaces or a specified interface.

To clear LLDP-related information, perform the following steps.

1. In privileged EXEC mode, clear the LLDP neighbor information.

   ```
   device# clear lldp neighbors
   ```

   This example clears the LLDP neighbor information for all interfaces.

2. Clear the LLDP statistics on an interface.

   ```
   device# clear lldp statistics interface ethernet 0/8
   ```

   This example clears the LLDP transmit and receive counters on the Ethernet interface 0/8.

3. Clear the LLDP statistics for all interfaces.

   ```
   device# clear lldp statistics
   ```

Password Recovery

Recovering the admin password from the root account

If you lose access to the SLX-OS admin account but you have access to the root account for the device, you can recover the password.

Perform the following steps to reset the admin password from the root account.

1. Open a session to access the device.
2. Log in as root.
3. Start the SLX-OS CLI.

   ```
   [root@device]# slxcli
   device#
   ```

4. Access global configuration mode.

   ```
   device# configure terminal
   ```

5. Reset the admin password.

   ```
   device(config)# username admin password password
   ```

   In this example, the admin password is reset to the default value of password.

   You can now use the admin account to manage the admin and user passwords by using normal password-management procedures.

VM root password recovery for Extreme device

By default, the root account on the virtual machine (VM) for Extreme device is disabled. To log into root, you can log into the SLX-OS CLI and enable the root account from global configuration mode by using **root enable** command. In rare cases, SLX-OS CLI may not be available to enable the root account.

The ability to enable the root account and recover the root credentials (password) depends on the bootenv environment variable. When the variable is set, it executes the root recovery logic based on the parameter set. The variable is not preserved across reboot. Every time a reboot occurs, the root account is disabled by default and this variable has to be set again to enable it unless the root account was not enabled from global configuration mode.

The root account access availability determines the method for password recovery:

- When the root account is disabled and the SLX-OS CLI is not available, you must recover the root login account. The password is also recovered.
- When the root account is enabled but you forget the password, you must recover the VM root password.

NOTE
The default password for the root account on the VM is fibranne.

Recovering the VM root login account

When you need to recover the VM root password, and the root account is disabled and SLX-OS CLI is not available, you must recover the VM root login account.

- To perform the recovery process on the device, you must have access to the console prompt.

For VM root login recovery, perform the following steps.

1. Access the device from the Privileged EXEC mode and reboot the device.

   ```
   device# reload system

   Warning: This operation will cause the chassis to reboot and requires all existing telnet, secure
   telnet and SSH sessions to be
   restarted.
   Unsaved configuration will be lost. Please run `copy running-config startup-config` to save the
   current configuration if not done already.

   Are you sure you want to reboot the chassis [y/n]? y
   ```

   NOTE
   If SLX-OS CLI is not available, then power cycle the device.

2. Press the ESC key during the switch reboot. From the subsequent menu that is displayed, select **ONIE** option by pressing the arrow key.

   ```
   Brocade SLX-
   OS                                                                                            |
   Offline
   Diagnostic                                                                                        |
   ONIE
   ```

3. At the ONIE prompt, select **ONIE Rescue** option and then define the root login value for the root recover environment variable.

   ```
   ONIE:/ # bootenv VM_Root_Recover RootLogin
   ```

4. Reboot the device.

   ```
   ONIE:/ # reboot
   ```

Recovering the VM root password

If you forgot the password for the VM root account, you can recover the default password.

- To perform the recovery process on the device, you must have access to the console prompt.

For VM root password recovery, perform the following steps.

1. Access the device from the Privileged EXEC mode and reboot the device.

   ```
   device# reload system

   Warning: This operation will cause the chassis to reboot and requires all existing telnet, secure
   telnet and SSH sessions to be
   restarted.
   Unsaved configuration will be lost. Please run `copy running-config startup-config` to save the
   current configuration if not done already.

   Are you sure you want to reboot the chassis [y/n]? y
   ```

   NOTE
   If SLX-OS CLI is not available, then power cycle the device.

2. Press the ESC key during the switch reboot. From the subsequent menu that is displayed, select **ONIE** option by pressing the arrow key.

   ```
   Brocade SLX-
   OS                                                                                           |
   Offline
   Diagnostic                                                                                       |
   ONIE
   ```

3. At the ONIE prompt, select **ONIE Rescue** option and then define the root login value for the root recover environment variable.

   ```
   ONIE:/ # bootenv VM_Root_Recover RootPasswd
   ```

4. Reboot the device.

   ```
   ONIE:/ # reboot
   ```

Python Event-Management and Scripting

Python under Extreme operating systems

The Python interpreter installed with supported Extreme operating systems enables you to access a Python shell or to launch Python scripts. You can also define event handlers that run such scripts automatically upon specified conditions.

NOTE
SLX-OS is among the Extreme operating systems that support Python.

Python overview

Python is a high-level scripting language that also supports object-oriented programming. If you have previous programming experience, you can quickly learn how to write useful, simple Python scripts.

NOTE
For Python resources, refer to http://python.org.

Working interactively in the Python shell

Use this procedure to access a Python shell, within which you can use Python commands that call and manipulate Extreme operating system commands.

NOTE
The Python shell is accessible only to admin-role users.
Python syntax is case-sensitive.

1. In privileged EXEC mode, enter **python** to access the Python shell.

   ```
   device# python
   ```

   The `device#` prompt changes to a Python prompt:

   ```
   device# python
   Python 3.3.2 (default, Apr 11 2014, 13:05:18)
   [GCC 4.8.2] on linux
   Type "help", "copyright", "credits" or "license" for more information.
   ```

2. To exit the Python shell and return to the Extreme operating system prompt, enter either:
   - `exit()`
   - `Ctrl-D`

3. To run a Extreme operating system command from within the Python shell, enter the `CLI( )` command.

   ```
   >>> cmd_show_running_ve = CLI('show running-config interface ve')
   !Command: show running-config interface ve
   !Time: Mon Aug 22 16:53:13 2016
   ```

   The statement entered above does two things:

   - Runs the **show running-config interface ve** command and displays the result.
   - Assigns that command to a Python variable named `cmd_show_running_ve`

4. To run a series of Extreme operating system commands from within the Python shell, separate the commands with `\n`.

   ```
   >>> cmd_config_ve = CLI('configure \n interface ve 101-103')
   !Command: configure
    interface ve 101-103
   !Time: Mon Aug 22 16:53:13 2016
   ```

   NOTE

   There is a difference between running a sequence of Extreme operating system CLI commands in the Python shell rather than in the standard Extreme operating system interface. Whereas in the standard interface the result of a command is persistent, in the Python shell each `CLI( )` statement is independent of any preceding ones.

In the following example, the lines beginning with **#** are added for explanation.

```
device# python
Python 3.3.2 (default, Apr 11 2014, 13:05:18)
[GCC 4.8.2] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> cmd_show_running_ve = CLI('show running-config interface ve')
!Command: show running-config interface ve
!Time: Mon Aug 22 16:53:13 2016

% No entries found.
# The SLX-OS show running-config interface ve command is run,
# and that command is assigned to the Python variable cmd_show_running_ve.

>>> cmd_config_ve = CLI('configure \n interface ve 101-103')
# A series of three commands are run and assigned to the Python variable cmd_config_ve.
!Command: configure
 interface ve 101-103
!Time: Mon Aug 22 16:53:13 2016


>>> cmd_show_running_ve.rerun()
# The rerun() function appended to cmd_show_running_ve gives the following output:
!Command: show running-config interface ve
!Time: Mon Aug 22 16:53:13 2016

interface Ve 101
 shutdown
!
interface Ve 102
 shutdown
!
interface Ve 103
 shutdown
!
!
```

Python scripts

Python scripts enable you to manipulate and launch Extreme operating system commands, taking advantage of the power and flexibility of Python. Such scripts also support event handling.

The topics in this section guide you through the process of writing and testing Python scripts, copying them to supported devices, and running them with the **python** command from the command line.

Guidelines for writing Python scripts

The general guidelines for writing Python scripts to run under SLX-OS are as follows:

- Although previous experience programming in Python is helpful, experience in other high-level languages is enough to get you started with simple Python scripts.
- The Python developer either needs experience with SLX-OS CLI or access to a resource with such experience.
- To help decide which editor or integrated development environment (IDE) to use, refer to http://www.python.org.
- Make sure that the appropriate version of the Python interpreter is installed on your development computer. For the current version of SLX-OS, install Python 3.3.2.
- Make sure that in the *Extreme SLX-OS Command Reference* you are familiar with the **python** and the **CLI( )** topics.
- The script must include `from CLI import CLI`. This enables the **CLI( )** command, by which Python can interact with SLX-OS.
- Write the Python script and save it, with a `.py` suffix. Valid filenames range from 4 through 32 characters (including the suffix). The first character must be alphabetic.

Guidelines for RASLOG event-handler scripts

The additional guidelines for writing RASLOG event-handler scripts are as follows:

- The script must include `import json`.
- If an event-handler detects the log messages specified in the triggers and the trigger-function conditions are met, the script is executed. As part of this execution, the script is passed a `--raslog-triggers` argument with JSON-formatted dictionary containing the relevant message identifiers (MSGID) as keys and the message text as values.
- If the trigger-function is OR, only a single MSGID and message text are present.
- If the trigger function is AND, all relevant MSGIDs and messages are passed inside the json data structure.

NOTE
For sample scripts, refer to Python scripts and run-logs on page 125.

Testing Python-script statements

While developing a Python script, you can test Extreme operating system calls by entering them in the device Python command shell. After the script is stable, you copy it to the device and then test it further by running it from the command line.

1. In privileged EXEC mode, enter the **python** command to access the Python shell.

   ```
   device# python
   Python 3.3.2 (default, Apr 11 2014, 13:05:18)
   [GCC 4.8.2] on linux
   Type "help", "copyright", "credits" or "license" for more information.
   ```

   Note that the `device#` prompt changed to a >>> Python prompt:

2. Enter the script statements one at a time, verifying that they run as expected.

   ```
   >>> cmd_config_ve = CLI('configure \n interface ve 101-103')
   !Command: configure
    interface ve 101-103
   !Time: Mon Aug 22 16:53:13 2016
   ```

3. Make corrections as needed.

Copying Python files to the device

After writing and testing a Python script file, use one of these topics to copy it to device flash memory.

Copying a file from a USB device

Use this topic to copy a file from a USB stick to an Extreme device.

> ATTENTION
> The only supported USB device for this task is the Extreme USB stick shipped with the device.

1. Copy the Python script file to the USB stick.
2. Insert the USB stick into the device USB port and enter the **usb on** command.
3. In privileged EXEC mode, enter the **copy** command to copy the Python file from the USB stick to the device flash memory.

   ```
   device# copy usb://pythscript1.py flash://pythscript1.py
   ```

4. To display a list of files in the device flash memory, enter the **dir** command.

   ```
   device# dir
   total 32
   drwxr-xr-x 2  251 1011  4096  Aug 26 08:44 .
   drwxr-xr-x 3  251 1011  4096  Jul 20 07:50 ..
   -rw-r--r-- 1 root root  1051  Mar 24 16:09 create_po.py
   -rw-r--r-- 1 root sys    695  Aug 23 21:18 defaultconfig.cluster
   -rw-r--r-- 1 root root   410  Aug 26 04:06 defaultconfig.standalone
   -rw-r--r-- 1 root root 10042  Aug  4 00:01 ospfnsr1.cfg
   -rw-r--r-- 1 root root   410  Aug 26 01:28 startup-config

   16908197888 bytes total (10226114560  bytes free)
   ```

5. To display the contents of a Python file copied into the device flash memory, enter the **show file** command.

   ```
   device# show file pythscript1.py
   ```

Downloading a file from a network

Use this topic to download a file from a network location to the Extreme device.

1. Make sure that the Python script file is uploaded to an accessible network location.
2. In privileged EXEC mode, enter the **copy** command to copy the Python file from the network location to the device flash memory.

   ```
   device# copy ftp://MyUserID:MyPassword@10.10.10.10//pythscript1.py flash://pythscript1.py
   ```

   For other file-transfer options, refer to the *Extreme SLX-OS Command Reference* **copy** topic.
3. To display a list of files in the device flash memory, enter the **dir** command.

   ```
   device# dir
   total 32
   drwxr-xr-x 2  251 1011  4096  Aug 26 08:44 .
   drwxr-xr-x 3  251 1011  4096  Jul 20 07:50 ..
   -rw-r--r-- 1 root root  1051  Mar 24 16:09 create_po.py
   -rw-r--r-- 1 root sys    695  Aug 23 21:18 defaultconfig.cluster
   -rw-r--r-- 1 root root   410  Aug 26 04:06 defaultconfig.standalone
   -rw-r--r-- 1 root root 10042  Aug  4 00:01 ospfnsr1.cfg
   -rw-r--r-- 1 root root   410  Aug 26 01:28 startup-config

   16908197888 bytes total (10226114560  bytes free)
   ```

4. To display the contents of a Python file copied into the device flash memory, enter the **show file** command.

   ```
   device# show file pythscript1.py
   ```

Running Python scripts from the command line

The facility to run Python scripts from the Extreme operating system command line enables you to execute complex and repetitious tasks with accuracy and efficiency. This facility also enables you to validate scripts intended for event management.

CAUTION
Make sure that you test Python scripts according to standard quality assurance practices before deploying them.

In privileged EXEC mode, enter the **python** command, specifying the Python script file that you want to run.

```
device# python create_po.py
```

After the script runs, the SLX-OS prompt displays.

NOTE
The create_po.py script is discussed in "Script for assigning interfaces to port channels (create_po.py)."

Python scripts and run-logs

This section contains sample SLX-OS Python scripts and run-logs.

Script for assigning interfaces to port channels (create_po.py)

The `create_po.py` script is an example for automating common configuration tasks. This script runs the relevant **show running-config** commands before and after the following actions:

- VLAN configuration

- Port-channel configuration
- Adding interfaces to port channels

NOTE
Lines beginning with # are annotations.

```
#Required in all scripts for SLX-OS:
from CLI import CLI

slot = [0]
interfaces = [28, 29, 30, 31]
port_channel = 10
vlan_range = "101-105"

# Runs show running-config int vlan before the configuration, and assigns this SLX
# command to a Python variable named cmd_show_running_vlans.
cmd_show_running_vlans = CLI("show running-config vlan")

# Configures VLANs.  {} is a placeholder for the format (arg1, arg2, ... argN) variables:
cmd_configure_vlans = CLI("config \n vlan {}".format(vlan_range))

# Reruns show running-config int vlan, following definition of new VLANs:
cmd_show_running_vlans.rerun()

# Configures port channel (vLAG):
cmd_show_running_port_channels = CLI("show running-config int po")
cmd_configure_port_channel = CLI("config \n int po {} \n switchport \n switchport mode trunk \n switchport
trunk allowed vlan add {} \n switchport trunk tag native-vlan ; no shut".format(port_channel, vlan_range))

# Reruns show running-config int po, following configuration changes:
cmd_show_running_port_channels.rerun()

# Adds interfaces to port channel (vLAG)

for interface in interfaces:
    cmd_configure_interface = CLI("config \n int eth {}/{} \n channel-group {} mode active type standard \n
no shut".format(slot, interface, port_channel))
    cmd_show_running_int_tengig = CLI("show running-config int eth {}/{}".format (slot, interface))

# Runs show running-config:
cmd_show_running = CLI("show running-config")
```

Run-log (create_po.py)

A log upon running `create_po.py` was as follows:

```
SLX# python create_po.py                                                                      !Command: show
running-config vlan
!Time: Fri Dec 16 18:35:41 2016

vlan 1
!
vlan dot1q tag native

!Command: config
 vlan 101-105
!Time: Fri Dec 16 18:35:41 2016

!Command: show running-config vlan
!Time: Fri Dec 16 18:35:41 2016

vlan 1
!
vlan 101
!
vlan 102
!
vlan 103
```

```
!
vlan 104
!
vlan 105
!
vlan dot1q tag native

!Command: show running-config int po
!Time: Fri Dec 16 18:35:41 2016

% No entries found.

!Command: config
 int po 10
 switchport
 switchport mode trunk
 switchport trunk allowed vlan add 101-105
 switchport trunk tag native-vlan ; no shut
!Time: Fri Dec 16 18:35:41 2016


!Command: show running-config int po
!Time: Fri Dec 16 18:35:42 2016

interface Port-channel 10
 switchport
 switchport mode trunk
 switchport trunk allowed vlan add 101-105
```

Script illustrating the .get_output function (get_output.py)

The .get_output function returns—as a list—the output of the SLX-OS CLI commands assigned to a Python object.

This script displays a list of firmware version per slot.

> NOTE
>
> For devices without slots, running this script is equivalent to running **show version**.

```
#Required in all scripts for SLX:
from CLI import CLI
# Import the Python Regular Expressions (re) module:
import re
# Create Python objects:
slot_firmware = {}

cmd_show_ver = CLI("show ver", False)
# Using .get_output(), assign the result of show ver to a Python object named output:
output = cmd_show_ver.get_output()
for line in output:
    found = re.search(r'^(\S+)\s+(\S+)\s+(\S+)\s+ACTIVE.*$', line, re.M)
    if found:
        slot_firmware[found.group(1)] = found.group(3)

print("SLOT_FIRMWARE:\n")
for key in slot_firmware:
    print("\t", key, "\t=> ", slot_firmware[key])
```

Event-handler script

This is a typical script (partial) launched when a specific RASLOG message is generated.

> NOTE
>
> Lines beginning with # are annotations.

```
#!/usr/local/python/3.3.2/bin/python3
```

```
import getopt
import json
import sys
import io
import re
import pdb
from CLI import CLI

nsm_raslog = 'NSM-1020'
raslog_triggers = {}

# Argument processing:
options, remainder = getopt.gnu_getopt(sys.argv[1:], '', ['raslog-triggers='])

for opt, arg in options:
    if opt in ('--raslog-triggers'):
        output = io.StringIO(arg)
        raslog_triggers = json.load(output)

f = open("poout.txt", "a")
f.write ("testing:\n")

def shut_po_members(po_num):
    members = []
    poCLI = "show port-channel " + str(po_num) + " | begin Link:"
    output = CLI(poCLI, do_print=False).get_output()
# Print output
    for entry in output:
        str1 = "".join(entry)
        str1 = str1.lstrip()
# Print str1.startswith("Link")
        if (str1.startswith("Link") == True):
            phy = str1.split("Link: ")[1].split("(0x")[0]
            members.append(phy)
            CLI("config term\nint" + phy + "\nshutdown", do_print=False).get_output()
        str1 = ""
```

Python event-management

Python event management enables you to specify a Python script that runs automatically upon specified conditions.

You specify which RASlog message triggers the script.

Configuring an event-handler profile

Use this procedure to create an event-handler profile, define one or more triggers for it, and specify a Python script that runs upon one or more trigger events.

1. Enter **configure terminal** to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **event-handler** command.

   ```
   device(config)# event-handler eventHandler1
   ```

3. For each trigger that you need for a profile, enter the **trigger** command.

   ```
   device(config-event-handler-eventHandler1)# trigger 1 raslog LOG-1001
   ```

   The trigger event is RASlog message #LOG-1001.

4. Enter the **action python-script** command to specify a Python script that runs when the event-handler is triggered.

   ```
   device(config-event-handler-eventHandler1)# action python-script example.py
   ```

5. To add an event-handler-profile description, enter **description**, followed by the description text.

   ```
   device(config-event-handler-eventHandler1)# description This is a sample description.
   ```

CAUTION
Make sure that you test Python scripts according to standard quality assurance practices before deploying them.

The following example includes all of the steps.

```
device# configure terminal
device(config)# event-handler eventHandler1
device(config-event-handler-eventHandler1)# trigger 1 raslog LOG-1001
device(config-event-handler-eventHandler1)# action python-script example.py
device(config-event-handler-eventHandler1)# description This is a sample description.
```

The following example defines a trigger that uses POSIX extended REGEX to search for a match within a specified RASlog message ID.

```
device# configure terminal
device(config-event-handler-eventHandler1)# event-handler eventHandler2
device(config-event-handler-eventHandler2)# trigger 1 raslog NSM-1003 pattern Interface Ethernet 0/[1-9] is
link down
```

RASlog message NSM-1003 includes "**interface** *interface-name* is link down", indicating that an interface is offline because the link is down. The REGEX searches within such a message for an interface from 0/1 through 0/9.

Activating an event-handler

Use this procedure to activate one or more event-handlers on the device. If a trigger specified in the event-handler profile occurs, a designated Python script runs.

1. Enter **configure terminal** to access global configuration mode.

   ```
   device# configure terminal
   ```

2. Enter the **event-handler activate** command, specifying the event handler that you are activating.

   ```
   device(config)# event-handler activate eventHandler1
   ```

3. To activate an additional event handler, enter the **event-handler activate** command, specifying the additional event handler.

   ```
   device(config-activate-eventHandler1)# event-handler activate eventHandler2
   ```

4. To de-activate an event handler, enter the **no event-handler activate** command

   ```
   device(config-activate-eventHandler2)# no event-handler activate eventHandler2
   ```

The following example includes all of the activation steps.

```
device# configure terminal
device(config)# event-handler activate eventHandler1
device(config-activate-eventHandler1)# event-handler activate eventHandler2
```

Configuring event-handler options

After you activate an event handler, you can configure various options. For example, you can specify if the event-handler script runs more than once and how multiple triggers are handled.

1. Activate an event handler, as described in Configuring an event-handler profile on page 128:

   ```
   device# configure terminal
   device(config)# event-handler activate eventHandler1
   ```

2. To specify a delay from when a trigger is received until execution of the event-handler action, enter the **delay** command.

   ```
   device(config-activate-eventHandler1)# delay 60
   ```

   The above example specifies a delay of 60 seconds.

3. To specify multiple iterations of the action when a trigger is received:

   a) Enter the **iterations** command.

   b) To specify an interval between iterations, enter the **interval** command.

   ```
   device(config-event-handler-eventHandler1)# iterations 3
   device(config-activate-eventHandler1)# interval 30
   ```

   The above example sets the number of iterations to 3 and specifies an interval of 30 seconds between each iteration.

4. To specify a maximum number of minutes to wait for an action script to complete execution, enter the **action-timeout** command.

   ```
   device(config-activate-eventHandler1)# action-timeout 30
   ```

   The example sets the timeout to 30 minutes.

5. To limit action-recurrence upon multiple trigger-events, enter one of the following commands:

   - **trigger-mode only-once**—for the duration of a device configuration, the event-handler action is launched only once.

     ```
     device(config-activate-eventHandler1)# trigger-mode only-once
     ```

   - **trigger-mode on-first-instance**—as long as the device is running, the event-handler action is launched only once. Following a device restart, the event-handler action can be triggered again.

     ```
     device(config-activate-eventHandler1)# trigger-mode on-first-instance
     ```

6. If multiple triggers are defined, to specify that the action run only if all of the triggers occur, enter the **trigger-function AND time-window** command.

   ```
   device(config-activate-eventHandler1)# trigger-function AND time-window 120
   ```

   The above example specifies that the action run only if all triggers occur within 120 seconds.

Troubleshooting event-management

Use these topics to troubleshoot issues that arise during implementation of Python event-management.

Aborting an event-handler action

If needed, abort a Python script launched by an event-handler action.

1. In privileged EXEC mode, enter the **event-handler abort action** command.

   ```
   device# event-handler abort action eh1
   This operation will abort an event handler action that is currently running and may leave the device
   in an inconsistent state.  Do you want to continue? [y/n]:y
   ```

2. To confirm aborting the action, type `y`.

   ```
   Operation completed successfully.
   ```

   The Python script launched by the event-handler action is aborted.

Event-management show commands

There are several show commands that display event-management information, listed here with descriptions.

TABLE 24 Event-management show commands in the *Command Reference*

Command	Description
show event-handler activations	Displays operational data of activated event-handlers.
show running-config event-handler	Displays details of event-handler profiles defined on the device. You can display the results by Python-script action or trigger ID.

Diagnostic Commands

The following tables list and describe a variety of offline diagnostic commands, show commands, and utility commands that can be used in troubleshooting hardware.

For details, refer to the *Extreme SLX-OS Command Reference*.

The offline diagnostic utility, DiagOS, runs in a separate mode and context from SLX-OS. To enter offline diagnostic mode from the SLX-OS command prompt, enter the **reload diag-mode** command in privileged EXEC mode, as follows:

```
device# reload diag-mode
```

The switch power-cycles and reboots in offline diagnostic mode. After the user logs in again, the new prompt shown below appears.

```
diag<~>#
```

> NOTE
> Check with the Extreme support team for the default login credentials for offline diagnostic mode. These credentials are different from the normal login credentials used for SLX-OS mode.

To exit offline diagnostic mode and return to SLX-OS mode, enter the **reload slxos-mode** command from the offline diagnostic prompt, as follows:

```
diag<~># reload slxos-mode
device#
```

Test logs are collected by the supportsave utility and persist across switch reboots. The maximum collective size of all log files is 10 Mbytes. Once offline diagnostic logs are collected from the switch, they can be found in the following supportsave directory: /slotH/slotH/diag/cmdlog.

> ATTENTION
> Remain in the default directory once in offline diagnostic mode. Otherwise the following diagnostic commands will fail.

The following files are generated: portledtest_log, portloopbacktest_log, prbstest_log, turboramtest_log.

> ATTENTION
> Do not abort testing. All tests must be allowed to run to completion.

TABLE 25 Diagnostic configuration commands

Command	Description
diag burninerrclear	Clears the error logs stored in nonvolatile memory.
diag portledtest	Executes portLedTest to test LEDs on a single port or all ports on the device.
diag portloopbacktest	Executes portLoopbackTest to test the loopback interface on a single port or all ports on the device.
diag setcycle	Specifies the parameters for the system-verification test suite to be run next.
diag systemverification	Executes a system-verification test suite.

TABLE 25 Diagnostic configuration commands (continued)

Command	Description
diag turboramtest	Executes turboRamTest on the switch to check DDR SDRAM.

TABLE 26 Diagnostic show commands

Command	Description
show diag burninerrshow	Displays the error messages stored in the device's nonvolatile memory.
show diag burninstatus	Displays the entire system verification log that is stored in the device's nonvolatile memory.
show diag revision	Displays the current version of the diagnostic software.
show diag setcycle	Displays parameters set for system verification.
show diag sysinfo	Displays system hardware information.

TABLE 27 Diagnostic utility commands

Command	Description
reload	Triggers a power cycle of the switch.