# Extreme Fabric Automation

## Administration Guide, 2.1.0

9036616-00 Rev AC
April 2020

## Legal Notice

Extreme Networks, Inc. reserves the right to make changes in specifications and other information contained in this document and its website without prior notice. The reader should in all cases consult representatives of Extreme Networks to determine whether any such changes have been made.

The hardware, firmware, software or any specifications described or referred to in this document are subject to change without notice.

## Trademarks

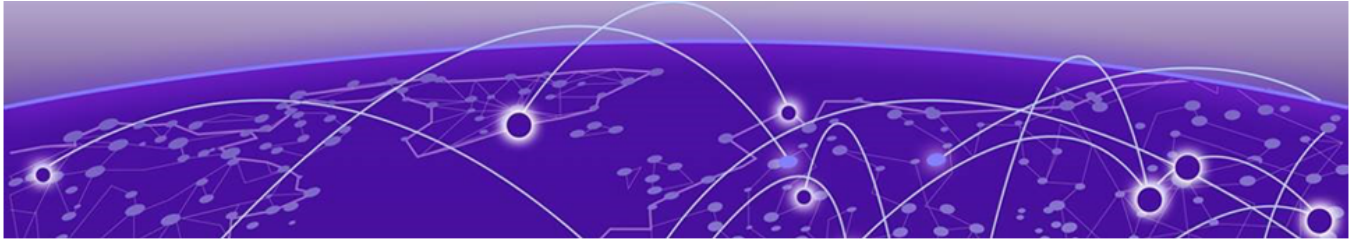
Extreme Networks and the Extreme Networks logo are trademarks or registered trademarks of Extreme Networks, Inc. in the United States and/or other countries.

All other names (including any product names) mentioned in this document are the property of their respective owners and may be trademarks or registered trademarks of their respective companies/owners.

For additional information on Extreme Networks trademarks, please see:
www.extremenetworks.com/company/legal/trademarks

## Open Source Declarations

Some software files have been licensed under certain open source or third-party licenses. End-user license agreements and open source declarations can be found at:
www.extremenetworks.com/support/policies/software-licensing



# Table of Contents

# Preface

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

## Text Conventions

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

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

**Table 1: Notes and warnings**

| Icon | Notice type | Alerts you to... |
|---|---|---|
|  | Tip | Helpful tips and notices for using the product. |
|  | Note | Useful information or instructions. |
|  | Important | Important features or instructions. |

**Table 1: Notes and warnings (continued)**

| Icon | Notice type | Alerts you to... |
|---|---|---|
|  | Caution | Risk of personal injury, system damage, or loss of data. |
|  | Warning | Risk of severe personal injury. |

**Table 2: Text**

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

**Table 3: Command syntax**

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

## Documentation and Training

Find Extreme Networks product information at the following locations:

Current Product Documentation

Release Notes

Hardware/software compatibility matrices for Campus and Edge products

Supported transceivers and cables for Data Center products

Other resources, like white papers, data sheets, and case studies

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

## Getting Help

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

Extreme Portal

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

The Hub

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

Call GTAC

For immediate support: (800) 998 2408 (toll-free in U.S. and Canada) or 1 (408) 579 2826. For the support phone number in your country, visit: www.extremenetworks.com/support/contact

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

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

### Subscribe to Service Notifications

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

1. Go to www.extremenetworks.com/support/service-notification-form.
2. Complete the form (all fields are required).

3. Select the products for which you would like to receive notifications.

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

4. Select **Submit**.

## Providing Feedback

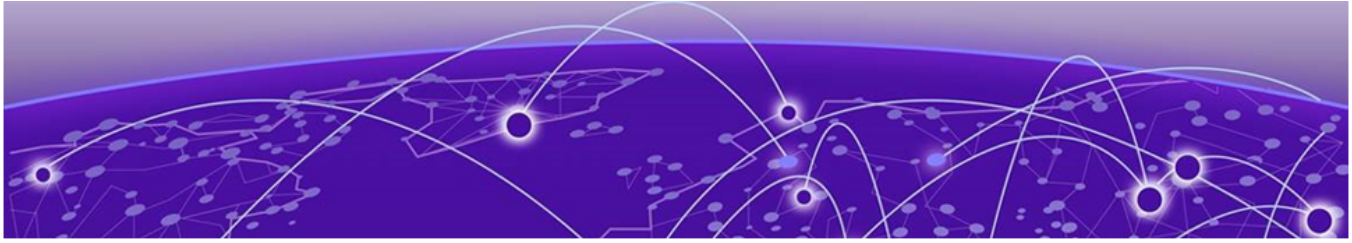
The Information Development team at Extreme Networks has made every effort to ensure the accuracy and completeness of this document. We are always striving to improve our documentation and help you work better, so we want to hear from you. We welcome all feedback, but we especially want to know about:

- Content errors, or confusing or conflicting information.
- Improvements that would help you find relevant information in the document.
- Broken links or usability issues.

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

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

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



# Extreme Fabric Automation



## Introduction to Extreme Fabric Automation

Extreme Fabric Automation (EFA) is a micro-services-based scalable automation application.

EFA orchestrates the following:

- The life cycle of
  - Small Data Center (small DC) fabrics based on Non-Clos topology
  - 3- or 5-stage IP Clos Fabric
- Tenant-aware Layer 2 and Layer 3 networks
- Integration with ecosystems that support HCI (Hyper Convergence Infrastructure) Service
  - VMware vCenter
  - OpenStack
  - Microsoft Hyper-V

The key tenets of this orchestration are as follows:

- Conformance to the EVD (Extreme Validated Design) for IP Fabrics: https://www.extremenetworks.com/resources/extreme-validated-design/extreme-ip-Fabric-architecture/
- Speed of provisioning
- Seamless installation and deployment mechanism
- High in performance, low in resource utilization, with minimal touch points
- Programmable containerized services, through an industry-standard Open API (https://www.openapis.org/)-based programmable interface
- Easy-to-use CLI commands to manage devices in an IP Fabric and in Tenant networks

EFA consists of core containerized services that interact with each other and with other infrastructure services to provide the core functions of IP Fabric automation.

| | |
|---|---|
| Asset Service | Provides the secure credential store and deep discovery of physical and logical assets of the managed devices, and publishes the asset refresh or change events to other services. |
| Fabric Service | Helps orchestrate and visualize the BGP and EVPN-based 3- and 5-stage Clos networks. |
| Tenant Service | Helps manage the Tenants, Tenant Networks, and end points, fully leveraging the knowledge of assets and the underlying fabric. |
| Inventory Service | Inventory Service is a REST layer on top of device inventory details, with the ability to filter data based on certain fields. Inventory Service securely stores device credentials in encrypted form and makes them available to different components, such as Fabric Services and the Tenant App. |

The following figure illustrates the application functionality in provisioning and discovery.

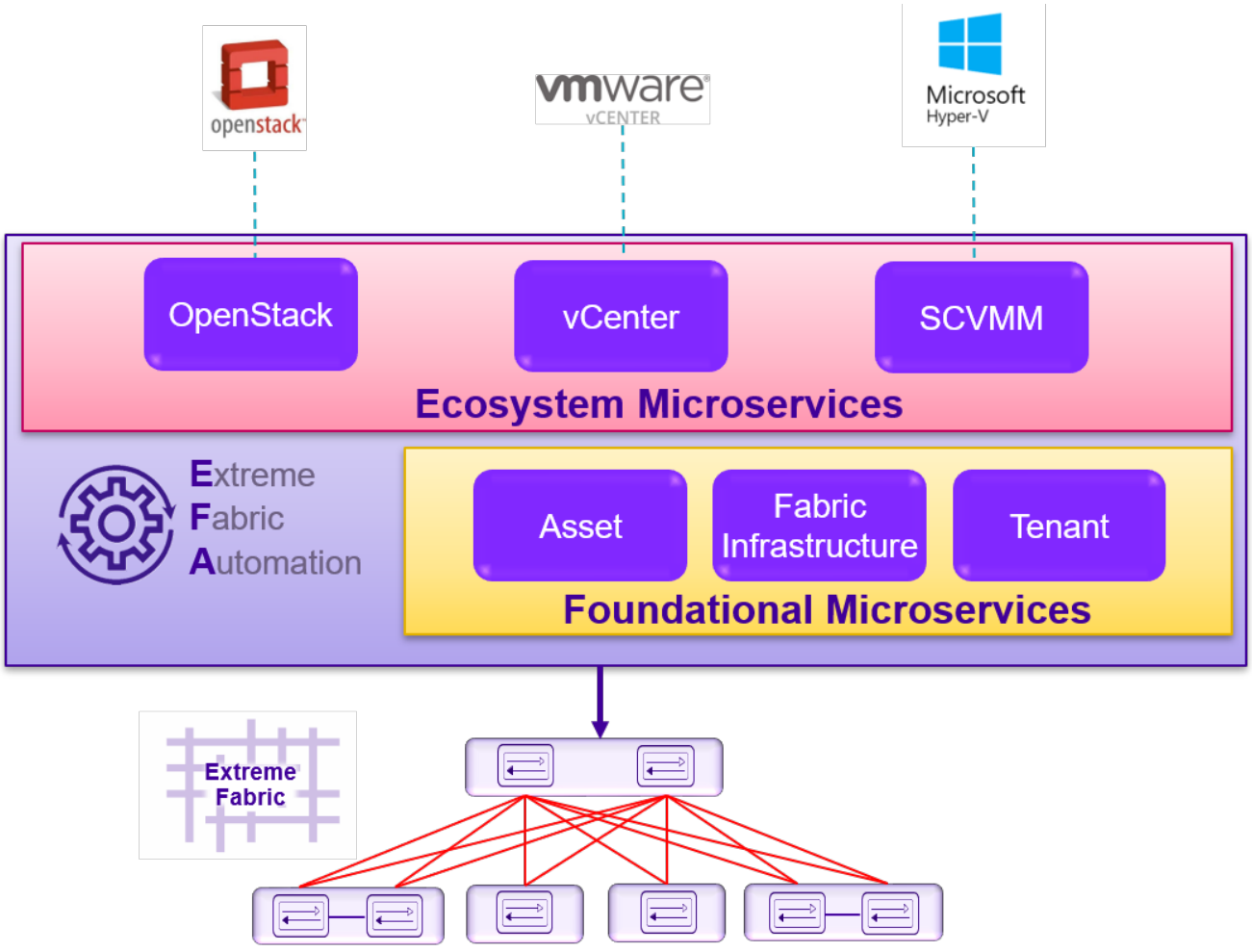



**Figure 1: Fabric Automation Microservices**

# Supported Platform Matrix

Extreme Fabric Automation is supported on the following platforms.

| Platform | SLXOS version | Role |
|---|---|---|
| SLX-9140 | 18s.1.01 /a /c, 18s.1.03 | Leaf, Small DC Fabric |
| SLX-9240 | 18s.1.01 /a /c, 18s.1.03 | Leaf, Spine, Super Spine |
| SLX-9150 | 20.1.x | Leaf, Small DC Fabric |
| SLX-9250 | 20.1.x | Leaf, Spine, Super Spine, Small DC Fabric |
| SLX-9030-48S | 18x.1.00 /a /b | Leaf |
| SLX-9030-48T | 18x.1.00 /a /b | Leaf |
| SLX-9540 | 18r.1.00aa /b /c/cc | Border Leaf |
| SLX-9540 | 20.1.x | Leaf, Border Leaf |
| SLX-9640 | 20.1.x | Border Leaf |
| SLX-9740 | 20.1.x | Spine, Super Spine, Universal Spine |
| SLX-9850 | 18r.1.00aa /b /c | Spine, Super Spine |

EFA provides seamless support for upgrade and downgrade of the devices across pre-20.1.x and 20.1.x images to keep the device and application configuration in sync.

# EFA Feature Overview

EFA features allow you to orchestrate the automation of Fabric infrastructure, the Tenant Services life-cycle management of 3- and 5-stage IP Clos DC Fabrics, and the automation of Small Data Center Fabric topology.

## Fabric Service

Fabric Service is responsible for automating the Fabric BGP underlay and EVPN overlay. By default, the EVPN overlay is enabled but can be disabled before provisioning if desired. Fabric Service exposes the CLI and REST API to clients for automating the Fabric underlay and overlay configuration.

Fabric Service features include:

- Small Data Center Topology (non-Clos support)
- Support for 3- and 5-stage Clos DC Fabrics
- Support for MCT configuration
- Support for ecosystem integration: Openstack, VMware vCenter, Microsoft Hyper-V

Underlay automation includes Interface Configurations (IP Numbered), BGP Underlay for Spine and Leaf, BFD, and MCT configurations. Overlay automation includes EVPN and Overlay Gateway configuration. Fabric Service is deployed with Inventory Service and Tenant Service.

## Tenant Service

Tenant Service exposes the CLI and REST API for automating the tenant network configuration on the Clos and Non-CLOS fabric. Tenant network configuration includes VLAN, BD, VE, EVPN, VTEP, VRF, and Router BGP configuration on the necessary Fabric devices in order to provide Layer 2-extension and Layer 3-extension across the Fabric.

Tenant Service features include:

- Streamlined Tenant provisioning
- Tenant provisioning on overlay Clos Fabric
- Tenant provisioning on overlay non-Clos Fabric

## Inventory Service

Inventory Service is a REST layer on top of device inventory details, with the capability to filter data based on certain fields. Inventory Service will also securely store credentials of devices in encrypted form and make it available to different components like Fabric Service and Tenant Service.

Inventory Service supports Device Replacement and Device Compare.

- Device replacement with the same configuration
- Device replacement with different configuration
- Device compare

Inventory Service supports the `execute-cli` option for pushing configuration commands that are not included in the EFA CLI to devices. Examples include configuring SNMP parameters or OSPF configurations. This provides the ability to use EFA for any SLX-OS command, and as well as push the same configuration to multiple devices.

## EFA Deployment on TPVM

TPVM (Third-Party Virtual Machine) is a guest VM that resides on Extreme SLX devices. You can run EFA 2.1.0 from the SLX 9150 or SLX 9250 TPVM. In this context, EFA leverages the K3S Kubernetes cluster as an underlying infrastructure for the EFA services deployment. The K3S cluster is a single instance and an important component for supporting high availability. A maximum of 24 devices is supported. These can be 24 devices in a single fabric or 24 devices across multiple fabrics.

## EFA Deployment on an External VM

EFA can be deployed on an external Virtual Machine when support for more than 24 devices is required. It may also be desirable based on the preference of where tools are deployed within the Data Center. Running EFA on the TPVM or an external VM provides added deployment flexibility.

## Fabric Service - Brownfield Support

A Brownfield deployment is one in which the installation and configuration of new software must coexist with legacy software systems. EFA supports Brownfield deployment with the following:

- Ability to migrate the devices configured through Embedded Fabric (legacy EFA) to the newer EFA application.

- Ability to migrate the fabric deployed through an older version of EFA to a newer version of EFA. For instance, if an older EFA server is dismantled, and a newer EFA version or the same EFA version is installed on a different server.
- Ability to migrate from EFA on the TPVM to EFA on an external server.

You can fully or partially migrate the fabric being configured through the use of the SLX CLI or out-of-band means, provided there are no conflicts with EFA fabric settings. Fabric Service learns and fetches the configuration on the devices through the Inventory Service, performs pre-validation checks, and generates appropriate errors for deviations in the configuration.

Brownfield scenarios are not supported for Tenant Services.

# EFA Ecosystem Integration

Administrators can use EFA to integrate with several orchestration ecosystems.

EFA provides one-touch integration points with these ecosystems, providing deep insight into VMs, vSwitches, port groups, and hosts, and the translation of these into IP Fabric networking constructs.

## VMware vCenter

- Registration of 1 or more vCenter servers in EFA
- Updates for vCenter asset details
- Lists of information about vCenter servers
- Delete or unregister vCenter servers
- Inventory integration
- Tenant Service integration Dynamic updates from vCenter and from EFA services

## OpenStack

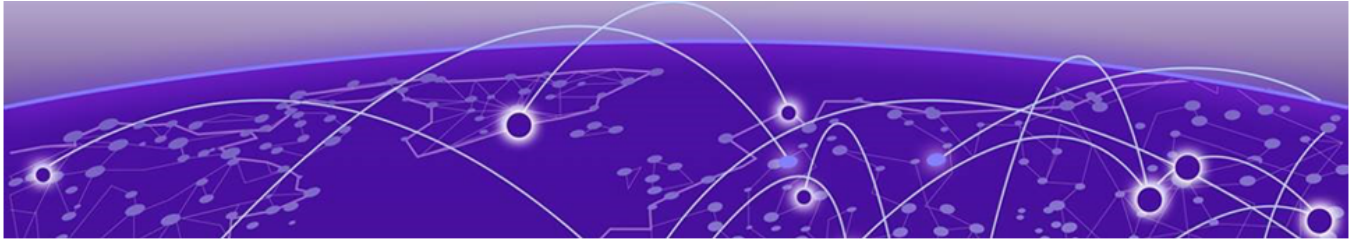
The OpenStack plugin package for ML2 and ML3 includes the following.

| | |
|---|---|
| ML2 Plugin | • CRUD operations on Network and Port<br>• LAG Support<br>• Provider Network (default, PT) |
| Trunking (VLAN) | Trunking using virtio ports |
| SRIOV-VF | Network Operations using SRIOV-VF Passthrough – Intel, Mellanox |
| SRIOV-PF | Network Operations using SRIOV-VF Passthrough – Intel, Mellanox |
| Layer 3 (E-W) | East West Traffic using virtio ports (Neutron Router, Router Interface, Subnet CRUD operations) |
| VMotion | Virtual Machine Migration |
| BD Support | Support for BD-enabled in Tenant Service |
| Multi VIM Support | Multiple Tenants managed from OSS |

| | |
|---|---|
| Multi-Segment Support | Multiple segments using SRIOV (PF/VF)+ Virtio (DHCP) |
| CEP Support | |

## Microsoft Hyper-V

- SCVMM (System Center Virtual Machine Manager) server discovery
- SCVMM server update
- Periodic polling of registered SCVMM servers
- SCVMM server list
- SCVMM server delete and deregister
- Network event handling



# Installation and Upgrade



## Install EFA on an External Linux Server

You can install EFA on an external VM.

1. Download the *.tar.gz image and untar it.

```
# tar -xfz single-node-deployment-efa-2.x.x.tar.gz
```

2. Change directory to single-node-deployment.

```
cd efa-2.x.x
```

3. Move efa-2.x.x to `/efa`.
4. Verify that the following prerequisites are met.
   - CPU: 4 cores
   - Storage: 50 GB
   - RAM: 8 GB
   - OS: Ubuntu 16.04+
5. Run the application installation script.

```
# source deployment.sh
```

## Deploy the OVA for EFA

Open Virtual Appliance (OVA) is an OVF file packaged with base image Ubuntu 16.04, and installed with EFA. OVA is also compatible with VMware ESXi servers and can be deployed with VMware products.

Use the OVA image for new installations. Take the following steps to deploy the OVA using VirtualBox.

1. Download the `EFA_v2.1.0_<build_number>.ova` file.
2. Start Oracle VirtualBox.
3. Select **File** > **Import Appliance**.
4. Select the `*.ova` file that you downloaded and then select **Open**.

5. Start the VM.

   The credentials for the OVA installation are:

   - Admin/Password: admin/password
   - Root/Password : root/dca123

6. Sign in to the VM as the admin user and then use **sudo** to run commands.

   The new admin user is added in the build.

## Upgrade or Reinstall EFA

You can upgrade or reinstall EFA.

1. Remove the existing `/efa` installation folder.
2. Follow the installation steps in Install EFA on an External Linux Server on page 17.

   The Installer includes the following packages and does not require internet connectivity.

   - Postgres database
   - Rabbitmq
   - Goinventory service
   - GoFabric service
   - GoTenant service
   - GoSwitch service
   - Konga
   - Kong-API-gateway
   - Metricbeats ELK Dashboard Stats/healthcheck monitor (available in ELK dashboard)
   - Filebeat-deployment
   - Kibana
   - Logstash
   - Elasticsearch
   - vCenter
   - Openstack
   - HyperV services

   Note

   If the previous deployment stack is running, the deployment script presents the following options:

   - Remove the current stack. Select this option to remove the entire stack.
   - Upgrade or Redeploy. If you are running the deployment script from the new tar ball, select this option to upgrade without wiping out the current database. You have this same option if the script is run from the same folder, in which case the stack is redeployed.
   - Quit. Select this option if you do not want to change the current stack.

# Rest APIs for EFA

When EFA is installed, the REST API guide is available as an HTML reference: `http://<host_ip>:docs`.

The API guide is a good reference to help integrate with other automation tools. The REST API is specified by OpenAPI and Swagger. The API guide is not available with TPVM installations.

For more information, see the selection of API guides on the Extreme Networks website: https://www.extremenetworks.com/support/documentation-api/extreme-fabric-automation-2-1-0/.

# EFA Installation and Deployment on TPVM

TPVM (Third-Party Virtual Machine) is a general server that resides on the Extreme SLX 9150 and the SLX 9250. When deploying EFA in the TPVM, no other applications can be run on that TPVM.

EFA in the context of a TPVM deployment is a micro service based fabric automation engine which leverages the K3S Kubernetes cluster as an underlying infrastructure for the EFA services deployment. The EFA application can be deployed using a single SLX command to install and upgrade on the TPVM Platform. The EFA application binary is shipped with the SLX 9150 and SLX 9250, along with the binaries for SLX-OS and the TPVM. Decoupling EFA from SLX-OS allows for upgrades to EFA without a need to upgrade SLX-OS or the TPVM. EFA can be deployed on one of the SLX devices in the fabric to manage the fabric.

The pre-packaged EFA application can be found under the folder `/efaboot` in the SLX OS. Additionally, for an incremental EFA image upgrade, an EFA Image (Tar file) can be copied to the `/efaboot` directory on SLX before the deployment.

## Requirements

TPVM must be installed and running on the SLX platform. The TPVM installation and configuration can be accomplished using the **`tpvm deploy`** command on the SLX device. The **`tpvm deploy`** command allows the user to install and configure TPVM from a single command, with the user providing the necessary input.

```
tpvm deploy mgmt admin-pwd allow-pwdless confirm-pwd [dhcp ]gw ipaddr
```

See the *Extreme SLX-OS Command Reference* for specific information about using this command.

## Install EFA on TPVM

The following procedure describes how to perform a new installation on a TPVM.

EFA on TPVM is only supported on the SLX 9150 and SLX 9250 platforms.

1. On the SLX where TPVM is planned to be run, verify that TPVM is set up for EFA deployment:
   - Validate that TPVM is running
     ```
     # show tpvm status
     ```
   - Validate that TPVM has an assigned IP address
     ```
     # show tpvm ip-address
     ```
   - Validate that the SSH keys are uploaded
     ```
     # show tpvm status
     ```
   - Validate that passwordless access is configured
     ```
     # show tpvm status
     ```
2. Enter SLX Linux mode.
   ```
   # start-shell
   # cd /efaboot
   ```
3. Copy the EFA tar file to SLX.
   ```
   # scp <efa-bundle>
   ```
4. Deploy EFA on TPVM from SLX shell.
   ```
   # efa deploy
   ```
5. Verify the status of EFA.
   ```
   # show efa status
   ```

## Upgrade EFA on TPVM

If a new version of the SLX firmware is required or if no SLX firmware is present, the EFA .tar file must be copied to the `/efaboot` directory.

1. Verify that TPVM is set up for EFA deployment.
   - Validate that TPVM is running
     ```
     # show tpvm status
     ```
   - Validate that TPVM has an assigned IP address
     ```
     # show tpvm ip-address
     ```
   - Validate that the SSH keys are uploaded
     ```
     # show tpvm status
     ```
   - Validate that passwordless access is configured
     ```
     # show tpvm status
     ```
2. Determine whether more than one EFA version is available in the SLX `/efaboot` directory.
   a. If no version is available, the installer stops
   b. If more than one version is available, you have the option to pick a version
   c. If only one version is available, the installer picks up that version
3. Determine whether the TPVM already has a version of EFA installed.
   a. If the same version is already installed, the installer stops

b. If no EFA is installed, the installer continues with installation
c. If a different version is detected, the upgrade or downgrade will continue, depending on the detected version.

4. Copy the EFA tar file to SLX (only if SLX firmware upgrade is required).

```
# start-shell
# cd /efaboot
# scp <efa-bundle
```

5. Deploy EFA on TPVM from SLX.

```
# efa deploy
```

6. Verify the status of deployed EFA.

```
# show efa status
```

## Verify EFA Status

Use the **show efa status** command to verify EFA status.

The output of the command shows the K3S installation and the health of the Fabric service, Asset service, Tenant service, Database service, and Messaging service.

Run the **show efa status** command.

```
device# show efa status
       NAME   STATUS   ROLES    AGE     VERSION
    TPVM   Ready    master   6m59s   v1.14.5-k3s.1
    admin@10.24.51.226's password:
    NAME                              READY      STATUS    RESTARTS   AGE
    pod/godb-service-wk57h            1/1         Running  0          6m11s
    pod/gofabric-service-8v8b2        1/1         Running  3          6m12s
    pod/goinventory-service-4kggf     1/1         Running  3          6m12s
    pod/gotenant-service-xcqf6        1/1         Running  3          6m12s
    pod/rabbitmq-0                    1/1         Running  0          6m12s
    pod/rabbitmq-1                    1/1         Running   0         4m51s
```

## Uninstall EFA

Use the **no efa deploy** command to uninstall EFA.

Take the following step to uninstall the current instance of EFA.

Run the **no efa deploy** command.

```
device# no efa deploy
```

# Ensuring that the System is Ready

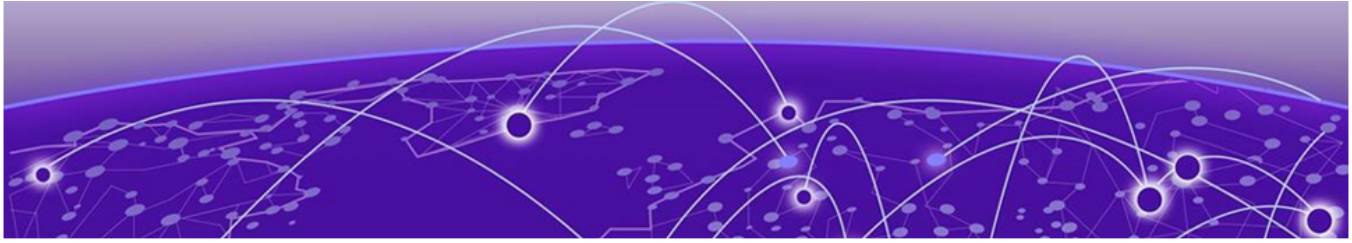
After any of the following scenarios, wait 10 minutes for EFA microservices to be operational before you run EFA commands.

- Powering on the OVA
- Rebooting the OVA
- Rebooting the TPVM
- Rebooting the SLX (which also reboots the TPVM)
- Rebooting the server on which the TAR is installed

Run the following command to verify that all PODs are in a running state.

```
# k3s kubectl get pods -n efa

NAME                                      READY   STATUS    RESTARTS   AGE
goswitch-service-958fcfb4f-qddnw          1/1     Running   4          72d
godb-service-57bd99747-f4cxb              1/1     Running   4          83d
efa-api-docs-6bb5dbcc74-br485             1/1     Running   4          72d
filebeat-service-86ddd654b6-z9zhr         1/1     Running   4          72d
goopenstack-service-554c57548f-bjwtb      1/1     Running   8          72d
logstash-service-6c49f8dd85-mngd4         1/1     Running   4          72d
rabbitmq-0                                1/1     Running   7          72d
govcenter-service-f6b49d9b9-s24wk         1/1     Running   19         72d
gohyperv-service-854654f6b9-m9mv8         1/1     Running   20         72d
goinventory-service-59d9b798d8-s9wn6      1/1     Running   20         72d
gotenant-service-55fd8889d8-g8rgb         1/1     Running   19         72d
gofabric-service-69d8995fc6-swnqw         1/1     Running   19         72d
elasticsearch-service-5cdc874b5d-f6rjh    1/1     Running   4          72d
kibana-service-7748b6db9c-lbm9w           1/1     Running   6          72d
metricbeat-service-76c4874887-mbm7h       1/1     Running   32         72d
```



# Fabric Infrastructure Provisioning



## Introduction to Fabric Service

Fabric Service is responsible for automating the Fabric BGP underlay and EVPN overlay. By default, the EVPN overlay is enabled but can be disabled before provisioning if desired. Fabric Service exposes the CLI and REST API to clients for automating the Fabric underlay and overlay configuration.

Fabric Service features include:

- Small Data Center Topology (non-Clos support)
- Support for 3- and 5-stage Clos Fabrics
- Support for MCT configuration
- Support for Eco-System Integration; Openstack, VMWare vCenter, Microsoft Hyper-V/SCVMM

Underlay automation includes Interface Configurations (IP Numbered), BGP Underlay for Spine and Leaf, BFD, and MCT configurations. Overlay automation includes EVPN and Overlay Gateway configuration. Fabric Service is deployed with Inventory Service and Tenant Service.

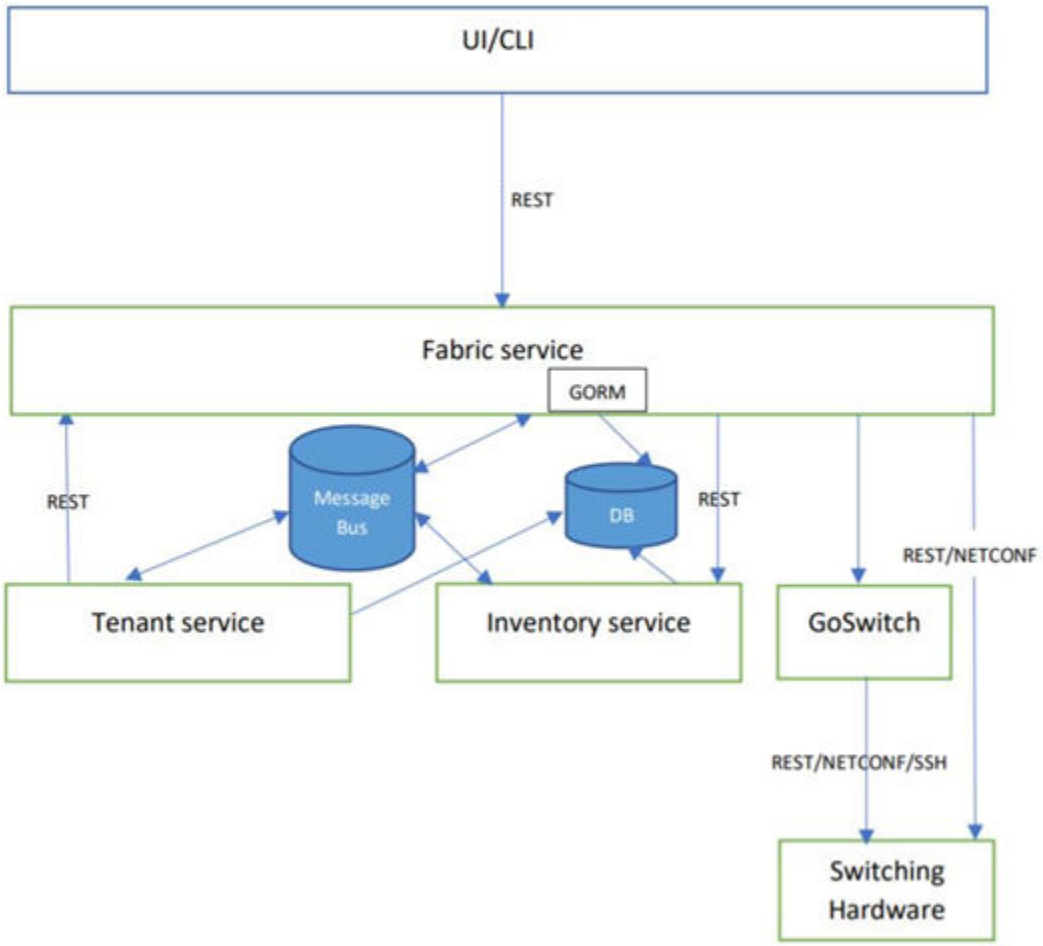



**Figure 2: Fabric Service Architecture**

The following platforms are supported.

| Platform | SLXOS version | Role |
|---|---|---|
| SLX-9140 | 18s.1.01 /a /c, 18s.1.03 | Leaf /Small DC Fabric |
| SLX-9240 | 18s.1.01 /a /c, 18s.1.03 | Leaf/Spine/Super Spine |
| SLX-9150 | 20.1.x | Leaf/Small DC Fabric |
| SLX-9250 | 20.1.x | Leaf/Spine/Super Spine/Small DC Fabric |
| SLX-9030-48S | 18x.1.00 /a /b | Leaf |
| SLX-9030-48T | 18x.1.00 /a /b | Leaf |
| SLX-9540 | 18r.1.00aa /b /c/cc, 20.1.x | Leaf/Border leaf |
| SLX-9640 | 20.1.x | Border Leaf |
| SLX-9850 | 18r.1.00aa /b /c | Spine/Super Spine |

# IP Fabric and Clos Orchestration Overview

A Fabric is a logical container for holding a group of devices. Here it denotes a collection of switches that are connected in a fabric topology and on which one can configure underlay and overlay.

Fabric service provides following features:

- 3-stage Clos automation
- 5-stage Clos automation
- Small Data Center automation
- Multi-Fabric automation
- Fabric topology view
- Fabric validation, error reporting, and recovery
- Single-homed leaf or multi-homed (MCT) leaf

Fabric CLIs/REST APIs provide the following:

- Mechanism to create a Fabric composed of multiple DC points of delivery (PoDs).
- Mechanism to configure Fabric settings. Fabric settings are collections of settings that control the various parameters of the Fabric being managed, for example, L2/L3 MTU, BGP maximum paths.
- Mechanism to fetch per-device errors occurring during Fabric configuration, for which the user can take corrective or remedial actions.

Errors occurring on the device during Fabric creation are tagged against the devices and can be retrieved from the CLI/REST APIs for use in taking corrective or remedial actions.

## SLX Device Prerequisites for Fabric Service

The following items are required before configuring Fabric Automation.

- Management IP addresses must be configured on all switches.
- SLX devices must have the appropriate firmware version. Refer to the Supported Platform Matrix.
- SLX 9850: Fabric links must be enabled manually , through `no shut`.
- SLX 9540: The appropriate TCAM profile must be set and the switch rebooted.

```
device# conf
Entering configuration mode terminal
device(config)# hardware
device(config-hardware)# profile tcam vxlan-ext
%Warning: To activate the new profile config, please run 'copy running-config startup-
config' followed by 'reload system'.
device(config-hardware)#
```

- Any breakout ports on SLX devices must be configured manually.
- Refer to the release specific SLX-OS Management Configuration Guide for configuration steps for each platform.

For information about Small Data Center topology (non-Clos support), see Introduction to Fabric Service on page 23.

## Automating Clos Fabric Provisioning

There are four steps to automating Clos Fabric Provisioning, whether it is a three-stage, five-stage, or Non-Clos Small Data Center topology: Create the fabric, register the devices, validate and add the devices, and provision the configuration on the devices.

1. Create the Fabric. Modify the default Fabric settings if necessary.
2. Register devices in the inventory with discovered Assets as follows:
   - Device details such as model, firmware, ASNs
   - Physical Interfaces, VEs
   - Logical Interfaces such as VLANs, BDs, port-channels
   - VRFs, BGP, MCT, EVPN, overlay
   - Stored device credentials, eliminating the need to specify device credentials in other provisioning commands
3. Add devices to the Fabric.
   - Clos physical topology (physical connections as per EVD) between devices is validated.
   - Based on device roles and topology, an intended configuration is generated.
4. Configure the Fabric and provision the underlay and overlay configuration on the devices.
   - Push the Laayer 2 and Layer 3 configurations necessary to form an IP Fabric down to the SLX devices.
   - Validate the configuration pushed to the devices.

# Fabric Services Overlay Overview

Extreme Fabric Automation (EFA) is a microservices-based application that manages the life cycle of IP Fabric CLOS and Small Data Center deployments. All of the microservices support REST APIs that are detailed by OpenAPI.

EFA offers unique flexibility in supporting multiple IP Fabric topologies based on a BGP underlay with a BGP/EVPN overlay:

- Small Data Center Fabric (non-Clos topology from a single switch pair up to four switch pairs)
- 3-stage Clos (Leaf / Spine)
- 5-stage Clos (Leaf / Spine / Super Spine

Tenant Network onboarding services are supported on all the topologies, allowing you to create connectivity for devices connected to the fabric, such as compute (servers), storage, and connectivity to external routers or gateways.

## Fabric Service

Fabric Service is responsible for automating the N-Stage CLOS underlay and overlay. By default, overlay is enabled and can be disabled. Fabric Service exposes the CLI and REST API to clients for automating the N-Stage CLOS overlay configuration.

Overlay configuration automation includes EVPN Configuration and Overlay Gateway Configuration.

Fabric Service is deployed with Inventory Service and helps orchestrate and visualize the BGP and EVPN-based 3- and 5-stage CLOS networks.

## Inventory Service

Inventory Service is a REST layer on top of device inventory details, with the ability to filter data based on certain fields. Inventory Service securely stores device credentials in encrypted form and makes them available to different components, such as Fabric and Tenant Services.

Inventory service supports the device replacement and compare feature.

- Device replacement with the same configuration
- Device replacement with different configuration
- Device compare

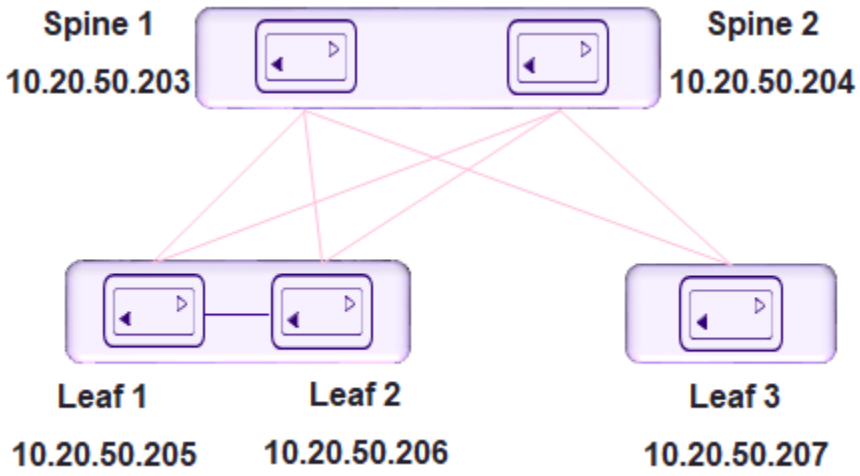
Inventory Service supports the **execute-cli** option for pushing specific configuration to devices. You can use **execute-cli** to push the same configuration on multiple devices.

## Configure 3-Stage Clos Automation

The 3-stage topology has two layers of devices: Leaf and Spine. All links between Leaf and Spine must be connected. Spine nodes should not be interconnected.

There are four steps to configuring a three-stage topology: create the fabric, register the devices, validate and add the devices, and provision the configuration on the devices.

The following is an example of a three-stage fabric.




1. Create the Fabric.
2. Add a device or devices to the Fabric.
3. Validate the Fabric Topology.

   During the addition of a device to a Fabric and during Fabric configuration, Clos topology validations are performed. If the validation contains errors, the errors are reported to the user. Any Fabric topology errors can be exported to a CSV or DOT file. The following topology validations are performed:

   - Leaf nodes must connect to all the Spine nodes.
   - A Spine node must connect to all the leaf nodes.

- A Border Leaf node connects to all the Spine nodes.
- A Spine node connects to all the Border Leaf .
- No more than two Leaf nodes connect to each other.
- No more than two Border Leaf nodes connect to each other.
- Border Leaf node and L node are not connected.
- Spine nodes are not connected to each other.
- Super-spine nodes are not connected to each other.
- If a Leaf node is marked as "multi-homed", then the node must have an MCT neighbor.
- If a Leaf node is marked as "single-homed", then the node is not connected to other Leaf nodes.
- If a Border Leaf node is marked as "multi-homed", then the node must have an MCT neighbor.
- If a Border Leaf node is marked as "single-homed", then the node is not connected to other Border Leaf nodes.
- Device role (such as Leaf, Border-leaf, Spine, or Super-spine) is validated for a given device (SLX 9840 cannot be added as a leaf).

4. Configure the devices on the Fabric.

## Configure 5-Stage Clos Automation

The five-stage topology has three layers of devices: Leaf, Spine, and Super Spine. All links between Leaf and Spine must be connected. Spine nodes should not be interconnected. Similarly, all the links between the Spine and Super-spine must be connected. A border Leaf can be directly connected to a Super-spine, but there should not be any connection between a border Leaf and a Spine.

There are four steps to configuring a five-stage topology: create the fabric, register the devices, validate and add the devices, and provision the configuration on the devices.

The following image is an example of a five stage fabric. The following steps describe how to configure a five-stage fabric topology.

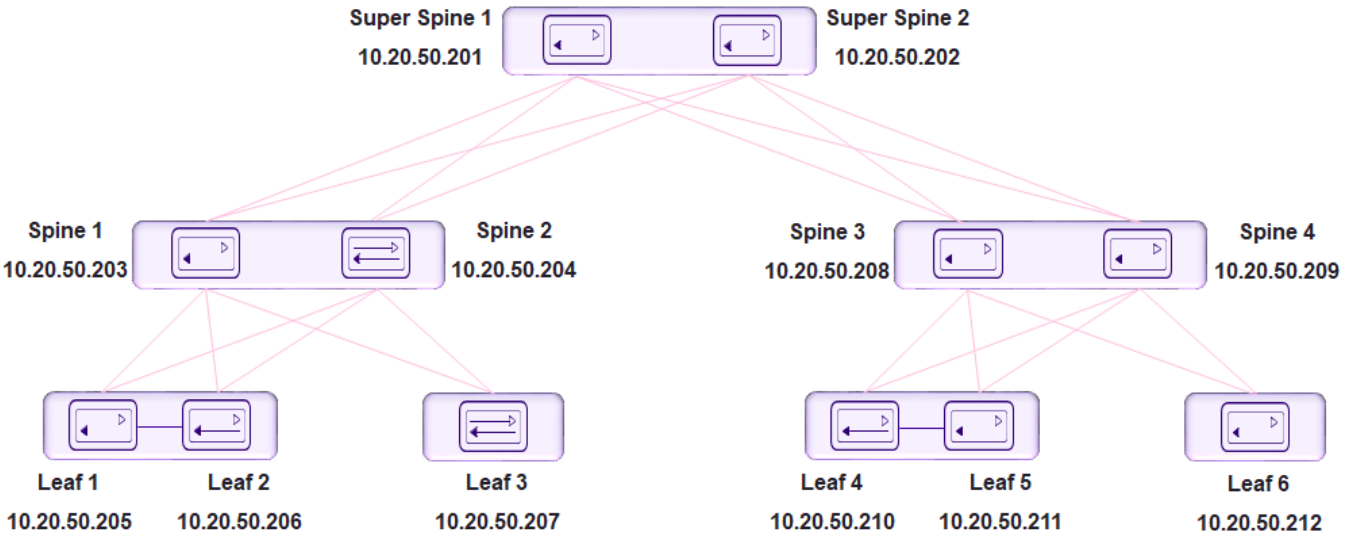



**Figure 3: Five-Stage Clos Automation**

Note

5-stage Clos can be built top to bottom or bottom to top. The following example builds from the top down.

1. Create the Fabric using the `efa fabric create` command.
2. Add a device or devices to the 5-stage Fabric using the `efa fabric device add`. The user must provide device credentials as part of this command if the devices are not already registered with the inventory.

   A device must be registered with Inventory Service before being added to a Fabric. Fabric Service supports IP numbered configuration. Each interface on a link between Leaf and Spine is assigned an IP address. eBGP peering uses these IP addresses.

   Multiple devices can be added to an existing fabric using the `efa fabric device add-bulk` command. If a username and password are provided, then the devices will be auto registered with the inventory service.
3. Use the `efa fabric configure` command to validate and configure the Fabric Topology.

   During the addition of a device to a Fabric and during Fabric configuration, Clos topology validations are performed. If the validation contains errors, the errors are reported to the user. Any Fabric topology errors can be exported to a CSV or DOT file.

   The following topology validations are performed:

   - Leaf nodes must connect to all the Spine nodes.
   - A Spine node must connect to all the Leaf nodes.
   - A Border Leaf node connects to all the Spine nodes.
   - A Spine node connects to all the Border Leaf nodes
   - No more than two Leaf nodes connect to each other.
   - No more than two Border Leaf nodes connect to each other.
   - Border Leaf node and Leaf node are not connected.
   - Spine nodes are not connected to each other.
   - Super-spine nodes are not connected to each other.

- If a Leaf node is marked as "multi-homed", then the node must have an MCT neighbor.
- If a Leaf node is marked as "single-homed", then the node is not connected to other Leaf nodes.
- If a Border Leaf node is marked as "multi-homed", then the node must have an MCT neighbor.
- If a Border Leaf node is marked as "single-homed", then the node is not connected to other Border Leaf nodes.
- Device role (such as Leaf, Border-leaf, Spine, and Super-spine) is validated for a given device (SLX 9840 cannot be added as a leaf).

## Fabric Topology View

Fabric topology view displays the physical Clos topology and the overlay VxLAN tunnels established between the Leaf nodes of the Fabric.

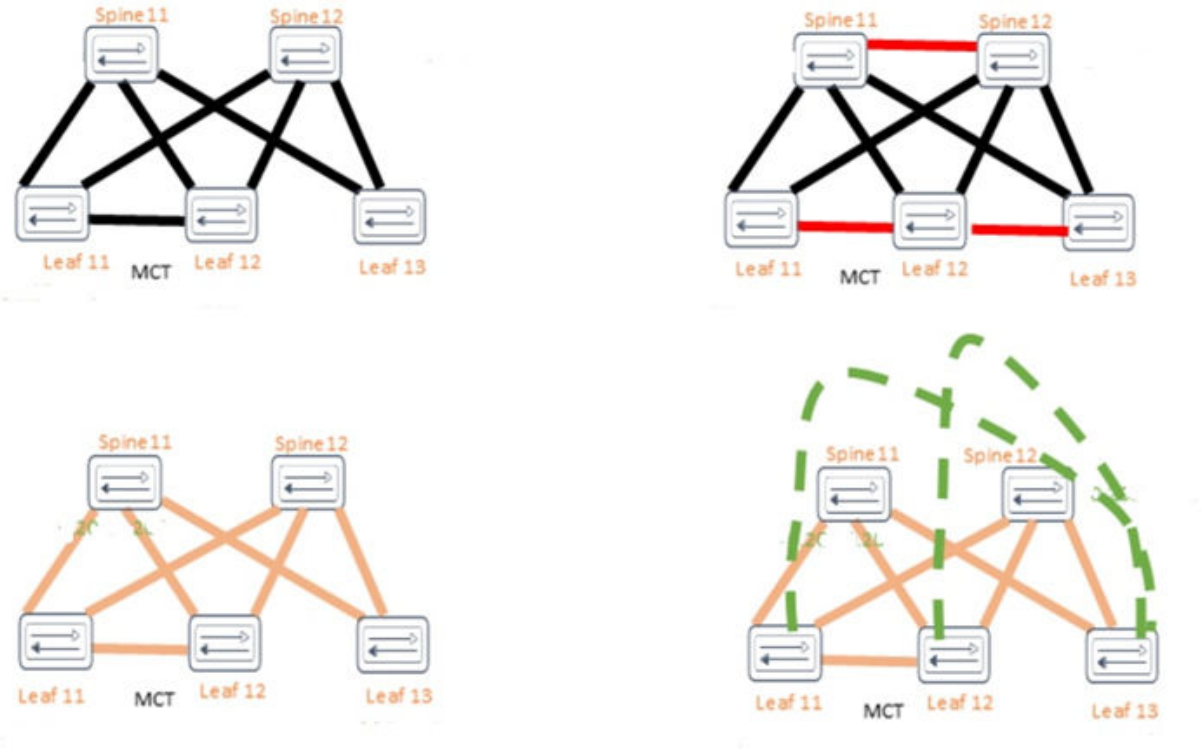



**Figure 4: 3-stage Clos Topology View**

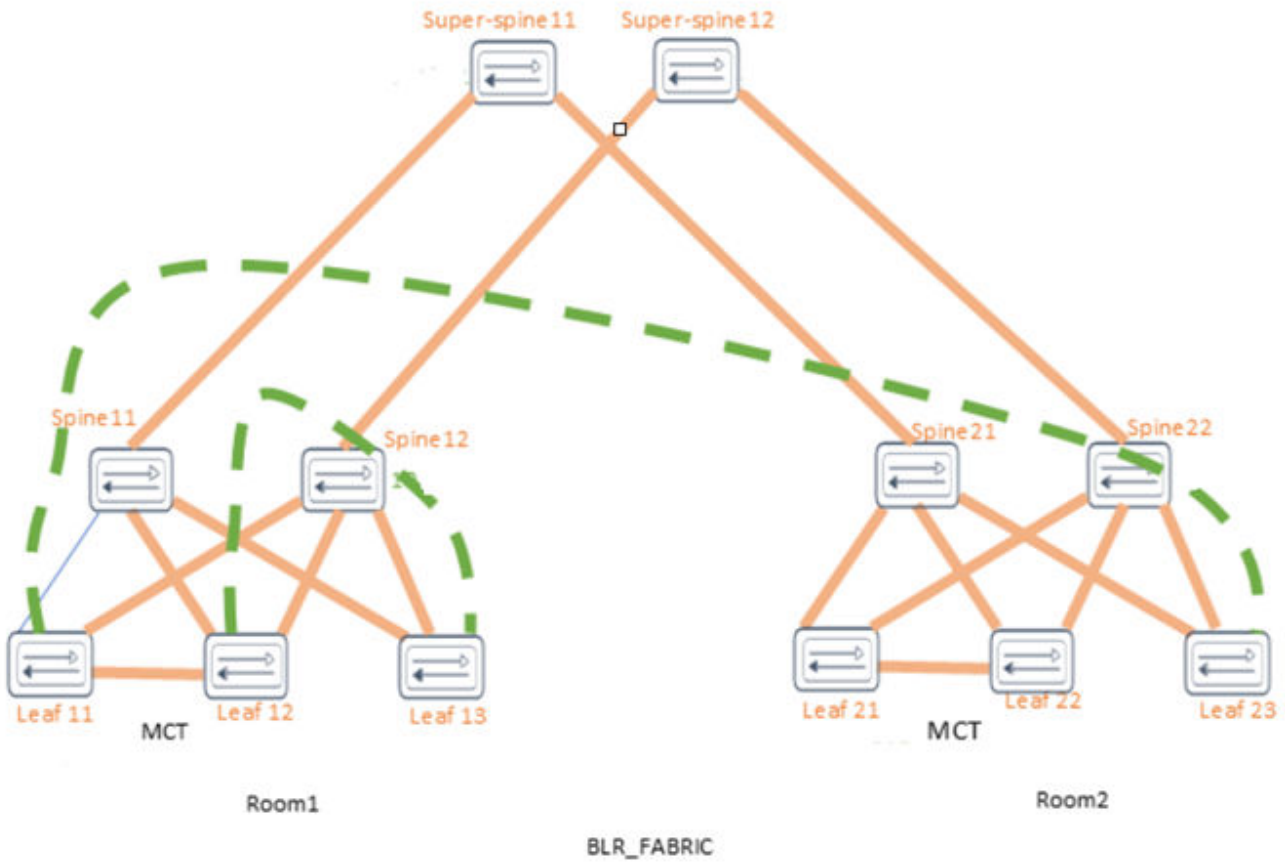



**Figure 5: 5-stage Clos Topology View**


**Figure 6: 3- stage and 5-stage Clos Topology View Key**

## Non-Clos Small Data Center Overview

New for this release, support for Small DC Fabric offers CLI commands and a REST API that is similar to that of Clos Fabric.

Non-Clos topology uses interconnected racks. Each rack consists of two devices connected via eth 0/46, eth 0/47, and eth 0/48. Non-Clos topology is used for small scale, VCS-like deployments. The maximum scale is four racks of two devices each, totalling eight devices.

Non-Clos fabric is supported on SLX 9150, SLX 9140, and SLX 9250 devices as follows:

- Single rack automation. Each rack consists of two node MCT pair.
- Multi-rack automation
- Multi-homed leaf (MCT)
- BGP neighborship
- Fabric topology view
- Fabric validation and troubleshooting

## Supported Non-Clos Topologies

Several small data center, non-Clos topologies are supported.

Each rack consists of 2 devices connected to each other via eth 0/46, eth 0/47, and eth 0/48. Devices in a rack form MCT (multi-chassis trunking).

**Table 4: Items in a rack**

| Items | Description |
|---|---|
| Intra-rack links | eth 0/46 and eth 0/47 are used for MCT and eth 0/48 is used for Layer 3 backup link |
| Inter-rack links | Links that interconnect the racks, also called fabric links |
| IBGP IPv4 neighbor | Established on the Layer 3 backup link |
| IBPG Layer 2 VPN EVPN neighbor | Established between the MCT nodes |
| EBGP IPv4 neighbors | Established on the Fabric links |
| EBGP Layer 2 VPN EVPN neighbors | Established from every node of every rack to every other remote rack node. The remote neighbor IP address is the remote device router ID. |

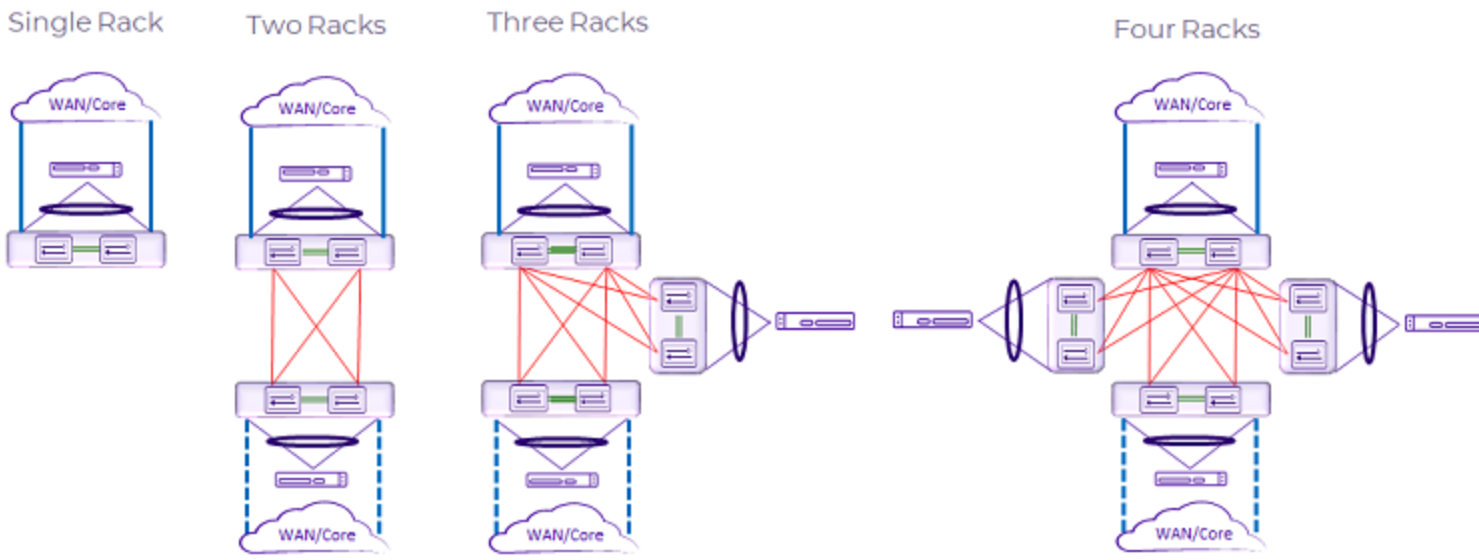



**Figure 7: Supported small data center topologies**

Small data centers are supported on SLX 9140, SLX 9150, and SLX 9250.

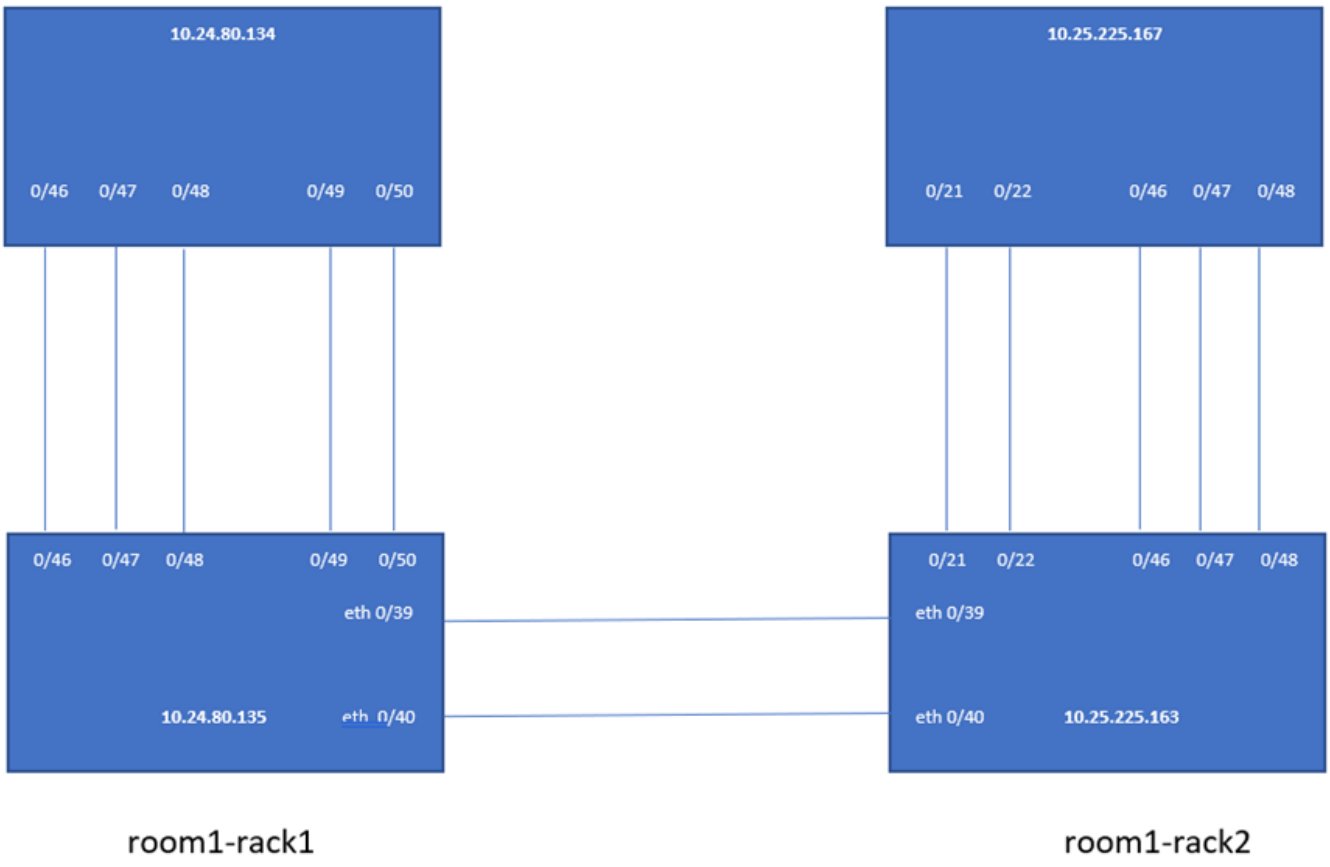



**Figure 8: Non-Clos multi-rack config automation**

## Configuration and Topology Validations

During the addition of a device to a fabric and during fabric configuration, Non-CLOS configuration and topology validations are performed. If the validation errors out, the same is reported for taking the appropriate corrective action. The encountered fabric errors can be exported to a .csv or .dot file. Below are the topology validations:

- Rack contains two devices.
- Rack contains SLX 9150, SLX 9140, SLX 9250 devices only.
- Maximum number of racks supported per fabric is four.
- Each rack is connected to at least another rack.
- Devices within a rack must be connected via 0/46, 0/47, and 0/48.

## Configuring Non-Clos Small Data Center Automation

Non-Clos topology is used for small scale, VCS-like deployments, and consists of four interconnected racks containing two devices each, for a maximum scale of eight devices. The devices are connected uisng eth 0/46, eth 0/47 and eth 0/48.

There are four steps to configuring a Non-Clos Small Data Center topology; Create the fabric, Register the devices, Validate and Add the devices, and Provision the configuration on the devices. The following steps describe how to configure a small data center topology.

1. Create the Fabric.

```
$ efa fabric create --name extr-fabric  --type non-clos
```

2. Add a device or devices to the Fabric. The user must provide device credentials as part of this command if the devices are not already registered with the inventory.

```
$ efa fabric device add --name extr-fabric --ip 10.24.80.134 --rack room1-rack1 --
username admin --password password
$ efa fabric device add --name extr-fabric --ip 10.24.80.135 --rack room1-rack1 --
username admin --password password

$ efa fabric device add --name extr-fabric --ip 10.25.225.163 --rack room1-rack2 --
username admin --password password
$ efa fabric device add --name extr-fabric --ip 10.25.225.167 --rack room1-rack2 --
username admin --password password
```

   A device must be registered with Inventory Service before being added to a Fabric. Fabric Service supports IP numbered configuration. Each interface on a link between leaf and spine is assigned an IP address. eBGP peering use these IP addresses.

   Multiple devices can be added to an existing fabric using the `efa fabric device add-bulk` command. If a username and password are provided, then the devices will be auto registered with the inventory service.

```
$ efa fabric device add-bulk --name extr-fabric --rack room1-rack1 --ip
10.24.80.134,10.24.80.135 --rack room1-rack2 --ip 10.25.225.163,10.25.225.167
```

3. Use the `efa fabric configure` command to validate and configure the Fabric Topology.

```
$ efa fabric configure --name extr-fabric
```

   During each of the steps; `create, add,` and `configure`, Clos topology validations are performed. If the validation contains errors, the errors are reported to the user. The encountered Fabric topology errors can be exported to a CSV/DOT file.

In Small Data Center Automation, the following topology validations are performed:

- A rack may contain up to 2 devices.
- A rack contains only SLX9140 devices.
- Maximum number of racks supported per fabric is 4.
- Each rack should be connected to at least one other rack.
- Devices within a rack must be connected via 0/46, 0/47, and 0/48.

If the addition of devices to a Fabric is successful, then the configuration is pushed to all the devices of the Fabric using the `efa fabric configure` command above.

Use the following commands to display detailed information about the Small Data Center topology and configuration.

- `efa fabric export`: Export fabric details to a .CSV file.
- `efa fabric show`: Display the details of the fabric.
- `efa fabric topology show physical`: Displays physical connectivity of the devices in a fabric.
- `efa fabric topology show underlay`: Displays the underlay connectivity - the BGP neighborship and the state of BGP sessions of the devices in a fabric.
- `efa fabric topology show overlay`: Displays the overlay connectivity of the devices in a fabric.

## Fabric Automation Configuration Commands

This section contains commands used to configure automation for 3-stage, 5-stage, and Non-Clos Small Data Center topologies. It is intended to provide additional details for the commands used in the configuration procedures.

## efa fabric clone

Cloning can expedite the deployment of fabrics across different sites / data centers. For fabrics in two different data centers to look exactly the same for disaster recovery purposes, create a clone for the source fabric.

*Syntax*

**efa fabric clone** [ **--source** *source-fabric-name* **--destination** *destination-fabric-name*]

*Parameters*

**--source**

Name of the fabric to be cloned.

**--destination**

New name of the cloned fabric.

*Usage Guidelines*

This command clones all the fabric properties - type, stage, description, fabric settings - but not the devices on the fabric.

*Examples*

The following example clones BLR_FABRIC into PUN_FABRIC.

```
efa fabric clone --source BLR_FABRIC --destination PUN_FABRIC
```

## efa fabric configure

Configures the underlay and overlay on all fabric devices.

*Syntax*

**efa fabric configure** [ **--name** *<fabric-name>* | **--force** ]

*Command Default*

If the --force option is used, all the devices will be removed and added back to the fabric. This can result in config remove and add on all the devices.

If the addition of devices to a Fabric is successful, the underlay and overlay is configured on all the devices of the Fabric using the `efa fabric configure` command.

*Parameters*

**--name**

Name of the fabric

**--force**

Force the configuration on the devices

*Examples*

```
efa fabric configure --name extr-fabric

efa fabric configure --name extr-fabric --force
```

## efa fabric create

Creates a fabric.

*Syntax*

**efa fabric create** { **--name** *<fabric-name>* | **--type** *< clos | non-clos >* |**--stage** *< 3 | 5 >* |**--description** *<description>* }

*Parameters*

**--name**

Name of the fabric.

**--description**

Description of the product.

**--type** *< clos | non-clos >*

Type of the fabric (default: clos).

**--stage**

Stage of the fabric [3 | 5] (default: 3).

*Examples*

```
efa fabric create --name extr-fabric  --type non-clos
```

## efa fabric delete

Deletes the fabric from inventory.

*Syntax*

**efa fabric delete** [ **--name** *fabric-name* ] [**--force** ]

*Parameters*

**--name**

Name of the fabric to be deleted.

**--force**

Forces the deletion of Fabric even if the Fabric has devices.

*Usage Guidelines*

Deletion of a fabric is not allowed if the fabric has one or more devices. You must delete all the devices from the fabric prior to deleting the fabric.

Forced deletion of a fabric removes the devices from fabric but not from inventory.

*Examples*

The following example deletes the fabric BLR_FABRIC.

```
efa fabric delete --name BLR_FABRIC
```

## efa fabric device add

Adds a device to an existing fabric.

*Syntax*

**efa fabric device add** { **--name** *fabric-name* **--ip** *device-ip* **--role** [ **leaf** | **spine** | **super-spine** | **border-leaf**] } [ **--leaf-type** [ **single-homed** | **multi-homed** ] **--hostname** *<hostname>* **--asn** *<local-asn>* **--vtep-loopback** *<id>* **--loopback** *<id>* **--pod** *<pod-name>* **--username** *<username>* **--password** *<password>* ]

*Command Default*

A device must be registered with Inventory Service before being added to a Fabric. Fabric Service supports IP numbered configuration. Each interface on a link between leaf and spine is assigned an IP address. eBGP peering use these IP addresses.

Device credentials must be provided as part of this command if the devices are not already registered with the inventory.

If user provides “username” and “password”, then the device will be auto registered with the inventory service.

If user doesn’t provide “username” and “password”, then user would need to explicitly register the device with the inventory service.

*Parameters*

**--name**

Name of the fabric

**--ip**

Device IP

**--role**

Device Role (leaf | spine | super-spine | border-leaf)

**--leaf-type**

Leaf Type (single-homed | multi-homed)

**--hostname**

Host Name

**--asn**

ASN

**--vtep-loopback**

VTEP Loopback

**--loopback**

Loopback Port Number

**--pod**

Name of the pod

**--rack**

Name of the rack

**--username**

Username for the device

**--password**

password for the device

*Examples*

```
efa fabric device add --name extr-fabric --ip 10.24.80.134,10.24.80.135  --rack room1-
rack1 --username admin --password password
```

## efa fabric device add-bulk

Adds multiple devices to an existing fabric.

*Syntax*

**efa fabric device add-bulk** { **--name** *<fabric-name>* | **--rack** *<rack name>* | **--ip** *<pair --of-ips>* | }[**--username** *<username>* | **--password** *<password>* ]

*Command Default*

If “username” and “password” are provided, the devices will be auto registered with the inventory service.

If “username” and “password” are not provided, the devices must be registered with the inventory service.

A single “three-stage-pod” and “five-stage-pod” can be provided per CLI execution.

*Parameters*

**--name**

Name of the fabric

**--leaf**

Comma separated list of leaf IP Address/Host names

**--border-leaf**

Comma separated list of borderLeaf IP Address/Host names

**--three-stage-pod**

Name of the leaf/spine pod

**--five-stage-pod**

Name of the super-spine pod

**--spine**

Comma separated list of spine IP Address/Host names

**--super-spine**

Comma separated list of super spine IP Address/Host names

**--username**

Username for the list of devices

**--password**

Password for the list of devices

*Examples*

```
efa fabric device add-bulk --name BLR_FABRIC --leaf 10.24.48.131,10.24.51.135 --border-
leaf 10.24.51.131,10.25.225.58
  --spine 10.24.80.139  --username admin --password password
```

## efa fabric device remove

Removes existing device from a fabric.

*Syntax*

**efa fabric device remove** { **--name** *<fabric-name>* | **--ip** *<list-of-device-ips>* }| [**--no-device-cleanup**]

*Command Default*

If the "--no-device-cleanup" option is used, the configuration pushed by the automation engine will not be cleaned up from the fabric devices. Removal of a device from fabric doesn't delete the device from inventory. The device from inventory must be deleted.

*Parameters*

**--name**

Name of the fabric

**ip**

Device IP

**--no-device-cleanup**

Do not clean up the configuration on the devices

*Examples*

```
efa fabric device remove --ip
10.24.48.131,10.24.51.135,10.25.225.58,10.24.51.131,10.24.80.139
 --name BLR_FABRIC --no-device-cleanup
```

## efa fabric settings update

Updates the fabric settings to overwrite the “default” fabric settings.

*Syntax*

**efa fabric settings update** [ **--attribute-type** *<attribute-value>* ]

*Parameters*

**--mtu**

The MTU size in bytes <Number:1548-9216>. The default value is 9216.

**--ip-mtu**

IPV4/IPV6 MTU size in bytes <Number:1300-9194>. The default value is 9100.

**--bfd-enable**

BFD enabled Yes/No. The default value is NO.

**--bfd-tx**

BFD desired min transmit interval in milliseconds <NUMBER: 50-30000>. The default value is 300.

**--bfd-rx**

BFD desired min receive interval in milliseconds <NUMBER: 50-30000>. The default value is 300.

**--bfd-multiplier**

BFD detection time multiplier <NUMBER: 3-50>. The default value is 3.

**--bgp-multihop**

Allow EBGP neighbors not on directly connected networks <Number:1-255>. The default value is 2.

**--max-paths**

Forward packets over multiple paths<Number:1-64>. The default value is 8.

**--allow-as-in**

Disables the AS_PATH check of the routes learned from the AS<Number:1-10>. The default value is 1.

**--p2p-link-range**

IP Address Pool to be used for Leaf to Spine links. The default value is 10.10.10.0/23.

**--loopback-ip-range**

IP Address Pool for Loopback interface, to be used for unnumbered and VTEP IP. The default value is 172.31.254.0/24.

**--rack-l3-backup-ip-range**

IP Address Pool for L3 Back up. The default value is 10.30.30.0/24.

**--loopback-port-number**

Loopback ID on the device to be used as donor IP interface for the link between Leaf and Spine. The default value is 1.

**--vtep-loopback-port-number**

Loopback ID on the device to be used as VTEP IP interface. The default value is 2.

**--leaf-asn-block**

ASN pool for Leaf nodes. The default value is 65000-65534.

**--spine-asn-block**

ASN pool for Spine nodes. The default value is 64512-64768.

**super-spine-asn-block**

ASN pool for Super-spine nodes. The default value is 64769.

**--rack-asn-block**

ASN Pool For Rack Nodes. The default value is 4200000000-4200065534.

**--leaf-peer-group**

Leaf Peer Group Name <WORD: 1-63>. The default value is leaf-group.

**--spine-peer-group**

Spine Peer Group Name <WORD: 1-63>. The default value is spine-group.

**--super-spine-peer-group**

Super-spine Peer Group Name <WORD: 1-63. The default value is spine-group.

**--rack-underlay-ebgp-peer-group**

Rack Underlay EBGP Peer Group Name <WORD: 1-63>. The default value is underlay-ebgp-group.

**--rack-overlay-ebgp-peer-group**

Rack Overlay EBGP Peer Group Name <WORD: 1-63>. The default value is overlay-ebgp-group.

**--anycast-mac-address**

IPV4 ANY CAST MAC address.mac address HHHH.HHHH.HHHH. The default value is 0201.0101.0101.

**--ipv6-anycast-mac-address**

IPV6 ANY CAST MAC address.mac address HHHH.HHHH.HHHH. The default value is 0201.0101.0102.

**--mac-aging-timeout**

MAC Aging Timeout <NUMBER: 0|60-100000>. The default value is 1800.

**--mac-aging-conversation-timeout**

MAC Conversational Aging time in seconds<NUMBER: 0|60-100000>. The default value is 300.

**--mac-move-limit**

MAC move detect limit <NUMBER: 5-500>. The default value is 20.

**--duplicate-mac-timer**

Duplicate Mac Timer. The default value is 5.

**--duplicate-mac-timer-max-count**

Duplicate Mac Timer Max Count. The default value is 3.

**--mctlink-ip-range**

IP Address Pool to be used for MCT peering. The default value is 10.20.20.0/24.

**--mct-port-channel**

Portchannel interface ID <NUMBER: 1-64> to be used as MCT peer-interface. The default value is 64.

**--rack-l3-backup-port**

Rack L3 Backup port <STRING: default '0/48'>to be used as rack 13 backup port. The default value is 48.

**--rack-mct-ports**

Rack MCT Port <STRING: default '0/46,0/47'>to be used as rack MCT port.

**--rack-ld-l3-backup-port**

Rack Low Density L3 Backup port (not applicable to SLX-9250) <STRING: default '0/32'>to be used as rack low density L3 backup port.

**--rack-ld-mct-ports**

Rack Low Density MCT Ports <STRING: default '0/30,0/31'>to be used as rack low density MCT port.

Rack L3 Backup port <STRING: default '0/48'>to be used as rack 13 backup port. The default value is 48.

**--control-vlan** *VLAN ID <NUMBER: 1-4090>*

VLAN ID to be used as MCT cluster control VLAN. The default value is 4090.

**--control-ve** *VE ID <NUMBER: 1-4090>*

VE ID to be used as MCT cluster control VE. The default value is 4090.

**--configure-overlay-gateway**

ConfigureOverlayGateway Enabled Yes/No. The default value is Yes.

**--vni-auto-map**

VTEP VLAN/BD to VNI Mode Auto (Yes/No). The default value is Yes.

## efa fabric show summary

Displays the summary of the fabric.

*Syntax*

**efa fabric show summary** [**--name** *<fabric-name>* ]

*Command Default*

Displays the summary of all the fabrics when the “--name” option is not provided.

Displays the summary of a given fabric when the “--name” option is provided.

*Parameters*

**--name** *<fabric-name>*

Name of the fabric

*Examples*

```
efa fabric show summary --name BLR_FABRIC
```

## efa fabric show

Displays the details of the fabric.

*Syntax*

**efa fabric show** [**--name** *fabric name*]

*Command Default*

Displays the details of all the fabrics when the “--name” option is not provided.

Displays the details of a given fabric when the “--name” option is provided.

*Parameters*

**--name** *<fabric-name>*

Name of the fabric

*Examples*

```
efa fabric show --name BLR_FABRIC
```

## efa fabric topology show physical

Displays physical connectivity of the devices in a fabric.

*Syntax*

**efa fabric topology show physical** [**--name** *<fabric name>*]

*Parameters*

**--name**

Name of the fabric

*Examples*

```
efa fabric topology show physical --name extr-fabric
```

## efa fabric topology show underlay

Displays the underlay connectivity of the devices in a fabric.

*Syntax*

**efa fabric topology show underlay** [**--name** *<fabric name>*]

*Parameters*

**--name**

Name of the fabric

*Examples*

```
efa fabric topology show underlay --name  extr-fabric
```

## efa fabric topology show overlay

Displays the overlay connectivity of the devices in a fabric.

*Syntax*

**efa fabric topology show overlay** [**--name** *<fabric name>* ]

*Parameters*

**--name**

Name of the fabric

*Examples*

```
efa fabric topology show overlay --name BLR_FABRIC
```

# Fabric Validation, Troubleshooting, and Error Recovery

## Fabric Service and Inventory Service Interactions

Fabric service interacts with Inventory service, using the following:

1. RabbitMQ message bus is used for event notifications to and from inventory service.

| Event from Fabric Service to Inventory Service | Action |
|---|---|
| FAB.Fabric_Created | Inventory will create a Fabric in its database. |
| FAB.Fabric_Deleted | Inventory will delete the Fabric from its database and remove the associations of devices from Fabric. |
| FAB.AddDevices_To_Fabric | Inventory will add the devices to the Fabric mapping. This message contains the IP address of devices. |
| FAB.DeleteDevices_From_Fabric | Inventory will delete the devices from Fabric mapping. This message contains the IP address of devices to be deleted. |
| FAB.ClearDevice_Config | Inventory will fetch the latest configuration for a given list of devices on which the clear config was done. |
| FAB.Fabric_Deployed | Inventory will fetch the latest configuration for a given list of devices on which the Fabric was (de)configured. |

| Event from Inventory Service to Fabric Service | Action |
|---|---|
| INV.Interfaces_Added | Fabric service creates new interfaces and generates the relevant configuration. |
| INV.Interfaces_Deleted | Fabric service deletes the corresponding interfaces and generates the relevant configuration. |
| INV.Interfaces_Updated | Fabric service generates the updated relevant configuration. |
| INV.Device_Link_Discovered | Fabric service generates the updated relevant configuration. |

| | |
|---|---|
| INV.Device_Link_Deleted | Fabric service generates the updated relevant configuration. |
| INV.Device_Updated | Fabric service generates the updated relevant configuration. |
| INV.Mct_Cluster_Added | Fabric service processes the request and generates the updated relevant configuration. |
| INV.Mct_Cluster_Deleted | Fabric service process the MCT cluster and generates the updated relevant configuration. |
| INV.Evpn_Created | Fabric service doesn't handle this event. |
| INV.Evpn_Deleted | Fabric service deletes the EVPN instance and generates the updated relevant configuration. |
| INV.OverlayGateway_Created | Fabric service creates the overlay gateway and generates the updated relevant configuration. |
| INV.OverlayGateway_Deleted | Fabric service deletes the overlay gateway and generates the updated relevant configuration. |

2. REST API is used for fetching the inventory data.

   Fabric service uses the REST APIs provided by inventory service to get the following information.

   a. Interfaces
   b. Device details including credentials
   c. Port-Channel
   d. LLDP Neighbors
   e. Router BGP
   f. EVPN
   g. MCT Cluster including control VLAN and control VE
   h. Management Cluster
   i. Overlay gateway

## Fabric Service and Tenant Service Interactions

1. Tenant Service needs the Fabric Settings information from Fabric Service (via REST).
2. Tenant Service can listen to all the events published to the Inventory Service.

## Fabric Service GoSwitch API

1. Fabric service uses GoSwitch for all configurations of the Create, Update, and Delete operations.
2. Fabric service uses GoSwitch for some of the configurations of the GET operations.

## Scalability

Any change in the topology will result in a configuration change only on the affected nodes. Configure the fabric to batch the configuration sent to the devices, based on one of the following strategies:

- POD level config push
- N number of device config push

The table below depicts the scalability numbers:

| Number of devices in fabric | Time taken by “fabric configure” |
|---|---|
| 3 (1 spine node + 1*2 node MCT pair) | 40 sec |
| 7 (1 spine node + 3*2 node MCT pair) | 1 min 18 sec |
| 32 (4 spine nodes + 14*2 node MCT pair) | 1 min 45 sec |

## efa fabric debug clear-config

Clears the underlay/overlay configuration from the device and recovers the device from erroneous conditions.

*Syntax*

**efa fabric debug clear-config** [ **--device** *device ip* | **--reference-fabric** *fabric name* ]

*Parameters*

**device**

The device IP address

**-reference-fabric**

Name of the fabric and to which device it will eventually belong.

*Examples*

```
$ efa fabric debug clear-config -device 10.24.4810.24.48.131,10.24.51.135,10.24.51.
131,10.25.225.58,10.24.80.139.131,10.24.51.135,10.24.51.131,10.25.225.58,10.24.80.139
```

## efa fabric debug config-gen-reason

Obtains the configuration generation reason for a particular fabric device.

*Syntax*

**efa fabric debug config-gen-reason** [ **-device** *device ip* | **-name** *fabric name* ]

*Parameters*

**device**

The device IP address

**name**

Name of the fabric to which the device belongs

*Examples*

```
efa fabric debug config-gen-reason --device 10.24.80.139 --name BLR_FABRIC
```

## efa fabric debug push-config

Pushes the configuration for the device from the fabric.

*Syntax*

**efa fabric debug push-config** [ **-device** *device ip* | **-name** *fabric name* ]

*Command Default*

If the device is not mentioned, the configuration for the entire fabric is Re-push.

*Parameters*

**device**

A comma separated list of Device IP addresses for which config needs to Re-push

**name**

Name of the fabric to which the device belongs

*Examples*

```
efa fabric debug push-config --device 10.25.225.58--name BLR_FABRIC
```

## efa fabric error show

Displays the name of the fabric, error types, and reasons for the errors.

*Syntax*

**efa fabric error show** [ **--name** *fabric name* | **--export** *file name* ]

*Command Default*

During device add to the fabric and during fabric configure, validation errors (topology and configuration) are reported for corrective action.

Errors occurring during the add, validate, and configure phase will be persisted in the DB.

A CLI will be provided that lists the device errors.

*Parameters*

**name**

Name of the fabric

**export**

Export fabric details to a csv file

*Examples*

```
efa fabric device error show --name BLR_FABRIC
```

## efa fabric execution show

Displays the REST API executions of Fabric service.

*Syntax*

**efa fabric execution show** [ **-id** *execution id* | **-limit** *number of executions* | **-status** *failed* | *succeeded* | *all* ]

*Parameters*

**id**

Filter the executions based on the execution id. "Limit" and "status" flags are ignored when the "id" flag is given.

**-limit**

Limit the number of executions to be listed. A value of "0" lists all the executions. The default is 10.

**-status**

Filter the executions based on the status (failed/succeeded/all). The default is "all".

*Examples*

```
efa fabric execution show
```

## efa supportsave

Collects the support-save of the Inventory, Tenant, and Fabric service logs, and their associated database dumps.

*Syntax*

**efa supportsave**

*Usage Guidelines*

The supportsave file is saved to `/var/log/efa/efa_<log-id>.logs.zip`.

*Examples*

This example collects the supportsave logs.

```
sudo efa supportsave
Version: 2.1.0
Build: GA
Time Stamp: 20-01-31:15:11:36
Support Save File: /var/log/efa/efa_1586909025.logs.zip
```

## efa fabric show-config

Displays the config of a given fabric.

*Syntax*

**efa fabric show-config** [ **--name** *fabric name* | **--device-role** *leaf* | *spine* | *super-spine* | *border-leaf* | **--ip** *ip address* ]

*Parameters*

**name**

Name of the fabric

**-device-role**

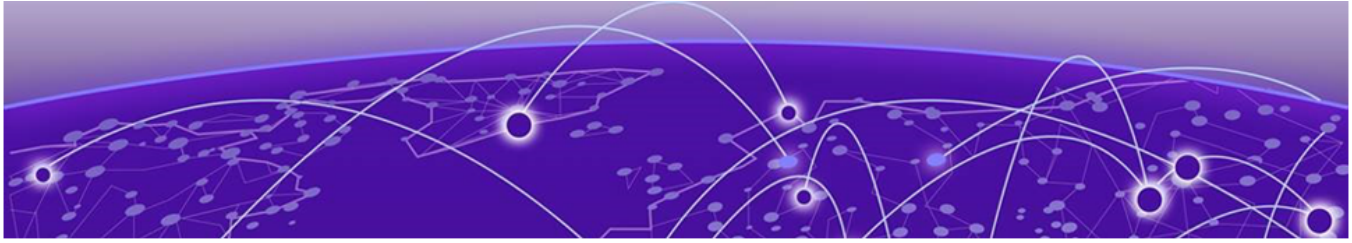
Role of devices for which config needs to show (leaf | spine | super-spine | border-leaf)

**ip**

The specific ip provided in conjuction with the device-role field.

*Examples*

```
efa fabric show-config --name BLR_FABRIC --device-role border-leaf --ip 10.25.225.58
```



# Tenant Services Provisioning



## Tenant Services Provisioning Overview

Tenant Services exposes the CLI and REST API for automating the Tenant network configuration on the Clos and non-Clos fabric.

Tenant network configuration includes VLAN, BD, VE, EVPN, VTEP, VRF, and Router BGP configuration on the necessary fabric devices to provide Layer 2-extension and Layer 3-extension across the fabric.

Tenant Services provisioning automates the Tenant configuration, which can be a subset of the combinations provided by the switching hardware.

Tenant Services supports multiple fabrics.

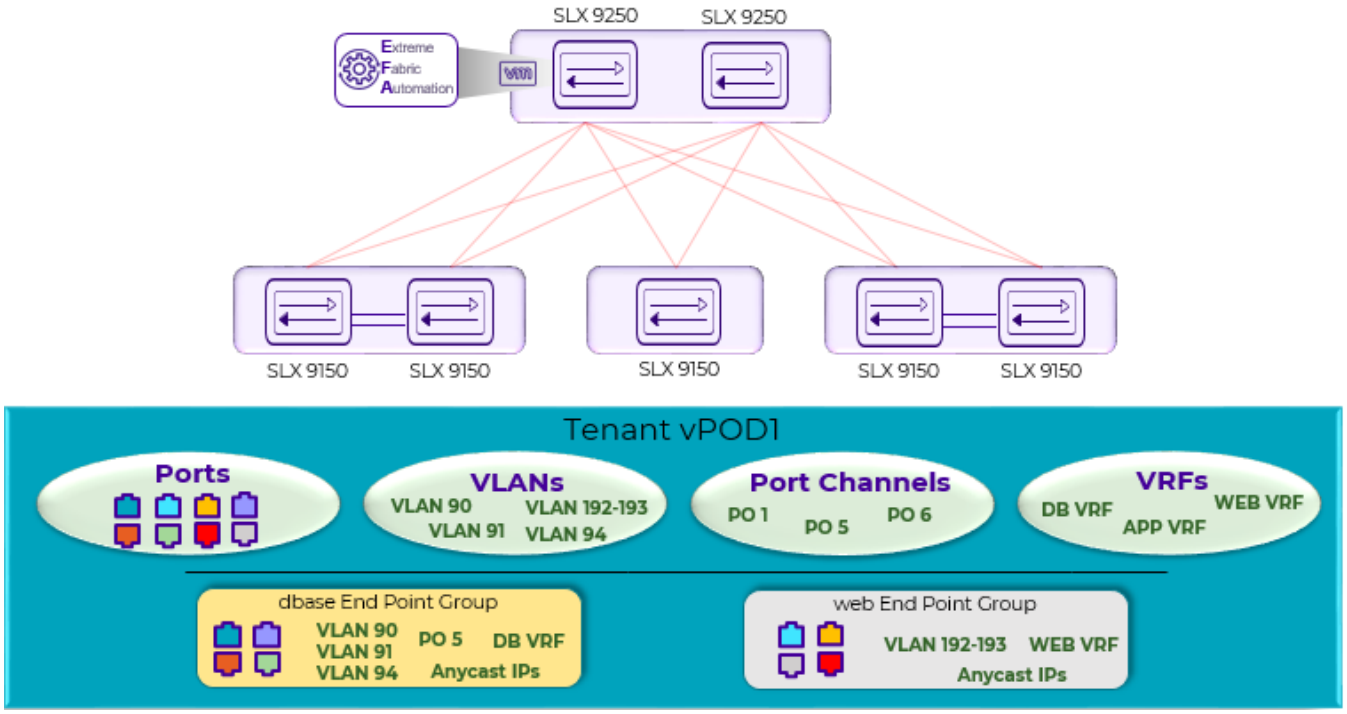



**Figure 9: Tenant Services Overview**

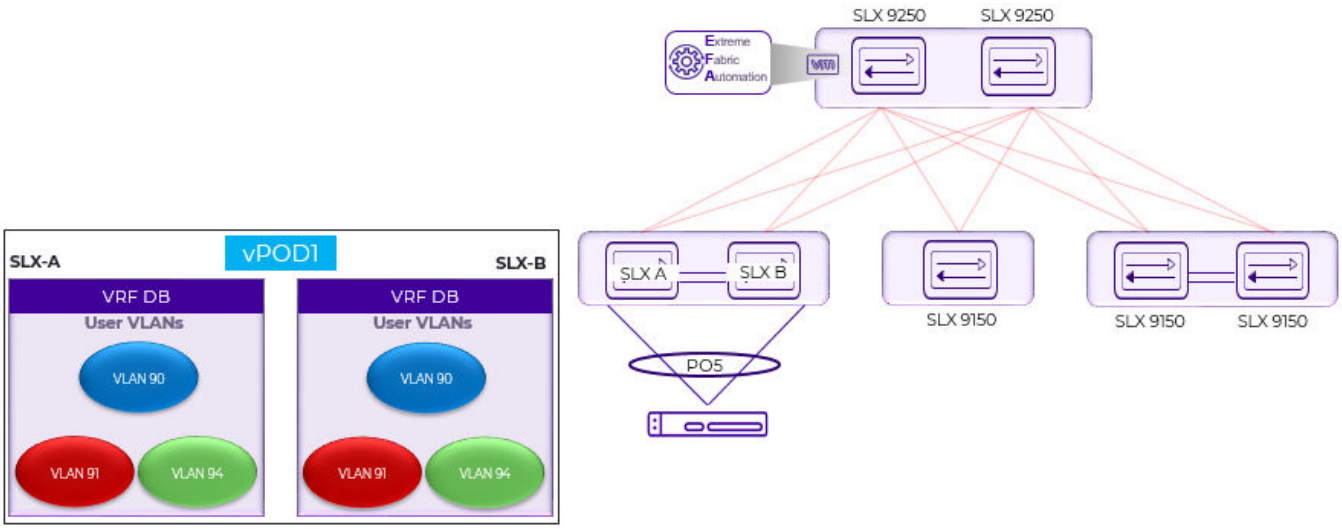



**Figure 10: Tenant Name vPOD1, VRF Name DB**

## Tenant

A Tenant is a logical construct that owns resources as follows:

- VLAN range: Ctags pertaining to which the traffic is expected to ingress and egress.

  Note
  Ctag (Customer VLAN tag) is used to identify the customer broadcast domain. In the IP Fabric network, it represents the customer and is mapped into a VXLAN tunnel thru a VNI (virtual network identifier). The VNI is the ID used to identify the VXLAN tunnel. With auto VNI mapping the Ctag ID equals the VNI. Users can also manually map Ctags to user-defined VNIs. These VNIs can be VLAN IDs (up to 4k) or to BD's (bridge domain) IDs.

- Device ports: Ports on which the traffic is expected to ingress and egress.

## Default Tenant

There is no longer a concept called "default tenant." The provisioning model and implementation are the same for "default" and "non-default" tenants.

## VLAN-based Tenant

For a VLAN based tenant, realization of network on the device is done using VLAN and switchport VLANs. Bridge domains are used for EVPN IRB.

## Bridge domain-based Tenant

For a BD based tenant, realization of network on the device is done using BD and BD-LIF. BD is used for EVPN IRB.

## Scalability

**Table 5: VNI scalability**

| VNI type | Scale |
|---|---|
| Non-auto VNI mapping | • The number of VNI (networks) supported per device = 8K [4K VLAN + 4K BD]<br>• The maximum number of VNI (networks) supported in the fabric = [8K * number of devices in the fabric]. |
| Auto VNI mapping | • The number of VNI (networks) supported per device = 8K [4K VLAN + 4K BD]<br>• The number of VNI (networks) supported per fabric = 8K |

## Event handling

Event handling specifies the scope of the tenant configuration on the devices.

Devices are added to the Tenant service only when the Fabric is provisioned on the devices.

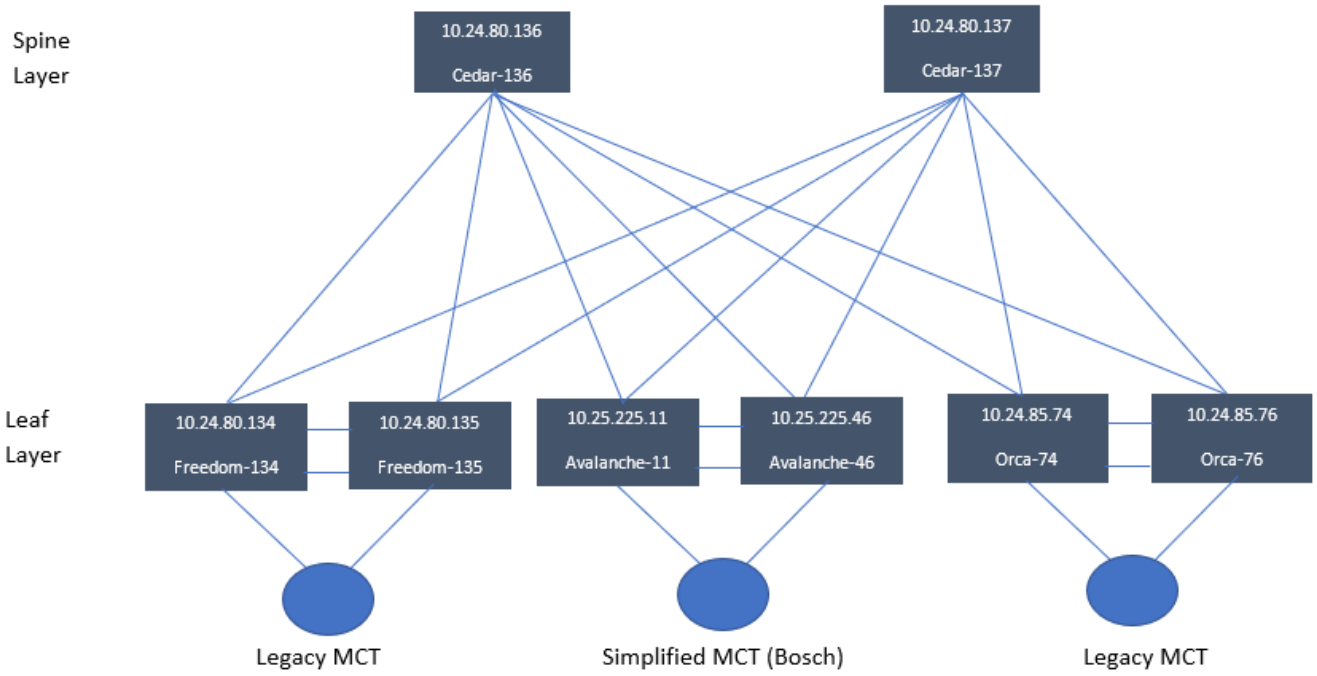
An event is an occurrence of a device being removed from the Fabric or from the Inventory.

- When a device is removed from the Fabric or Inventory, the device is cleaned up from Tenant Service and the Tenant configuration is removed from the device.
- User-created entities, such as Tenant, VRF, and EPG, are not deleted whereas references for ports/port-channels of deleted devices are removed.

## CLOS Fabric with Non-auto VNI Map

Auto VNI simplifies the mapping IDs by using the VLAN ID as the VNI ID (i.e. VLAN 100 = VNI 100). This works well in environments where overlapping VLANs are not being used. However, if two different tenants are using VLAN 100, VNI 100 cannot be used by both. At this point, manual mapping of VLAN to VNI is required. Extreme Fabric Automation simplifies this by allowing VNI ranges for tenants to automate "manual" mapping to work for overlapping VLANs.

The following figure shows 3-stage CLOS topology.




```
efa fabric create --name fabric1
efa fabric setting update --name fabric1 --vni-auto-map No
efa fabric device add-bulk --spine 10.24.80.136 --border-leaf
10.25.225.11,10.25.225.46 --leaf 10.24.80.134-
135,10.24.85.74,10.24.85.76 --username admin --password password --
name fabric1
efa fabric configure --name fabric1
```

**Figure 11: 3-stage CLOS Topology**

The following figure shows tenant constructs in CLOS fabric.


**Figure 12: Scope of Tenant Constructs in Fabric**

## CLOS Fabric with Auto VNI Map

- In CLOS fabric with auto VNI map, the VNI is statically derived using the VLAN ID or BD ID.
  - For the VLAN case, VNI = VLAN ID
  - For the BD case, VNI = 4096 + BD ID
  - User will not be able to reserve l2-vni-range or l3-vni-range for a given tenant.
  - User will not be able to provide a specific l2-vni/l3-vni in an EPG.
- VLAN Based Tenants:

  Multiple VLAN Based tenants cannot share the same VLAN, considering the multiple tenants cannot share the same VNI.
- BD Based Tenants:

  Multiple BD Based tenants can share the same VLAN, as the VLANs from each tenant will be mapped to a unique BD and further a unique VNI.

### Multi Tenancy

The following example shows a multi tenancy configuration.

```
efa tenant create --name tenant11 --vrf-count 10 --vlan-range 2-4090 --port
10.24.80.134[0/15-17],10.24.80.135[0/15-17],10.25.225.11[0/15-17],10.25.225.46[0/15-17],
10.24.85.74[0/15-17],10.24.85.76[0/15-17] --description Subscriber1

efa tenant show
+----------+-------------+-------------+------------+-----------+-----------
+-----------------------+
|   Name   | L2VNI-Range | L3VNI-Range | VLAN-Range | VRF-Count | Enable-BD |
Ports         |
+----------+-------------+-------------+------------+-----------+-----------
+-----------------------+
| tenant11 |             |             | 2-4090     | 10        | False     |
10.24.85.74[0/15-17]  |
|          |             |             |            |           |           |
10.24.80.135[0/15-17] |
|          |             |             |            |           |           |
10.25.225.11[0/15-17] |
|          |             |             |            |           |           |
10.25.225.46[0/15-17] |
```

```
|          |             |             |            |           |           |
10.24.80.134[0/15-17] |
|          |             |             |            |           |           |
10.24.85.76[0/15-17]  |
+----------+-------------+-------------+------------+-----------+-----------
+-----------------------+

efa tenant create --name tenant12 --vrf-count 10  --vlan-range 2-4090 --port
10.24.80.134[0/18-20],10.24.80.135[0/18-20],10.25.225.11[0/18-20],10.25.225.46[0/18-20],
10.24.85.74[0/18-20],10.24.85.76[0/18-20]
Tenant Creation Failed:
         Vlan (2) overlaps with Tenant (tenant11)

efa tenant create --name tenant21 --vrf-count 10 --enable-bd --port 10.24.80.134[0/21-25],
10.24.80.135[0/21-25],10.24.85.74[0/21-25],10.24.85.76[0/21-25],10.25.225.11[0/21-25],
10.25.225.46[0/21-25]

efa tenant create --name tenant22 --vrf-count 10 --enable-bd --port 10.24.80.134[0/26-30],
10.24.80.135[0/26-30],10.24.85.74[0/26-30],10.24.85.76[0/26-30],10.25.225.11[0/26-30],
10.25.225.46[0/26-30]

efa tenant show
+----------+-------------+-------------+------------+-----------+-----------
+-----------------------+
|   Name   | L2VNI-Range | L3VNI-Range | VLAN-Range | VRF-Count | Enable-BD |
Ports         |
+----------+-------------+-------------+------------+-----------+-----------
+-----------------------+
| tenant11 |             |             | 2-4090     | 10        | False     |
10.25.225.46[0/15-17] |
|          |             |             |            |           |           |
10.25.225.11[0/15-17] |
|          |             |             |            |           |           |
10.24.80.135[0/15-17] |
|          |             |             |            |           |           |
10.24.85.74[0/15-17]  |
|          |             |             |            |           |           |
10.24.85.76[0/15-17]  |
|          |             |             |            |           |           |
10.24.80.134[0/15-17] |
+----------+-------------+-------------+------------+-----------+-----------
+-----------------------+
| tenant21 |             |             | 2-4090     | 10        | True      |
10.24.85.74[0/21-25]  |
|          |             |             |            |           |           |
10.25.225.11[0/21-25] |
|          |             |             |            |           |           |
10.25.225.46[0/21-25] |
|          |             |             |            |           |           |
10.24.80.134[0/21-25] |
|          |             |             |            |           |           |
10.24.85.76[0/21-25]  |
|          |             |             |            |           |           |
10.24.80.135[0/21-25] |
+----------+-------------+-------------+------------+-----------+-----------
+-----------------------+
| tenant22 |             |             | 2-4090     | 10        | True      |
10.24.85.76[0/26-30]  |
|          |             |             |            |           |           |
10.25.225.11[0/26-30] |
|          |             |             |            |           |           |
10.24.80.135[0/26-30] |
|          |             |             |            |           |           |
10.25.225.46[0/26-30] |
```

```
|          |             |             |            |           |           |
10.24.85.74[0/26-30]  |
|          |             |             |            |           |           |
10.24.80.134[0/26-30] |
+----------+-------------+-------------+------------+-----------+-----------
+------------------------+

efa tenant epg create --name epg11 --tenant tenant11 --po po1115,po1215,po1315 --
switchport-mode trunk --switchport-native-vlan 11 --ctag-range 11-12

efa tenant epg create --name epg21 --tenant tenant21 --po po2121,po2221,po2321 --
switchport-mode trunk  --ctag-range 11-12

efa tenant epg create --name epg22 --tenant tenant21 --po po2122,po2222,po2322  --
switchport-mode trunk  --ctag-range 11-12

efa tenant epg show

================================================================================
Name          : epg11
Tenant        : tenant11
Description   :
Ports         :
POs           : po1315, po1215, po1115
Port Property : switchport mode     : trunk
              : native-vlan-tagging : false
NW Policy     : ctag-range          : 11-12

Network Property [Flags : * - Native Vlan]
+------+--------+------------+---------+-------------+-------------+
| Ctag | L2-Vni | Anycast-ip | BD-name |  Dev-state  |  App-state  |
+------+--------+------------+---------+-------------+-------------+
| 11*  | 11     |            |         | provisioned | cfg-in-sync |
+------+--------+------------+---------+-------------+-------------+
| 12   | 12     |            |         | provisioned | cfg-in-sync |
+------+--------+------------+---------+-------------+-------------+

================================================================================
================================================================================
Name          : epg21
Tenant        : tenant21
Description   :
Ports         :
POs           : po2121, po2221, po2321
Port Property : switchport mode     : trunk
              : native-vlan-tagging : false
NW Policy     : ctag-range          : 11-12

Network Property [Flags : * - Native Vlan]
+------+--------+------------+--------------+-------------+-------------+
| Ctag | L2-Vni | Anycast-ip |   BD-name    |  Dev-state  |   App-state |
+------+--------+------------+--------------+-------------+-------------+
| 11   | 4099   |            | Auto-BD-4099 | provisioned | cfg-in-sync |
+------+--------+------------+--------------+-------------+-------------+
| 12   | 4100   |            | Auto-BD-4100 | provisioned | cfg-in-sync |
+------+--------+------------+--------------+-------------+-------------+

================================================================================
================================================================================
Name          : epg22
Tenant        : tenant21
Description   :
Ports         :
POs           : po2122, po2222, po2322
```

```
Port Property : switchport mode     : trunk
              : native-vlan-tagging : false
NW Policy     : ctag-range          : 11-12

Network Property [Flags : * - Native Vlan]
+------+--------+------------+--------------+-------------+-------------+
| Ctag | L2-Vni | Anycast-ip |   BD-name    |  Dev-state  |  App-state  |
+------+--------+------------+--------------+-------------+-------------+
| 12   | 4102   |            | Auto-BD-4102 | provisioned | cfg-in-sync |
+------+--------+------------+--------------+-------------+-------------+
| 11   | 4101   |            | Auto-BD-4101 | provisioned | cfg-in-sync |
+------+--------+------------+--------------+-------------+-------------+

efa tenant epg create --name epg23 --tenant tenant21 --po po2122,po2322  --switchport-
mode trunk  --ctag-range 21-22 --bridge-domain 21:Auto-BD-4101 --bridge-domain 22:Auto-
BD-4102

efa tenant epg show

=========================================================================
Name          : epg11
Tenant        : tenant11
Description   :
Ports         :
POs           : po1315, po1215, po1115
Port Property : switchport mode     : trunk
              : native-vlan-tagging : false
NW Policy     : ctag-range          : 11-12

Network Property [Flags : * - Native Vlan]
+------+--------+------------+---------+-------------+-------------+
| Ctag | L2-Vni | Anycast-ip | BD-name |  Dev-state  |  App-state  |
+------+--------+------------+---------+-------------+-------------+
| 11*  | 11     |            |         | provisioned | cfg-in-sync |
+------+--------+------------+---------+-------------+-------------+
| 12   | 12     |            |         | provisioned | cfg-in-sync |
+------+--------+------------+---------+-------------+-------------+

=========================================================================
=========================================================================
Name          : epg21
Tenant        : tenant21
Description   :
Ports         :
POs           : po2121, po2221, po2321
Port Property : switchport mode     : trunk
              : native-vlan-tagging : false
NW Policy     : ctag-range          : 11-12

Network Property [Flags : * - Native Vlan]
+------+--------+------------+--------------+-------------+-------------+
| Ctag | L2-Vni | Anycast-ip |   BD-name    |  Dev-state  |   App-state |
+------+--------+------------+--------------+-------------+-------------+
| 11   | 4099   |            | Auto-BD-4099 | provisioned | cfg-in-sync |
+------+--------+------------+--------------+-------------+-------------+
| 12   | 4100   |            | Auto-BD-4100 | provisioned | cfg-in-sync |
+------+--------+------------+--------------+-------------+-------------+

=========================================================================
=========================================================================
Name          : epg22
Tenant        : tenant21
Description   :
Ports         :
```

```
POs           : po2122, po2222, po2322
Port Property : switchport mode     : trunk
              : native-vlan-tagging : false
NW Policy     : ctag-range          : 11-12

Network Property [Flags : * - Native Vlan]
+------+--------+------------+--------------+-------------+-------------+
| Ctag | L2-Vni | Anycast-ip |   BD-name    |  Dev-state  |  App-state  |
+------+--------+------------+--------------+-------------+-------------+
| 12   | 4102   |            | Auto-BD-4102 | provisioned | cfg-in-sync |
+------+--------+------------+--------------+-------------+-------------+
| 11   | 4101   |            | Auto-BD-4101 | provisioned | cfg-in-sync |
+------+--------+------------+--------------+-------------+-------------+

==========================================================================
==========================================================================
Name          : epg23
Tenant        : tenant21
Description   :
Ports         :
POs           :
Port Property : switchport mode     : trunk
              : native-vlan-tagging : false
NW Policy     : ctag-range          : 21-22

Network Property [Flags : * - Native Vlan]
+------+--------+------------+--------------+-------------+-------------+
| Ctag | L2-Vni | Anycast-ip |   BD-name    |  Dev-state  |  App-state  |
+------+--------+------------+--------------+-------------+-------------+
| 21   | 4101   |            | Auto-BD-4101 | provisioned | cfg-in-sync |
+------+--------+------------+--------------+-------------+-------------+
| 22   | 4102   |            | Auto-BD-4102 | provisioned | cfg-in-sync |
+------+--------+------------+--------------+-------------+-------------+

==========================================================================
```

## Configuring Tenant Services

Use the commands in this section to configure Tenant services.

Tenant Services configuration involves the following steps:

- Creating a Tenant
- Creating Port Channels
- Creating a VRF
- Creating an EPG

## efa tenant create

Creates tenant reserved resources like total number of L3 VNIs, VLANs, VRFs, and Bridge Domains for fabrics with non-auto VNI settings which can later be applied to an end point group.

*Syntax*

**efa tenant create** [ **--name** *tenant-name* | **-- description** *tenant-description* | **--l2-vni-start** *value* | **--l3-vni-start** *value* | **--vlan-range** *value* | **--vrf-count** *value* | **--enable-bd** |**port** *list of ports* ]

*Parameters*

**name** *tenant name*

Name of the tenant.

**description** *tenant description*

Describes the tenant.

**l2-vni-start** *value*

Contiguous Range of L2 VNIs in ascending order starting from l2-vni-start will be reserved for the tenant within the scope of a fabric.

**l3-vni-start** *value*

Contiguous Range of L3 VNIs in ascending order starting from l3-vni-start will be reserved for the tenant within the scope of a fabric.

**vlan-range** *value*

Range of Vlans to be reserved for the tenant.

**vrf-count int32** *value*

Number of VRFs reserved for the tenant.

**enable-bd**

Enable BD capability for networks created under this tenant.

**port** *value*

List of physical ports of devices which will be reserved for the asset. Example SW1_IP[0/1],SW2_IP[0/5].

*Examples*

```
efa tenant create --name vPOD1 --vrf-count 10 --vlan-range 2-4090 --port 172.31.254.13
[0/1],172.31.254.20 [0/1], --description Subscriber1
```

## efa tenant show

Displays the tenant details.

*Syntax*

**efa tenant show -- name** *tenant-name*

*Parameters*

**name** *tenant-name*

Specifies the name of the tenant.

*Examples*

This example shows the output of the command:

```
device# show efa status
Starting efa deploy one-touch CLI, please DO NOT hit CTRL+C
Step  1: Checking if TPVM is already deployed With necessary configurations for efa
commands to Run
Done
Step 2 Get IP Address assigned to TPVM to deploy the app
IP  Address of the TPVM: 10.20.63.187:
Done
Step 3: version details
==================================================
               EFA version details
==================================================
Version : v2.1.0
Build: 30
Time Stamp: 20-01-28:13:18:25
— Time Elapsed: 510.667Âµs —


Step 4: node details
==================================================
               k3s node details
==================================================
NAME   STATUS   ROLES    AGE     VERSION
tpvm   Ready    master   7m22s   v1.16.3-k3s.2


Step 5: EFA details
==================================================
             EFA application details
==================================================
NAME                                        READY   STATUS    RESTARTS   AGE
pod/efa-api-docs-5d457dc874-7rb7d           1/1     Running   0          4m5s
pod/godb-service-57bd99747-jj6xq            1/1     Running   0          4m3s
pod/rabbitmq-0                              1/1     Running   0          4m6s
pod/goinventory-service-55db6bc9c4-s57cd    1/1     Running   0          4m5s
pod/goopenstack-service-847777494c-vf58b    1/1     Running   4          4m1s
pod/gofabric-service-679f4c998-hw5nm        1/1     Running   0          4m5s
pod/gohyperv-service-8564db6f6c-5t7tf       1/1     Running   4          3m58s
pod/gotenant-service-7c785864b-m2r2r        1/1     Running   0          4m5s
pod/govcenter-service-6cc999d4b9-97npn      1/1     Running   5          3m59s

NAME                           TYPE        CLUSTER-IP      EXTERNAL-IP
PORT(S)                           AGE
service/gofabric-service       ClusterIP   10.43.62.102    <none>        8081/
TCP                           4m5s
```

```
service/goinventory-service   ClusterIP   10.43.65.52     <none>        8082/
TCP                           4m5s
service/gotenant-service      ClusterIP   10.43.90.204    <none>        8083/
TCP                           4m5s
service/efa-api-docs          ClusterIP   10.43.228.63    <none>        80/
TCP                           4m5s
service/rabbitmq              NodePort    10.43.9.29      <none>        15672:31672/TCP,
5672:30672/TCP   4m4s
service/db-service            NodePort    10.43.199.176   <none>        5432:30432/
TCP                    4m3s
service/goopenstack-service   NodePort    10.43.241.10    <none>        8085:30085/
TCP                    4m1s
service/govcenter-service     ClusterIP   10.43.241.91    <none>        8086/
TCP                           3m59s
service/gohyperv-service      ClusterIP   10.43.19.40     <none>        8087/
TCP                           3m58s

NAME                                  READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/efa-api-docs          1/1     1            1           4m5s
deployment.apps/godb-service          1/1     1            1           4m3s
deployment.apps/goinventory-service   1/1     1            1           4m5s
deployment.apps/goopenstack-service   1/1     1            1           4m1s
deployment.apps/gofabric-service      1/1     1            1           4m5s
deployment.apps/gohyperv-service      1/1     1            1           3m58s
deployment.apps/gotenant-service      1/1     1            1           4m5s
deployment.apps/govcenter-service     1/1     1            1           3m59s

NAME                                           DESIRED   CURRENT   READY   AGE
replicaset.apps/efa-api-docs-5d457dc874        1         1         1       4m5s
replicaset.apps/godb-service-57bd99747         1         1         1       4m3s
replicaset.apps/goinventory-service-55db6bc9c4 1         1         1       4m5s
replicaset.apps/goopenstack-service-847777494c 1         1         1       4m1s
replicaset.apps/gofabric-service-679f4c998     1         1         1       4m5s
replicaset.apps/gohyperv-service-8564db6f6c    1         1         1       3m58s
replicaset.apps/gotenant-service-7c785864b     1         1         1       4m5s
replicaset.apps/govcenter-service-6cc999d4b9   1         1         1       3m59s

NAME                         READY   AGE
statefulset.apps/rabbitmq    1/1     4m6s

device#
```

## efa tenant update

Allows changes to be made to a tenant after it has been created.

*Syntax*

**efa tenant update** [ -- name *tenant-name* | --**operation** *value* | --**description** *tenant-description* | --**l2-vni-range** *value* | --**l3-vni-range** *value* | --**vlan-range** *value* | --**vrf-count** *value* | --**enable-bd** –**port** *list-of-ports* ]

*Command Default*

*Parameters*

**name** *tenant-name*

Specifies name of tenant.

**description** *tenant-description*

Describes the tenant.

**l2-vni-range** *value*

Specifies the contiguous range of Layer 2 VNIs in ascending order that are reserved for the tenant within the scope of a fabric.

**l3-vni-range** *value*

Specifies the contiguous Range of L3 VNIs in ascending order will be reserved for the tenant within the scope of a fabric.

**vlan-range** *value*

Specifies the range of Vlans to be reserved for the tenant.

**vrf-count int** *value*

Specifies the number of VRFs tobe reserved for the tenant.

**enable-bd** *value*

Enables BD capability for networks created under this tenant.

**operation** *value*

Specifies operation code. Valid values are desc-update | vni-update | port-add | port-delete | vlan-update | num-vrf-update | enable-bd-update

**port** *value*

Lists physical ports of devices which will be reserved for the asset. Example: SW1_IP[0/1],SW2_IP[0/5]

*Examples*

Example:

```
tenant update --name tenant11 --operation desc-update --description tenant11Desc

tenant update --name tenant11 --operation enable-bd-update --enable-bd

tenant update --name tenant11 --operation vni-update --l2-vni-range 10002-14190  --l3-vni-
range 14191-14200

tenant update --name tenant11 --operation vlan-update --vlan-range 2-4090

--vlan-range 2-4090tenant update --name tenant11 --operation num-vrf-update --vrf-count 10

tenant update --name tenant11 --operation port-add --port 10.24.80.134[0/15-17],
10.24.80.135[0/15-17],10.25.225.11[0/15-17],10.25.225.46[0/15-17],10.24.85.74[0/15-17],
10.24.85.76[0/15-17
```

## efa tenant delete

Deletes a tenant.

*Syntax*

**efa tenant delete** [ -- **name** *tenant-name* --**force** ]

*Command Default*

*Parameters*

**name** *tenant-name*

Specifies a tenant and a tenant name.

**force**

Forces the deletion on the tenant if the option is provided.

*Examples*

The following example deletes a tenant:

```
efa tenant delete --name tenant11
```

## efa tenant po create

Creates a portchannel.

*Syntax*

**efa tenant po create** [ **--name** *tenant-name* | **--tenant** *tenant-name* | **--speed** *speed-value* | **--negotiation** *neg-value* | **--port** *port-list* | **--number** *po-id* ]

*Parameters*

**name** *PortChannel Name*

Specifies the portchannel name.

**tenant** *tenant name*

Specifies the tenant name.

**speed** *value*

Configure speed for the portchannel and its Member ports. Valid values are 100Mbps|1Gbps|10Gbps|25Gbps|40Gbps|100Gbps.

**negotiation** *value*

Configure LACP Negotiation mode for portchannel. Valid values are active|passive|static.

**port** *value*

Specifies device ip along with ethernet port details. Example: SW1_IP[0/1],SW2_IP[0/5]

**number** *value*

Portchannel interface number generated by the service.

*Examples*

```
efa tenant po create --name po5 --port 172.31.254.13 [0/1],172.31.254.20 [0/1], --
negotiation active --tenant vPOD1
```

## efa tenant po show

Specifies the portchannel of all tenants, a given tenant, or a given po.

*Syntax*

**efa tenant po show** [ **--name** *portchannel name* **--tenant** *tenant name* ]

*Parameters*

**name** *portchannel-name*

Specifies portchannel name.

**tenant** *tenant -name*

Specifies the tenant name.

*Examples*

Example:

```
efa tenant po show
+--------+----------+----+--------+-------------+--------------------+
|  Name  |  Tenant  | ID | Speed  | Negotiation |       Ports        |
+--------+----------+----+--------+-------------+--------------------+
| po1115 | tenant11 | 3  | 10Gbps | active      | 10.24.85.76[0/15]  |
|        |          |    |        |             | 10.24.85.74[0/15]  |
+--------+----------+----+--------+-------------+--------------------+
| po1215 | tenant11 | 1  | 10Gbps | active      | 10.24.80.134[0/15] |
|        |          |    |        |             | 10.24.80.135[0/15] |
+--------+----------+----+--------+-------------+--------------------+
| po1315 | tenant11 | 1  | 10Gbps | active      | 10.25.225.11[0/15] |
|        |          |    |        |             | 10.25.225.46[0/15] |
+--------+----------+----+--------+-------------+--------------------+
```

## efa tenant po update

Updates a portchannel.

*Syntax*

**efa tenant po update** [ **--name** *po-name* | **--tenant** *tenant-name* | **--operation** *port-add* | *port-delete* | **--port** *port-list* ]

*Parameters*

**name** *portchannel name*

Specifies portchannel.

**tenant** *tenant name*

Specifies tenant name.

**operation** *operation name*

Adds or deletes operation on the ports. Valid options are port-add | port-delete.

**port** *value*

Specifies device ip along with ethernet port details. Example: SW1_IP [0/1], SW2_IP[0/5]

*Examples*

```
efa tenant po update --name po1115 --tenant tenant11 --operation port-add --port
10.24.85.76[0/15]
efa tenant po update --name po1115 --tenant tenant11 --operation port-delete --port
10.24.85.76[0/15]
```

## efa tenant po delete

Deletes a portchannel.

*Syntax*

**efa tenant po delete** [ **--name** *tenant name* **--force** ]

*Parameters*

**name** *tenant name*

Specifies portchannel name or comma-separated portchannel names. Ex: po1 or po1,po2,po3.

**force**

Forces the portchannel deletion if the option is provided.

*Examples*

```
efa tenant po delete --name po1115,po1215,po1315 --tenant tenant11
```

## efa tenant vrf create

Creates a VRF.

*Syntax*

**efa tenant vrf create** [ **--name** *vrf name* **--tenant** *tenant name* **--rt-type** *value* **--rt** *value* ]

*Parameters*

**name** *VRF name*

Specifies the name of the VRF.

**tenant** *tenant name*

Specifies the name of the tenant.

**rt-type** *value*

Route Target VPN Community. Valid values are both|import|export.

**rt** *value*

A unique number setting for forming Route Target and Route Distinguisher.

*Examples*

```
efa tenant vrf create --name DB --tenant vPOD1
```

## efa tenant vrf show

Displays the VRF information of all tenants, a selected tenant, or a selected VRF.

*Syntax*

**efa tenant vrf show** [ **--name** *vrf name* **--tenant** *tenant name* ]

*Parameters*

**name** *vrf name*

Name of the VRF.

**tenant** *tenant name*

Name of the tenant.

*Examples*

```
efa tenant vrf show
+--------+----------+--------+--------+--------+----------------+
|  Name  |  Tenant  | L3-VNI | IRB-BD | IRB-VE |  Route Target  |
+--------+----------+--------+--------+--------+----------------+
| blue11 | tenant11 |        |        |        | import 100:100 |
|        |          |        |        |        | export 100:100 |
|        |          |        |        |        | import 200:200 |
|        |          |        |        |        | export 200:200 |
|        |          |        |        |        | import 300:300 |
|        |          |        |        |        | export 400:400 |
+--------+----------+--------+--------+--------+----------------+
| red11  | tenant11 |        |        |        | import 101:101 |
|        |          |        |        |        | export 101:101 |
|        |          |        |        |        | import 201:201 |
|        |          |        |        |        | export 201:201 |
|        |          |        |        |        | import 301:301 |
|        |          |        |        |        | export 401:401 |
+--------+----------+--------+--------+--------+----------------+
| red21  | tenant21 |        |        |        | import 101:103 |
|        |          |        |        |        | export 101:103 |
|        |          |        |        |        | import 201:203 |
|        |          |        |        |        | export 201:203 |
|        |          |        |        |        | import 301:303 |
|        |          |        |        |        | export 401:403 |
+--------+----------+--------+--------+--------+----------------+
| blue21 | tenant21 |        |        |        | import 100:102 |
|        |          |        |        |        | export 100:102 |
|        |          |        |        |        | import 200:202 |
|        |          |        |        |        | export 200:202 |
|        |          |        |        |        | import 300:302 |
|        |          |        |        |        | export 400:402 |
```

## efa tenant vrf delete

Deletes a VRF.

*Syntax*

**efa tenant vrf delete** [ **--name** *vrf name* **--tenant** *tenant name* ]

*Command Default*

*Parameters*

**name** *vrf name*

Specifies the name of the VRF.

**tenant** *tenant name*

Specifies the name of the tenant.

**rt-type** *value*

Specifies route target VPN community. Valid values are both|import|export.

**rt** *value*

Specifies a unique number for setting for forming Route Target and Route Distinguisher.

*Examples*

```
efa tenant vrf delete --name red21 --tenant tenant21
efa tenant vrf delete --name blue11,red11 --tenant tenant11
```

## efa tenant epg create

Creates an End Point Group (EPG).

*Syntax*

**efa tenant epg create** [**--name** *epg-name* |**--tenant** *tenant name* |**--description** {**port** *port-list* **po** *port-channel list* } **--switchport mode** *value* **--switchport-native-vlan-tagging --ctag range** *value* **--vrf***value* **l3 vni** *value* **l2 vni** *list of ctag:l2-vni* **--anycast-ip** *list of ctag:anycast-ip* **--bridge-domain** *list of ctag:bridge-domain* **--switchport-native-vlan** *value* ]

*Parameters*

**name**

Name of the EPG

**tenant**

Name of the tenant.

**description**

Description of the EPG.

**port**

Device ip along with ethernet port details.

**po**

List of portchannels.

**switchport-mode**

Configures switch port mode on the interfaces. Valid values are `access, trunk, trunk-no-default-native.`

**switchport-native-vlan-tagging**

Enable the native vlan characteristics on the ports of this endpoint group. Valid only if mode is set to trunk.

**switchport-native-vlan**

Configures native vlan on the interfaces. Valid values are 2 through 4090 corresponding to the value of its Ctag-range.

**ctag-range**

Customer vlan range in comma and hyphen separated format.

**vrf**

VRF to which these networks are attached.

**l3-vni**

L3 VNI to be used for this VRF.

**l2-vni**

L2 VNI to be used for this network in the format `ctag:l2-vni.`

**anycast-ip**

Ipv4 anycast address in the format `ctag:anycast-ip.`

**bridge-domain**

Bridge domain name in the format `ctag:bridge-domain.`

*Examples*

```
efa tenant epg create --name dbase --tenant vPOD1 --po po5 --switchport-mode trunk --ctag-
range 90-94 --anycast-ip
 90:192.168.90.254/24,91:192.168.91.254/24,94:192.168.94.254/24 --vrf DB
```

## efa tenant epg error show

Errors reported to the user for epg create or update operation in the DB are shown by this command.

**efa tenant error show --name** *epg-name* |**--tenant** *tenant name*

**name**

Name of the EPG

**tenant**

Name of the tenant.

```
efa tenant epg error show --name e4 --tenant T10
```

## efa tenant epg update

Updates an existing End Point Group (epg).

*Syntax*

**efa tenant epg update** [**--name** *epg-name***--tenant** *tenant-name* **--operation** *value* **--port** *port list* **--po** *portchannel list* **--switchport mode***value* **--switchport-native-vlan-tagging** *value* **--ctag range** *value* **--vrf***value* **--l3 vni** *value* **--l2 vni** *ctag:l2-vni* **--anycast-ip***list of ctag:anycast-ip* **--bridge-domain** *list of ctag:bridge-domain* **--switchport-native-vlan** *value* ]

*Parameters*

**--name**

Name of the EPG

**--tenant**

Name of the tenant

**--operation** *value*

Defines the operation to be performed. Valid values are `port-group-add, port-group-delete, ctag-range-add, ctag-range-delete, vrf-add, vrf-delete.`

**port**

Port or ports on the device where the tenant network is configured. For example, SW1_IP[0/1],SW2_IP[0/5].

**po**

List of portchannels where the tenant network is configured. Example; po1 or po1,po2.

**switchport-mode**

Configures Switch port mode on the interfaces. Valid values are access | trunk | trunk-no-default-native

**switchport-native-vlan-tagging**

Enable the native vlan characteristics on the ports of this endpoint group. Valid only if mode is set to trunk.

**switchport-native-vlan**

Configures native vlan on the interfaces. Valid values are 2 through 4090.

**ctag-range**

Customer vlan range in comma and hyphen separated format.

**vrf**

VRF to which these networks are attached.

**l3-vni**

L3 VNI to be used for this VRF.

**l2-vni**

L2 VNI to be used for this network in the format `ctag:l2-vni.`

**anycast-ip**

Ipv4 anycast address in the format `ctag:anycast-ip.`

**bridge-domain**

Bridge domain name in the format `ctag:bridge-domain.`

*Usage Guidelines*

An empty EPG is an EPG without any network-policy, network-property, or port-property.

An EPG can be created with a port-property and without a port-group. But an EPG cannot be created with a port-group and without a port-property.

ARP suppression is enabled for all the possible broadcast domains(VLAN/BD) on the device.

CEP is handled by replicating all the tenant configuration on the MCT neighbor except for the endpoint configuration, since the endpoint doesn't exist on the MCT neighbor.

The EPG update for a bridge domain-based EPG is similar to an update to a VLAN-based EPG. During a port-group add/delete operation, the logical interface configurations will be created/deleted for the existing ctags, and the corresponding bridge-domains.

During a ctag-range-add or delete operation, the logical interface and bridge-domain configurations are updated on the EPG.

During vrf-add or delete operation, the corresponding L3 configurations will be added or deleted to the EPG.

Event handling sets the corresponding tenant networks to the `cfg-refreshed` state. However, there is no way to re-push the refreshed configuration onto the devices.

*Examples*

The following example is an EPG update for a VLAN Based L3 EPG : port-group-delete

```
efa tenant epg show
================================================================================
Name          :epg11
Tenant        :tenant11
Description   :
Ports         :
POs           : po1115, po1315, po1215
Port Property : switchport mode     : trunk
        :native-vlan-tagging : false
NW Policy     : ctag-range          : 211-212
        : vrf                : blue11
        : l3-vni             : 14191
Network Property [Flags : * - Native Vlan]
+------+--------+---------------+---------+-------------+-------------+
| Ctag | L2-Vni |  Anycast-ip   | BD-name | Dev-state  |  App-state  |
+------+--------+---------------+---------+-------------+-------------+
| 212  | 10003  | 10.10.12.1/24 |         | provisioned | cfg-in-sync|
```

```
+------+--------+---------------+---------+-------------+-------------+
| 211* | 10002  | 10.10.11.1/24 |         | provisioned | cfg-in-sync|
+------+--------+---------------+---------+-------------+-------------+
efa tenant epg update --operation port-group-delete --name epg11 --tenant tenant11 --po
po1315
efa tenant epg show
================================================================================
Name          :epg11
Tenant        :tenant11
Description   :
Ports         :
POs           : po1115, po1215
Port Property : switchport mode     : trunk
        :native-vlan-tagging : false
NW Policy     : ctag-range          : 211-212
        : vrf                 : blue11
        : l3-vni              : 14191
Network Property [Flags : * - Native Vlan]
+------+--------+---------------+---------+-------------+-------------+
| Ctag | L2-Vni |  Anycast-ip   | BD-name | Dev-state  |  App-state  |
+------+--------+---------------+---------+-------------+-------------+
| 212  | 10003  | 10.10.12.1/24 |         | provisioned | cfg-in-sync|
+------+--------+---------------+---------+-------------+-------------+
| 211* | 10002  | 10.10.11.1/24 |         | provisioned | cfg-in-sync|
+------+--------+---------------+---------+-------------+-------------+
```

The following example is an EPG update for a VLAN Based L3 EPG : port-group-add

```
efa tenant epg update --operation port-group-add --name epg11 --tenant tenant11 --po
po1315
efa tenant epg show
================================================================================
Name          :epg11
Tenant        :tenant11
Description   :
Ports         :
POs           : po1115, po1315, po1215
Port Property : switchport mode     : trunk
        :native-vlan-tagging : false
NW Policy     : ctag-range          : 211-212
        : vrf                 : blue11
        : l3-vni              : 14191
Network Property [Flags : * - Native Vlan]
+------+--------+---------------+---------+-------------+-------------+
| Ctag | L2-Vni |  Anycast-ip   | BD-name | Dev-state  |  App-state  |
+------+--------+---------------+---------+-------------+-------------+
| 212  | 10003  | 10.10.12.1/24 |         | provisioned | cfg-in-sync|
+------+--------+---------------+---------+-------------+-------------+
| 211* | 10002  | 10.10.11.1/24 |         | provisioned | cfg-in-sync|
+------+--------+---------------+---------+-------------+-------------+
```

## efa tenant epg show

Shows EPG details for all flags or specific flags.

*Syntax*

**efa tenant epg show** [ **--name** *epg-name***--tenant** *tenant-name*

*Parameters*

**name**

Name of the EPG

**tenant**

Tenant Name

*Examples*

The following is an example of `efa tenant epg show`:

```
efa tenant epg show
================================================================================
Name           :epg11
Tenant         :tenant11
Description    :
Ports          :
POs            : po1115, po1215
Port Property : switchport mode     : trunk
         :native-vlan-tagging : false
NW Policy      : ctag-range          : 211-212
         : vrf                 : blue11
         : l3-vni              : 14191
Network Property [Flags : * - Native Vlan]
+------+--------+---------------+---------+-------------+-------------+
| Ctag | L2-Vni |  Anycast-ip   | BD-name | Dev-state  |  App-state  |
+------+--------+---------------+---------+-------------+-------------+
| 212  | 10003  | 10.10.12.1/24 |         | provisioned | cfg-in-sync|
+------+--------+---------------+---------+-------------+-------------+
| 211* | 10002  | 10.10.11.1/24 |         | provisioned | cfg-in-sync|
+------+--------+---------------+---------+-------------+-------------+
```

## efa tenant epg split

Splits an EPG.

*Syntax*

**efa tenant epg split --tenant** *tenant name* **--source-epg** *source epg name* [**--destination-epg** *name* **--destination-epg-description** *description* **--destination-epg-ctag-range** *ctag range*]

*Parameters*

**--tenant**

Tenant name

**--source-epg**

Source EPG name

**--destination-epg**

Destination EPG name

**--destination-epg-description**

Destination EPG description

**--destination-epg-ctag-range**

Destination EPG ctag range

*Usage Guidelines*

*Examples*

The following is an example of using the `epg split` flag.

```
efa tenant epg show
================================================================================
Name           : epg11
Tenant         : tenant11
Description    :
Ports          :
POs            : po1315, po1215, po1115
Port Property : switchport mode     : trunk
        : native-vlan-tagging : false
NW Policy  : ctag-range          :211-212
               : vrf                : blue11
               : l3-vni             : 14191
Network Property [Flags : * - Native Vlan]
+------+--------+---------------+---------+-------------+-------------+
| Ctag | L2-Vni |  Anycast-ip   | BD-name |  Dev-state  | App-state   |
+------+--------+---------------+---------+-------------+-------------+
| 211  | 10002  | 10.10.11.1/24 |         | provisioned | cfg-in-sync |
+------+--------+---------------+---------+-------------+-------------+
| 212  | 10003  | 10.10.12.1/24 |         | provisioned | cfg-in-sync |
+------+--------+---------------+---------+-------------+-------------+

================================================================================
efa tenant epg split --tenant tenant11 --source-epg epg11 --destination-epg epg12 --
destination-epg-ctag-range 212

efa  tenant epg show
```

```
================================================================================
Name          : epg12
Tenant        : tenant11
Description   :
Ports         :
POs           : po1315, po1215, po1115
Port Property : switchport mode     : trunk
        : native-vlan-tagging : false
NW Policy  : ctag-range          : 212
             : vrf                 : blue11
             : l3-vni              : 14191
Network Property [Flags : * - Native Vlan]
+------+--------+---------------+---------+-------------+-------------+
| Ctag | L2-Vni |  Anycast-ip   | BD-name |  Dev-state  |  App-state  |
+------+--------+---------------+---------+-------------+-------------+
| 212  | 10003  | 10.10.12.1/24 |         | provisioned | cfg-in-sync |
+------+--------+---------------+---------+-------------+-------------+
================================================================================
Name          : epg11
Tenant        : tenant1
Description   :
Ports         :
POs           : po1315, po1215, po1115
Port Property : switchport mode     : trunk
             : native-vlan-tagging : false
NW Policy  : ctag-range          : 211
             : vrf                 : blue11
             : l3-vni              : 14191
Network Property [Flags : * - Native Vlan]
+------+--------+---------------+---------+-------------+-------------+
| Ctag | L2-Vni |  Anycast-ip   | BD-name |  Dev-state  |  App-state  |
+------+--------+---------------+---------+-------------+-------------+
| 211  | 10002  | 10.10.11.1/24 |         | provisioned | cfg-in-sync |
+------+--------+---------------+---------+-------------+-------------+
```

## efa tenant epg delete

Deletes an End Point Group (epg)

*Syntax*

**efa tenant epg delete --name --force** *true/false* **--tenant** *tenant name*

*Parameters*

**--name**

EPG name or comma separated EPG names to be deleted.

**--force**

Forces the deletion of any underling End Point Group tied to the tenant.

Note

ny EPG tied to a tenant should first be deleted before attempting to delete the tenant, as this removes any port level configurations on the switch that are defined by the EPG. Failure to do this will result in EFA returning an error, however using the –force option will override this error and delete the underlying EPG's configuration on the switch

**--tenant**

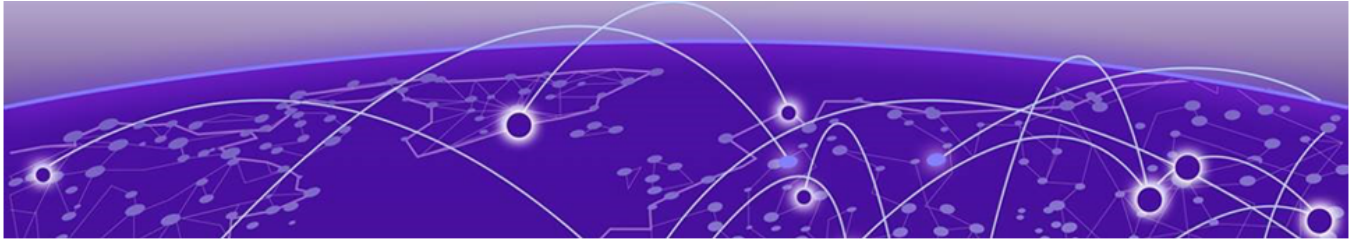
Tenant name

*Usage Guidelines*

*Examples*

The following example deletes epg 11 on tenant 11.

```
efa tenant epg delete --name epg11 --tenant tenant11
```



# Brownfield Fabric Services



## Brownfield Fabric Services Overview

A Brownfield deployment is one in which the installation and configuration of new software must coexist with legacy software systems.

New features for this release include:

- Ability to migrate the devices configured through Embedded Fabric (legacy EFA) to the newer EFA application.
- Ability to migrate the fabric deployed through older version of EFA to newer version of EFA. Here the scenario is user dismantled the older EFA server and installs the newer EFA version or the same EFA version on a different server.

With this feature, you can migrate the fabric being configured fully or partially through the use of the SLX CLI or out-of-band means, provided there are no conflicts with EFA Babric settings. Brownfield scenarios are not supported for Tenant Services. This is planned for future releases. Fabric Service learns and fetches the configuration on the devices through the Inventory Service, performs pre-validation checks, and generates appropriate errors for deviations in the configuration.

## Import or Migrate Embedded Fabric (Legacy EFA) to EFA

As part of migration, all the fabric configuration pushed to the devices through embedded fabric (legacy EFA) are retrieved through the database dump file which is copied from TPVM. Alternatively, you can connect to the TPVM server directly and run the import CLI command to migrate.

The legacy EFA supports only 3 stage CLOS fabrics, so only 3 stage CLOS can be imported to EFA.

In the fabric service back end the migration process involves the following steps:

1. Depending on the location of legacy EFA database file, the file is SCPed from TPVM server to the host where EFA is running first and then uploaded to fabric container at /tmp location.

2. Fetch all the devices that are part of legacy EFA DB and register these switches with inventory service of EFA. If any of the device registration fails, then other devices which were successfully registered are also removed from inventory.
3. Fetch the fabric settings from legacy EFA DB and create a new fabric (with user provided name) in EFA. If fabric already exists with devices added to it, then an appropriate error is returned. Ensure that the new fabric doesn't have any devices added to it.
4. Devices are added to the fabric and fabric pre-validation is done. If fabric device addition or pre-validation fails, an error is displayed, also fabric created previously is deleted and devices are removed from the fabric and deregistered from inventory. Configurations on the device are not cleaned and all the configurations that existed prior to import are retained.
5. Post-validation phase is run where fabric configuration is compared with configuration fetched from inventory. If the data does not match, an error is displayed.
6. On successful migration, devices are added in the fabric with provisioning state as "CFG-READY".
7. Issue "configure fabric" to move all the devices to "CFG-IN-SYNC" state.

## efa fabric import

Imports a fabric.

*Syntax*

**efa fabric import**

*Command Default*

If efa.db (Embedded fabric DB file) is available on the host where EFA application is run then it can be imported by providing the local file path. In this case server IP and server credentials are not required.

If efa.db is present on the TPVM user has to provide the db file path on TPVM for e.g /var/efa/efa.db. Also, user needs to enter the TPVM server IP address and server credentials.

If db file path is incorrect or if the application is unable to open the db file then an appropriate error is returned.

If connection to TPVM server fails or unable to locate the DB file using the remote file path then an appropriate error is returned to the user.

*Parameters*

**db-file**

EFA dbfile name including full path

**fabric-name**

Name with which fabric will get imported in the EFA

**server-ip**

Remote server IP

**username**

Remote server username

**password**

Remote server password

*Examples*

Local DB file import

```
efa fabric import --db-file /root/efa/efa.db --fabric-name default

Successfully imported fabric to EFA
```

Remote DB file import

```
efa fabric import --db-file /var/efa/efa.db --fabric-name efa_fabric -server-ip
10.24.85.180 -username admin-password password

Successfully imported fabric to EFA
```

```
efa fabric import --db-file /root/efa/efa.db --fabric-name default

 Leaf Device 10.25.225.172 not connected to Spine Device
        10.24.80.137 [Failed]
```

```
Spine Device 10.24.80.137 not connected to Leaf Device
       10.25.225.172 [Failed]
```

# Pre-validation of Configuration

This section covers the use case where the devices have a preexisting configuration which may have been configured either through Embedded Fabric/CLI/EFA or some out-of-band means.

When devices with preexisting configuration are added to EFA, fabric service performs certain validations before adding devices to the EFA fabric. If any of the configuration that is retrieved from the devices does not fall under the fabric settings range, an error is displayed. You can perform corrective actions to add such devices to the EFA fabric.

## Global Device Configuration

| Use Case | Valid | Invalid |
|---|---|---|
| L2 MTU | If the value fetched from the device is same as that configured in fabric settings. | If the value does not match, an error is displayed. |
| IP MTU | If the value fetched from the device is same as that configured in fabric settings. | If the value does not match, an error is displayed. |
| ASN | If the value fetched from the device is within the ASN range configured in fabric settings. | If the value is out of range of what is configured in fabric settings.<br>If the value is conflicting with existing device which is already added to Fabric. |

## Interface Configuration

| Use Case | Valid | Invalid |
|---|---|---|
| Interface IP Address | Received an IP address within “Link IP range” of fabric settings. | • If IP address received is out of range, a validation error is displayed.<br>• If IP address received is within the range but is already in use/reserved by fabric, a validation error is displayed. |
| Loopback Interface ID | Loopback Interface ID is reconciled | |
| VTEP Loopback Interface ID | VTEP Loopback interface ID is reconciled | |
| Loopback Interface IP Address | Loopback IP address is within “Loopback IP range” of fabric settings. | If the loopback IP address is out of range of “Loopback IP range” of fabric settings, an error is displayed. |

| Use Case | Valid | Invalid |
|---|---|---|
| VE IP address | VE IP address is within “MCT Link IP Range” of fabric settings. | • If IP address received is out of range, a validation error is displayed.<br>• If IP address received is within the range but is already in use/reserved by fabric, a validation error is displayed. |
| Static IP Route (Applicable for SLX 9540 and SLX 9640) | If the IP route is same as fabric intended configuration. | If the nexthop does not point to VE IP. |

## MCT Configuration

MCT peer and VE validations are platform specific.

| Use Case | Valid | Invalid |
|---|---|---|
| MCT Cluster Name | MCT Cluster Name is different from fabric name in fabric properties. MCT Cluster Name learned from device is used while configuring the device, so that the cluster name is reconciled. | |
| Cluster Control VLAN | Cluster Control VLAN matches with “Control VLAN” of fabric settings. | If the VLAN does not match, an error is displayed. |
| Cluster Control VE | Cluster Control VE matches with “Control VE” of fabric settings | If the VE does not match, an error is displayed. |
| MCT Peer IP | Peer IP address is within IP range of “MCT Link IP range” of fabric settings. | If Peer IP address is out of IP range, an error is displayed. |
| MCT Peer Interface | Peer Interface type matches with fabric settings and ID is reconciled. | |

## Overlay Gateway Configuration

| Use Case | Valid | Invalid |
|---|---|---|
| Overlay Gateway Name | Gateway Name is reconciled. | |
| VNI Auto Map | VNI auto mapping setting configured on the device matches with fabric settings. | If gateway is in activated state and VNI Auto Map setting is different from the fabric settings, an error is displayed. |
| Overlay Gateway Interface | Gateway Interface, for example loopback 2 is reconciled. | |

## EVPN Configuration

| Use Case | Valid | Invalid |
|---|---|---|
| EVPN Name | EVPN Name is reconciled | |
| MAC Aging Timeout (check based on device capability, applies to SLX 9140 and SLX 9240) | Field value is overwritten by the fabric settings value. | |
| MAC Aging Conversation Timeout (check based on device capability, applies to SLX 9140 and SLX 9240) | | |
| MAC Move Limit (check based on device capability) | | |
| ArpAgingTimeout (check based on device capability) | | |
| Duplicate Mac Timer | | |
| Duplicate MAC Timer MAX Count | | |

## BGP Configuration

To pre-validate the BGP configuration, the BGP configuration must be prepared similar to the add device phase. Once the BGP configuration is computed, the configuration retrieved from the device is compared against it.

| Use Case | Valid | Invalid |
|---|---|---|
| Router ID | If the generated router id matches the one received from the device. | If the router id does not match, an error is displayed. |
| BFD Enable/Disable | If the BFD value from device matches the one that is computed from fabric settings. While configuring the fabric, the values computed by fabric service override the ones on the device. | NA |
| BFD Tx/Rx Timer Values | If the values from device match with the ones that are generated. While configuring the fabric, the values computed by fabric service override the ones on the device. | NA |
| Network Address | If the value is within “Loopback IP Range” of fabric settings or matches the computed value. | If the value is out of range or clashes with another IP neighbor already stored in fabric DB, an error is displayed. |

| Use Case | Valid | Invalid |
|---|---|---|
| EVPN Neighbor IP Address | If the neighbor IP address falls in range of “Link IP range” of fabric settings. There may be a case where the neighbor IP address is valid but neighbor is not part of fabric. You can ignore such validation as that configuration is a no-op for fabric. | If the value is out of range or clashes with another IP neighbor already stored in fabric DB, an error is displayed. |
| Remote ASN | If the ASN is within the range of fabric settings and not already in use by another neighbor. | If the ASN is out of range or already reserved. |
| Peer group name | If the peer group name matches the peer group that is computed. | If the value does not match, an error is displayed. |

## BGP Tables

The following BGP tables help in computing the diffs for the events from the inventory service.

- Router BGP table
- BGP peer group table
- BGP IP address family table
- BGP IP neighbor address table
- BGP EVPN address family table
- BGP EVPN neighbor address table

All BGP tables handle the DB migration so that upgrade from older EFA to newer EFA works.

For more information on the attributes of each table, refer to Database schema section or fabric_schema.sql file.

## BGP Events

The following BGP events from inventory service are handled as part of event handling.

### BGP Router Delete

When router BGP delete message is received, fabric passes through all the IP and EVPN neighbors, peer group tables and the entries corresponding to the device for which router BGP delete message is received and mark the entries as ‘create’ to configure the router BGP and its related neighbors on the device.

### BGP IP Neighbor Delete

When BGP IP neighbor delete message is received, fabric passes through all the IP neighbors deleted and which exists in fabric database for a given neighbor IP or remote ASN and mark the entries as ‘create’ to configure the deleted IP neighbors on the device.

### BGP IP Neighbor Update

When BGP IP neighbor update message is received, fabric passes through all the IP neighbors matching the neighbor IP for the device in the database. If any of the fabric managed attribute in the IP neighbor table is changed, fabric marks the entries as 'update' and pushes the configuration back to the device.

### BGP EVPN Neighbor Delete

When BGP EVPN neighbor delete message is received, fabric passes through all the EVPN neighbors deleted and which exists in fabric database for a given neighbor IP or remote ASN and mark the entries as 'create' to configure back the deleted EVPN neighbors on the device.

### BGP EVPN Neighbor Update

When BGP EVPN neighbor update message is received, fabric passes through all the EVPN neighbors matching the neighbor IP for the device in database. If any of the fabric managed attribute in the EVPN neighbor table is changed, fabric marks the entries as 'update' and pushes the configuration back to the device.

### Peer Group Delete

Peer Group Delete message is received only when there are no IP/EVPN neighbors associated with it. If there are no IP/EVPN neighbors associated with it, fabric marks the Peer Group as 'delete'.

### Peer Group Update

When Peer group attributes such as BFD and remote ASN change, inventory sends a peer group update message. The fabric processes this message and checks if the peer group exists in the database. If the peer group exists and there are changes to the attributes, the fabric pushes the peer group configuration with fabric intended configuration back to the device.

### BGP IP Address Family Delete

BGP IP Address Family Delete message is received when the IP address-family for a device is deleted through CLI or out-of-band means. When fabric receives this message, it passes through all the IP neighbors associated with that address-family and marks the entries as 'create config' to restore all the deleted IP neighbors on the device.

### BGP EVPN Address Family Delete

BGP EVPN Address Family Delete message is received when the EVPN address-family for a device is deleted through CLI or out-of-band means. When fabric receives the message, it passes through all the EVPN neighbors associated with that address-family and marks the entries as 'create config' to restore all the deleted EVPN neighbors on the device.

## Limitations

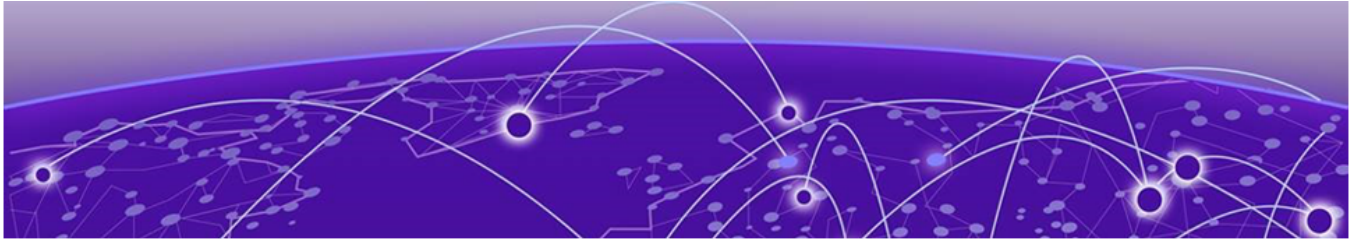
- Migration of older version of EFA to newer version of EFA is similar to adding devices to newer fabric and configuring fabric. If there are no changes in fabric configuration, devices migrate to the newer

version of EFA successfully. Limitation on pre-validations and post-validations apply. Similar limitations apply to fabric configured through CLI or out-of-band means.

- As part of pre-validations, ASN, IP address of interface, loopback interface IP, and MCT VE IP validations are done. Other pre-validations mentioned in Pre-validation of Configuration are NOT handled.
- Post-validation is NOT handled.
- If you enter a fabric name that already exists with added devices, import fails.
- If the existing fabric type mismatches with the fabric to be imported, import fails.
- If the fabric name entered is “default” and import fails, the fabric properties rollback to the original state.
- Non-CLOS Embedded fabric import is not supported.
- If the user explicitly changes the device credentials which are not in-sync with Embedded fabric database, the device registration fails.

The following are not supported:

- Devices with preexisting configuration and configured through out-of-band means such as CLI and not through Embedded fabric.
- Devices configured through Embedded fabric and firmware upgraded to SLX 20.1.1 release containing simplified MCT feature. Import fails if you perform firmware upgrade before importing. Import the fabric first and then perform firmware upgrade.
- Brownfield configurations reconcilement.



# Logging and ELK Integration

Logging and ELK Integration on page 101

## Logging and ELK Integration

In the EFA ecosystem, ELK (Elasticsearch, Logstash, Kibana) is implemented in the same network as the Application stack.

### URLs to access the ELK stack

Elasticsearch: `http://<host_ip>:9200`

Kibana: `http://<host_ip>:5601`

### Sample log

```
@timestamp:December 13th 2018, 22:18:12.929 source
:/var/log/dcapp/fabric/fabric.log offset:513,560 message:{"level":"info","msg":
"Fabric service Health status OK ","time":"2018-12-12T18:03:04Z"} prospector.type:log
json.level:info json.msg:Fabric service Health status OK json.time:2018-12-12T18:03:04Z
beat.name:5d2a1a83ed27 beat.hostname:5d2a1a83ed27 beat.version:6.2.2 _id:
YdN4qGcBzheJSFbXB7U5 _type:doc _index:filebeat-6.2.2-2018.12.13 _score:1
```

**Table 6: Log tags**

| Tag | Description |
|---|---|
| source | Provides the information about which service the log belongs to. |
| level | Provides the level of log, for example, whether a log is “Error” or “Info” or “Warning”. |
| _id | Each log is numbered with a unique ID. |
| json.msg | Contains details about the operation or error message in this field. |
| timestamp | Details about when the operation was performed. Gives exact time of log creation. |

### Infra level

```
# docker logs k3s
```

To obtain a *<container-id>*, run **`docker ps`**.

The ELK stack is deployed as part of the deployment, which helps analyze the application-specific logs. Logs for the services are available in the host at `/var/log/dcapp`.

### Application level

The ELK stack helps analyze the application-specific logs. Logs for the services are available in the host at `/var/log/dcapp`.

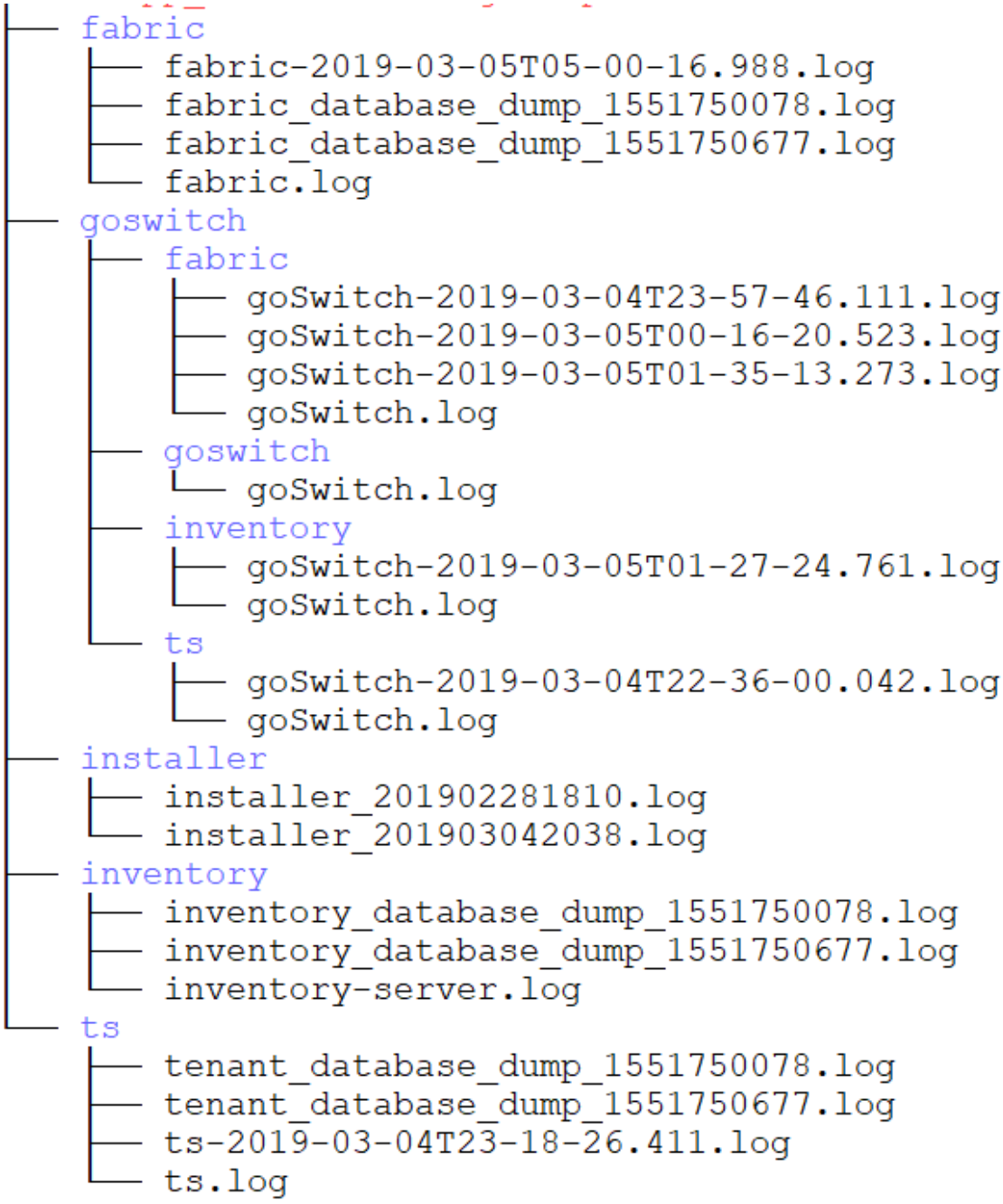



**Figure 13: Application level log**

Logs are visualized on a Kibana dashboard. The following is an example.

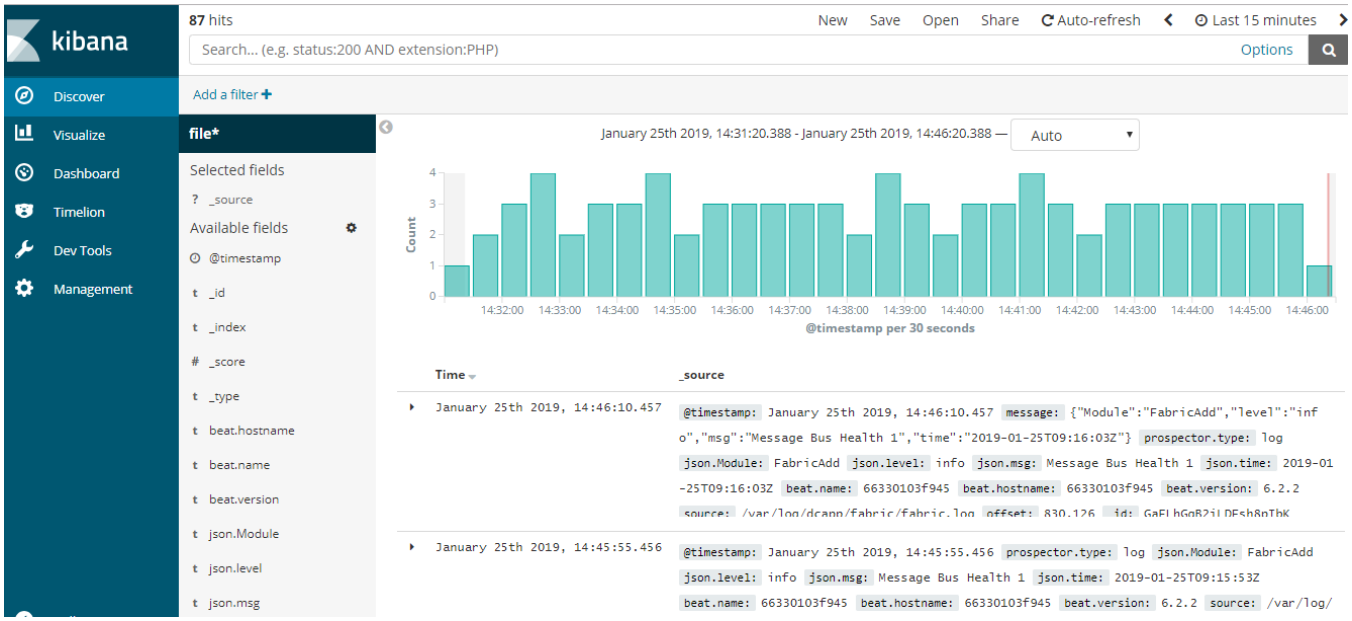



**Figure 14: Kibana discover**

# SLX-OS Firmware Download with Maintenance Mode



## SLX-OS Firmware Download with Maintenance Mode

With the firmware download with maintenance mode feature, you can download firmware on one or more devices in the IP Fabric with the smallest possible disruption to data path traffic. Both CLOS and Non-CLOS fabrics are supported. Maintenance mode feature is supported only on SLX devices running SLXOS 20.1.x.

New features for this release include:

- Firmware download with maintenance mode
  - Asynchronously launched operation
  - Sanity and pre-install script check
  - Configuration to set convergence timeout, enable, and disable
  - Persist the running configuration so that device configuration and maintenance mode configuration is preserved after reboot

    - Firmware download with no commit option to allow restoration of firmware to the previous version
    - During maintenance mode, no configurations will be allowed on the device.
- Firmware host registration, with support for register, update, delete, and list operations
- Firmware download preparation, with support for add, remove, and list operations
- Firmware download execute to start firmware download with maintenance mode asynchronous operation
- Firmware download show, to display a table of devices in the Fabric and their corresponding status

### Limitations

- Switch firmware must be SLX-OS 20.1.x or later to support firmware download with maintenance mode for a hitless firmware upgrade.
- This feature assumes an existing host which contains SLX-OS firmware images ready to be downloaded from.
- This feature should not be launched on the same switch where EFA TPVM is deployed.

### Supported Devices

The SLX-OS firmware download with maintenance mode is supported on the following SLX devices running SLX-OS 20.1.x.

- SLX 9540
- SLX 9640
- SLX 9150-48Y
- SLX 9150-48T
- SLX 9250

## Hitless Firmware Upgrade

A hitless firmware upgrade utilizes the maintenance mode switch feature to gracefully divert traffic away from the switch to alternate paths. The switch can be put into maintenance mode and a firmware upgrade can be performed. The switch can safely be rebooted and the new firmware activated without traffic loss. The switch can be taken out of maintenance mode and allow traffic on the newly upgraded switch.

### Upgrade Super-Spine Firmware in CLOS

1. Enabling maintenance mode on a super-spine involves BGP. The graceful_shutdown parameter will be sent to all the super-spine's underlay neighbors (all connected spines). Each neighbor will process the graceful_shutdown and refresh their routes to use the alternate path. Maintenance mode is enabled on the first super-spine and traffic is diverted over to the second super-spine.
2. The running-configuration is saved on the first super-spine to preserve all current configurations including the maintenance mode enable configuration.
3. The firmware on the first super-spine can now be upgraded and rebooted for firmware activation without traffic loss.

4. Once the new firmware is activated, maintenance mode can be disabled. The graceful_shutdown parameter is removed from all the underlay neighbors and traffic to the first super-spine is allowed again.
5. The running-config is persisted again to ensure the maintenance mode disabled state is retained.

   The same process can be carried out on the second super-spine to upgrade the firmware without traffic loss.

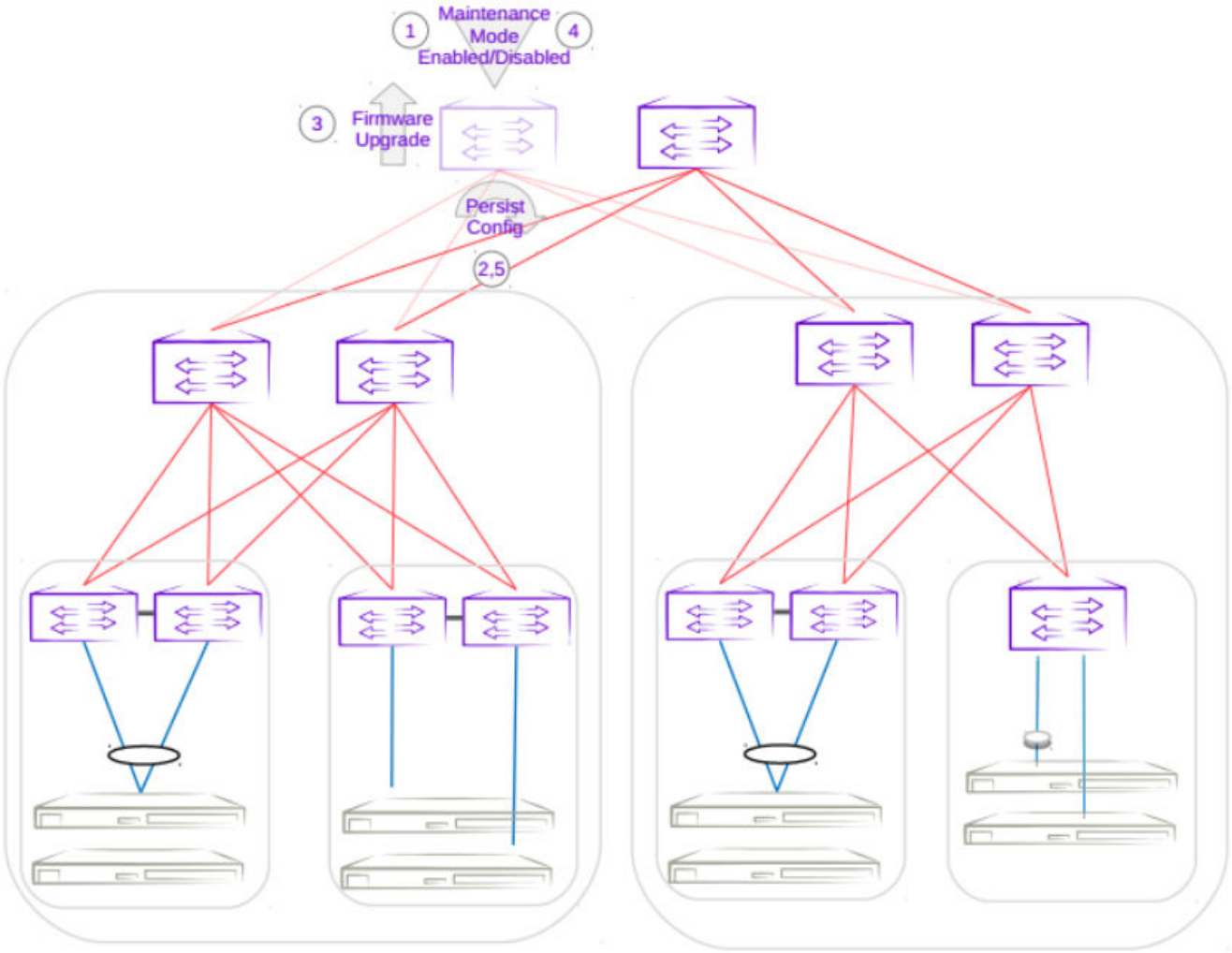



**Figure 15: First super-spine firmware upgrade with maintenance mode**

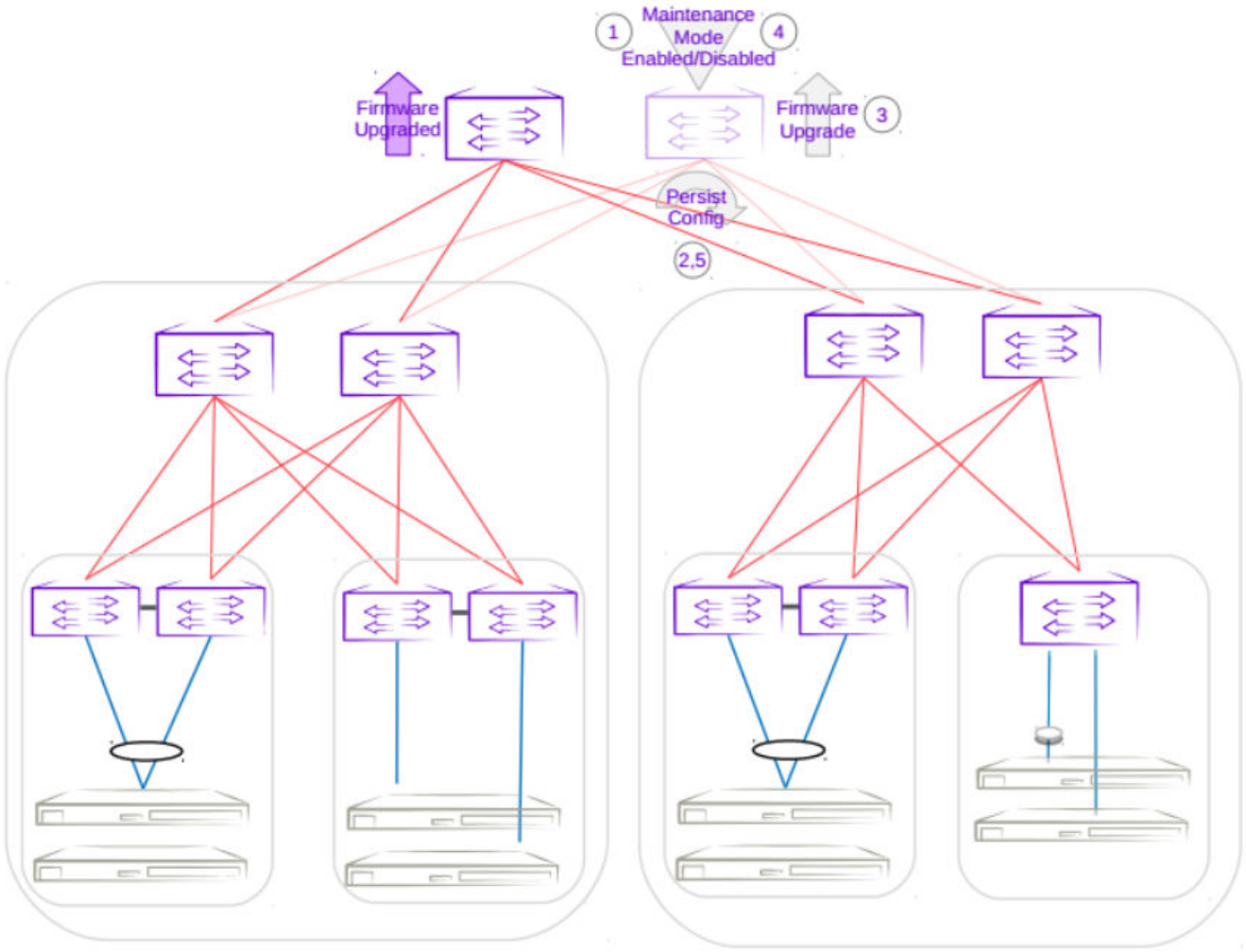



**Figure 16: Second super-spine firmware upgrade with maintenance mode**

## Upgrade Spine in CLOS

1. Enabling maintenance mode on a spine also involves BGP. The graceful_shutdown parameter will be sent out to all the spine's underlay neighbors (all leafs in the pod and super-spines). The neighbors will no longer send traffic to the first spine going into maintenance mode and redirect traffic to an alternate path.
2. The running-configuration is saved on the first spine to preserve all current configurations including the maintenance mode enable configuration.
3. The firmware on the first spine can now be upgraded and rebooted for firmware activation without traffic loss.
4. After the firmware is upgraded, the maintenance mode is disabled to allow traffic again through the upgraded spine.
5. The running-config is saved again to ensure the maintenance mode config remains disabled.

   The same process can be carried out on the second spine to upgrade the firmware without traffic loss.

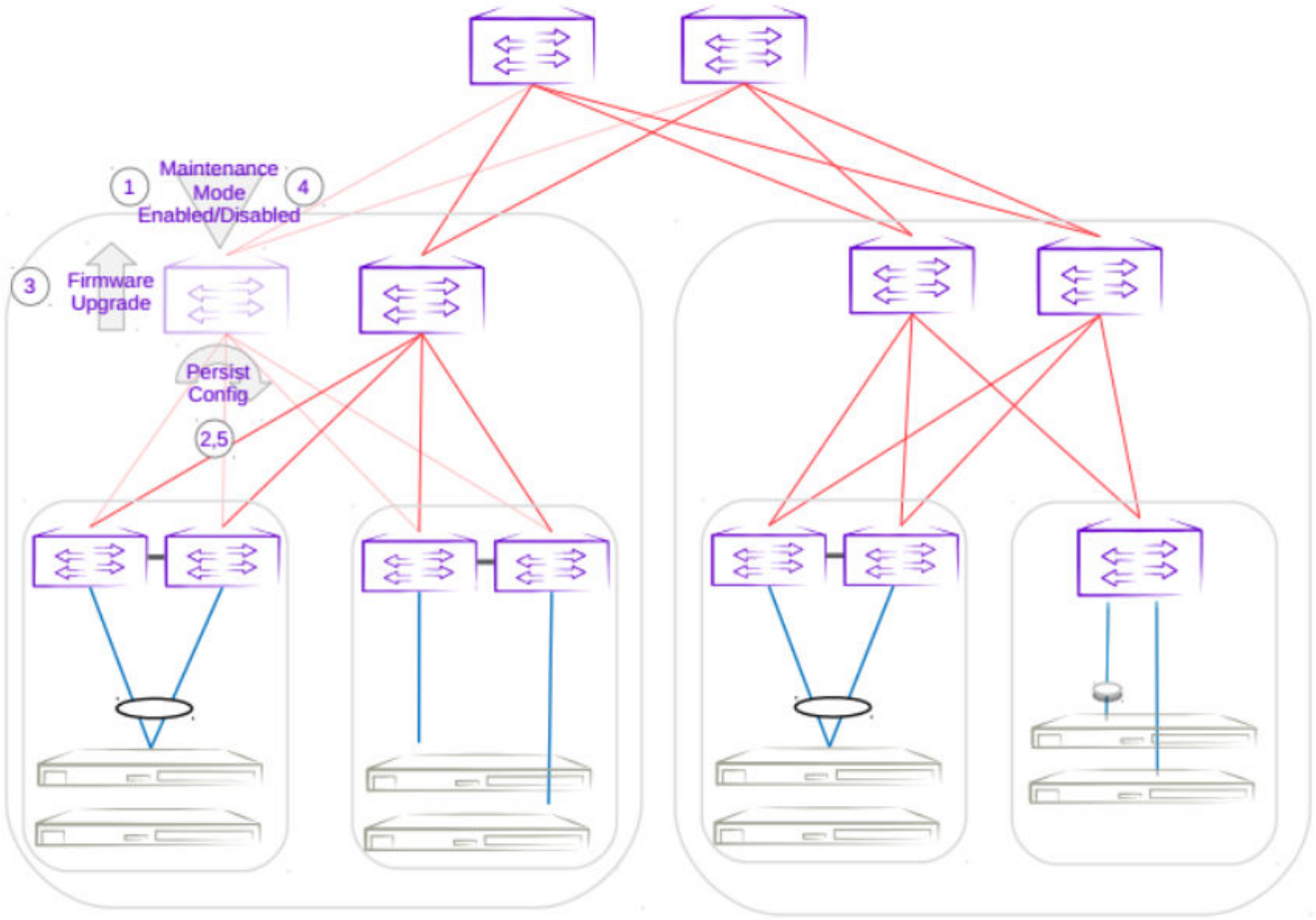



**Figure 17: First spine firmware upgrade with maintenance mode**

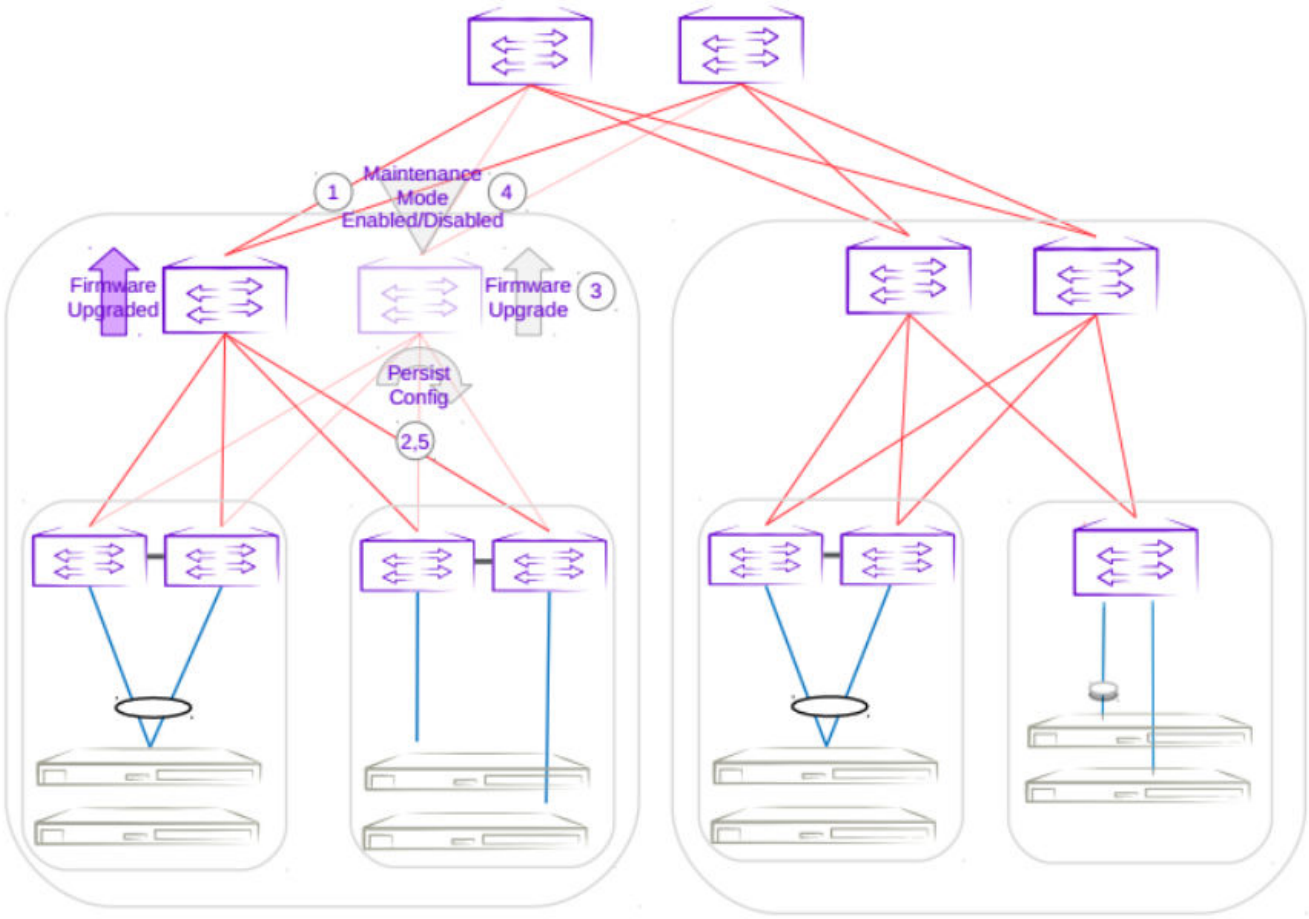



**Figure 18: Second spine firmware upgrade with maintenance mode**

## Upgrade MCT Leaf Pair with Dual Homed Servers in CLOS

1. Enabling maintenance mode on an MCT leaf involves BGP and MCT/NSM. The graceful_shutdown parameter will be sent out to all the leaf's underlay neighbors (all spines in the pod). The neighbors will no longer send traffic to the MCT leaf going into maintenance mode and redirect traffic from spines to the peer MCT leaf. MCT will instruct the peer leaf to become the designated forwarder, ICL will be shut down, and CCE ports for clients will also be shut down. Traffic from dual-homed servers will be redirected to the peer leaf. With maintenance mode enabled, traffic is completely redirected to the peer leaf.
2. The running-configuration is saved on the first MCT leaf to preserve all current configurations including the maintenance mode enable configuration.
3. The firmware on the MCT leaf can now be upgraded and rebooted for firmware activation without traffic loss.
4. After the firmware is upgraded, the maintenance mode is disabled to allow traffic again through the upgraded MCT leaf.
5. The running-config is saved again to ensure the maintenance mode config remains disabled.

   The same process can be carried out on the second MCT leaf to upgrade the firmware without traffic loss.

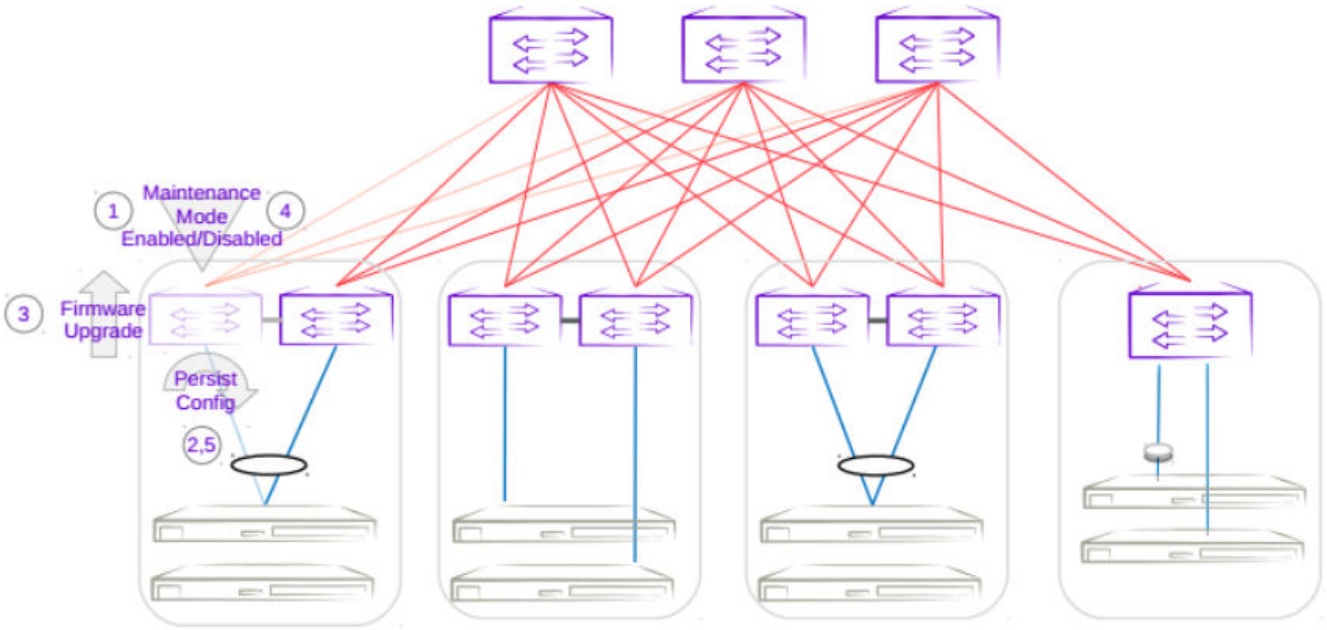



**Figure 19: First MCT leaf firmware upgrade with maintenance mode**

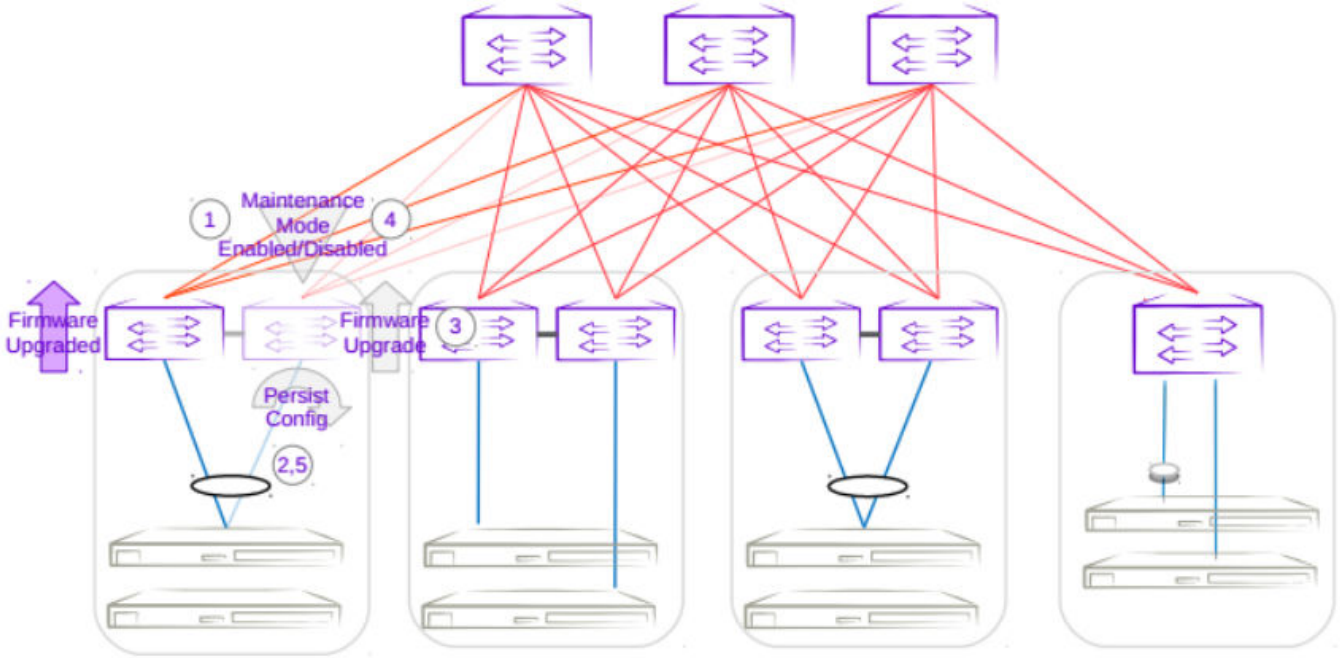



**Figure 20: Second MCT leaf firmware upgrade with maintenance mode**

## Upgrade Three Rack Centralized in Non-CLOS

1. Enabling maintenance mode on one of the leafs in the centralized MCT leaf pair follows the same behavior as the MCT leaf pair in CLOS topology. The only difference is the iBGP L3 backup link between MCT leaf pairs. Maintenace mode will result in the traffic being redirected to the peer leaf in the centralized MCT leaf pairs.

2. The running-configuration is saved on the first MCT leaf to preserve all current configurations including the maintenance mode enable configuration.
3. The firmware on the MCT leaf can now be upgraded and rebooted for firmware activation without traffic loss.
4. After the firmware is upgraded, the maintenance mode is disabled to allow traffic again through the upgraded MCT leaf.
5. The running-config is saved again to ensure the maintenance mode config remains disabled.

   The same process can be carried out on the second MCT leaf to upgrade the firmware without traffic loss.

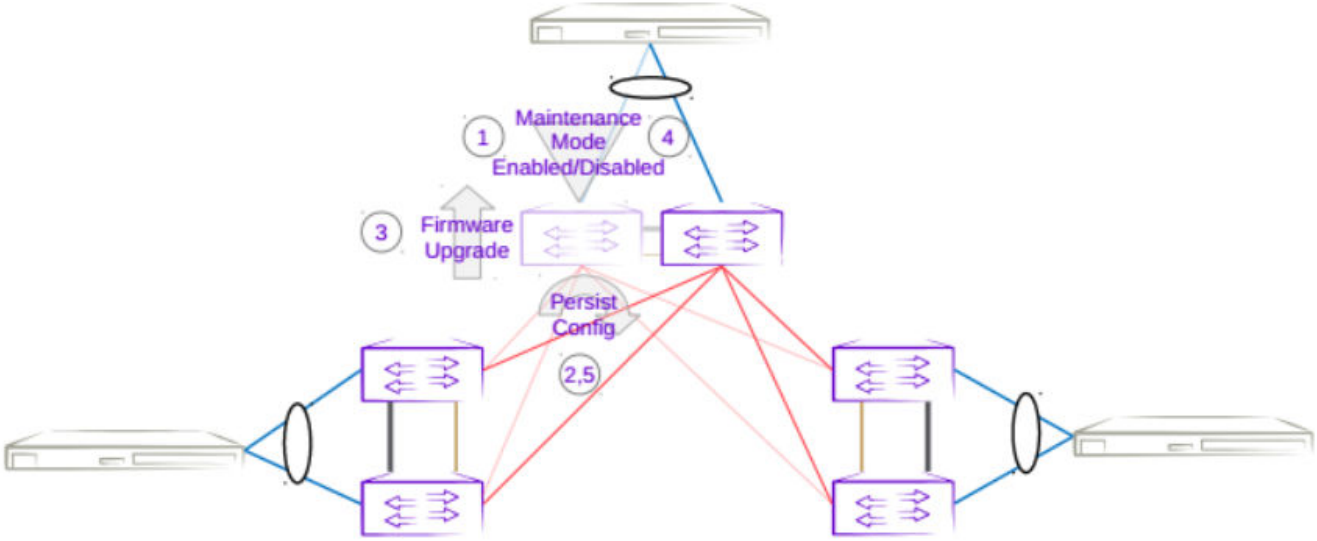



**Figure 21: Three rack centralized first MCT leaf firmware upgrade with maintenance mode**

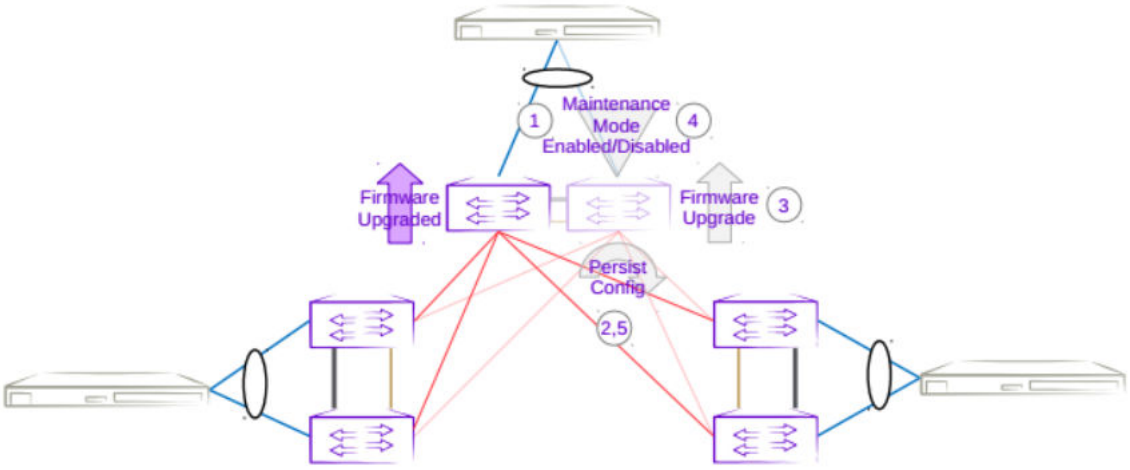



**Figure 22: Three rack centralized Second MCT leaf firmware upgrade with maintenance mode**

## Upgrade Three Rack Ring in Non-CLOS

1. Enabling maintenance mode on one of the leafs in a three rack ring MCT leaf pair follows the same behavior as the MCT leaf pair in CLOS topology. The only difference is the iBGP L3 backup link between MCT leaf pairs. Maintenace mode will result in the traffic being redirected to the peer MCT leaf.
2. The running-configuration is saved on the first MCT leaf to preserve all current configurations including the maintenance mode enable configuration.
3. The firmware on the MCT leaf can now be upgraded and rebooted for firmware activation without traffic loss.
4. After the firmware is upgraded, the maintenance mode is disabled to allow traffic again through the upgraded MCT leaf.
5. The running-config is saved again to ensure the maintenance mode config remains disabled.

   The same process can be carried out on the second MCT leaf to upgrade the firmware without traffic loss.

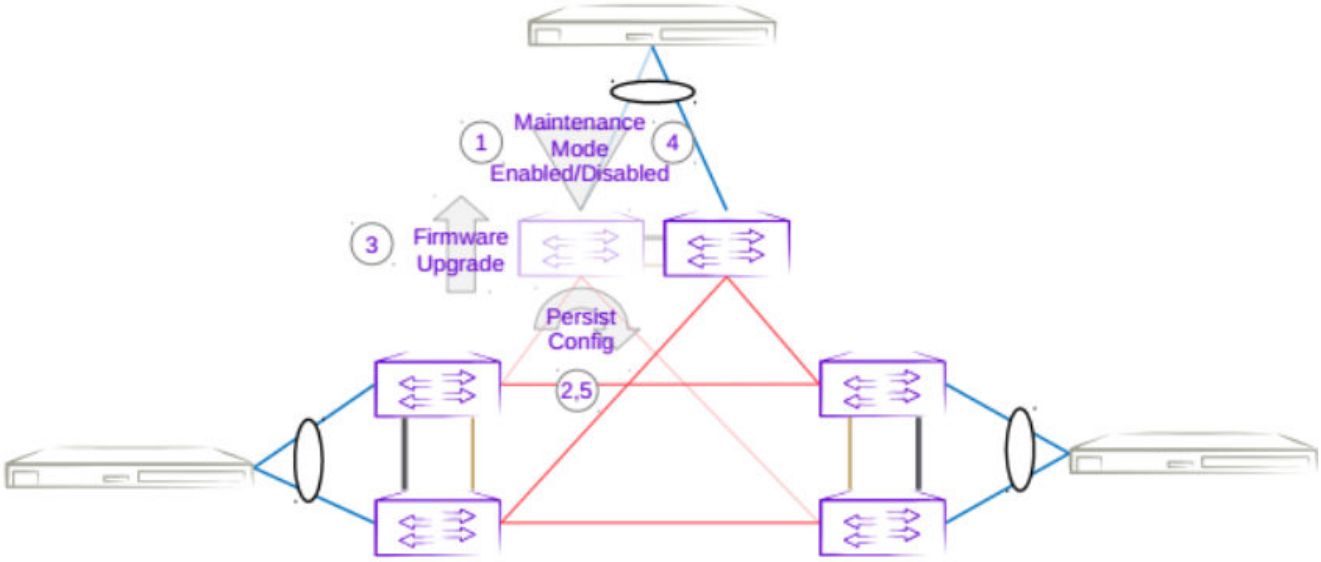



**Figure 23: Three rack ring first MCT leaf firmware upgrade with maintenance mode**

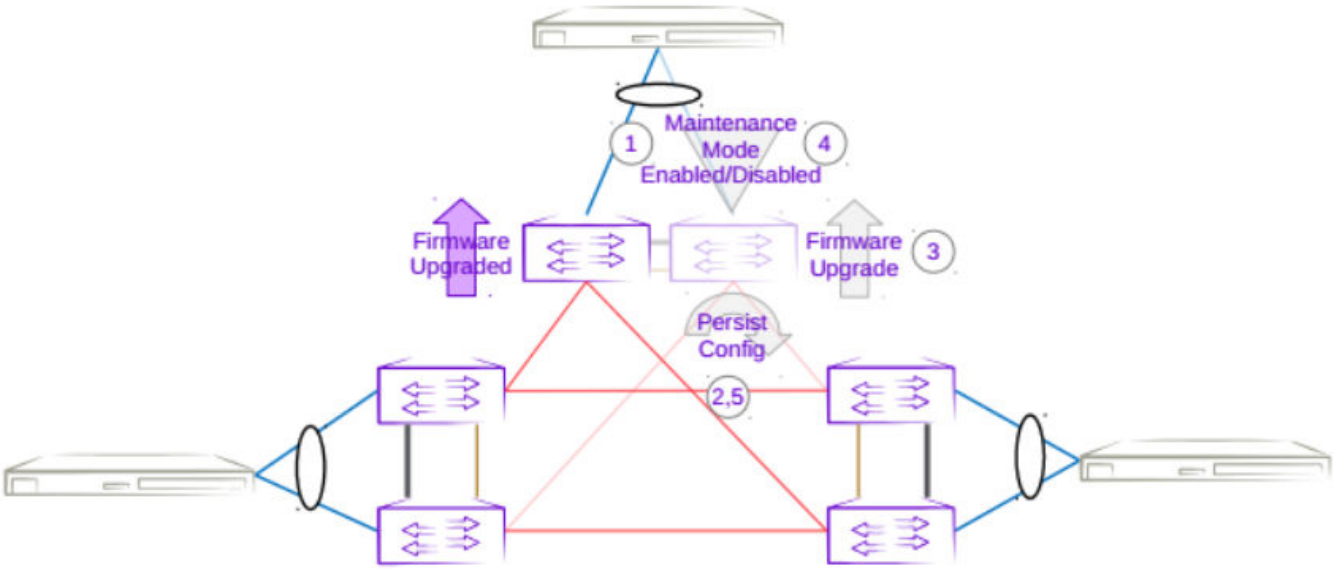



**Figure 24: Three rack ring second MCT leaf firmware upgrade with maintenance mode**

# Traffic Loss

## Single Leaf

Traffic loss is expected when upgrading a single leaf which is not in an MCT pair. Since there are no alternate paths for the single leaf, maintenance mode will not be enabled. Only the configuration will be persisted, and a firmware upgrade will be carried out. A traffic loss warning will be flagged when upgrading a single non-MCT leaf.

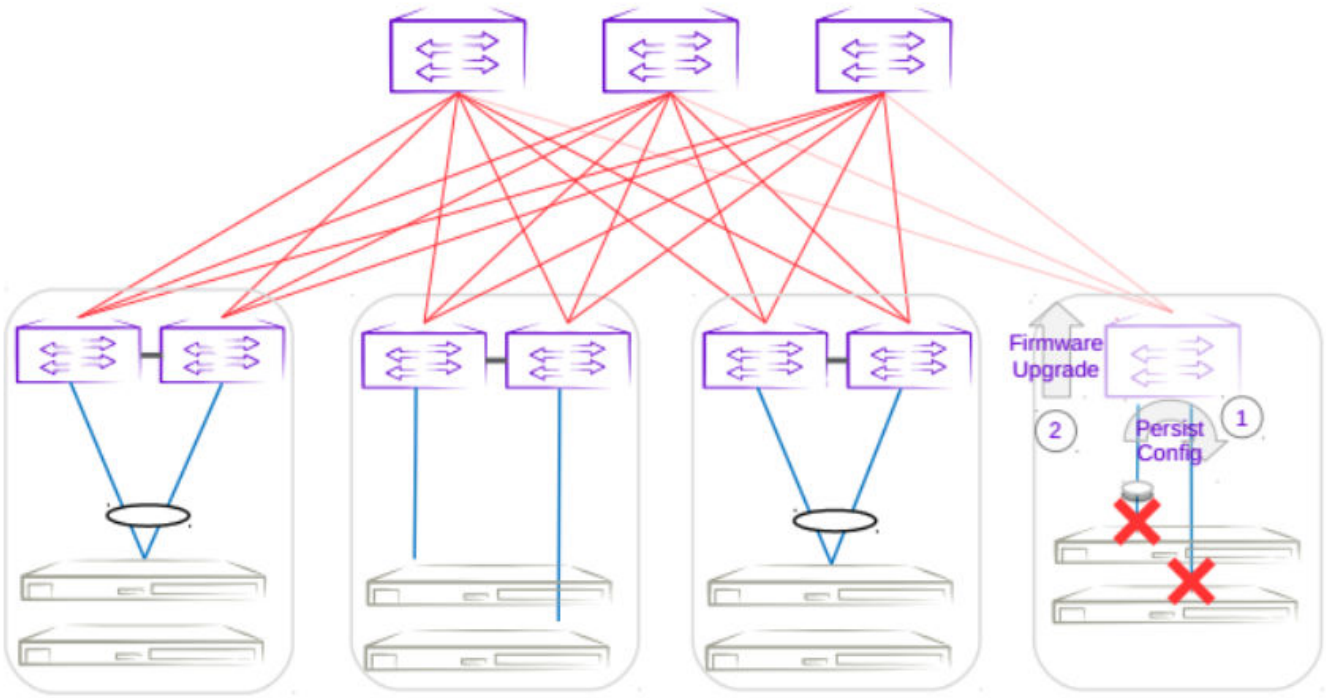



**Figure 25: Single leaf traffic loss**

## Single-Homed Server

Traffic loss is also expected for any singled-homed server. Detecting single-homed servers are not in the scope of this feature so a generic warning will be provided at the start of a firmware download.

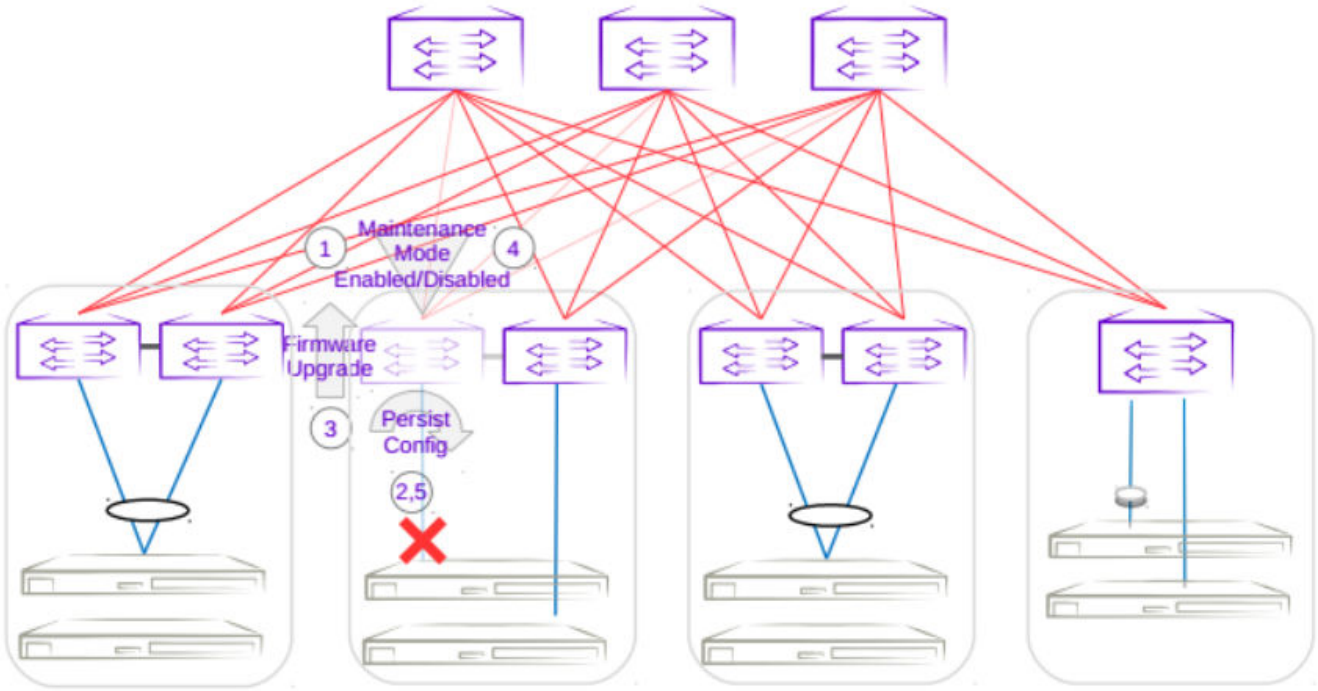



**Figure 26: Single-homed server traffic loss**

## Non-Redundant Spine or Super-Spine

This is not a typical deployment, but traffic loss is expected in this scenario. Since no alternate paths exist for non-redundant switches, maintenance mode will not be enabled for this case. A traffic loss warning will be flagged when upgrading non-redundant switches.

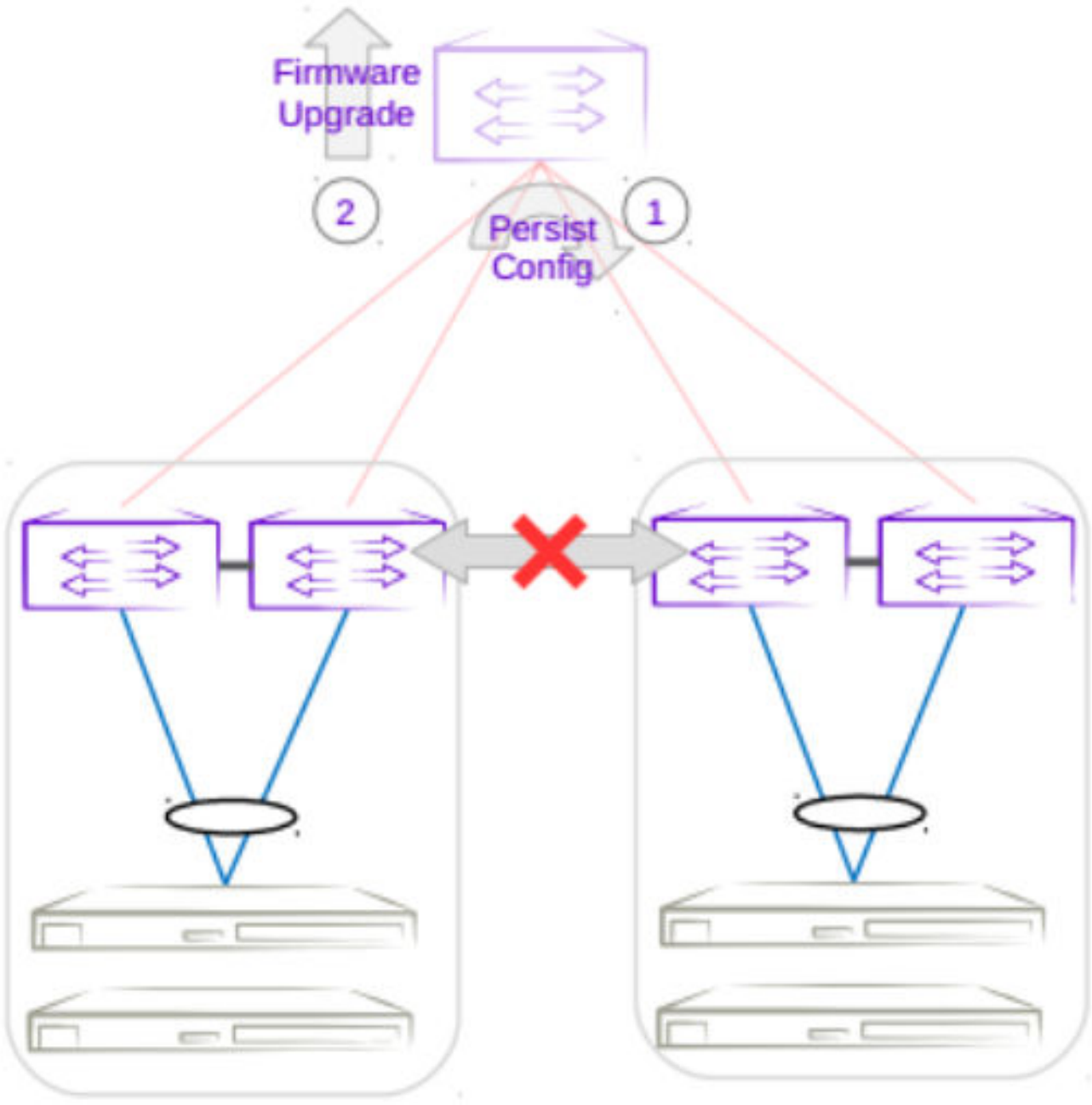



**Figure 27: Non-redundant spine traffic loss**

## Firmware Download with Maintenance Mode Operations

The following operations are performed per device when a firmware download is executed.

1. Firmware Sanity Check
2. Maintenance Mode Enable
3. Persist Config
4. Firmware Download

5. Maintenance Mode Disable
6. Persist Config
7. Firmware Commit

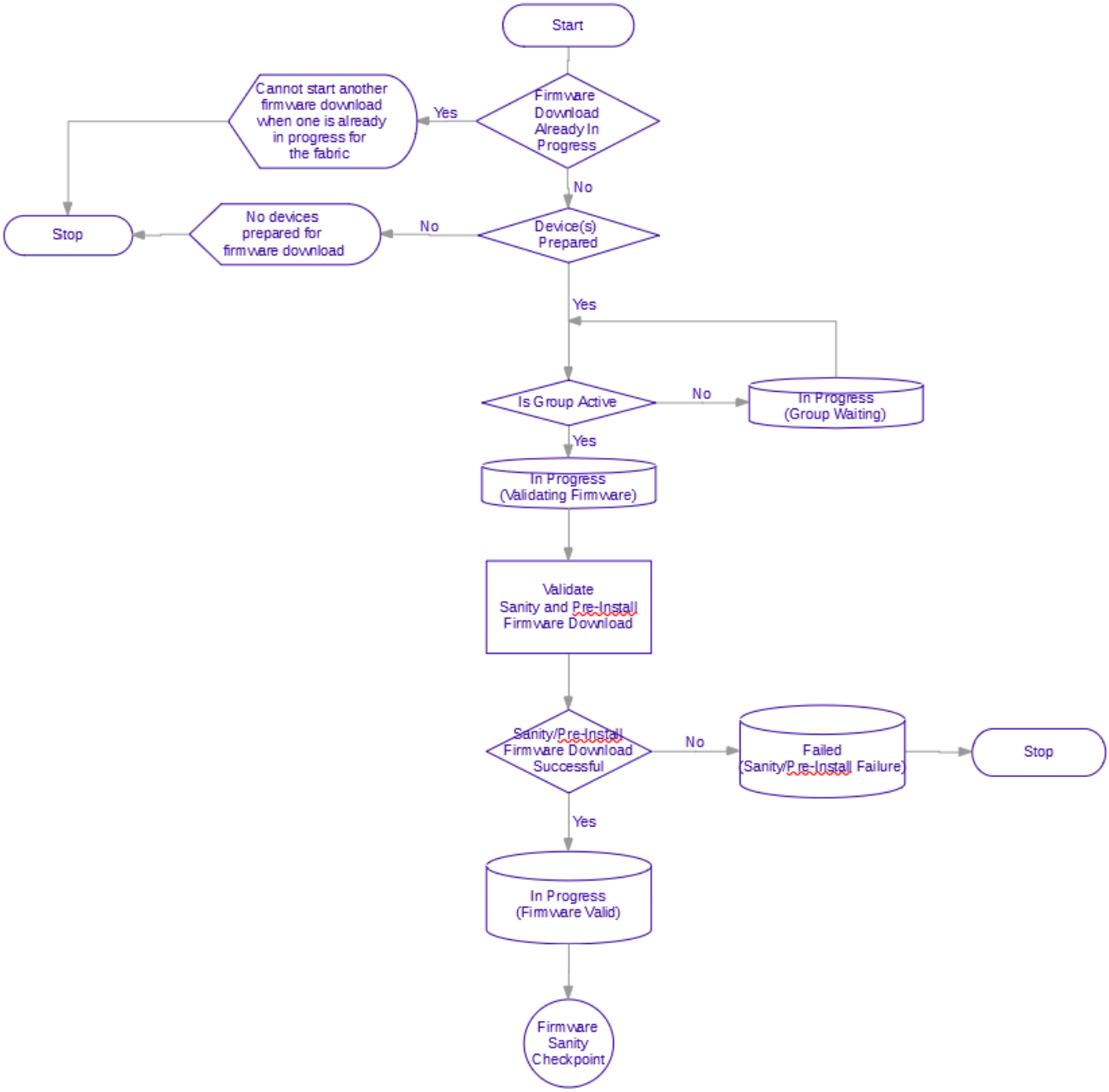



**Figure 28: Firmware sanity check flowchart**

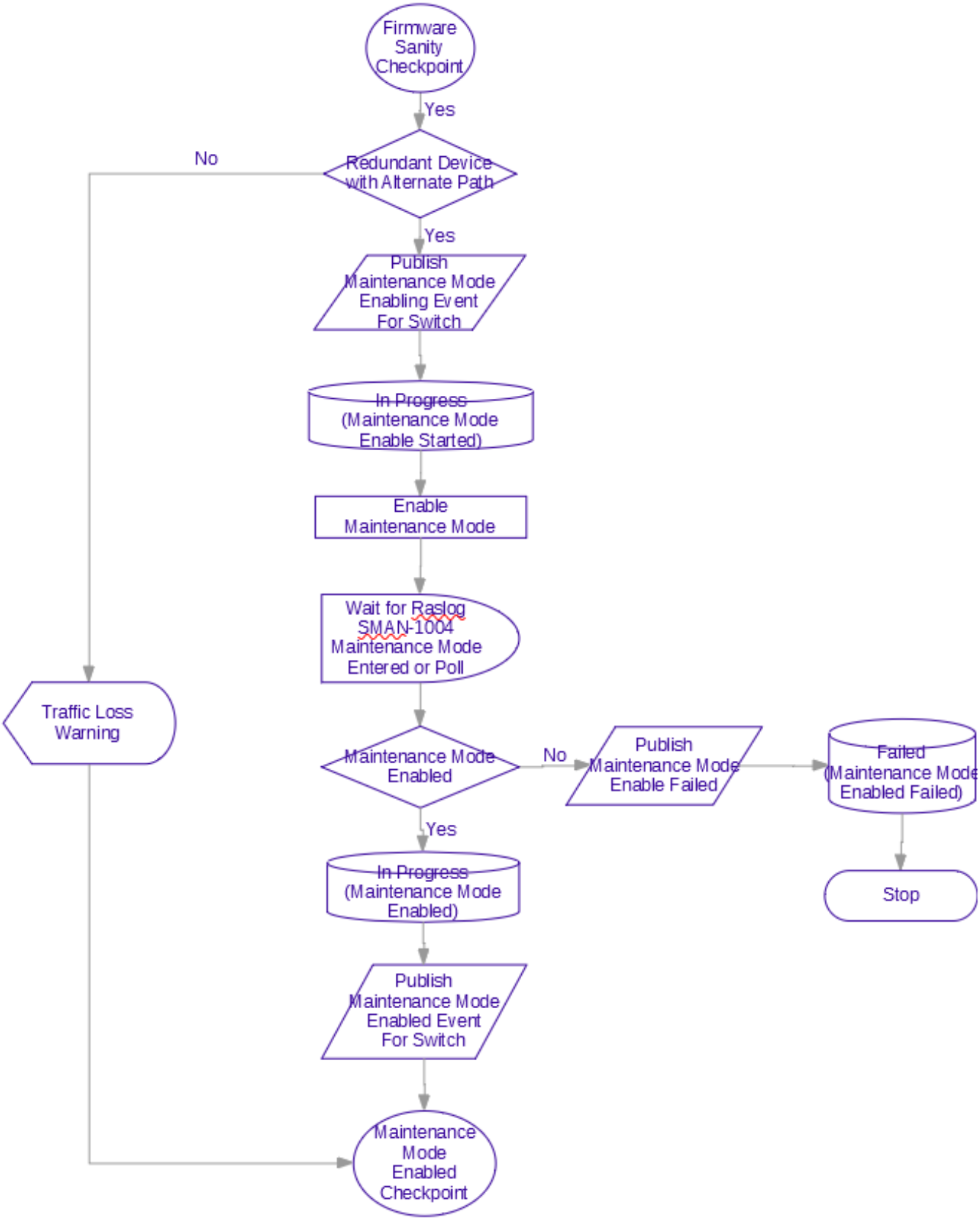



**Figure 29: Maintenance mode enable flowchart**

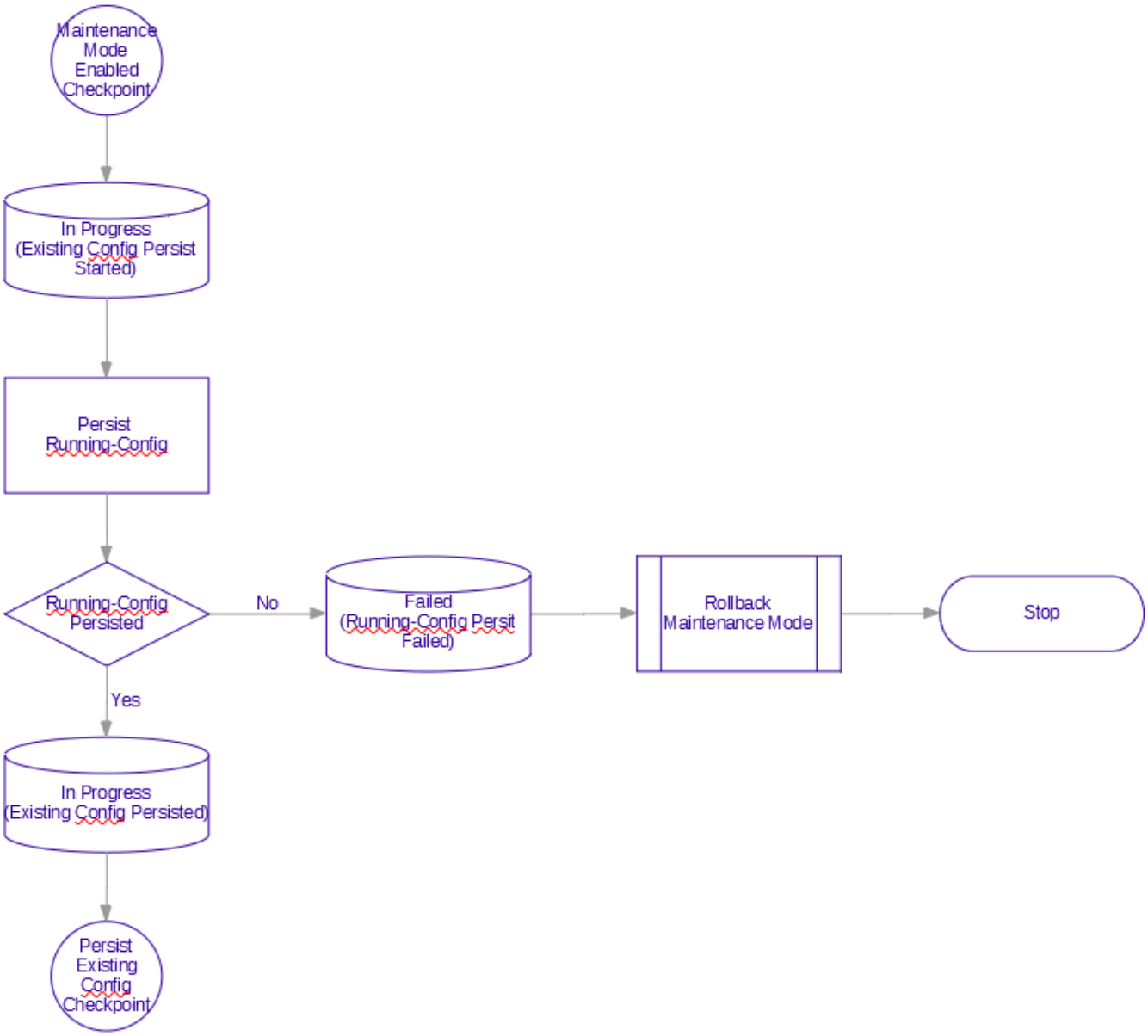



**Figure 30: Persist existing config flowchart**

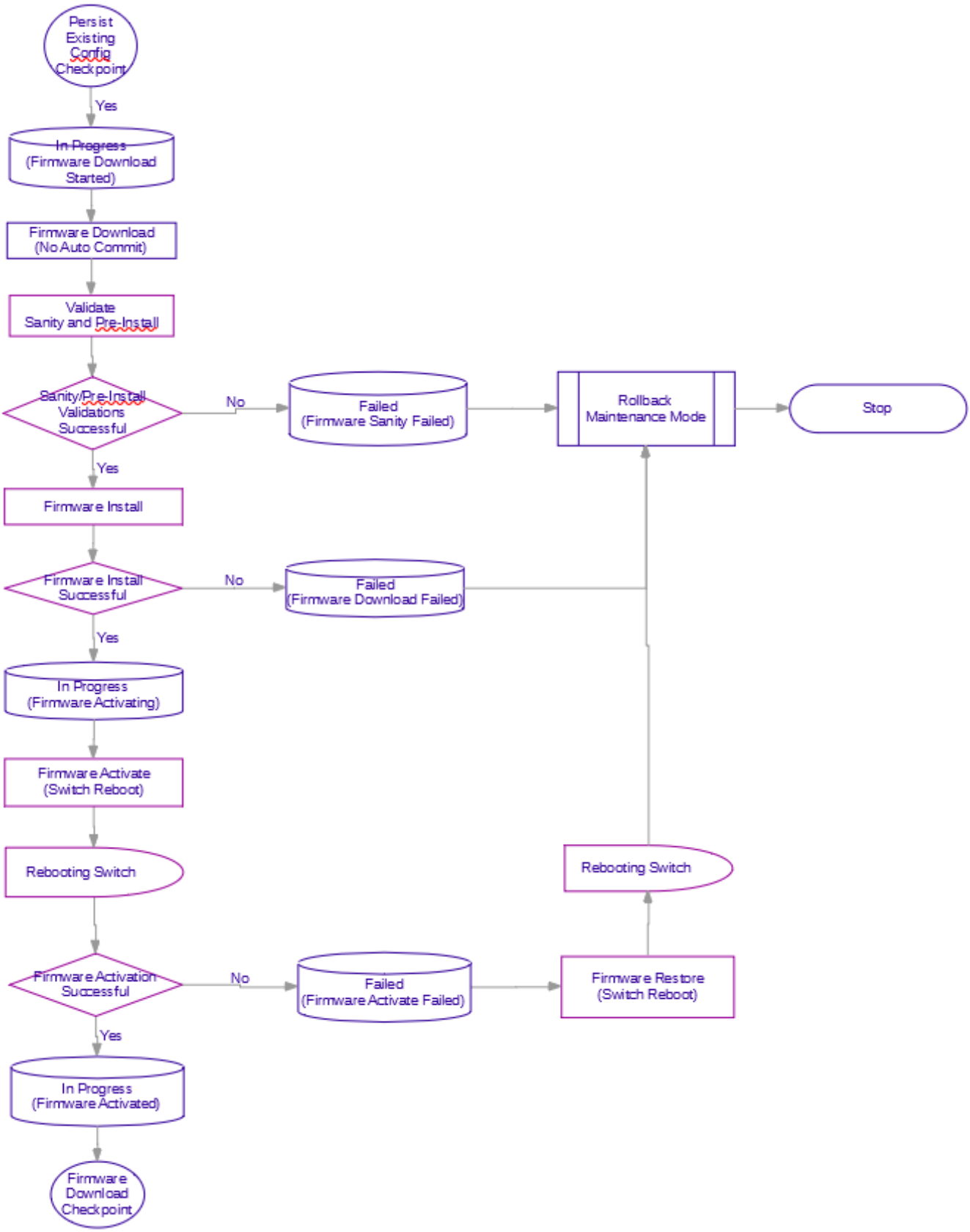



**Figure 31: Firmware download flowchart**

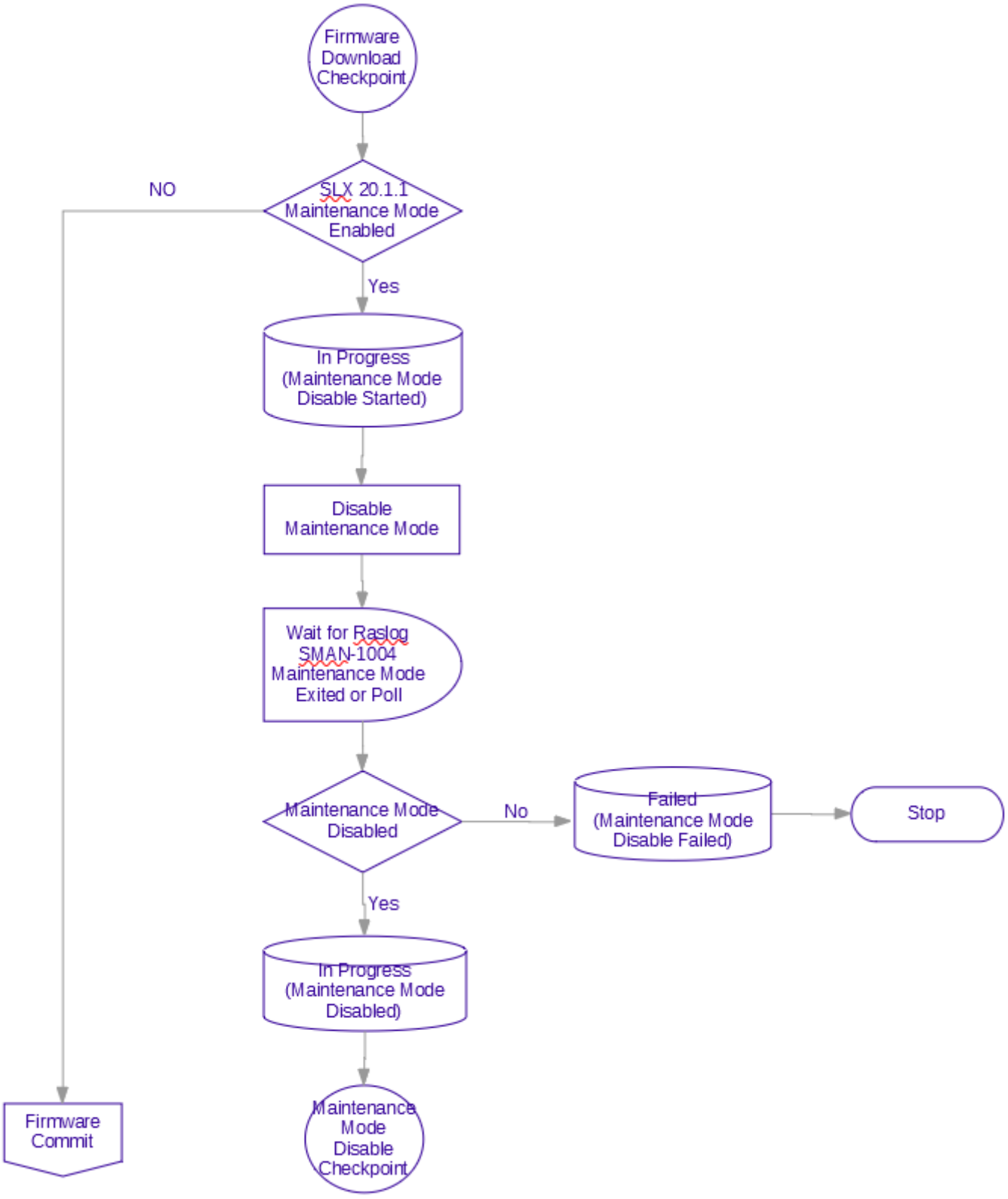



**Figure 32: Maintenance mode disable flowchart**

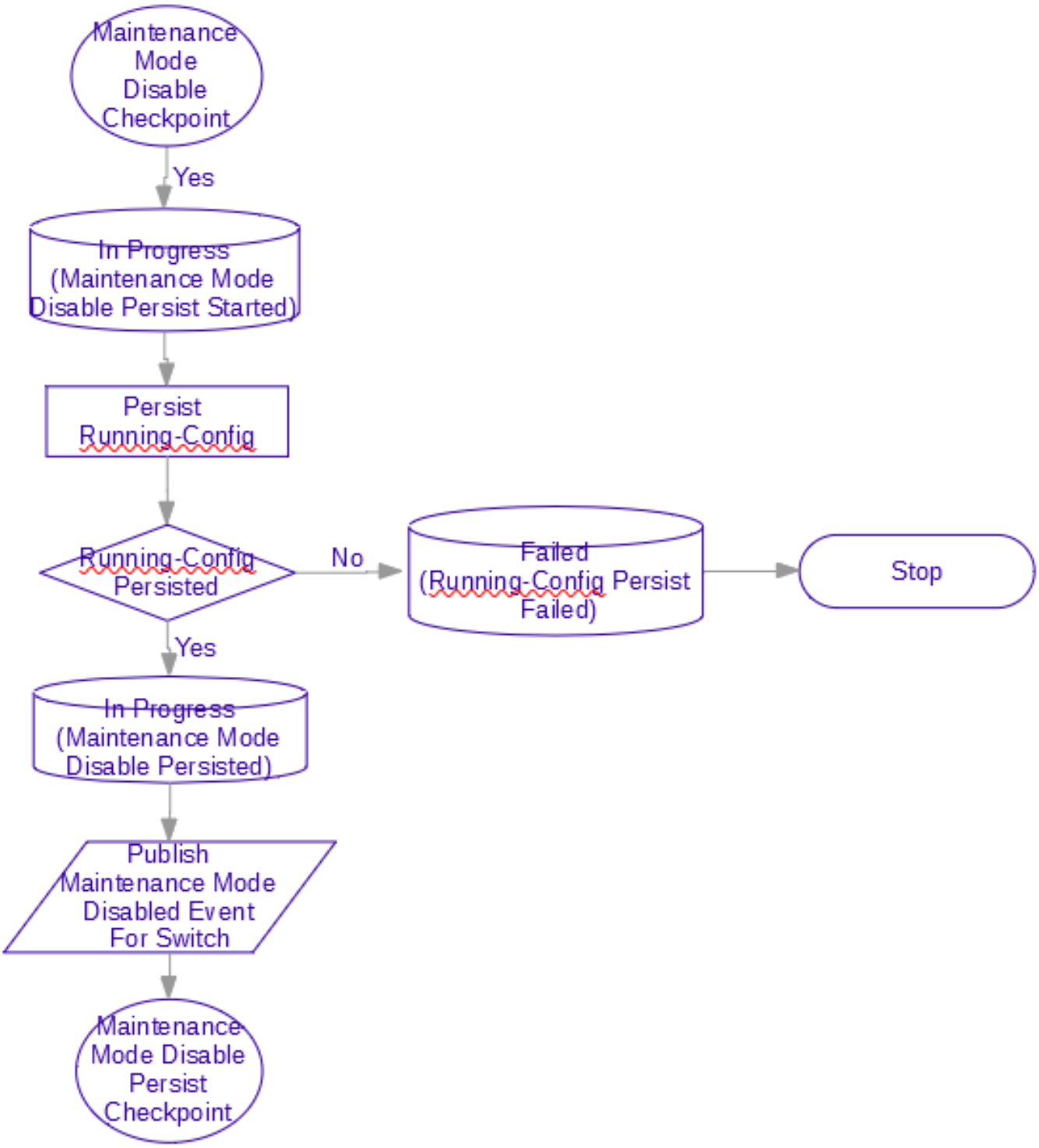



**Figure 33: Persist maintenance mode disable config flowchart**

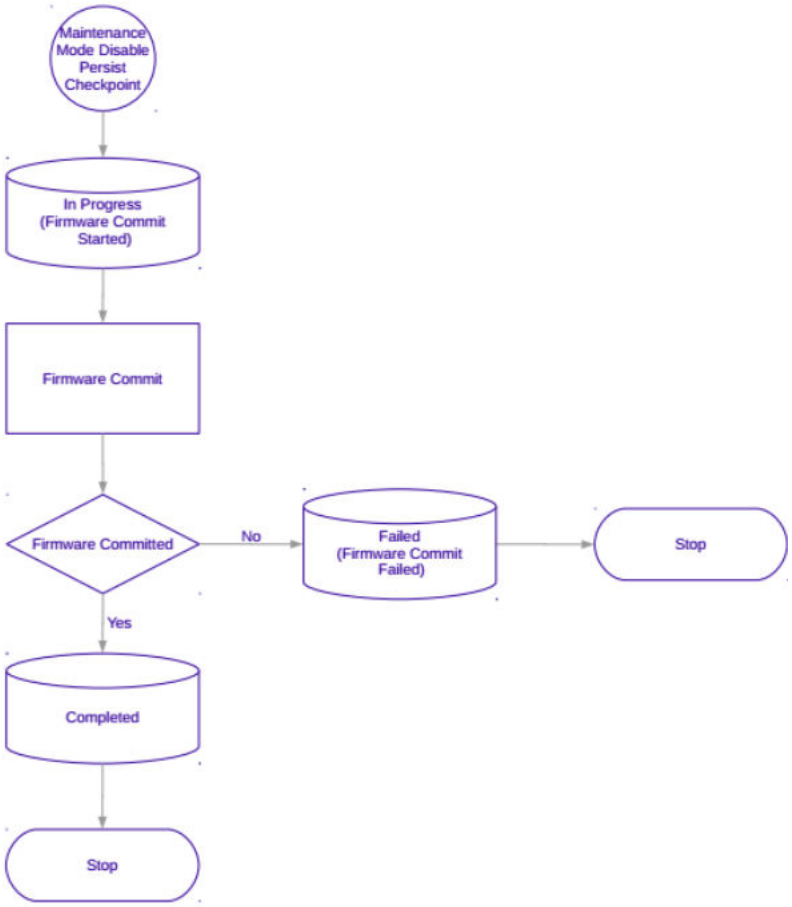



**Figure 34: Firmware commit flowchart**

## Foundation and Ecosystem Services Interaction

When maintenance mode is enabled, configuration changes to the switch are not allowed. This feature will notify other services by publishing the following messages.

| Message | Details | Maintenance Mode State |
|---|---|---|
| Maintenance Mode Fabric Enabling | Published just prior to issuing the maintenance mode on one or more devices in the fabric when no devices had maintenance mode enabled previously. | At least one device in the fabric is enabling or has maintenance mode enabled. |
| Maintenance Mode Fabric Disabled | Published when maintenance mode disable (no enable) operation has completed on devices in the fabric and all devices in the fabric have maintenance mode disabled. | All devices in the fabric have maintenance mode disabled. |
| Maintenance Mode Device Enabling | Published just prior to issuing the maintenance mode enable configuration to the switch. | Disabled |

| Message | Details | Maintenance Mode State |
|---|---|---|
| Maintenance Mode Device Enable Failed | Published when maintenance mode enable operation on the switch has failed or timed out. | Disabled |
| Maintenance Mode Device Enable Success | Published when maintenance mode enable operation on the switch is successful. | Enabled |
| Maintenance Mode Device Disabling | Published just prior to issuing the maintenance mode disable (no enable) configuration to the switch. | Enabled |
| Maintenance Mode Device Disable Failed | Published when maintenance mode disable (no enable) operation on the switch has failed or timed out. | Enabled |
| Maintenance Mode Device Disable Success | Published when maintenance mode disable (no enable) operation on the switch is successful. | Disabled |

# efa inventory firmware-host register

Registers a firmware host which will be used to download firmware builds to the devices.

## Syntax

**efa inventory firmware-host register** [**--ip** *<firmware host ip address>* | **--protocol** *< scp | ftp | sftp | http >* |**--username** *<username credentials to login to the firmware download host>* | **--password** *<password credentials to login to the firmware download host>* ]

## Command Default

Simple connectivity test to the firmware-host by given IP. The complete firmware-host sanity check will be performed later when a device is prepared and again when firmware-download is executed.

## Parameters

**--ip**

Firmware host IP address

**--protocol**

Protocol < scp | ftp | sftp | http >

**--username**

Username credentials to login to the firmware download host

**--password**

Password credentials to login to the firmware download host

# efa inventory firmware-host update

Updates the login credentials or file transfer protocol to be used by a switch when downloading firmware from the firmware host.

## Syntax

**efa inventory firmware-host update** [ **--ip** *<firmware host ip address>* | **--protocol** *< scp | ftp | sftp | http >* | **--username** *<username credentials to login to the firmware download host>* | **--password** *<password credentials to login to the firmware download host>* ]

## Parameters

**--ip**

Firmware host IP address

**--protocol**

Protocol < scp | ftp | sftp | http >

**--username**

Username credentials to login to the firmware download host

**--password**

Password credentials to login to the firmware download host

# efa inventory firmware-host delete

Removes a firmware host.

## Syntax

**efa inventory firmware-host delete --ip** *<firmware host ip address>*

## Parameters

**--ip**

Firmware host IP address

# efa inventory firmware-host list

Displays all registered firmware hosts.

## Syntax

**efa inventory firmware-host list**

## Parameters

**list**

Firmware hosts list

# efa inventory device firmware-download prepare add

Prepares a device for a firmware download. Firmware host sanity validations will be done at this time. If the validations are successful, then the device will be prepared.

## Syntax

**efa inventory device firmware-download prepare add** [ **--ip** *<device ip address>* | **--firmware-host** *<firmware download host ip address>* | **--firmware-directory** *<path to the target firmware build>* ]

## Command Default

Traffic loss expected for non-redundant devices (single non-MCT leaf, spine, or super spine).

This command will only accept one device ip address and allow only one outstanding prepared device.

## Parameters

**--ip**

Device IP address

**firmware-host**

Firmware download host IP address

**--firmware-directory**

Path to the target firmware build

## Usage Guidelines

Firmware host must be registered.

Device IPs belong to the same fabric (for clos topologies).

Do not allow devices to be prepared if a firmware-download is in progress.

Allow a device to be prepared after a firmware-download has completed or in an unprepared state.

Do not allow for both MCT leaf pairs to be prepared together.

Do not allow for all spines in the same pod of the same fabric to be prepared together.

Do not allow for all super-spines in the fabric to be prepared together.

Firmware sanity check performed for the registered firmware-host and firmware-directory on the given device.

## Examples

```
efa inventory device firmware-download prepare --ip 10.24.12.122 --firmware-host
10.31.2.101 --firmware-directory /proj/buildsjc/sre/SQA/slxos/20.1.1/20.1.1_bld60
```

```
efa inventory device firmware-download prepare --ip 10.24.12.122 --firmware-host
10.31.2.101 --firmware-directory /proj/buildsjc/sre/SQA/slxos/20.1.1/20.1.1_bld60


Fabric Name: stage5

+--------------+-----------+-------+--------------+-------+------+------------------
+---------------+-----------------------------------------------------
+--------------------------------+

|  IP Address  | Host Name | Model | Chassis Name |  ASN  | Role | Current Firmware |
Firmware Host |                Firmware Directory                |       Last Update
Time        |

+--------------+-----------+-------+--------------+-------+------+------------------
+---------------+-----------------------------------------------------
+--------------------------------+

| 10.24.12.122 | SLX       |  3010 | SLX9150-48XT | 65001 | Leaf | 20.1.1_bld59     |
10.31.2.101 | /proj/buildsjc/sre/SQA/slxos/20.1.1/20.1.1_bld60 | 2019-10-21
06:30:15.736483-07 |

+--------------+-----------+-------+--------------+-------+------+------------------
+---------------+-----------------------------------------------------
+--------------------------------+

Firmware Download Prepare Details

Prepare Device Firmware Download [Success]

10.24.12.122 [Succeeded]
```

# efa inventory device firmware-download prepare remove

Removes the device from the prepared list.

## Syntax

**efa inventory device firmware-download prepare remove** [ **--fabric** *<fabric name>* |**--ip** *<device ip address>* ]

## Command Default

This command will unprepare the single device if it is prepared.

## Parameters

**--fabric**

Name of the fabric

**--ip**

IP address of the device

# efa inventory device firmware-download prepare list

Prepares a list of all devices for firmware download.

## Syntax

**efa inventory device firmware-download prepare list --fabric** *<fabric name>*

## Command Default

Only a single prepared device will be allowed.

## Parameters

**--fabric**

Name of the fabric

## Examples

```
efa inventory device firmware-download prepare list --fabric stage5

 Fabric Name: stage5

+--------------+-----------+-------+--------------+-------+------------+------------------
+---------------+-----------------------------------------------------
+--------------------------------+

|  IP Address  | Host Name | Model | Chassis Name |  ASN  |    Role    | Current Firmware
| Firmware Host |                Firmware Directory                |       Last Update
Time        |

+--------------+-----------+-------+--------------+-------+------------+------------------
+---------------+-----------------------------------------------------
+--------------------------------+

| 10.24.12.153 | SLX       |  3012 | SLX9250      | 64512 | SuperSpine | 20.1.1_bld59
|   10.31.2.101 | /proj/buildsjc/sre/SQA/slxos/20.1.1/20.1.1_bld60 | 2019-10-21
06:30:01.424591-07 |

| 10.24.12.146 | SLX       |  3012 | SLX9250      | 64521 | Spine      | 20.1.1_bld59
|   10.31.2.101 | /proj/buildsjc/sre/SQA/slxos/20.1.1/20.1.1_bld60 | 2019-10-21
06:30:01.424591-07 |

| 10.24.12.148 | SLX       |  3012 | SLX9250      | 64523 | Spine      | 20.1.1_bld59
|   10.31.2.101 | /proj/buildsjc/sre/SQA/slxos/20.1.1/20.1.1_bld60 | 2019-10-21
06:30:01.424591-07 |

| 10.24.12.121 | SLX       |  3010 | SLX9150-48XT | 65001 | Leaf       | 20.1.1_bld59
|   10.31.2.101 | /proj/buildsjc/sre/SQA/slxos/20.1.1/20.1.1_bld60 | 2019-10-21
06:30:01.424591-07 |

| 10.24.12.123 | SLX       |  3010 | SLX9150-48XT | 65002 | Leaf       | 20.1.1_bld59
|   10.31.2.101 | /proj/buildsjc/sre/SQA/slxos/20.1.1/20.1.1_bld60 | 2019-10-21
06:30:01.424591-07 |

| 10.24.12.125 | SLX       |  3010 | SLX9150-48XT | 65003 | Leaf       | 20.1.1_bld59
```

```
|   10.31.2.101 | /proj/buildsjc/sre/SQA/slxos/20.1.1/20.1.1_bld60 | 2019-10-21
06:30:01.424591-07 |

| 10.24.12.127 | SLX       |  3010 | SLX9150-48XT | 65004 | Leaf       | 20.1.1_bld59
|   10.31.2.101 | /proj/buildsjc/sre/SQA/slxos/20.1.1/20.1.1_bld60 | 2019-10-21
06:30:01.424591-07 |

+--------------+-----------+-------+--------------+-------+------------+------------------
+---------------+-----------------------------------------------------
+--------------------------------+
```

# efa inventory device firmware-download execute

Starts the firmware upgrade with maintenance mode for devices in prepared state and clear out the prepared list.

## Syntax

**efa inventory device firmware-download execute --fabric** *<fabric name>*

## Command Default

General warning for traffic loss for single-homed servers if any leafs or non-CLOS devices are prepared for firmware-download.

## Parameters

**--fabric**

Name of the fabric

## Usage Guidelines

One or more devices are prepared.

Only allow one outstanding firmware-download execution per fabric.

# efa inventory device firmware-download show

Shows the progress and status the executed firmware-download.

## Syntax

**efa inventory device firmware-download show --fabric** *<fabric name>*

## Parameters

**--fabric**

Name of the fabric

## Examples

```
efa inventory device firmware-download show --fabric stage5

Fabric Name: stage5

+-----------+-------+--------------+-------+------------+------------------
+-----------------+--------------+--------------------------
+--------------------------------+

| Host Name | Model | Chassis Name |  ASN  |    Role    | Current Firmware | Target
Firmware | Update State |          Status          |       Last Update Time        |

+-----------+-------+--------------+-------+------------+------------------
+-----------------+--------------+--------------------------
+--------------------------------+

| SLX       |  3012 | SLX9250      | 64512 | SuperSpine | 20.1.1_bld60     |
20.1.1_bld60    | Completed    |                          | 2019-10-21 06:30:01.424591-07
|

| SLX       |  3012 | SLX9250      | 64513 | SpineSpine | 20.1.1_bld60     |
20.1.1_bld60    | Completed    |                          | 2019-10-21 06:30:09.744543-07
|

| SLX       |  3012 | SLX9250      | 64521 | Spine      | 20.1.1_bld60     |
20.1.1_bld60    | Completed    |                          | 2019-10-21 06:30:16.563591-07
|

| SLX       |  3012 | SLX9250      | 64522 | Spine      | 20.1.1_bld60     |
20.1.1_bld60    | Completed    |                          | 2019-10-21 06:30:15.736483-07
|

| SLX       |  3012 | SLX9250      | 64523 | Spine      | 20.1.1_bld60     |
20.1.1_bld60    | Completed    |                          | 2019-10-21 06:30:17.665491-07
|

| SLX       |  3012 | SLX9250      | 64524 | Spine      | 20.1.1_bld60     |
20.1.1_bld60    | Completed    |                          | 2019-10-21 06:30:20.244241-07
|

| SLX       |  3010 | SLX9150-48XT | 65001 | Leaf       | 20.1.1_bld60     |
20.1.1_bld60    | Completed    |                          | 2019-10-21 06:30:25.441725-07
|

| SLX       |  3010 | SLX9150-48XT | 65001 | Leaf       | 20.1.1_bld59     |
```

```
20.1.1_bld60    | In Progress  | Maintenance Mode Enabled | 2019-10-21 06:34:25.211744-07
|

| SLX       |  3010 | SLX9150-48XT | 65002 | Leaf       | 20.1.1_bld59     |
20.1.1_bld60    | Not Prepared |                            |
|

| SLX       |  3010 | SLX9150-48XT | 65002 | Leaf       | 20.1.1_bld59     |
20.1.1_bld60    | Not Prepared |                            |
|

| SLX       |  3010 | SLX9150-48XT | 65003 | Leaf       | 20.1.1_bld59     |
20.1.1_bld60    | Not Prepared |                            |
|

| SLX       |  3010 | SLX9150-48XT | 65003 | Leaf       | 20.1.1_bld59     |
20.1.1_bld60    | Not Prepared |                            |
|

| SLX       |  3010 | SLX9150-48XT | 65004 | Leaf       | 20.1.1_bld59     |
20.1.1_bld60    | Not Prepared |                            |
|

| SLX       |  3010 | SLX9150-48XT | 65004 | Leaf       | 20.1.1_bld59     |
20.1.1_bld60    | Not Prepared |                            |
|

+-----------+-------+--------------+-------+------------+------------------
+-----------------+--------------+--------------------------
+---------------------------------+
```

# efa inventory device firmware-download error show

Displays detailed error status

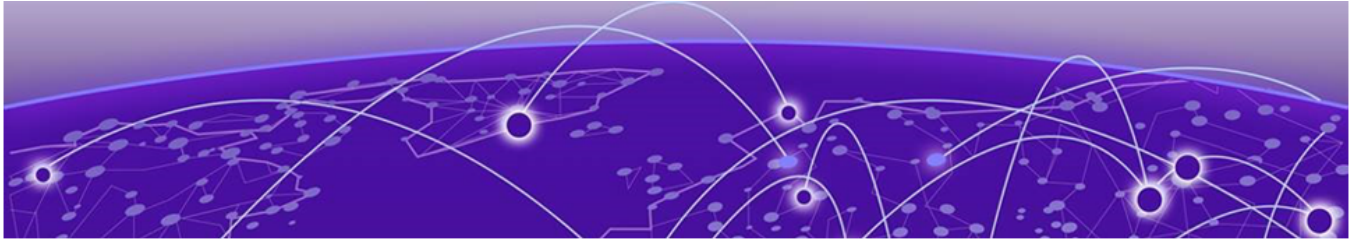
## Syntax

**efa inventory device firmware-download error show --fabric** *<fabric name>*

## Parameters

**--fabric**

Name of the fabric



# Database Backup and Restore



## Backup and Restore Overview

You can backup and restore the DCApp services, such as goInventory-service, goFabric-service, and goTenant-service.

Use the backup and restore feature in case the DCApp database becomes corrupted or you want to revert to a previous configuration. This is a two-step process:

1. Backup the database
2. Restore the database

## Backup the Database

All three Inventory, Fabric, and Tenant Service databases are backed up as part of supportSave, so users can run the **`bash efa_backup`** command to back up the databases.

*Example:*

```
admin@TPVM:/apps/efa$ bash efa_backup.sh

Provide your backup absolute directory path: /apps/backup
v2.1.0-33
backing "dcapp_asset" database to /apps/backup directory
backing "dcapp_fabric" database to /apps/backup directory
backing "dcapp_tenant" database to /apps/backup directory
backing "dcapp_vcenter" database to /apps/backup directory
backing "dcapp_hyperv" database to /apps/backup directory
backing "efa_openstack" database to /apps/backup directory
admin@TPVM:/apps/efa$ cd ..
admin@TPVM:/apps$ ls
backup  bin  efa  efadata  efa_logs  efa-v2.1.0-23.tar.gz  efa-v2.1.0-33.tar.gz
k3s.yaml  lost+found  rancher
admin@TPVM:/apps$ tar cvzf backup.tar.gz backup
backup/
backup/dcapp_tenant.dump
backup/dcapp_hyperv.dump
backup/dcapp_fabric.dump
backup/efa_openstack.dump
backup/dcapp_vcenter.dump
```

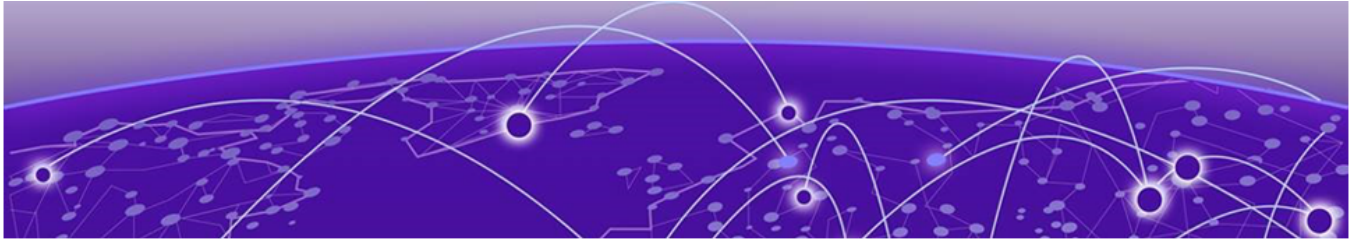
```
backup/efa_version
backup/dcapp_asset.dump
admin@TPVM:/apps$ ls
backup         bin  efadata   efa-v2.1.0-23.tar.gz  k3s.yaml    rancher
backup.tar.gz  efa  efa_logs  efa-v2.1.0-33.tar.gz  lost+found
admin@TPVM:/apps$

Copy to /efatboot
```

## Restore the database

The following command restores the database.

```
admin@TPVM:/apps/efa$ bash efa_restore.sh
EFA installed on TPVM...
please provide database back-up directory: /home/admin/backup
stopping service gofabric-service
deployment.apps "gofabric-service" deleted
stopping service gotenant-service
deployment.apps "gotenant-service" deleted
stopping service goinventory-service
deployment.apps "goinventory-service" deleted
stopping service goopenstack-service
```



# Inventory Service



## Inventory Service Overview

The Inventory Service API allows information from EFA's database to be queried through EFA's north-bound REST API. The Inventory Service API supports the capability to filter data based on specified fields. .

Examples of this filtering capability are:

- Get all devices with a specified role
- Get all devices with a specified firmware version
- Get all Fabric-connected interfaces for a particular device
- Get all connected ports for a particular device
- Get all interfaces that participate in a particular PO

Inventory data made available through the Inventory Service REST API conforms to the OpenAPI standard. Further details about this API can be found at https://www.extremenetworks.com/support/documentation-api/extreme-fabric-automation-2-1-0/, API documentation, and EFA's OpenAPI Specification file, can also be accessed directly through the EFA server at `http://<host_ip>:docs`

### Supporting Hardware

Inventory Service is supported on several SLX devices.

- SLX 9140
- SLX 9240
- SLX 9540
- SLX 9850

### Inventory Update Messaging

The Inventory Service dynamically updates devices, the RASLog receiver, the database, and the Fabric Service when changes occur in the inventory.

The Inventory Service dynamic messaging occurs for changes in the inventory, such as the following examples.

- LLDP changes
- Status changes for physical and logical interfaces
- Slot and card changes, such as power up, power down, removal, and addition
- Port flapping
- IP address being assigned to interfaces

## Replace a Device with the Same Configuration

You can replace a device that has the same configuration as the new device without affecting the network or the configuration.

1. Ensure that the configuration from the old device is copied to the new device.
2. Ensure that the device information in EFA is current.

   Do not perform this step if all configuration changes to the device are done through EFA.

   ```
   efa inventory device update ip -<IP address of the old device>
   ```

   **Note**

   - This command ensures that the Asset service has the latest information.
   - If the replaced device is not reachable or responding and the configuration being copied to the replacement device does not match the details in the Asset service, see Replace a Device with a Different Configuration on page 142.

3. Replace the device.

   ```
   efa inventory device replace ip -<IP address of the new device>
   ```

This example shows typical output for a successful device replacement.

```
efa inventory device replace ip -10.x.x.x

| ID | IP Address | Host Name| Model| Chassis Name| Firmware |  ASN  | Role | Fabric |
-------------------------------------------------------------------------------------
| 7  | 10.x.x.x   | Fre-135  | 3001 | BR-SLX9140  | 18s.1.01a| 65000 |      |        |
```

This example shows sample output for a failure.

```
efa inventory device replace ip -10.x.x.x
Key                           Reason
-------------------------------------------------------------------
Interface IPs Updated         Interface ethernet 0/5 has IP 1.1.1.1/31 but no previous IP
                              was assigned for device 10.x.x.x
VRF Interface Mapping Added   true
```

This example shows sample output for a different failure.

```
efa inventory device replace ip -10.x.x.x
Key                           Reason
-------------------------------------------------------------------
Local AS updated
Interface IPs Updated         Interface ethernet 0/28 has IP 10.10.10.56/31 but no
                              previous IP was assigned for device 10.x.x.x. Interface
ethernet
                              0/25:1 has IP 10.10.10/85/31 but no previous IP was
assigned for
                              device 10.x.x.x
```

```
Added Interfaces                loopback 1
Global Added                    64512
VRF updated                     true
VRF Interface Mapping Added     true
```

## Replace a Device with a Different Configuration

You can replace a device when the configuration on the old device does not match the configuration on the new device.

1. Remove the old device from EFA.

   ```
   efa fabric device remove –name <Fabric name> --ip <IP address of the device>
   ```

   This command removes the device (decommissioned from the Fabric), cleans up all relevant neighbors, and cleans up the sub-configurations from MCT and BGP configurations.
2. Add the new device to EFA.

   ```
   dca fabric device add-bulk –name <Fabric name> --leaf <IP address of the
   device if it is a leaf> --spine <IP address of the device if it is a spine>
   --super-spine <IP address of the device if it is a super spine>
   ```
3. Verify that the device shows up in the Fabric.

   ```
   efa fabric show
   ```
4. (Optional) View the cause of an application or device error.

   ```
   efa fabric error show
   ```
5. (Optional) View the cause of a configuration refresh.

   ```
   efa fabric debug config-gen-reason
   ```
6. Configure the Fabric.

   ```
   efa fabric configure -name <Fabric name>
   ```
7. Rerun all Tenant configurations.

## Compare a Device

You can view the configurations on the device that are out of sync with the configurations in the Asset service.

This helper utility displays a summary of the information in the Asset database.

View a summary of the information in the Asset database.

```
efa inventory device compare –ip <IP address of the device>
```