

Extreme Fabric Automation 2.0.1 Administration Guide v1.0

September 2019

9036378-01 Rev AA

Copyright Statement and Legal Notices

Copyright © 2019 Extreme Networks, Inc. All Rights Reserved.

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

Contents

Copyright Statement and Legal Notices	2
Legal Notice	2
Trademarks	2
Open Source Declarations	2
Preface	7
Contacting Extreme Technical Support	8
Extreme resources	8
Document feedback	8
Introduction	10
Supported Platform Matrix	11
Installation	12
New installation	12
OVA installation	13
Upgrade/reinstall	13
REST API Guide	15
IP Fabric/Clos Orchestration	16
Overview	16
Day-0 prerequisites on SLX devices	16
Automating Clos Fabric provisioning	17
Automating 3-stage Clos	18
Automating 5-stage Clos	19
Fabric Validation, Error Reporting, and Recovery	21
Get Fabric summary	21
Get Fabric details	22
Get Fabric device config gen reason	22
Get physical topology	23
Get underlay topology	24
Get overlay topology	25
Get device config	25
Clear device config	27
Get Fabric execution	28
Get Fabric error	29

Get supportSave	30
Tenant Network Provisioning	31
Overview	31
Day-0 prerequisites For Tenant Network Provisioning	31
Tenant network creation.....	32
Tenant network provisioning	34
Network creation	35
Network deletion	36
Advanced CLIs.....	37
Tenant network provisioning using Default-Tenant and Default-Asset.....	37
Network creation	37
Network deletion	39
Tenant network provisioning using User-Tenant and User-Asset.....	39
Network creation	39
Network deletion	40
Tenant Error Reporting and Recovery	42
Tenant network error reporting (Advanced CLIs)	42
Tenant network workflow-error reporting	45
Logging/ELK Integration.....	46
URLs to access the ELK stack:	46
Sample log	46
Infra level.....	46
Application level.....	47
Container stats and host stats	48
Container overview	49
CPU and memory usage	49
Appendix A: Fabric Service CLIs	51
3-Stage Clos automation	51
Create a Fabric	51
Add a device to the Fabric	52
Add multiple devices to a Fabric.....	53
Validate the Fabric topology.....	54
Configure Fabric on the device	54
Remove multiple devices from the Fabric	55
Fabric settings	56

5-stage Clos automation	58
Multi-Fabric automation	61
Create a Fabric	61
Delete a Fabric	62
Clone a Fabric.....	62
Appendix B: Asset Service CLIs	63
Device register.....	63
Device delete.....	64
Device update	64
Key-value store.....	65
Create KV store	65
Delete KV store	65
List KV store	66
Examples	66
Persist device configuration	68
Execute CLI	69
List registered devices	71
Compare and Replace device	72
Appendix C: Tenant Service CLIs.....	72
Automating L2 and L3 extension between racks	73
Create a VRF.....	74
Create Tenant network (using workflow)	74
Automating VLAN scoping at the ToR level.....	77
Automating VLAN scoping at the port level within a ToR	78
Update operations	79
Update Tenant network.....	79
Update endpoint group	79
Update Port-channel.....	81
Update Asset.....	81
Update Tenant	82
Get operations	84
Get Tenant network (workflow CLI).....	84
Get Tenant network (Advanced CLI)	85
Get endpoint group	86
Get port-channel group	86

Get VRF	88
Get Tenant	89
Get Asset	89
Delete operations	89
Delete Tenant network (workflow CLI)	89
Delete Tenant network (Advanced CLI)	89
Delete end point group	90
Delete port-channel	91
Delete VRF	91
Delete Asset	91
Delete Tenant	91
Bulk operations	92
Bulk create	92
Bulk delete	93
Appendix D: Database Backup and Restore	94
Overview	94
Backup the Database	94
Restore the database	94
Appendix E: SLX-OS Device Firmware Update Overview	98
Persist the configuration	98
Back up the DCA databases	98
SLX-OS firmware update on the device	98
Update the inventory	98
Verify the firmware update	98
Appendix F: Device Replacement and Compare	100
Device replacement with the same configuration	100
Device replacement with different configuration	101
Device compare	102
Execute-cli	102

Document history

Version	Summary of changes	Publication date
1.0	Initial release	23 September 2019

Preface

Contacting Extreme Technical Support

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

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

GTAC (Global Technical Assistance Center) for immediate support

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

Email: support@extremenetworks.com. To expedite your message, enter the product name or model number in the subject line.

GTAC Knowledge - Get on-demand and tested resolutions from the GTAC Knowledgebase, or create a help case if you need more guidance.

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

Support Portal - Manage cases, downloads, service contracts, product licensing, and training and certifications.

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

Your Extreme Networks service contract number and/or serial numbers for all involved Extreme Networks products

A description of the failure

A description of any action(s) already taken to resolve the problem

A description of your network environment (such as layout, cable type, other relevant environmental information)

Network load at the time of trouble (if known)

The device history (for example, if you have returned the device before, or if this is a recurring problem)

Any related RMA (Return Material Authorization) numbers

Extreme resources

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

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

Document feedback

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

You can provide feedback in two ways:

- Use our short online feedback form at <http://www.extremenetworks.com/documentation-feedback-pdf/>

- Email us at internalinfodev@extremenetworks.com

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

Introduction

Extreme Fabric Automation (EFA) is also known as Data Center Automation Application (DCA) is a microservices-based scalable application written in GoLang that orchestrates the following installations:

- 3-stage IP Clos Fabric
- 5-stage IP Clos Fabric
- Tenant Aware Networks

NOTE: This document was originally a “Controlled Release” document titled *Data Center Automation Application (DCA), 2.0.0 Administration Guide v3.0*. Its content has been subsequently updated.

This release addresses only the automation of Fabric infrastructure and Tenant Services lifecycle management of 3- and 5-stage IP Clos DC Fabrics. It does not address the automation of Small Data Centers or 7-stage IP Clos Fabrics.

The key tenets of this orchestration are as follows:

- Conformance to the EVD (Extreme Validated Design) for IP Fabric):
<https://www.extremenetworks.com/resources/extreme-validated-design/extreme-ip-fabric-architecture/>
- Speed of provisioning
- Seamless installation/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 Tenant Networks.

EFA comprises three core containerized services that interact with each other and other infrastructure services to provide the core functions of IP Fabric automation:

Asset Service	Provides the secure credential store and deep discovery of physical/logical Assets of the managed devices, and publishes the Asset refresh/change events to other services
Fabric Service	Helps orchestrate and visualize the BGP/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

The following figure illustrates the application’s Docker-based functionality in provisioning and discovery.

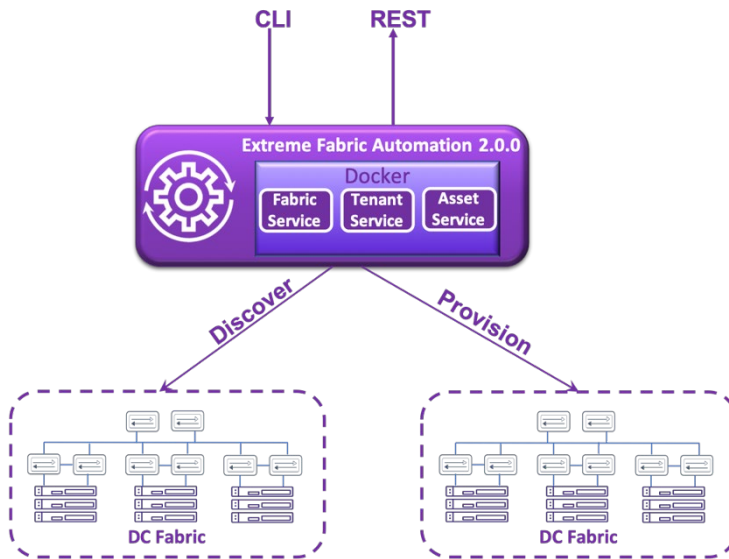


Figure 1. Docker-based provisioning and discovery

Supported Platform Matrix

This application is supported on the following platforms.

Platform	Role	SLXOS version
SLX-9140	Leaf	18s.1.01 /a /c,18s.1.03
SLX-9240	Leaf/spine/super-spine	18s.1.01 /a /c,18s.1.03
SLX-9850	Spine/super-spine	18r.1.00aa /b /c
EN-SLX-9030-48S	Leaf	18x.1.00 /a /b
EN-SLX-9030-48T	Leaf	18x.1.00 /a /b
SLX-9540	Leaf/border leaf	18r.1.00aa /b /c

Installation

Note the following requirements.

Prerequisites	<ul style="list-style-type: none">• Minimum System Requirements<ul style="list-style-type: none">- CPU: 4 cores- Storage: 50 GB- RAM: 8 GB- OS: Ubuntu 16.04+• Software (installed by deployment script)<ul style="list-style-type: none">- Docker CE: v18.06.1~ce~3-0~ubuntu- Docker-compose: v1.22- Docker-compose file: version 3- postgresql-client: v9.6
----------------------	--

New installation

Do the following to install the application for the first time, referring to following flow.

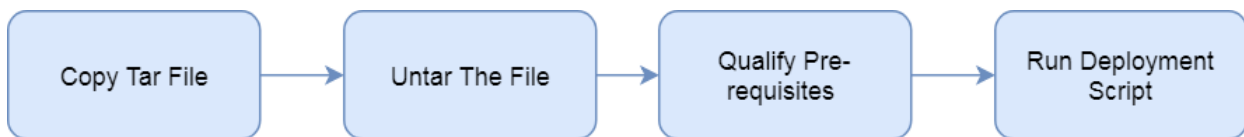


Figure 2. New installation flow

1. Download image (*.tar.gz) and untar it.
tar -xzf single-node-deployment-<TAR_VERSION>.tar.gz
2. Change directory to single-node-deployment.
cd single-node-deployment
3. Check for pre-requisites.
 - CPU: 4 cores
 - Storage: 50 GB
 - RAM: 8 GB
 - OS: Ubuntu 16.04+
4. Run the App Installation script.
source deployment.sh

OVA installation

Open Virtual Appliance (OVA) is an OVF file packaged with base image Ubuntu 16.04, storage = 50 G, RAM = 8 GB, and preinstalled with EFA. OVA is also compatible with VMware ESXi servers, so it can be deployed with VMware products.

To deploy the OVA using VirtualBox, follow the steps below:

1. Download the *EFA_v2.0.0_<build_number>.ova* file.
2. Start Oracle VirtualBox.
3. Choose **File -> Import Appliance**.
4. Select the downloaded *.ova file and click **Open**.
5. After importing the OVA file, start the VM.

NOTE: Use the OVA image for new installations. For existing deployments, refer to the Upgrade/reinstall section.

User credentials:

There are three credentials; the default user is “ubuntu”.

- 1) user/password: ubuntu/ubuntu
- 2) user/password: xmc/xmc123
- 3) root/password : root/dca123

Upgrade/reinstall

Do the following to upgrade or reinstall the application, referring to the following flow chart.

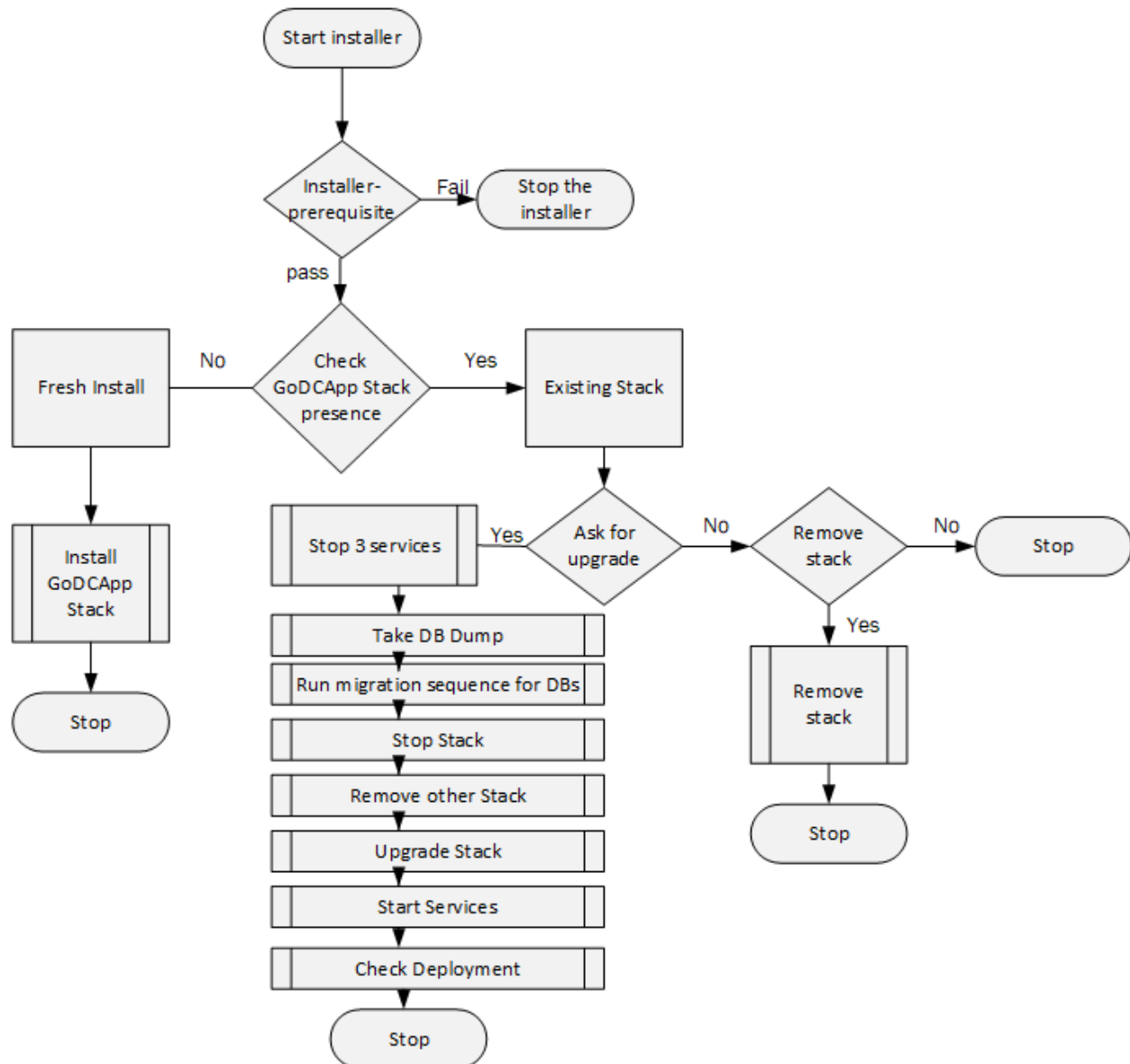


Figure 3. Upgrade/reinstall flow chart

1. Change directory to single-node-deployment.
cd single-node-deployment
2. Run the deployment script.
source deployment.sh
This script will bring up the following components:
 - a. Postgres database
 - b. Rabbitmq
 - c. Goinventory service
 - d. GoFabric service
 - e. GoTenant service

- f. GoSwitch service
- g. Konga
- h. Kong-API-gateway
- i. Metricbeats ELK Dashboard Stats/healthcheck monitor (available in ELK dashboard)
- j. Filebeat-deployment
- k. Kibana
- l. Logstash
- m. Elasticsearch

If the previous deployment stack is already running, the script will present the user with three options:

1. Remove the current stack: The user can simply remove the entire stack with this option.
2. Upgrade or Redeploy: If users are running the deployment.sh script from the new tar-ball , they can choose to upgrade the setup without wiping out the current database volume. Similarly, they will have the same option if the script is re-run from the same folder, in which case the stack will only get re-deployed.
3. Quit: No change in the current stack.

REST API Guide

Once EFA is installed the API guide is available as a HTML reference. <http://<host ip>:8002>. The API guide is a good reference to help integrate with other automation tools. The REST API is specified via OpenAPI/Swagger.

IP Fabric/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 Clos topology and on which one can configure underlay and/or overlay.

Fabric service provides following features:

1. 3-stage Clos automation
2. 5-stage Clos automation
3. Multi-Fabric automation
4. Fabric topology view
5. Fabric validation, error reporting, and recovery
6. 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 would be tagged against the devices and can be retrieved from the CLI/REST APIs for use in taking corrective or remedial actions.

Day-0 prerequisites on SLX devices

- Management IP addresses must be configured on all switches.
- SLX devices should have the appropriate firmware version as indicated in the “Supported Platform Matrix” section.
- On the SLX 9850, Fabric links must be enabled manually , through “no shut”.
- On the SLX 9540, the appropriate TCAM profile must be set and the switch rebooted, as follows.

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


- Breakout ports, if any, on SLX devices must be configured manually. Refer to the appropriate SLX-OS Management Guide for configuration steps for different platforms.
- Follow the EVD guidelines for Fabric and ICL port connections on leaf nodes for the following:
 - SLX 9140, SLX 9540: 0/49 -- 0/54
 - SLX 9030: 0/49 -- 0/52

Automating Clos Fabric provisioning

This consists of four basic steps.

- 1) Create the Fabric.
 - a. Modify default Fabric settings if required.
- 2) Register devices with inventory (autodiscovery of various “Assets” of the registered device), 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
 - a. Clos physical topology (physical connections as per EVD) between devices is validated.
 - b. Based on device roles and topology, an intended configuration is generated.
- 3) Add devices to the Fabric.
 - a. Clos physical topology (physical connections as per EVD) between devices is validated.
 - b. Based on device roles and topology, an intended configuration is generated.
- 4) Configure the Fabric
 - a. Provision the underlay and overlay configuration on the devices
 - i. Basically, push all the L2 and L3 configurations necessary to form an IP Fabric down to the SLX devices.
 - ii. Validate the configurations pushed to the devices.

Automating 3-stage Clos

This topology has two layers of devices, leaf and spine. All the links between the leaf and spine must be connected. Spine nodes should not be interconnected with each other. Refer to the following figure.

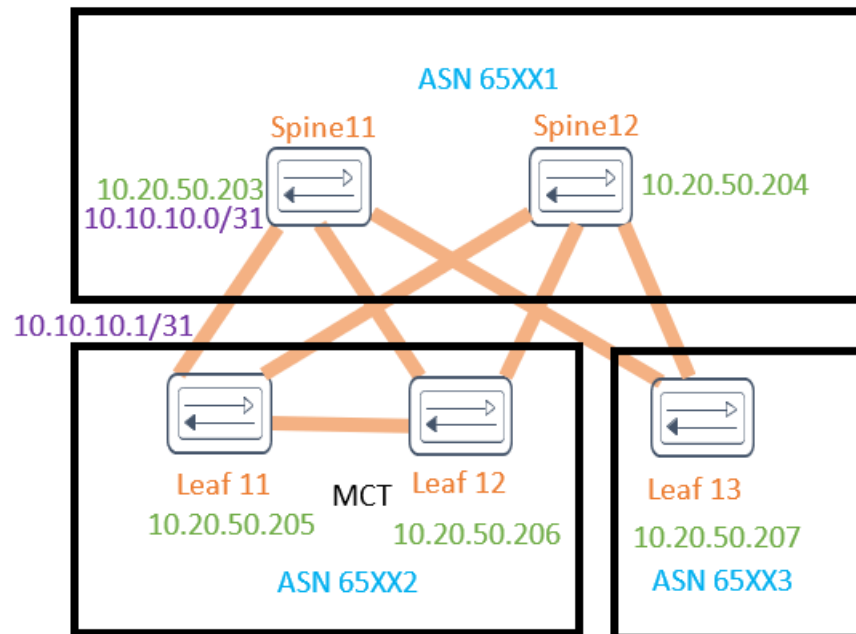


Figure 4. Automating 3-stage Clos

1. Create a 3-stage Fabric.

```
$ dca fabric create --name stage3 --stage 3
```

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

```
$ dca fabric device add-bulk --name stage3 --leaf  
10.20.50.205,10.20.50.206,10.20.50.207 --spine  
10.20.50.203,10.20.50.204 --username admin --password  
password
```

NOTE: There is also an option to add/register single device at a time. Refer to [Add a device to the Fabric](#).

3. Configure the Fabric.

```
$ dca fabric configure --name stage3
```

Automating 5-stage Clos

This topology allows three types of devices to be added: leaf, spine, and super-spine. All the links between the leaf and spine must be connected. 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.

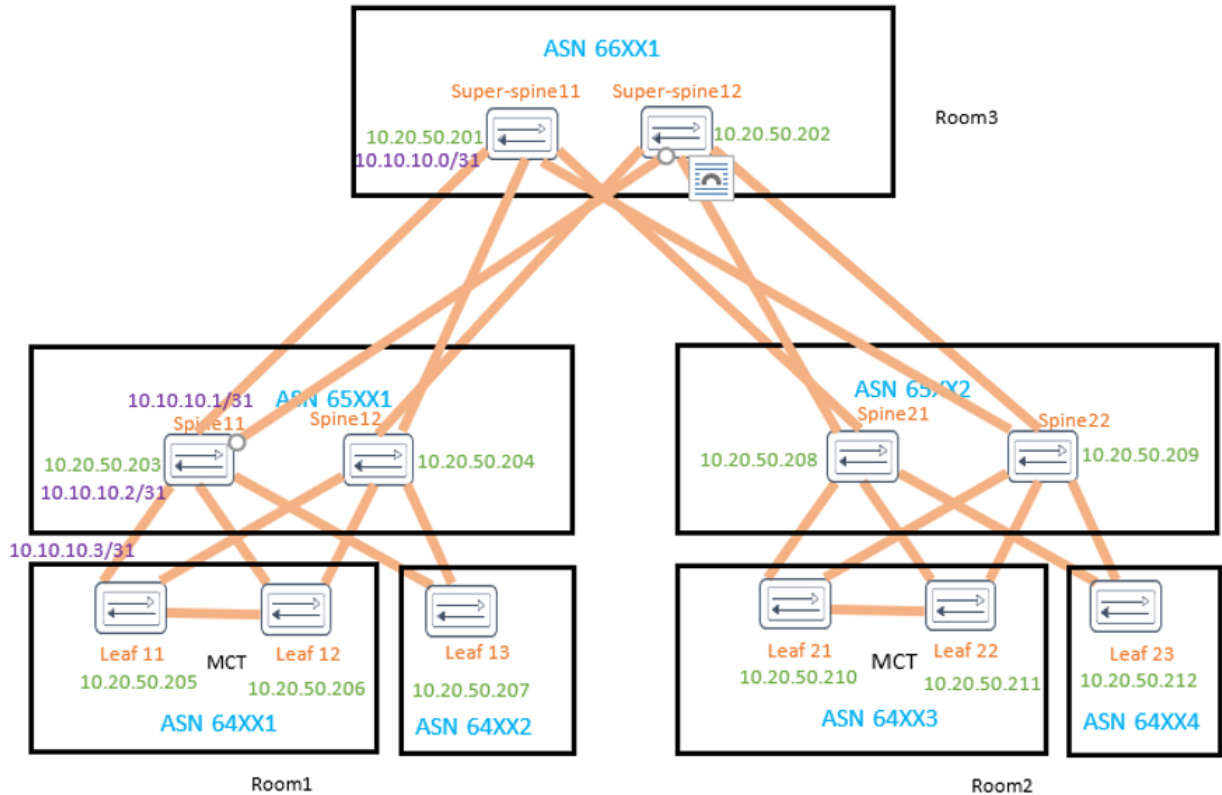


Figure 5. Automating 5-stage Clos

1. Create a 5-stage Fabric.

```
$ dca fabric create --name stage5 --stage 5
```

2. Add devices/PoDs to the 5-stage Fabric. The user must provide device credentials as part of this command if the devices are not already registered with the inventory.

- a. Add devices in PoD "Room1" and associate it to PoD "Room3".

```
$ dca fabric device add-bulk --name stage5 --leaf
10.20.50.205,10.20.50.206-207 --spine 10.20.50.203-204 --super-
spine 10.20.50.201-202 --three-stage-pod Room1 --five-stage-pod
Room3 --username admin --password password
```

b. Add an additional PoD "Room2".

```
$ dca fabric device add-bulk --name stage5 --leaf  
10.20.50.210-212 --spine 10.20.50.208-209 --three-stage-pod Room2  
--username admin --password password
```

3. Configure the Fabric.

```
$ dca fabric configure --name stage5
```

Fabric Validation, Error Reporting, and Recovery

This section lists a variety of functions and their related commands.

Get Fabric summary

CLI	dca fabric show summary [--name <Fabric-name>]
Behavior	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.
Field Description	--name string: Name of the Fabric.

Example:

```
$ dca fabric show summary -name fabric1
+-----+-----+
|                NAME                | VALUE |
+-----+-----+
| Fabric Name                         | fabric1 |
| Fabric Stage                         | 3      |
| Fabric Description                    |        |
| Number Of Pods                       | 0      |
| Number Of Single Homed Leaf Nodes    | 0      |
| Number Of Multi Homed Leaf Nodes     | 6      |
| Number Of Spine Nodes                 | 1      |
| Number Of Super Spine Nodes          | 0      |
| Number Of not provisioned Nodes      | 7      |
| Number Of Provisioned Nodes          | 0      |
| Number Of Provisioned Failed Nodes   | 0      |
| Number Of config ready Nodes         | 5      |
| Number Of config in sync Nodes       | 0      |
| Number Of config generation error Nodes | 2      |
| Number Of config refreshed Nodes     | 0      |
+-----+-----+
```

Get Fabric details

CLI	dca fabric show [--name <Fabric-name>]
Behavior	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.
Field Description	--name string: Name of the Fabric. --export string: Export Fabric details to CSV file.

Example:

```
$ dca fabric show --name stage3
```

```
Fabric Name: fabric1, Fabric Description: , Fabric Stage: 3
```

IP ADDRESS	POD	HOST NAME	ASN	ROLE	DEVICE STATE	APP STATE	CONFIG GEN REASON	PENDING CONFIGS	VTLB ID	LB ID
10.24.80.136		Ced-136	64512	spine	provisioned	cfg refreshed	DD	SYSP-U,BGP-C,INTIP-C	NA	1
10.24.80.134		SLX	65000	leaf	provisioned	cfg in-sync	NA	NA	2	1
10.24.80.135		SLX	65000	leaf	provisioned	cfg in-sync	NA	NA	2	1
10.25.225.11		Avalanche-01	65001	leaf	not provisioned	cfg ready	DA	SYSP-C,MCT-C,MCT-PA,BGP-C,INTIP-C,EVPN-C,0-C	2	1
10.25.225.46		Avalanche-02	65001	leaf	not provisioned	cfg ready	DA	SYSP-C,MCT-C,MCT-PA,BGP-C,INTIP-C,EVPN-C,0-C	2	1

CONFIG GEN REASON:
LD - Link Delete, LA - Link Add, IU - Interface Update
MD - MCT Delete, OD - Overlay Gateway Delete, OU - Overlay Gateway Update, ED - Evpn Delete
DD - Dependent Device Update, DA - Device Add, DR - Device ReAdd, ASN - Asn Update, HN - HostName Update, NA - Not Applicable

PENDING CONFIGS:
MCT - MCT Cluster, 0 - Overlay Gateway, SYSP - System Properties, INTIP - Interface IP
C/D/U - Create/Delete/Update, PA/PD - Port Add/Port Delete

For App or Device Error/Failure reason, run "dca fabric error show" for details
For config refresh reason, run "dca fabric debug config-gen-reason" for details

Get Fabric device config gen reason

This debug CLI can be used to get the configuration generation reason for a particular Fabric device.

CLI	dca fabric debug config-gen-reason --device <device-ip> --name <Fabric-name>
Field Description	--device: Device IP address --name: Name of the Fabric to which the device belongs.

Example:

IP ADDRESS	DEVICE ROLE	INTERFACE TYPE	INTERFACE NAME	LINK STATE	REMOTE IP ADDRESS	REMOTE DEVICE ROLE	REMOTE INTERFACE TYPE	REMOTE INTERFACE NAME
10.25.225.11	leaf	ethernet	0/20	added	10.25.225.46	leaf	ethernet	0/20
10.25.225.11	leaf	ethernet	0/19	added	10.25.225.46	leaf	ethernet	0/19
10.25.225.11	leaf	ethernet	0/49	added	10.24.80.136	spine	ethernet	0/31

config generate reason [Success]

Get physical topology

CLI	dca fabric topology show physical --name <Fabric name>
Behavior	Displays all the physical connectivity of the devices in a Fabric.
Field Description	--name string: Name of the Fabric.

Example:

```
$ dca fabric topology show physical --name stage5
```

SOURCE NODE IP	SOURCE NODE ROLE	DESTINATION NODE IP	DESTINATION NODE ROLE	SOURCE NODE INTERFACE	DESTINATION NODE INTERFACE	SOURCE DEVICE MULTI HOPPED	DESTINATION DEVICE MULTI HOPPED
10.20.50.210	Spine	10.20.50.203	Leaf	0/1:1	0/1	false	true
10.20.50.210	Spine	10.20.50.203	Leaf	0/1:2	0/2	false	true
10.20.50.210	Spine	10.20.50.203	Leaf	0/1:3	0/3	false	true
10.20.50.210	Spine	10.20.50.203	Leaf	0/1:4	0/4	false	true
10.20.50.210	Spine	10.20.50.204	Leaf	0/2:1	0/1	false	true
10.20.50.210	Spine	10.20.50.204	Leaf	0/2:2	0/2	false	true
10.20.50.210	Spine	10.20.50.204	Leaf	0/2:3	0/3	false	true
10.20.50.210	Spine	10.20.50.204	Leaf	0/2:4	0/4	false	true
10.20.50.210	Spine	10.20.50.212	Leaf	0/11:1	0/1	false	true
10.20.50.210	Spine	10.20.50.212	Leaf	0/11:2	0/2	false	true
10.20.50.210	Spine	10.20.50.212	Leaf	0/11:3	0/3	false	true
10.20.50.210	Spine	10.20.50.212	Leaf	0/11:4	0/4	false	true
10.20.50.210	Spine	10.20.50.213	Leaf	0/12:1	0/1	false	true
10.20.50.210	Spine	10.20.50.213	Leaf	0/12:2	0/2	false	true
10.20.50.210	Spine	10.20.50.213	Leaf	0/12:3	0/3	false	true
10.20.50.210	Spine	10.20.50.213	Leaf	0/12:4	0/4	false	true
10.20.50.210	Spine	10.18.124.50	SuperSpine	0/19	0/19	false	false
10.20.50.210	Spine	10.20.51.238	SuperSpine	0/20	0/20	false	false
10.20.50.210	Spine	10.20.50.203	Leaf	0/21:1	0/21	false	true
10.20.50.210	Spine	10.20.50.204	Leaf	0/21:2	0/21	false	true
10.20.50.210	Spine	10.20.50.212	Leaf	0/22	0/49	false	true
10.20.50.210	Spine	10.20.50.213	Leaf	0/23	0/49	false	true
10.20.50.210	Spine	10.18.124.46	SuperSpine	0/26	1/1	false	false
10.20.50.210	Spine	10.18.124.10	SuperSpine	0/28	2/1	false	false
10.20.50.210	Spine	10.20.50.203	Leaf	0/29	0/49	false	true
10.20.50.210	Spine	10.20.50.204	Leaf	0/30	0/49	false	true
10.20.50.213	Leaf	10.20.50.210	Spine	0/1	0/12:1	true	false
10.20.50.213	Leaf	10.20.50.210	Spine	0/2	0/12:2	true	false
10.20.50.213	Leaf	10.20.50.210	Spine	0/3	0/12:3	true	false
10.20.50.213	Leaf	10.20.50.210	Spine	0/4	0/12:4	true	false
10.20.50.213	Leaf	10.20.50.211	Spine	0/19	0/11:1	true	false
10.20.50.213	Leaf	10.20.50.211	Spine	0/20	0/11:2	true	false
10.20.50.213	Leaf	10.20.50.211	Spine	0/21	0/11:3	true	false
10.20.50.213	Leaf	10.20.50.211	Spine	0/22	0/11:4	true	false
10.20.50.213	Leaf	10.20.50.212	Leaf	0/25	0/25	true	true
10.20.50.213	Leaf	10.20.50.212	Leaf	0/37	0/37	true	true
10.20.50.213	Leaf	10.20.50.212	Leaf	0/38	0/38	true	true
10.20.50.213	Leaf	10.20.50.212	Leaf	0/39	0/39	true	true
10.20.50.213	Leaf	10.20.50.212	Leaf	0/40	0/40	true	true
10.20.50.213	Leaf	10.20.50.210	Spine	0/49	0/23	true	false
10.20.50.213	Leaf	10.20.50.212	Leaf	0/50	0/50	true	true
10.20.50.213	Leaf	10.20.50.211	Spine	0/52	0/23	true	false
10.20.50.207	Leaf	10.20.50.208	Leaf	0/33	0/33	true	true
10.20.50.207	Leaf	10.20.50.208	Leaf	0/34	0/34	true	true
10.20.50.207	Leaf	10.20.50.208	Leaf	0/35	0/35	true	true
10.20.50.207	Leaf	10.20.50.208	Leaf	0/36	0/36	true	true
10.20.50.207	Leaf	10.20.51.238	SuperSpine	0/49	0/3	true	false
10.20.50.207	Leaf	10.18.124.50	SuperSpine	0/50	0/3	true	false
10.20.50.207	Leaf	10.18.124.46	SuperSpine	0/51	2/41	true	false
10.20.50.207	Leaf	10.18.124.10	SuperSpine	0/52	2/41	true	false
10.20.51.238	SuperSpine	10.20.50.207	Leaf	0/3	0/49	false	true
10.20.51.238	SuperSpine	10.20.50.208	Leaf	0/4	0/49	false	true
10.20.51.238	SuperSpine	10.20.50.210	Spine	0/19	0/20	false	false
10.20.51.238	SuperSpine	10.20.50.211	Spine	0/20	0/20	false	false
10.20.50.212	Leaf	10.20.50.211	Spine	0/52	0/21	true	false

--- Time Elapsed: 20.278194357s ---

Get underlay topology

CLI	dca fabric topology show overlay --name <Fabric name>
Behavior	Displays underlay connectivity, as well as BGP neighborhood and the state of BGP sessions of the devices in a Fabric.
Field Description	--name string: Name of the Fabric.

Example:

```
$dca fabric topology show underlay --name stage5
```

SOURCE DEVICE IP	DESTINATION DEVICE IP	SOURCE DEVICE ROUTER ID	NEIGHBOR IP	UNDERLAY STATE
10.20.50.210	10.20.50.201	172.31.254.201	10.10.10.16	ESTAB
10.20.50.210	10.18.124.10	172.31.254.201	10.10.10.23	ESTAB
10.20.50.210	10.20.50.212	172.31.254.201	10.10.10.97	ESTAB
10.20.50.210		172.31.254.201	10.10.10.137	ESTAB
10.20.50.210		172.31.254.201	10.10.10.138	ESTAB
10.20.50.210		172.31.254.201	10.10.10.155	ESTAB
10.20.50.210		172.31.254.201	10.10.10.164	ESTAB
10.20.50.210		172.31.254.201	10.10.11.52	ESTAB
10.20.50.210		172.31.254.201	10.10.11.54	ESTAB
10.20.50.210		172.31.254.201	10.10.11.74	ESTAB
10.20.50.210		172.31.254.201	10.10.11.78	ESTAB
10.20.50.210		172.31.254.201	10.10.11.82	ESTAB
10.20.50.210	10.20.50.203	172.31.254.201	10.10.11.90	ESTAB
10.20.50.210	10.20.50.204	172.31.254.201	10.10.11.96	ESTAB
10.20.50.210	10.20.50.204	172.31.254.201	10.10.11.98	ESTAB
10.20.50.210		172.31.254.201	10.10.11.108	ESTAB
10.20.50.210		172.31.254.201	10.10.11.120	ESTAB
10.20.50.210		172.31.254.201	10.10.11.127	ESTAB
10.20.50.210		172.31.254.201	10.10.11.147	ESTAB
10.20.50.210		172.31.254.201	10.10.11.176	ESTAB
10.20.50.210		172.31.254.201	10.10.11.209	ESTAB
10.20.50.210	10.20.50.201	172.31.254.201	10.10.11.215	ESTAB
10.20.50.210	10.20.50.202	172.31.254.201	10.10.11.221	ESTAB
10.20.50.210	10.20.50.201	172.31.254.201	10.10.11.222	ESTAB
10.20.50.210	10.20.50.206	172.31.254.201	10.10.11.236	ESTAB
10.20.50.210	10.20.50.205	172.31.254.201	10.10.11.242	ESTAB
10.20.50.204		172.31.254.204	10.10.11.53	ESTAB
10.20.50.204		172.31.254.204	10.10.11.55	ESTAB
10.20.50.204		172.31.254.204	10.10.11.58	ESTAB
10.20.50.204		172.31.254.204	10.10.11.63	ESTAB
10.20.50.204		172.31.254.204	10.10.11.65	ESTAB
10.20.50.204		172.31.254.204	10.10.11.75	ESTAB
10.20.50.204		172.31.254.204	10.10.11.109	ESTAB
10.20.50.204		172.31.254.204	10.10.11.126	ESTAB
10.20.50.204		172.31.254.204	10.10.11.128	ESTAB
10.20.50.204	10.20.50.211	172.31.254.204	10.10.11.142	ESTAB
10.20.50.204	10.20.50.204	172.31.254.204	10.10.11.144	ESTAB
10.20.50.204		172.31.254.204	10.10.11.146	ESTAB
10.20.50.204		172.31.254.204	10.20.20.46	ESTAB
10.20.50.214		172.31.254.215	10.10.11.203	ESTAB
10.20.50.214		172.31.254.215	10.10.11.226	ESTAB
10.20.50.212	10.20.50.211	172.31.254.205	10.10.11.95	ESTAB
10.20.50.212	10.20.50.211	172.31.254.205	10.10.11.97	ESTAB
10.20.50.212	10.20.50.211	172.31.254.205	10.10.11.99	ESTAB
10.20.50.212	10.20.50.211	172.31.254.205	10.10.11.101	ESTAB
10.20.50.212		172.31.254.205	10.20.20.49	ESTAB

--- Time Elapsed: 17.268608655s ---

Get overlay topology

CLI	dca fabric topology show overlay --name <Fabric name>
Behavior	Displays overlay connectivity of the devices in a Fabric.
Field Description	--name string: Name of the Fabric.

Example:

```
$dca fabric topology show overlay --name stage5
```

```

-----
| OVERLAY ECAP TYPE | SOURCE LEAF IP | DESTINATION LEAF IP | SOURCE VTEP IP | DESTINATION VTEP IP | OVERLAY ADMIN STATE | OVERLAY OPER STATE | OVERLAY BFD STATE |
-----
| vxlan             | 10.24.80.63    | 10.24.80.60,10.24.80.61 | 172.31.254.41 | 172.31.254.36 | up | up | down |
| vxlan             | 10.24.80.60,10.24.80.61 | 10.24.80.63 | 172.31.254.36 | 172.31.254.41 | up | up | down |
-----
--- Time Elapsed: 6.16503575s ---

```

Get device config

CLI	dca fabric show-config --name <Fabric name> [--device-role <string>]
Behavior	Displays the device configuration in JSON format.
Field Description	--name string: Name of the Fabric --device-role string: Filters the config based on device-role (super-spine/spine/leaf/all) (default "all")

Example:

```
$ dca fabric show-config --name stage5
```

Output Snippet:

```

{{
  "Fabric_name": "stage5",
  "switches": [
    {
      "switch": "10.20.50.212",
      "role": "Leaf",
      "bgp": {
        "local_as": "65002",
        "network": "172.31.254.22/32",
        "max_paths": "8",
        "bfd_rx": "0",
        "bfd_tx": "0",
        "bfd_multiplier": "0",
        "peer_groups": [
          {

```

```

        "name": "podA-spine-group",
        "description": "To Spine",
        "bfd": "",
        "remote_as": "64512"
    }
],
"neighbors": [
    {
        "remote_ip": "10.10.11.61",
        "remote_as": "0",
        "peer_group": "podA-spine-group",
        "bgp_multihop": ""
    },
    {
        "remote_ip": "10.10.11.63",
        "remote_as": "0",
        "peer_group": "podA-spine-group",
        "bgp_multihop": ""
    },
    {
        "remote_ip": "10.10.11.65",
        "remote_as": "0",
        "peer_group": "podA-spine-group",
        "bgp_multihop": ""
    },
    },
    <Output truncated>

    {
        "interface_name": "0/37",
        "interface_type": "ethernet",
        "interface_ip_address": "",
        "interface_description": ""
    },
    {
        "interface_name": "0/51",
        "interface_type": "ethernet",
        "interface_ip_address": "",
        "interface_description": ""
    },
    <Output truncated>
},
{
    "switch": "10.20.50.213",
    "role": "Leaf",
    "bgp": {
        "local_as": "65002",
        "network": "172.31.254.93/32",
        "max_paths": "8",
        "bfd_rx": "0",
        "bfd_tx": "0",
        "bfd_multiplier": "0",
        "peer_groups": [
            {
                "name": "podA-spine-group",
                "description": "To Spine",
                "bfd": "",
                "remote_as": "64512"
            }
        ]
    }
}

```

```

    }
  ],
  "neighbors": [
    {
      "interface_name": "1",
      "interface_type": "loopback",
      "interface_ip_address": "172.31.254.27",
      "interface_description": ""
    },
    {
      "interface_name": "2",
      "interface_type": "loopback",
      "interface_ip_address": "172.31.254.23",
      "interface_description": ""
    }
  ]
}
]
}}
-- Time Elapsed: 4.8378569s --

```

Clear device config

CLI	dca fabric debug clear-config --device <device-ip>
Behavior	Clears the underlay/overlay configuration from the device, to recover the device from erroneous conditions.
Field Description	--device <device-ip> : Device IP address. --username string : Username of device. --password string : Password of device.

Example:

```

$ dca fabric debug clear-config --device
10.24.80.134,10.24.80.135,10.24.80.136,10.25.225.163,10.25.225.167,10.
25.225.172,10.25.225.11,10.25.225.46,10.24.85.74,10.24.85.76

```

Get Fabric execution

CLI	dca fabric execution show [--id <execution id> --limit <num-of-executions> --status { failed succeeded all }]
Behavior	Displays the REST API executions of Fabric Service.
Field Description	<p>--id string : Filters the executions based on execution id. "limit" and "status" flags are ignored when "id" flag is given.</p> <p>--limit int32 : Limits the number of executions to be listed. Value "0" lists all the executions. (default = 10)</p> <p>--status string : Filters the executions based on the status(failed/succeeded/all) (default "all")</p>

Example:

```
$ dca fabric execution show
```

ID	COMMAND	STATUS	START TIME	END TIME
1eb53796-02a5-42b5-9b98-b05385e31f5d	fabric configure:Validate Fabric	Completed(81.858301ms)	2019-03-05T21:39:45Z	2019-03-05T21:39:46Z
a63048f7-b3c8-4c6e-92ed-5eablec79baa	fabric configure:Validate Fabric	Completed(56.87023ms)	2019-03-05T21:39:42Z	2019-03-05T21:39:42Z
17d81ee6-bd2d-4ee2-a626-a386160d56bb	fabric show --name	Completed(93.57931ms)	2019-03-05T19:45:08Z	2019-03-05T19:45:08Z
84ae086a-5ccb-4328-a567-be6a5c759772	fabric show	Completed(192.620108ms)	2019-03-05T19:45:01Z	2019-03-05T19:45:01Z
36673930-b8a5-405c-b78f-ac96bd844d8a	fabric show	Completed(168.251829ms)	2019-02-16T17:57:16Z	2019-02-16T17:57:17Z
12f68337-fc9e-46fe-9e50-8e64172622c7	fabric	Completed(1m2.25919261s)	2019-02-16T17:42:57Z	2019-02-16T17:43:59Z
b28134dd-62fb-43b2-971e-37e5030f099f	configure:ConfigureFabric fabric configure:Validate Fabric	Completed(86.467771ms)	2019-02-16T17:42:57Z	2019-02-16T17:42:57Z
b7209e8a-2ed3-4904-8d2d-b920de3fe801	fabric configure:Validate Fabric	Completed(71.951742ms)	2019-02-16T17:41:51Z	2019-02-16T17:41:51Z
8e90fa5d-fe9d-411d-8210-0f577e23e847	fabric device add	Completed(944.123529ms)	2019-02-16T17:41:50Z	2019-02-16T17:41:51Z
7157069d-2d31-4268-8d5f-a0fd9d03b025	fabric device remove	Completed(473.375627ms)	2019-02-16T17:40:56Z	2019-02-16T17:40:57Z

--- Time Elapsed: 14.481365ms ---

Get Fabric error

Any validation errors (topology and configuration errors) are reported to the user for corrective action, when a device is added to the Fabric and also during Fabric configuration. Any errors occurred during add, validate and configure phases persist in the DB and the following CLI lists the device errors.

CLI	dca fabric error show --name <Fabric-name> [--export <file-name>]
Behavior	Displays Fabric validation errors.
Field Description	--name string : Name of the Fabric. --export string : Exports Fabric details to a CSV file.

Example:

```
$ dca fabric error show -name stage3
```

```
+-----+-----+-----+-----+-----+
| FABRIC NAME | IP ADDRESS | ROLE | ERROR TYPE | ERROR REASON |
+-----+-----+-----+-----+-----+
| stage3      | 10.26.10.158 | Leaf | CONFIG ERROR | operation: Poll for management |
|             |              |      |              | cluster status Management     |
|             |              |      |              | Cluster is not operational.   |
|             |              |      |              | Polling timed out             |
| stage3      | 10.26.10.157 | Leaf | CONFIG ERROR | operation: Poll for management |
|             |              |      |              | cluster status Management     |
|             |              |      |              | Cluster is not operational.   |
|             |              |      |              | Polling timed out             |
+-----+-----+-----+-----+-----+
--- Time Elapsed: 96.122581ms ---
```

Get supportSave

CLI	dca supportsave
Behavior	Collects supportSave data, which comprises of the Inventory, Tenant and Fabric service logs and the database dump of all the aforementioned services. It also includes the installer logs.

Example:

```
$ dca supportsave
Version : 2.0.0
Time Stamp: 19-01-29:17:20:28
Support Save File: /var/log/dcapp/dcapp_1548966881.logs.zip
-- Time Elapsed: 1.140725647s --
```

Tenant Network Provisioning

Overview

Tenant Services provides a unified means to create and manage the Tenants and their associated networks on the LEAF nodes of underlay and/or overlay IP Fabric. The Tenant network provisioning is allowed only on the LEAF nodes of the Fabric.

Tenant Network Provisioning provides following features on 3/5-Stage Clos Fabric:

1. L2 and L3 extension between racks
2. VLAN scoping at the ToR level
3. VLAN scoping at the port level within a ToR
4. Multi-homed leaf using Multi-Chassis Trunk (MCT)

Tenant CLIs/REST APIs provide the following:

- A mechanism to create a Tenant network on various endpoints spread across multiple leaf nodes in the Fabric.

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

Day-0 prerequisites For Tenant Network Provisioning

- Management IP addresses must be configured on all switches.
- SLX devices must have the appropriate firmware version, as indicated in the “Supported Platform Matrix” section.
- 3/5-stage Clos Fabrics must be provisioned prior to Tenant network provisioning.

Tenant network creation

Tenant network creation involves the following steps to configure, manage, and associate networks in the Fabric:

1) Creating an Asset

Creating an Asset includes reserving the devices and their physical ports from the existing leaf devices within a Fabric.

- a. An Asset contains physical ports from the devices belonging to the same Fabric.

Default-Asset is auto created by Tenant Service, which owns all the physical ports in the data-center. Ports from Default-Asset can be carved out and assigned to other user assets. The ports are released back to Default-Asset once the user assets are deleted.

2) Creating a Tenant

- a. A Tenant is a logical entity with reserved resources being L2VNI, L3VNI, and Assets created in Step 1.
- b. If the Fabric type is overlay, based on the VNI setting in the Fabric, the VNI values can be set while the Tenant is created.
- c. An Asset cannot be part of multiple Tenants.
- d. Only one Asset from a Fabric can be owned by a Tenant.

Default-Tenant is auto created by Tenant Service, which owns all the resources (VNIs) of the data-center and owns Default-Asset. Resources from Default-Tenant can be carved out and assigned to user tenants. The resources are released back to Default-Tenant once the user tenants are deleted.

3) Creating a port-channel

- a. A port-channel can be created by providing the port-channel name and the interfaces.
- b. Provisioning of a port-channel includes creating the port-channel with Tenant service and pushing the configurations to the devices.
- c. Port Channel creation is allowed on the following:
 - Interfaces spanning across MCT leaf nodes (dual-homed)
 - Interfaces originating from same leaf device (single-homed)

4) Creating an endpoint group (EPG)

- a. An EPG is a logical entity that is a collection of physical or port-channel interfaces that belong to one Asset, with same set of port properties being ctag, switchport mode, and endpoint tracking.
- b. EPG creation involves configuring switching mode (access/trunk), ctag and endpoint tracking on the interfaces belonging to this endpoint group.
- c. No configurations are pushed to the device as part of EPG creation.
- d. If the EPG is being created with port-channel interfaces, the port-channel must be created before it is assigned to EPG.

5) Creating a VRF

VRF creation is required for an L3 Tenant network. VRFs that are created are not provisioned on the devices until the associated Tenant network is created.

6) Creating a Tenant network

- a. A Tenant network is an entity that uses various forwarding constructs such as VLAN/bridge domain and VRFs to connect multiple EPGs across the leaf nodes in a Fabric.
- b. All the L2/L3 configurations necessary to bring up the Tenant network are pushed to the devices.

Tenant network provisioning

The following illustrates automating L2 and L3 extension between the leaf devices and between racks (network creation).

NOTE: Bridge domain, IRB-BD, IRB-VE(L3VNI), VRF RD, RT, port-channel IDs, and MCT client IDs are auto-generated by the system

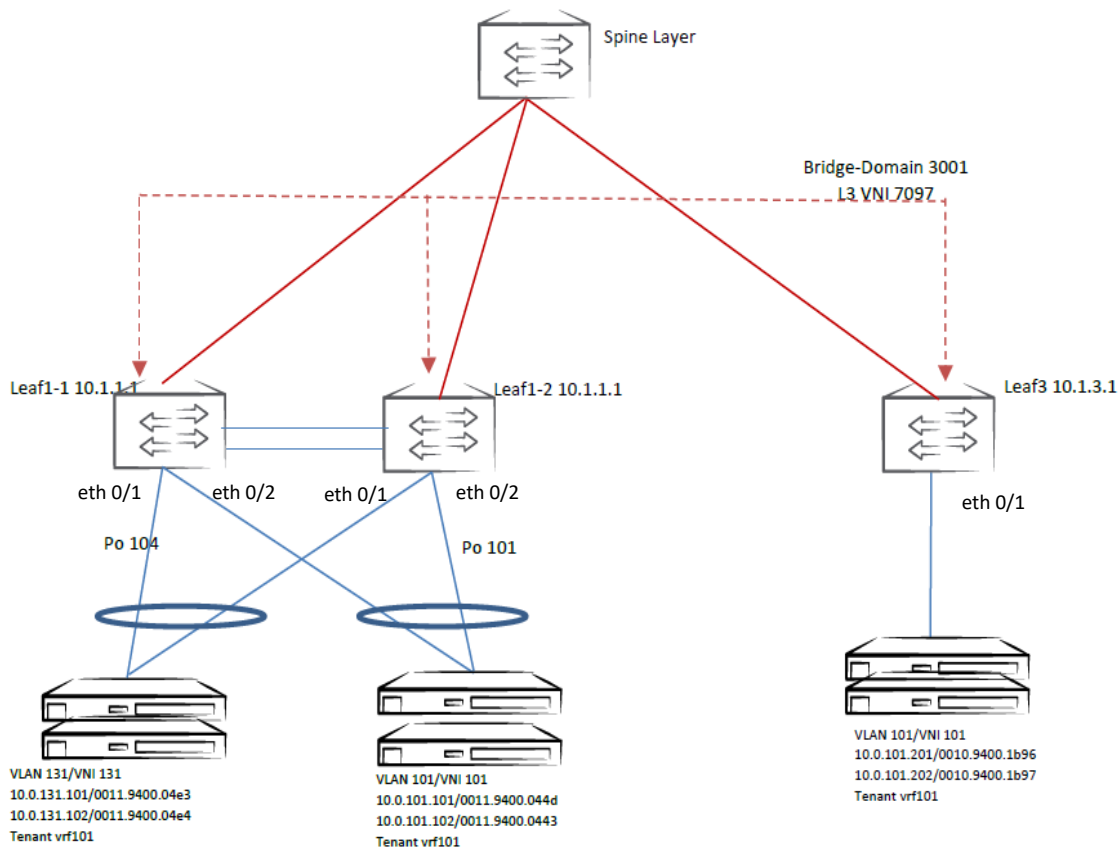


Figure 6. Automating L2/L3 extension between the racks

The following illustrates automating L2 and L3 extension between the leaf devices between racks (network creation).

Network creation

Do the following to create a network.

1. Create a VRF (only for L3 network).

```
$ dca tenant vrf create --name vrf101
```

NOTE: VRF properties such as route target value and type are auto-allocated if not provided.

2. Create a network that includes the creation of port-channels, EPG, and network.

Network -- Ctag - 101

```
$ dca tenant workflow network create --name network101
--po-name po101 --po-speed 10Gbps --po-ports
10.1.1.1[0/2],10.1.1.2[0/2] --ports 10.1.1.3[0/1] --ctag 101
--enable-bd --vrf vrf101 --anycast-ip 10.0.101.254/22
```

The above command does the following:

- Creates port-channel “po101” with member ports as “Eth 0/2” from 10.1.1.1 and 10.1.1.2
- Creates EPG with interfaces “po101” and “Eth 0/1” from 10.1.1.3 and associates it to ctag 101.
- Associates “vrf101” to this network.
- Configures anycast IP between MCT pairs.

Network -- Ctag - 131

```
$ dca tenant workflow network create --name network131
--po-name po131 --po-speed 10Gbps --po-ports
10.1.1.1[0/1],10.1.1.2[0/1] --ctag 131 --enable-bd --vrf
vrf101 --anycast-ip 10.0.131.254/22
```

The above command does the following:

- Creates port-channel “po131” with member ports as “Eth 0/1” from 10.1.1.1 and 10.1.1.2
- Creates EPG with interfaces “po131” and associates it to ctag 131.
- Associates “vrf101” to this network.
- Configures anycast IP between MCT pairs.

NOTE: There is an advanced CLI option whereby user can create a port-channel, EPG, and network by using individual CLIs. Refer to the section “Advanced CLI”.

Network deletion

Do the following to delete a network.

1. Delete single or list of Tenant Network by providing the Tenant network names

```
$ dca tenant workflow network delete --name network101,network131
```

2. Delete VRF

```
$ dca tenant vrf create --name vrf101
```

Advanced CLIs

This section presents advanced CLIs for network creation and deletion.

Tenant network provisioning using Default-Tenant and Default-Asset

Network creation

1. Create port-channels.

```
$ dca tenant po create --name po104 --port 10.1.1.1[0/1],10.1.1.2[0/1]
--speed 10Gbps --negotiation active
```

```
$ dca tenant po create --name po101 --port 10.1.1.1[0/2],10.1.1.2[0/2]
--speed 10Gbps --negotiation active
```

2. Create an EPG.

```
$ dca tenant epg create --name epg1 --po po104 --ctag 131
```

```
$ dca tenant epg create --name epg2 --po po101 --port
10.1.3.1[0/1] --ctag 101
```

NOTE: EPG creation assigns the same CTAG to all the ports/port-channels. If multiple CTAGs for the same ports/port-channels must be associated, then multiple EPGs and their corresponding CTAGs must be created accordingly, as in the following example.

```
$ dca tenant epg create --name epg1 --po po104 --ctag 131
```

```
$ dca tenant epg create --name epg2 --po po104 --ctag 132
```

3. Create a VRF.

```
$ dca tenant vrf create --name vrf101
```

NOTE: VRF router target value and type are auto-allocated if not provided.

4. Create a Tenant network by providing the EPG names and network properties

```
$ dca tenant network create --name network131
--epg epg1 --enable-bd --vrf vrf101
--anycast-ip 10.0.131.254/22
```

```
$ dca tenant network create --name network101
--epg epg2 --enable-bd --vrf vrf101
--anycast-ip 10.0.101.254/22
```

NOTE: The **--enable-bd** command determines whether the configurations to be pushed onto the devices are VLAN or bridge-domain based.

Network deletion

1. Delete a Tenant network. L2/L3 configurations are deleted from the device.

```
$ dca tenant network delete --name network131
```

NOTE: If a VRF is mapped to multiple networks, it is deleted only if there are no networks associated to it.

2. Delete an EPG.

```
$ dca tenant epg delete --name epg2
```

3. Delete a VRF.

```
$ dca tenant vrf delete --name vrfRed
```

4. Delete port-channels.

```
$ dca tenant po delete --name po104
```

Tenant network provisioning using User-Tenant and User-Asset

Network creation

1. Create an Asset.

```
$ dca tenant asset create --name Asset-Group-1 --port  
10.1.1.1[0/1-2],10.1.1.2[0/1-2],10.1.1.3[0/1]
```

2. Create a Tenant.

```
$ dca tenant tenant create --name Tenant-Sales --asset  
Asset-Group-1 --l2-vni-count 50 --l3-vni-count 50
```

3. Create port-channels.

```
$ dca tenant po create --name po104 --port  
10.1.1.1[0/1],10.1.1.2[0/1] --speed 10Gbps --negotiation active  
--tenant Tenant-Sales
```

```
$ dca tenant po create --name po101 --port  
10.1.1.1[0/2],10.1.1.2[0/2] --speed 10Gbps --negotiation active  
--tenant Tenant-Sales
```

4. Create an EPG.

```
$ dca tenant epg create --name epg1 --po po104 --ctag 131
--tenant Tenant-Sales

$ dca tenant epg create --name epg2 --po po101 --
port 10.1.3.1[0/1] --ctag 101 --tenant Tenant-Sales
```

5. Create a Tenant network by providing the EPG names and network properties

```
$ dca tenant network create --name network131
--epg epg1 --enable-bd --vrf vrf101
--anycast-ip 10.0.131.254/22 --tenant Tenant-Sales

$ dca tenant network create --name network101
--epg epg2 --enable-bd --vrf vrf101
--anycast-ip 10.0.101.254/22 --tenant Tenant-Sales
```

Network deletion

1. Delete a Tenant network. L2/L3 configurations are deleted from the device.

```
$ dca tenant network delete --name network131 --tenant
Tenant-Sales
```

NOTE: If a VRF is mapped to multiple networks, it is deleted only if there are no networks associated to it.

2. Delete an EPG.

```
$ dca tenant epg delete --name epg2 --tenant Tenant-Sales
```

3. Delete a VRF.

```
$ dca tenant vrf delete --name vrfRed --tenant Tenant-Sales
```

4. Delete port-channels.

```
$ dca tenant po delete --name po104 --tenant Tenant-Sales
```

5. Delete Tenant.

```
$ dca tenant delete --name Tenant-Sales
```

6. Delete Asset.

```
$ dca tenant asset delete --name Asset-Group-1
```


Tenant Error Reporting and Recovery

Tenant network error reporting (Advanced CLIs)

The appropriate error messages are displayed during configuration changes. Note the following examples.

1. Any validation or topology errors are reported in real time for corrective action. No entry in the DB is persisted for invalid/validation errors. All the errors are displayed through the CLI output.

Example:

```
$ dca tenant network create --name net2 --epg net2-epg-3334
Network net3 Creation Failed:
  EndpointGroup: net2-epg-3334 not found
--- Time Elapsed: 12.6931ms ---
```

2. All device-level errors are displayed through CLI if the Tenant network fails during configuration or deconfiguration phase. Any errors that occur during add and delete phases are persisted in the DB and will displayed through the CLI. The errors can also be viewed by means of `'dca tenant network show --name <network name> --config'`.

Example:

```
$ dca tenant network create --name net2 --epg net2-epg-333
Network net2 Creation Failed:
  Device: 10.24.51.135 Port: ethernet:0/33 Error: CreateSwitchPortMode failed. 'Remove L3 configuration from the interface'
Execute CLI to get network config: 'dca tenant network show --name net2 --config'
--- Time Elapsed: 24.1968408s ---
```

3. Recovery of a Tenant network failed during creation

Tenant network creation

```
$ dca tenant network create --name net2 --epg net2-epg-333
Network net2 Creation Failed:
  Device: 10.25.225.58 Port: ethernet:0/33 Error: CreateSwitchPortMode failed. 'Remove L3 configuration from the interface'
Execute CLI to get network config: 'dca tenant network show --name net2 --config'
--- Time Elapsed: 12.7654903s ---
```

Show Tenant network

This CLI shows the configurations that are pushed and also displays device specific errors.

Output:

```
$ dca tenant network show --name net2 --config
```

Tenant Network Info:

Network Name	net2
Endpoint Groups	net2-epg-333
BD Enabled	false
Vrf	
Device	10.25.225.58
Vlan	333
Port	ethernet-0/33

Network net2 State Failed:

```
Device: 10.25.225.58 Port: ethernet:0/33 Error: CreateSwitchPortMode failed. 'Remove L3 configuration from the interface'  
--- Time Elapsed: 12.6914ms ---
```

Reason of failure

- The tenant network creation failed since the port 0/33 on device 10.25.225.58 was a router interface. The configuration present on the device conflicts with tenant network configuration.

Recovery steps

- Delete the network
- Take the corrective action (by removing the conflicting configurations from the device)
- Recreate the tenant network

Delete Tenant

This CLI removes the configurations that are pushed to the devices and deletes the tenant network.

```
$ dca tenant network delete --name net2
```

Network Deletion Succeeded.

```
--- Time Elapsed: 11.6106628s ---
```

Tenant network re-creation

```
$ dca tenant network create --name net2 --epg net2-epg-333
```

Network Creation Succeeded.

```
Execute CLI to get network config: 'dca tenant network show --name net2 --config'
```

```
--- Time Elapsed: 14.7537428s ---
```

```
$ dca tenant network show --name net2 --config
```

Tenant Network Info:

Network Name	net2
Endpoint Groups	net2-epg-333
BD Enabled	false
Vrf	
Device	10.25.225.58
Vlan	333
Port	ethernet-0/33
Vlan	333
SwitchportMode	trunk

```
--- Time Elapsed: 15.1713ms ---
```

4. Recovery of a Tenant network failed during deletion

If a tenant network deletion fails (for example, due to any conflicting configurations pushed to the device manually), once the corrective action is taken, do the following.

- Delete request can be rerun by means of `'dca tenant network delete --name <network-name>'`

5. Force delete of a Tenant network

A Tenant network deletion can fail due to miscellaneous reasons such as bad configurations on the devices or device becoming unreachable. Such networks can be deleted by using the `--force` option, which will clean up the Tenant network from the Tenant service and try to delete the configurations from the devices (any errors during this will be ignored).

Example:

```
$ dca tenant network delete --name net2 --force
Network Deletion Succeeded.
--- Time Elapsed: 11.6106628s ---
```

Tenant network workflow-error reporting

Workflow-error reporting is similar to that provided through Advanced CLIs, where the validation and topology errors are displayed through the CLI without creating an entry in the DB. All the device-level configuration or deconfiguration errors are displayed through the **network show** CLIs.

Example:

```
$ dca tenant workflow network create --name net1 --ports
10.25.225.58[0/33] --ctag 33333
Validation Failed:
  --ctag should be a value between 1 and 4090
--- Time Elapsed: 0s ---
```

Example:

```
$ dca tenant workflow network create --name net1 --ports
10.25.225.58[0/33] --ctag 333
Operation Failed:
  Network already exists
--- Time Elapsed: 9.7462ms ---
```

Please refer to step 3, 4&5 of section Tenant network error reporting (advanced CLIs) for recovering a failed tenant network

Logging/ELK Integration

In the DCA ecosystem, ELK (Elasticsearch, Logstash, Kibana) is implemented in the same network as that of 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
```

Ws	TAG	INFO
Who	source	Provides the information about which service the log belongs to.
Why	level	Provides the level of log, for example, whether a log is "Error" or "Info" or "Warning".
Where	_id	Each log is numbered with a unique ID.
What	json.msg	Contains details about the operation or error message in this field.
When	timestamp	Details about when the operation was performed. Gives exact time of log creation.

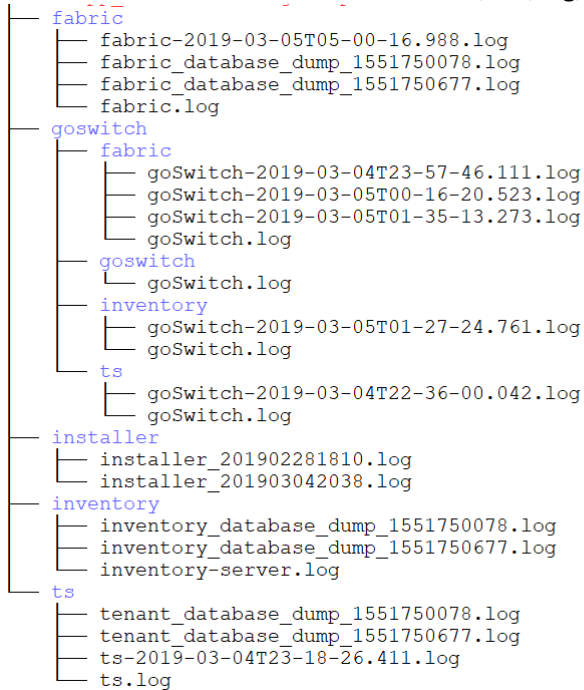
Infra level

```
# docker logs <container-id>
```

NOTE: To obtain a <container-id>, execute **docker ps**.

Application level

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



Logs are visualized on a Kibana dashboard. Below are some sample application-specific logs.

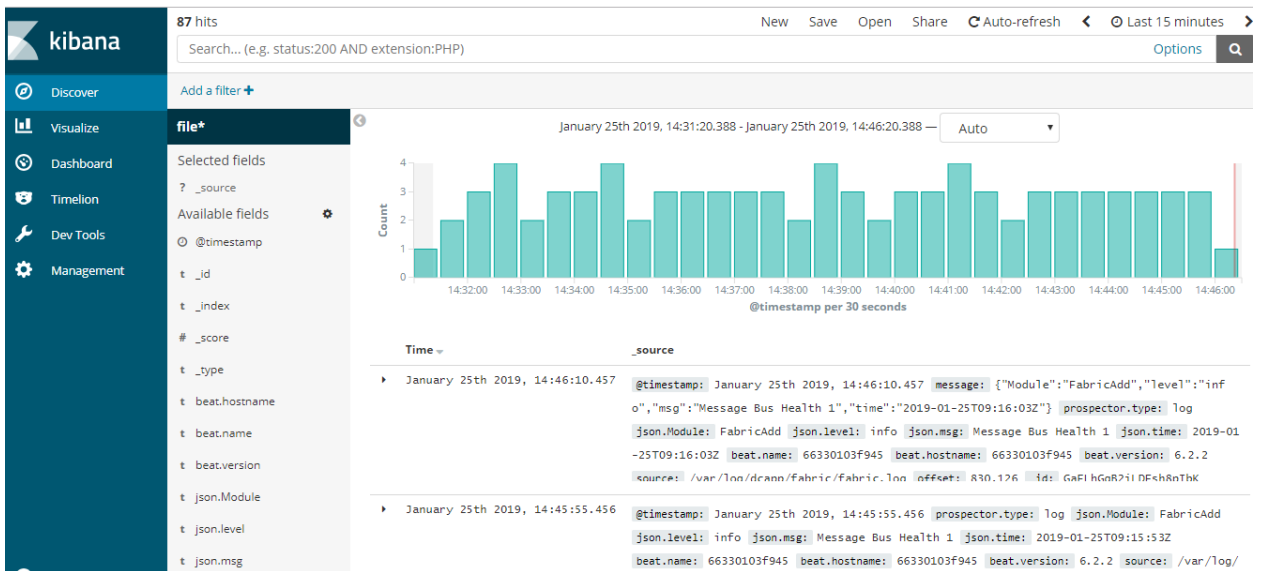
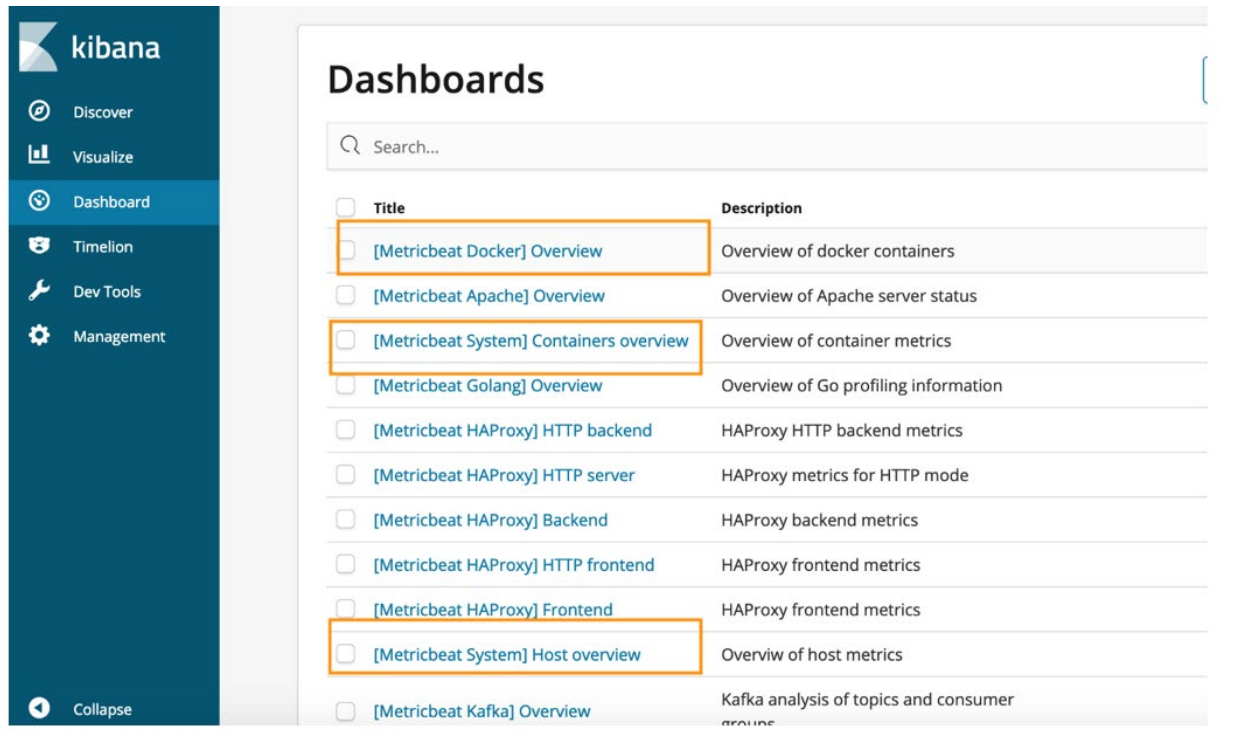


Figure 6. Kibana dashboard

Container stats and host stats

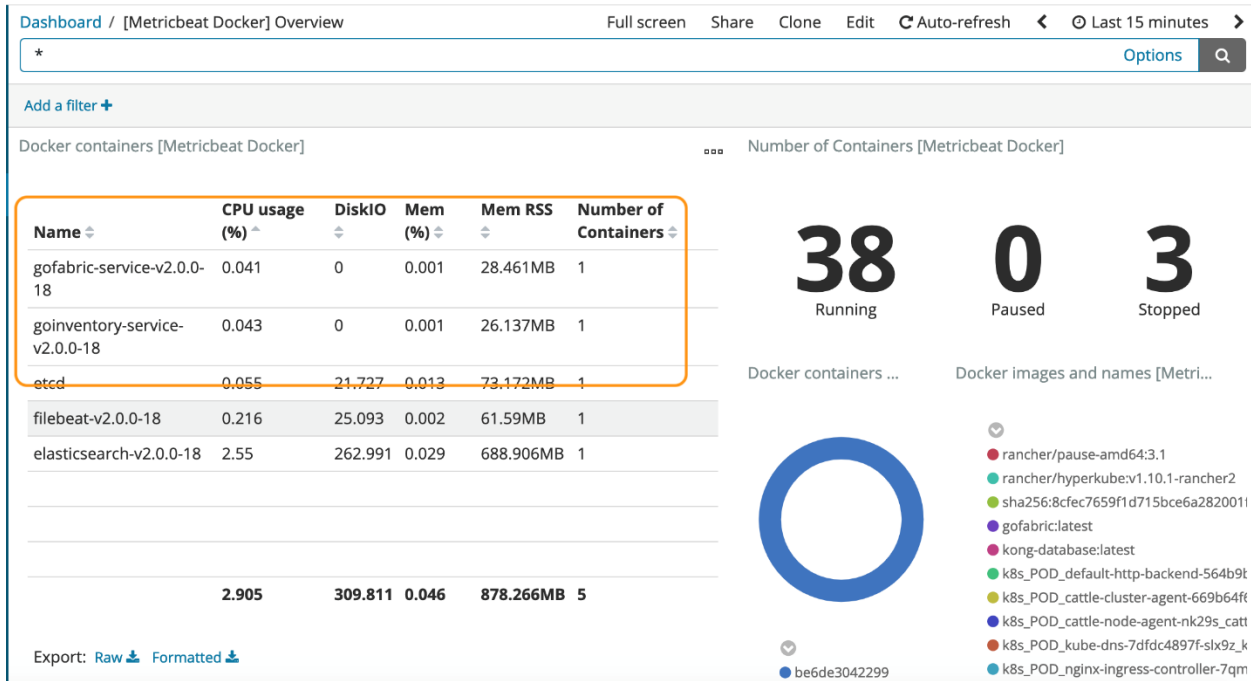
Metricbeat integration helps track the containers and host stats. This can help identify the memory leak and CPU usage per container or host. The Kibana stack launched as part of the DCAPP provides a number of Metricbeat dashboards.

Log in to Kibana and click on “Dashboard” to view the Metricbeat dashboards:



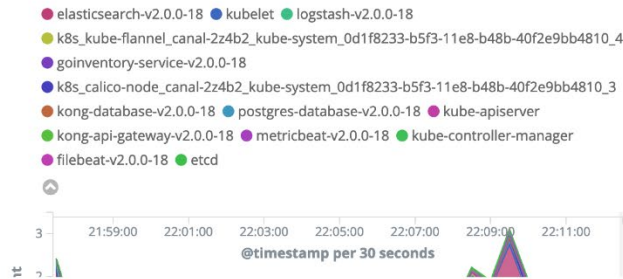
The following screenshots that show the container overview, CPU usage, and memory usage for containers.

Container overview

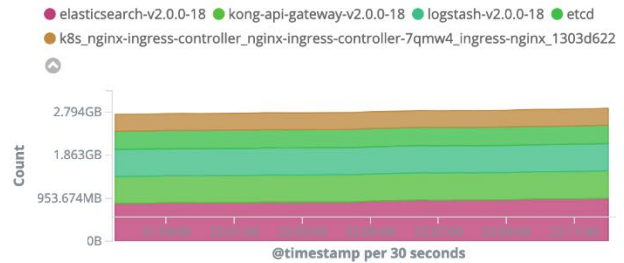


CPU and memory usage

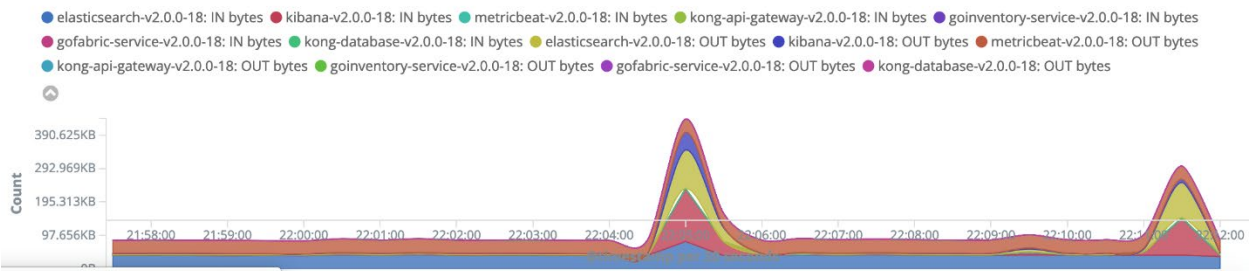
CPU usage [Metricbeat Docker]

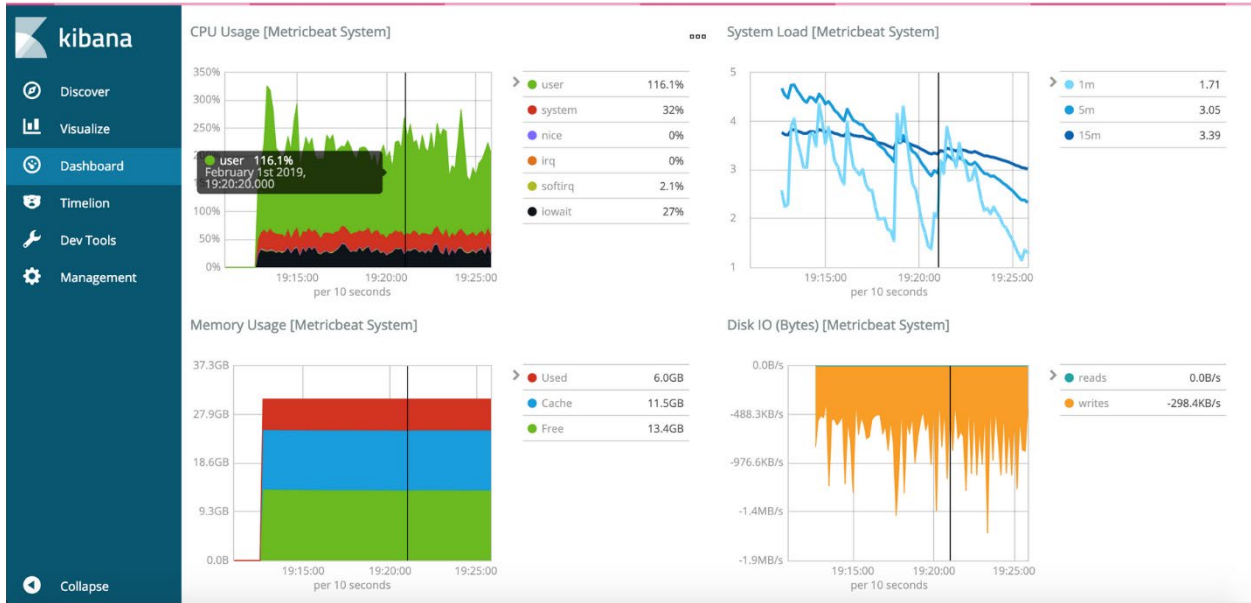


Memory usage [Metricbeat Docker]



Network IO [Metricbeat Docker]





Appendix A: Fabric Service CLIs

This appendix presents the CLI used to execute the implementation of various topologies.

3-Stage Clos automation

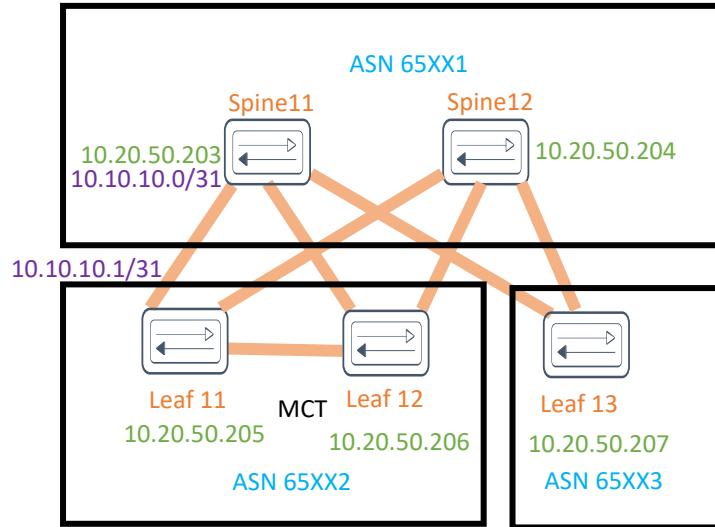


Figure A1. 3-stage Clos automation

Create a Fabric

CLI	dca fabric create --name <fabric-name> [--stage { 3 5 } --description <description>]
Field Description	--name string : Name of the Fabric. --description string : Description of the Fabric. --stage int : Stage of the Fabric. Default = 3.

Example:

```
$ dca fabric create --name BLR_FABRIC_1 --stage 3 --description
BLR_FABRIC_1
```

Add a device to the Fabric

A device must be registered with Inventory Service before being added to a Fabric. Fabric Service supports IP numbered configuration.

- IP numbered: Each interface on every link between leaf and spine is assigned an IP address, and eBGP peering use these IP addresses. Number of IP addresses per device = Number of links.

CLI	dca fabric device add --name <i><fabric-name></i> --ip <i><device-ip></i> --role { leaf spine super-spine } [leaf-type { single-homed multi-homed } --hostname <i><hostname></i> --asn <i><local-asn></i> --vtep-loopback <i><id></i> --loopback <i><id></i> --pod <i><name></i> --username <i><username></i> --password <i><password></i>]
Behavior	Adds a device to an existing Fabric. If user provides “username” and “password”, then the device is autoregistered with Inventory Service. If user does not provide “username” and “password”, then user must register the device explicitly with Inventory Service.
Field Description	--name string : Name of the Fabric --ip string : Device IP address --role : Device role: leaf, spine, or super-spine --leaf-type : Leaf type: single-homed or multi-homed --hostname string : Host name --asn string : ASN --vtep-loopback string : VTEP loopback ID --loopback string : Loopback port number --pod string : Name of the PoD --username string : Username for the device --password string : Password for the device

Example:

```
$ dca fabric device add --name BLR_FABRIC_1 --ip 10.20.50.203 --role spine --hostname Spine11 --username admin --password password
$ dca fabric device add --name BLR_FABRIC_1 --ip 10.20.50.204 --role spine --hostname Spine12 --username admin --password password
$ dca fabric device add --name BLR_FABRIC_1 --ip 10.20.50.205 --role leaf --leaf-type multi-homed --hostname Leaf11 --username admin --password password
$ dca fabric device add --name BLR_FABRIC_1 --ip 10.20.50.206 --role leaf --leaf-type multi-homed --hostname Leaf12 --username admin --password password
$ dca fabric device add --name BLR_FABRIC_1 --ip 10.20.50.207 --role leaf --hostname Leaf13 --username admin --password password
```

Add multiple devices to a Fabric

CLI	Fabric device add-bulk --name <Fabric-name> [--leaf <list-of-leaf-ips> --spine <list-of-spine-ips> --super-spine <list-of-super-spine-ips> --three-stage-pod <pod-name> --five-stage-pod-name <pod-name> --username <username> --password <password>]
Behavior	Adds multiple devices to an existing Fabric. If user provides “username” and “password”, then the devices are autoregistered with Inventory Service. If user does not provide “username” and “password”, then user must register the devices explicitly with Inventory Service. A single “three-stage-pod” and “five-stage-pod” can be provided per CLI execution.
Field Description	<p>--name <i>string</i> : Name of the Fabric</p> <p>--leaf <i>string</i> : Comma separated list of leaf IP addresses/host names</p> <p>--three-stage-pod <i>string</i> : Name of the leaf/spine PoD</p> <p>--five-stage-pod <i>string</i> : Name of the super-spine PoD</p> <p>--spine <i>string</i> : Comma separated list of spine IP addresses/host names</p> <p>--super-spine <i>string</i> : Comma separated list of super spine IP addresses/host names</p> <p>--username <i>string</i> : Username for the list of devices</p> <p>--password <i>string</i> : Password for the list of devices</p>

Example:

```
$ dca fabric device add-bulk --name BLR_FABRIC_1 --leaf
10.20.50.205,10.20.50.206,10.20.50.207 --spine
10.20.50.203,10.20.50.204 --username admin --password password
```

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 errors out, errors are reported to the user, based on which user can correct the topology. The encountered Fabric topology errors can be exported to a CSV/DOT file as well. Below are the topology validations:

1. Leaf nodes must be connected to all the spine nodes.
2. A spine node must be connected to all the leaf nodes.
3. No more than two leaf nodes must be connected to each other.
4. Spine nodes must not be connected to each other.
5. Super-spine nodes must not be connected to each other.
6. If a leaf node is marked as "multi-homed", then the node must have an MCT peer.
7. If a leaf node is marked as "single-homed", then the node must not be connected to other leaf nodes.
8. Device role (leaf/spine/super-spine) must be validated for a given device model (e.g. the SLX 9840 cannot be added as a leaf.)

NOTE: A management cluster is brought up on the MCT nodes forming the leaf pairs. Once the management cluster is operational, a logical VTEP is instantiated on the MCT nodes through the management cluster principal node. A logical VTEP IP needs to be consistent on both the MCT nodes. Transitioning MCT (multi-homed) nodes to non-MCT (single-homed) nodes and vice versa is not allowed and is an error condition.

Configure Fabric on the device

If the addition of devices to a Fabric is successful, then the underlay/overlay configuration can be configured on all the devices of the Fabric by means of the following CLI.

CLI	dca fabric configure --name <fabric-name> [--force]
Behavior	Configures underlay/overlay on the FDabric devices. If the --force option is used, then the all the devices are removed and added back to the Fabric, which can result in a configuration remove and add on all the devices.
Field Description	--name string : Name of the Fabric --force : Forces the configuration on the devices

Example:

```
$ dca fabric configure --name BLR_FABRIC_1
$ dca fabric configure --name BLR_FABRIC_1 --force
```

Remove multiple devices from the Fabric

Removal of device(s) from the Fabric results in deconfiguration of the underlay/overlay from the device and the device membership is removed from the Fabric.

CLI	dca fabric device remove --name <fabric-name> --ip <list-of-device-ips> [--no-device-cleanup]
Behavior	Removes existing device(s) from a Fabric. If the --no-device-cleanup option is used, then the configuration pushed by the automation engine is not cleaned up from the Fabric device(s). Removal of a device from the Fabric does not delete the device from inventory. He user must explicitly delete the device from inventory.
Field Description	--name string : Name of the Fabric --ip string : Device IP --no-device-cleanup : Prevents cleanup of the configuration on the devices

Example:

```
$ dca fabric device remove --name BLR_FABRIC_1 --ip 10.20.50.205,  
10.20.50.206,10.20.50.207
```

Fabric settings

The user can update the Fabric settings to overwrite the “default” Fabric settings, by using the following CLI. Defaults are shown in parentheses ().

CLI	dca fabric settings update --attribute-type <attribute-value>
Attributes	<p>mtu : The MTU size in bytes <Number:1548-9216> (9216)</p> <p>ip-mtu : IPV4/IPV6 MTU size in bytes <Number:1300-9194> (9100)</p> <p>bfd-enable : BFD enabled Yes/No (No)</p> <p>bfd-tx : BFD desired min transmit interval in milliseconds <NUMBER: 50-30000> (300)</p> <p>bfd-rx : BFD desired min receive interval in milliseconds <NUMBER: 50-30000> (300)</p> <p>bfd-multiplier : BFD detection time multiplier <NUMBER: 3-50> (3)</p> <p>bgp-multihop : Allow EBGP neighbors not on directly connected networks <Number:1-255> (2)</p> <p>max-paths : Forward packets over multiple paths<Number:1-64> allow-as-in : Disables the AS_PATH check of the routes learned from the AS<Number:1-10> (8)</p> <p>p2p-link-range: IP address pool to be used for leaf-to-spine links (10.10.0.0/23)</p> <p>loopback-ip-range: IP Address Pool for Loopback interface, to be used for unnumbered and VTEP IP (172.31.254.0/24)</p> <p>loopback-port-number : Loopback ID on the device to be used as donor IP interface for the link between leaf and spine. (1)</p> <p>vtep-loopback-port-number : Loopback ID on the device to be used as VTEP IP interface (2)</p> <p>leaf-asn-block : ASN pool for leaf nodes (4200000000-4200010000)</p> <p>spine-asn-block : ASN pool for spine nodes (4210000000-4210005000)</p> <p>super-spine-asn-block : ASN pool for super-spine nodes (4220000000-4220001000)</p> <p>leaf-peer-group : Leaf peer group name <WORD: 1-58> (leaf-group) spine-peer-group : Spine peer group name <WORD: 1-58> (spine-group)</p> <p>super-spine-peer-group : Super-spine peer group name <WORD: 1-58> (super-spine-group)</p>

	<p>anycast-mac-address : IPV4 ANY CAST MAC address.mac address HHHH.HHHH.HHHH (0201.0101.0101)</p> <p>ipv6-anycast-mac-address : IPV6 ANY CAST MAC address.mac address HHHH.HHHH.HHHH (0201.0101.0102)</p> <p>conversational-arp-aging-timeout: Determines how long an ARP entry stays in cache (300)</p> <p>mac-aging-timeout : MAC Aging Timeout <NUMBER: 0 60-100000> (1800)</p> <p>mac-aging-conversation-timeout : MAC Conversational Aging time in seconds<NUMBER: 0 60-100000> (300)</p> <p>mac-move-limit : MAC move detect limit <NUMBER: 5-500> (20)</p> <p>duplicate-mac-timer : Duplicate MAC timer (5)</p> <p>duplicate-mac-timer-max-count : Duplicate MAC timer maximum count (3)</p> <p>mctlipk-ip-range : IP address pool to be used for MCT peering (10.20.20.0/24)</p> <p>mct-port-channel : Port-channel interface ID <NUMBER: 1-64> to be used as MCT peer-interface (64)</p> <p>control-vlan : VLAN ID <NUMBER: 1-4090> to be used as MCT cluster control VLAN (4090)</p> <p>control-ve : VE ID <NUMBER: 1-4090> to be used as MCT cluster control VE (4090)</p> <p>configure-overlay-gateway : ConfigureOverlayGateway Enabled Yes/No (Yes)</p> <p>vni-auto-map : VTEP VLAN/BD to VNI Mode Auto (Yes/No) (Yes)</p>
--	---

5-stage Clos automation
 Refer to the following figure.

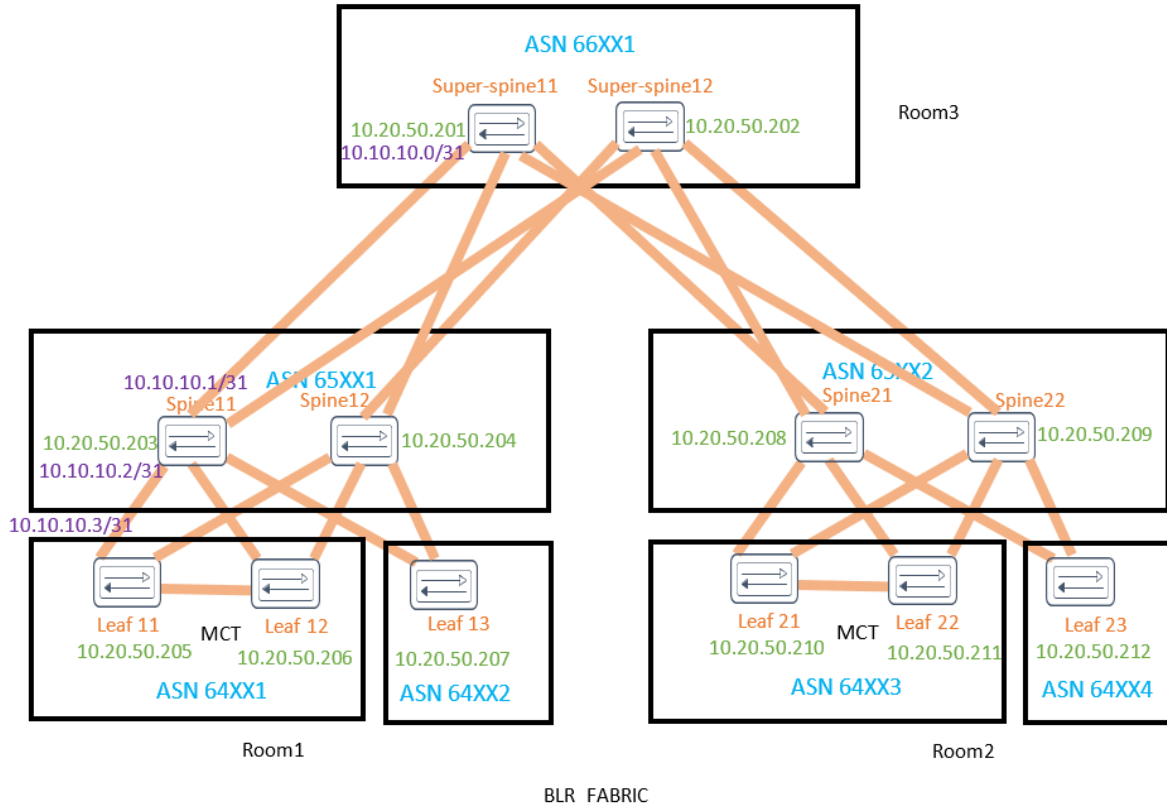


Figure A2. 5-stage Clos automation

Examples:

Non-bulk CLI

```
$ dca fabric create --name BLR_FABRIC --stage 5

$ dca fabric device add --name BLR_FABRIC --ip 10.20.50.201 --role
super-spine --hostname Super-spine11 --pod Room3 --username admin --
password password

$ dca fabric device add --name BLR_FABRIC --ip 10.20.50.202 --role
super-spine --hostname Super-spine12 --pod Room3 --username admin --
password password

$ dca fabric device add --name BLR_FABRIC --ip 10.20.50.203 --role
spine --hostname Spine11 --pod Room1 --username admin --password
password
```

```

$ dca fabric device add --name BLR_FABRIC --ip 10.20.50.204 --role
spine --hostname Spine12 --pod Room1 --username admin --password
password

$ dca fabric device add --name BLR_FABRIC --ip 10.20.50.205 --role
leaf --hostname Leaf11 --leaf-type multi-homed --pod Room1 --username
admin --password password

$ dca fabric device add --name BLR_FABRIC --ip 10.20.50.206 --role
leaf --hostname Leaf12 --leaf-type multi-homed --pod Room1 --username
admin --password password

$ dca fabric device add --name BLR_FABRIC --ip 10.20.50.207 --role
leaf --hostname Leaf13 --leaf-type single-homed --pod Room1 --username
admin --password password

$ dca fabric device add --name BLR_FABRIC --ip 10.20.50.208 --role
spine --hostname Spine21 --pod Room2 --username admin --password
password

$ dca fabric device add --name BLR_FABRIC --ip 10.20.50.209 --role
spine --hostname Spine22 --pod Room2 --username admin --password
password

$ dca fabric device add --name BLR_FABRIC --ip 10.20.50.210 --role
leaf --hostname Leaf21 --leaf-type multi-homed --pod Room2 --username
admin --password password

$ dca fabric device add --name BLR_FABRIC --ip 10.20.50.211 --role
leaf --hostname Leaf22 --leaf-type multi-homed --pod Room2 --username
admin --password password

$ dca fabric device add --name BLR_FABRIC --ip 10.20.50.212 --role
leaf --hostname Leaf23 --leaf-type single-homed --pod Room2 --username
admin --password password

$ dca fabric configure --name BLR_FABRIC

```

Bulk CLI

```

$ dca fabric create --name BLR_FABRIC --stage 5

$ dca fabric device add-bulk --name BLR_FABRIC --leaf
10.20.50.205,10.20.50.206-207 --spine 10.20.50.203-204 --super-spine
10.20.50.201-202 --three-stage-pod Room1 --five-stage-pod Room3

$ dca fabric device add-bulk --name BLR_FABRIC --leaf 10.20.50.210-212
--spine 10.20.50.208-209 --three-stage-pod Room2

$ dca fabric configure --name BLR_FABRIC

```


Multi-Fabric automation

Fabric Service supports the automation of multiple Clos Fabrics. The following shows two disjoint 3-stage Clos networks to be automated by Fabric Service.

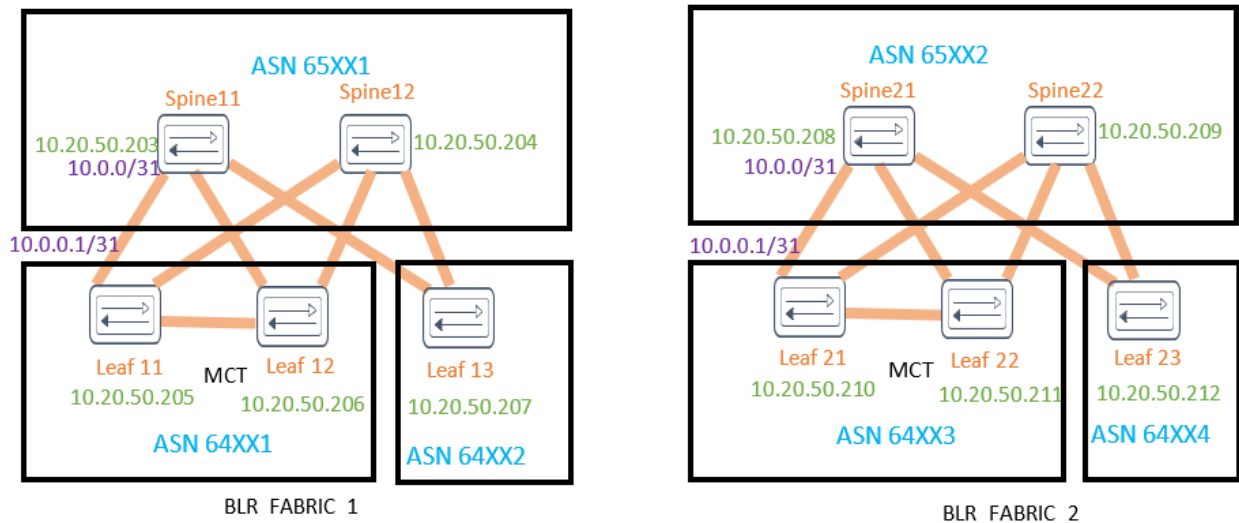


Figure A3. Multi-Fabric automation

Create a Fabric

CLI	dca fabric create --name <fabric-name> [--stage {3 5 } --description <description>]
Behavior	Creates a Fabric.
Field Description	--name string : Name of the Fabric --description string : Description of the Fabric --stage int : Stage of the Fabric (Default = 3)

Example:

```
$ dca fabric create --name BLR_FABRIC_1 --stage 3 --description
BLR_FABRIC_1
```

Delete a Fabric

CLI	dca fabric delete --name <fabric-name> [--force]
Behavior	Deletes a Fabric. Deletion of Fabric is not allowed if the Fabric has one or more devices. User must delete all the devices from the Fabric prior to deleting the Fabric. Forced deletion of Fabric removes the devices from Fabric but not from inventory.
Field Description	--name <i>string</i> : Name of the Fabric --force : Forces the deletion of Fabric even if the Fabric has devices

Example:

```
$ dca fabric delete --name BLR_FABRIC_1
```

Clone a Fabric

CLI	dca fabric clone { --source <source-fabric-name> --destination <destination-fabric-name> }
Behavior	Creates a clone of a source Fabric. Clones all the Fabric properties (type, stage, description, Fabric settings), but not the devices.
Field Description	--source <i>string</i> : Specifies name of the Fabric to be cloned --destination <i>string</i> : Specifies name of the cloned Fabric

Example:

```
$ dca fabric clone --source BLR_FABRIC_1 --destination  
SJ_FABRIC_1
```

Appendix B: Asset Service CLIs

Asset Service exposes the following operations to Fabric Service through CLI and REST endpoints:

- Device registration
- Device deletion
- Device update
- Key-value store
- Persist device configuration
- Execute CLI
- List registered device
- Compare and Replace

NOTE: Device registration operations are combined with Fabric Service CLIs and are not additionally required to provision an IP Fabric.

Device register

CLI	dca inventory device register --ip <string> --username <string> --password <string>
Behavior	Registers devices to Asset Service. This includes discovery of Assets such as physical interface, logical interfaces, LLDP links, BGP, VRF, Overlay, MCT information.
Field Description	<p>--ip string Comma separated range of device IP addresses. Example: 1.1.1.1-3,1.1.1.2,2.2.2.2</p> <p>--username string : Username to connect to the device</p> <p>--password string : Password to connect to the device</p>

Example:

```
$ dca inventory device register --ip 10.24.85.74,10.24.85.76,10.24.80.136-137
--username admin --password password
+-----+-----+-----+-----+-----+-----+-----+-----+
| ID | IP Address | Host Name | Model | Chassis Name | Firmware | Status | Reason |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 3 | 10.24.80.137 | SLX | 3000 | BR-SLX9240 | 18s.1.01a | Success | |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | 10.24.85.74 | SLX | 3006 | EN-SLX-9030-48S | 18x.1.00a | Success | |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 2 | 10.24.85.76 | SLX | 3007 | EN-SLX-9030-48T | 18x.1.00a | Success | |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 4 | 10.24.80.136 | SLX | 3000 | BR-SLX9240 | 18s.1.01a | Success | |
+-----+-----+-----+-----+-----+-----+-----+-----+
Device Details
--- Time Elapsed: 17.4935791s ---
```

Device delete

CLI	dca inventory device delete --ip <string>
Behavior	Deletes a device from Asset service, and also sends notification to Fabric Service and other services.
Field Description	<p>--ip string Comma separated range of device IP addresses. Example: 1.1.1.1-3,1.1.1.2,2.2.2.2</p> <p>--fabric string Fabric name</p>

Example:

```
$ dca inventory device delete --ip 10.24.85.76,10.24.85.74,10.24.80.136-137
+-----+-----+-----+-----+-----+-----+-----+-----+
| ID | IP Address | Host Name | Model | Chassis Name | Firmware | Fabric | Status | Reason |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | 10.24.85.74 | SLX      | 3006 | EN-SLX-9030-48S | 18x.1.00a |      | Success |      |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 2 | 10.24.85.76 | SLX      | 3007 | EN-SLX-9030-48T | 18x.1.00a |      | Success |      |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 3 | 10.24.80.137 | SLX      | 3000 | BR-SLX9240      | 18s.1.01a |      | Success |      |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 4 | 10.24.80.136 | SLX      | 3000 | BR-SLX9240      | 18s.1.01a |      | Success |      |
+-----+-----+-----+-----+-----+-----+-----+-----+
Device Details
--- Time Elapsed: 245.9984ms ---
```

Device update

Device update ensures that the inventory database is in sync with the device. Asset Service acts on the Fabric events and auto update occurs when the device is provisioned or unprovisioned.

The user must manually update the device in some scenarios. In those cases the user has the option to update one device at a time or all devices in a Fabric. Also, single device update can be used to update the device credentials by means of the **--username** and **--password** options.

After a successful device update, Inventory Service sends a notification of the changed event for all the Assets that have been modified.

CLI	dca inventory device update --ip <string> [--username <string> --password <string>] --fabric <string>
Behavior	Updates Asset information for a given device or all devices in the Fabric. This CLI can be used to update device credentials.
Field Description	<p>--ip string : IP Address of the device to update Asset details of</p> <p>--fabric string : Devices from the Fabric to update Asset details of</p> <p>--username string : Username to connect to the device</p> <p>--password string : Password to connect to the device</p>

Key-value store

Asset Service provides a key-value (KV) store where the user can store key-value pairs required for integration. The values stored here can be encrypted. DCA uses this KV store to store the device credentials. If the user needs to update the device credentials of an existing Fabric, then the KV store CLIs is use.

Create KV store

CLI	dca inventory kvstore create --key <string> --value <string> [--encrypt]
Behavior	Creates a key-value pair, with option to encrypt the values.
Field Description	--key string Key name --value string Value for the key --encrypt Specifies encryption

Delete KV store

CLI	dca inventory kvstore delete --key <string>
Behavior	Deletes a key-value pair.
Field Description	--key string Key name

List KV store

CLI	dca inventory kvstore list [--key <string> --prefix <string>] [--decrypt]
Summary	Displays all key-value pairs, and also filters views based on key, prefix, with option to decode the encrypted value.
Field Description	--decrypt : Decrypts the secret fields --key string : Returns the KV pair matching the key --prefix string : Retrieves list of KV pairs matching the prefix

Examples

These examples create or update key-value pairs with and without encryption.

```
$ dca inventory kvstore create --key test --value password
```

```
+-----+-----+-----+
| Key | Value | Secret |
+-----+-----+-----+
| test | password | false |
+-----+-----+-----+
Key Value Pair
--- Time Elapsed: 57.9627ms ---
```

```
$ dca inventory kvstore create --key test2 --value secret --encrypt
```

```
+-----+-----+-----+
| Key | Value | Secret |
+-----+-----+-----+
| test2 | 8c9iGAiW4qwkyKPFiJoqXg== | true |
+-----+-----+-----+
Key Value Pair
--- Time Elapsed: 56.5338ms ---
```

These examples list key-value pairs.

```
$ dca inventory kvstore list
```

Key	Value	Secret
switch.10.20.50.205.password	aC6p+gbmOIki6KOQJ6rSDw==	true
switch.10.20.50.205.user	admin	false
switch.10.20.50.212.password	aC6p+gbmOIki6KOQJ6rSDw==	true
switch.10.20.50.212.user	admin	false
switch.10.20.50.213.password	aC6p+gbmOIki6KOQJ6rSDw==	true
switch.10.20.50.213.user	admin	false
switch.1.2.2.2.user	root	false
test	password	false
test2	8c9iGAiW4qwkyKPFiJoqXg==	true

Keystore Details

--- Time Elapsed: 51.9304ms ---

```
$ dca inventory kvstore list --key test
```

Key	Value	Secret
test	password	false

Keystore Details

--- Time Elapsed: 33.0059ms ---

```
$ dca inventory kvstore list --key test2
```

Key	Value	Secret
test2	8c9iGAiW4qwkyKPFiJoqXg==	true

Keystore Details

--- Time Elapsed: 30.6644ms ---

```
$ dca inventory kvstore list --prefix test --decrypt
```

```
+-----+-----+-----+
| Key   | Value  | Secret |
+-----+-----+-----+
| test  | password | false  |
+-----+-----+-----+
| test2 | secret  | true   |
+-----+-----+-----+
Keystore Details
--- Time Elapsed: 62.9851ms ---
```

This example deletes key-value pairs.

```
$ dca inventory kvstore delete --key test
```

```
Key test deleted successfully.
--- Time Elapsed: 61.9647ms ---
```

Persist device configuration

Device configurations configured by Fabric Service and Tenant Service do not automatically persist on the devices. To persist the running configuration on a device, Asset Service provides CLI/REST endpoints.

CLI	dca inventory device running-config persist [--ip <list of ips> --fabric <string>]
Behavior	Execute CLI on the device.
Field Description	--ip string : Comma separated list of device IP address/hostnames --fabric string : Devices from the Fabric

These examples persist configurations.

```
$ dca inventory device running-config persist --ips
10.20.50.212,10.20.50.213
```

```
Persist Device(s) Running-Config[success]
+-----+-----+-----+-----+
| IP Address | Device Name | Fabric | Status |
+-----+-----+-----+-----+
| 10.20.50.212 | Leaf-1-3    | stage5 | Success |
+-----+-----+-----+-----+
| 10.20.50.213 | Leaf-1-4    | stage5 | Success |
+-----+-----+-----+-----+
Persist Running-Config Details
--- Time Elapsed: 12.2902836s ---
```

```
$ dca inventory device running-config persist --fabric stage5
```

```
Persist Device(s) Running-Config[success]
+-----+-----+-----+-----+
| IP Address | Device Name | Fabric | Status |
+-----+-----+-----+-----+
| 10.20.50.212 | Leaf-1-3    | stage5 | Success |
+-----+-----+-----+-----+
| 10.20.50.213 | Leaf-1-4    | stage5 | Success |
+-----+-----+-----+-----+
Persist Running-Config Details
--- Time Elapsed: 11.4899986s ---
```

Execute CLI

Certain operations that are not supported via the Fabric and Tenant services can still be accomplished by the `execute-cli` option. To execute a cli on the devices, the Asset Service provides CLI/REST endpoints.

CLI	dca inventory device execute-cli [--ip <list of ips> --fabric <string>][--command <comma separated string> --config]
Behavior	Persists the running configuration on the device to the startup configuration.
Field Description	<p>--ip string : Comma separated list of device IP address/hostnames</p> <p>--fabric string : Devices from the Fabric</p> <p>--command Comma/Semi-colon separated list of CLI commands to execute on the device(s)</p> <p>--config Flag to indicate whether the command is a config command or not.</p>

These examples persist configurations.

```
$ dca inventory device execute-cli --ip 10.24.80.134,10.24.80.135 --
command "show version"
```

IP Address	Host Name	Fabric	Command	Status	Reason	Output
10.24.80.134	SLX		show version	Success		<pre> SLX# show version SLX-OS Operating System Software SLX-OS Operating System Version: 18s.1.01 Copyright (c) 2017-2019 Extreme Networks Inc. Firmware name: 18s.1.01a Build Time: 08:15:03 Mar 5, 2019 Install Time: 02:12:43 Mar 21, 2019 Kernel: 2.6.34.6 Host Version: Ubuntu 14.04 LTS Host Kernel: Linux 3.14.17 Control Processor: QEMU Virtual CPU version 2.0.0 System Uptime: 13days 10hrs 39mins 15secs Slot Name Primary/Secondary Versions Status ----- SW/0 SLX-OS 18s.1.01a ACTIVE* 18s.1.01a </pre>
10.24.80.135	SLX		show version	Success		<pre> SLX# show version SLX-OS Operating System Software SLX-OS Operating System Version: 18s.1.01 Copyright (c) 2017-2002 Extreme Networks Inc. Firmware name: 18s.1.01a Build Time: 08:15:03 Mar 5, 2019 Install Time: 23:17:45 Mar 10, 2002 Kernel: 2.6.34.6 Host Version: Ubuntu 14.04 LTS Host Kernel: Linux 3.14.17 Control Processor: QEMU Virtual CPU version 2.0.0 System Uptime: 16days 20hrs 20mins 6secs Slot Name Primary/Secondary Versions Status ----- SW/0 SLX-OS 18s.1.01a ACTIVE* 18s.1.01a </pre>

List registered devices

CLI	dca inventory device list [--orphan --fabric <string> --role { leaf spine super-spine } --ips <list of ips>]
Behavior	Displays the devices registered to Asset Service. Displays all devices if these are not specified.
Field Description	--orphan Lists devices not associated to a Fabric --fabric string Fabric name --role Specifies device role --ips stringArray Comma-separated device IPs

These examples list inventories.

```
$ dca inventory device list
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+
| IP Address | Host Name | Model | Chassis Name | Firmware | ASN | Role | Fabric |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 10.20.50.205 | LEAF-2-1 | 3001 | BR-SLX9140 | 18s.1.01 | | | |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 10.20.50.212 | Leaf-1-3 | 3006 | EN-SLX-9030-48S | 18x.1.00 | 65002 | Leaf | stage5 |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 10.20.50.213 | Leaf-1-4 | 3006 | EN-SLX-9030-48S | 18x.1.00 | 65002 | Leaf | stage5 |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

Device Details

```
--- Time Elapsed: 85.9462ms ---
```

```
$ dca inventory device list --orphan
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+
| IP Address | Host Name | Model | Chassis Name | Firmware | ASN | Role | Fabric |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 10.20.50.205 | LEAF-2-1 | 3001 | BR-SLX9140 | 18s.1.01 | | | |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

Device Details

```
--- Time Elapsed: 56.0216ms ---
```

```
$ dca inventory device list --fabric stage5 --role leaf
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+
| IP Address | Host Name | Model | Chassis Name | Firmware | ASN | Role | Fabric |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 10.20.50.212 | Leaf-1-3 | 3006 | EN-SLX-9030-48S | 18x.1.00 | 65002 | Leaf | stage5 |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 10.20.50.213 | Leaf-1-4 | 3006 | EN-SLX-9030-48S | 18x.1.00 | 65002 | Leaf | stage5 |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

Device Details

```
--- Time Elapsed: 87.9664ms ---
```

```
$ dca inventory device list --ips 10.20.50.212,10.20.50.213
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+
| IP Address | Host Name | Model | Chassis Name | Firmware | ASN | Role | Fabric |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 10.20.50.212 | Leaf-1-3 | 3006 | EN-SLX-9030-48S | 18x.1.00 | 65002 | Leaf | stage5 |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 10.20.50.213 | Leaf-1-4 | 3006 | EN-SLX-9030-48S | 18x.1.00 | 65002 | Leaf | stage5 |
+-----+-----+-----+-----+-----+-----+-----+-----+
Device Details
```

Compare and Replace device

Compare cli is a tool to indicate to the user if the switch configuration has drifted from what the application has stored

CLI	dca inventory device compare --ip <IP Address of the device>
Behavior	Displays if there are any configuration drifts from what the Application has saved or whether the configuration on the device is current
Field Description	--ip string Device IP for which we want to check the current configuration

```
$ dca inventory device compare --ip 10.24.80.134 |
Device 10.24.80.134 configuration is current in the application.
--- Time Elapsed: 14.996360091s ---
```

Replace CLI is meant to be used in case of RMA. This will help the users replace the device from the tool, if the fabric and tenant configuration on the new device are the same as the device its replacing

CLI	dca inventory device replace --ip <IP Address of the device>]
Behavior	Allows users to replace the Device if the configuration in the Fabric and Tenant service match what the application has stored.
Field Description	--ip string IP of the Device to be replaced

```
dca inventory device replace --ip=10.24.80.134
+-----+-----+-----+-----+-----+-----+-----+-----+
| ID | IP Address | Host Name | Model | Chassis Name | Firmware | ASN | Role | Fabric |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 37 | 10.24.80.134 | SLX | 3001 | BR-SLX9140 | 18s.1.01a | | | |
+-----+-----+-----+-----+-----+-----+-----+-----+
Device Details
```


Appendix C: Tenant Service CLIs

This appendix presents the CLIs used to execute the implementation of various topologies.

NOTE: The samples provided here are using WORKFLOW. The same functionality can be achieved by using the Advanced CLIs.

Automating L2 and L3 extension between racks

This use case addresses the L2 and L3 extension of networks across leaf nodes in an IP Fabric

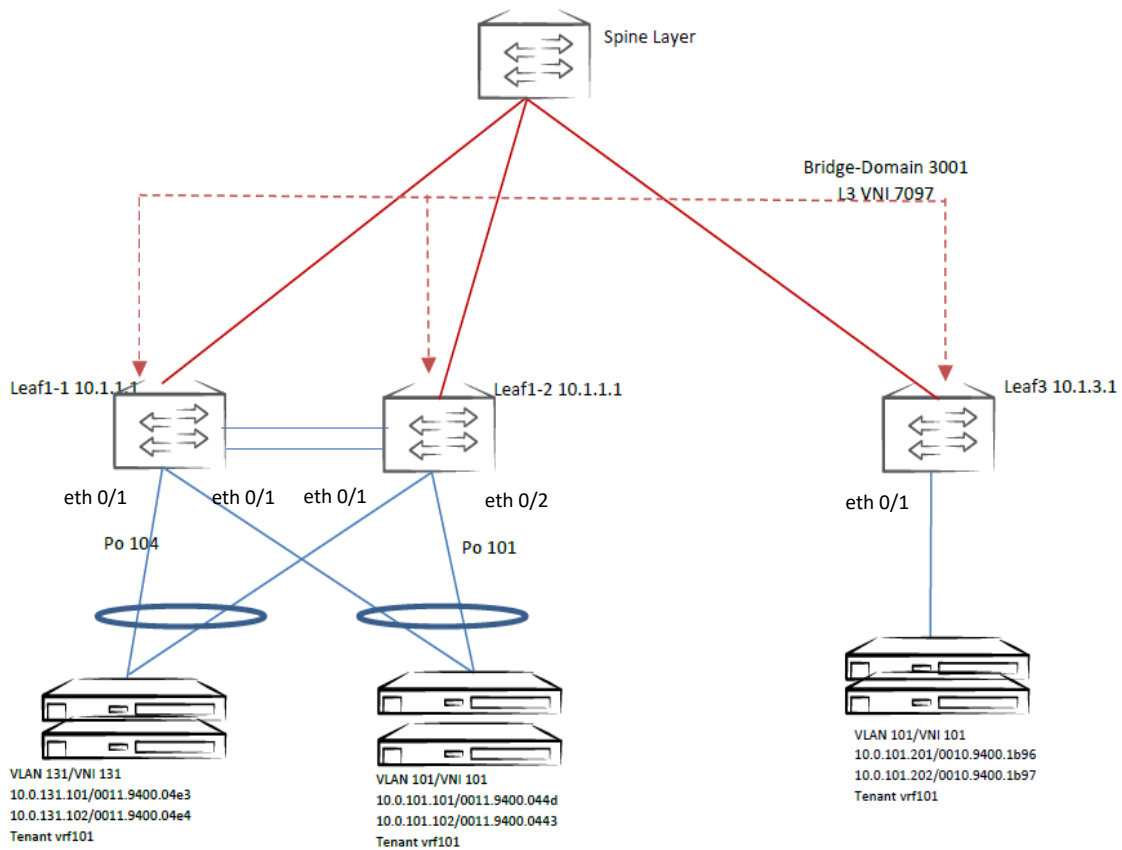


Figure A4. L2 and L3 extension between racks

Create a VRF

Default values are shown in parentheses (). The VRF RD value is auto-generated based on the router-id configured on the device. The RT can be either provided as input or auto-generated if not provided).

CLI	dca tenant vrf create --name <vrf-name> [--rt-type { import export both } --rt <rt >
Field Description	--name string : Name of the vrf. --rt-type string : Route Target VPN Community. Valid values are both import export. (both) --rt string : A unique number for setting for forming Route Target and Route Distinguisher. Accepted format is nn:nn. --tenant string Name of the tenant

Example:

```
dca tenant vrf create --name --vrf vrf101
```

Create Tenant network (using workflow)

Default values are shown in parentheses ().

CLI	dca tenant workflow network create --name <network-name>
Behavior	Creates port-channels and the channel groups on the interfaces. Creates EPGs with the port-channel and the Advanced ports. Creates the Tenant network.
Field Description	--name string : Name of the Tenant network --tenant string : Name of the tenant (default "default-tenant") --po-name stringArray : Name of the port-channel --po-speed stringArray : Speed for the port-channel and its member ports. <100Mbps 1Gbps 10Gbps 25Gbps 40Gbps 100Gbps> --po-ports stringArray : Comma-separated list of ports forming a port-channel. Supported format is < ipaddress [ports] > . Ports can be single or comma-separated ports or a range of ports. --po-ctag stringArray : Ctag value for port-channels. <1-4090> --ports stringArray : Comma-separated list of ports. Supported format < ipaddress [ports]> . Ports can be single or comma-separated ports or a range of ports. --port-ctag stringArray : Ctag value for ports. --ctag string : Ctag appearing on all the ports and port-channels. --mode string : Configures switchport mode on the interfaces. < trunk access trunk_no_default_active >(trunk) --enable-bd : Enables bridge domain if the option is provided. --enable-ept : Enables endpoint tracking on the interfaces if the option is provided. --vrf string : Name of the VRF of which this network is a part. --anycast-ip string : Configures anycast IPv4 address on the interface Ve.

Example:

```
$ dca tenant workflow network create --name network131 --po-name
po131 --po-speed 10Gbps --po-ports Leaf-1-ip[0/1], Leaf-2-ip[0/1]
--ctag 131 --enable-bd --vrf vrf101 --anycast-ip 10.0.131.254/22
```

```
$ dca tenant workflow network create --name network101 --po-name
po101 --po-speed 10Gbps --po-ports Leaf-1-ip[0/2], Leaf-2-ip[0/2]
--ports Leaf-3-ip[0/1] --ctag 101 --enable-bd --vrf vrf101
--anycast-ip 10.0.101.254/22
```

Examples: (supported port formats)

```
$ dca tenant workflow network create --name network101 --po-name
po101 --po-speed 10Gbps --po-ports Leaf-1-ip[0/2-10], Leaf-2-
ip[0/2-10] --ports Leaf-3-ip[0/1-10] --ctag 101 --enable-bd --vrf
vrf101 --anycast-ip 10.0.101.254/22
```

```
$ dca tenant workflow network create --name network101 --po-name
po101 --po-speed 10Gbps --po-ports Leaf-1-ip[0/2,0/3,0/4], Leaf-2-
ip[0/2,0/3-4] --ports Leaf-3-ip[0/1-2,0/3] --ctag 101 --enable-bd
--vrf vrf101 --anycast-ip 10.0.101.254/22
```

NOTE: To turn on more features on port-channel, EPGs, and Tenant networks, Advanced CLIs must be used.

Tenant network creation goes through the following stages before completion:

- Validation of the resources allocated to the Tenant network.
- Allocation of free resources such as BD, IRB VE , MCT client ID , VRF RD&RT.
- Pushing the configuration to the SLX devices.
- If the configuration is successful on all the devices, the Tenant network is moved to “READY” state.
- Any device-level and port-level failures are marked as “ERROR” state.

Automating VLAN scoping at the ToR level

This use case addresses how multiple CTAGs can be mapped to a bridge domain at the ToR level.

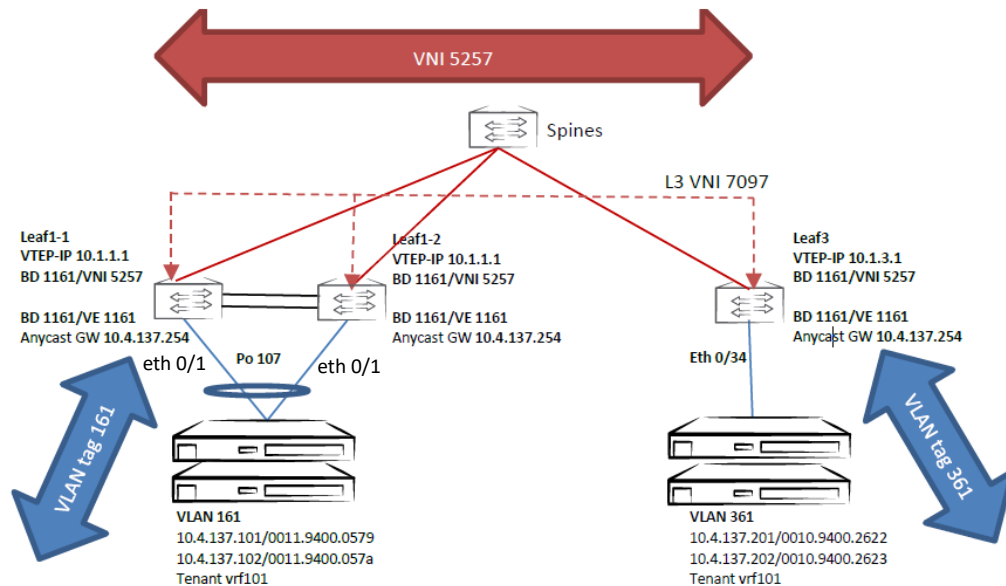


Figure A4. VLAN scoping at ToR

Example:

```
$ dca tenant vrf create --name vrf101
$ dca tenant workflow network create --name netToR --po-name po107
--po-speed 10Gbps --po-ports Leaf-1-ip[0/1], Leaf-2-ip[0/1] --po-
ctag 161 --ports Leaf-3-ip[0/34] --ctag 361 --vrf vrf101
--anycast-ip 10.4.137.254/22 --enable-bd
```

Automating VLAN scoping at the port level within a ToR

This use case addresses how multiple CTAGs can be mapped to a bridge-domain at the port level within a ToR level.

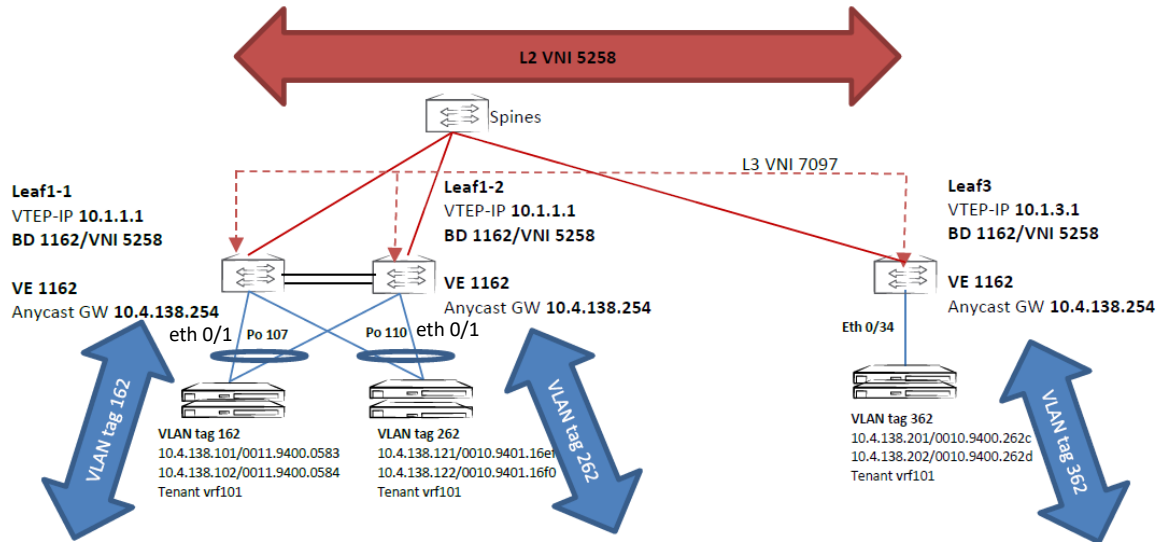


Figure A4. VLAN scoping at the port level with a ToR

Example:

```
$ dca tenant vrf create --name vrf101

$ dca tenant workflow network create --name netPortToR --po-name
po162 --po-speed 10Gbps --po-ports Leaf-1-ip[0/1], Leaf-2-ip[0/1]
--po-ctag 162 --po-name po262 --po-speed 10Gbps --po-ports Leaf-1-
ip[0/2], Leaf-2-ip[0/2] --po-ctag 262 --ports Leaf-3-ip[0/34]
--ctag 362 --vrf vrf101 --anycast-ip 10.4.138.254/22 --enable-bd
```

Update operations

All the update operations are supported only through the Advanced CLIs

Update Tenant network

Default values are shown in parentheses ().

CLI	dca tenant network update --name <network-name>
Behavior	Based on the operation type, the delta L2/L3 configurations will be added or removed from the devices
Field Description	--name string : Name of the Tenant network. --tenant string : tenant name --op-code string : Update operation type to be performed. < epg_add epg_delete router_add router_delete > --epg strings : Comma-separated EPG names to be attached to the network. --vrf string : Name of the VRF. --anycast-ip string : Configures anycast IPv4 address on the interface Ve.

- Add a new EPG to the existing Tenant network:

```
$ dca tenant network update --name netPortToR --op-code epg_add
--epg epg399
```

- Delete an existing EPG from the Tenant network:

```
$ dca tenant network update --name netPortToR --op-code
epg_delete --epg epg300
```

- Update an existing L2 Tenant network to L3 Tenant network:

```
$ dca tenant network update --name netPortToR --op-code
router_add --vrf vrf101 --anycast-ip 10.4.138.254/22
```

Update endpoint group

Default values are shown in parentheses ().

CLI	dca tenant epg update --name <epg-name>
Behavior	Add or delete ports to the existing EPG. If the EPG is part of any Tenant network, based on the op-code type, the delta configurations are added or deleted from the devices.
Field Description	--name string : Name of the end point group. --tenant string : tenant name --op-code string : Add or delete operation on the ports. < port_add port_delete po_add po_delete > --po stringArray : List of port channels on which tenant network will be configured. Example: po1 or po1, po2 --port stringArray : List of physical ports of device on which Tenant network will be configured. --ept-status : Enable or disable endpoint tracking on the EPG. < enable disable >.(disable)

	--force	: Force delete the configurations from devices
--	----------------	--

- Add a new port to the existing EPG:

```
$ dca tenant epg update --name epg-101 --op-code port_add --port
Leaf-3-ip[0/10]
```

- Delete an existing port from an existing EPG:

```
$ dca tenant epg update --name epg-101 --op-code port_delete
--port Leaf-3-ip[0/2]
```

Update Port-channel

Default values are shown in parentheses ().

CLI	dca tenant po update --name <portchannel-name>
Behavior	Add or delete ports to the existing port channel group. Based on the op-code type, the delta configurations will be added or deleted from the interfaces.
Field Description	--name string : Name of the port channel --tenant string :tenant name --op-code string : Add or Delete the ports from the port channel < add delete> --port stringArray :Device ip along with ethernet port details

- Add a new port to the existing EPG:

```
$ dca tenant po update --name po-101 --op-code add --port Leaf-1-
ip[0/3]
```

- Delete an existing port from an existing EPG:

```
$ dca tenant po update --name po-101 --op-code delete --port
Leaf-3-ip[0/1]
```

Update Asset

Default values are shown in parentheses ().

CLI	dca tenant asset update --name <asset-name>
Behavior	Add or delete ports to the existing asset group. Based on the op-code type, the asset ports are added/deleted from the existing asset. If --force option is provided along with op-code delete , the port is deleted from all the epgs/pos/tenant networks.
Field Description	--name string Name of the Asset --port stringArray List of physical ports of devices which will be reserved for the asset. Example SW1_IP[0/1],SW2_IP[0/5] --op-code string Operation code. Valid values are add delete. --force Force the Update on Asset if the option is provided

- Add a new port to the existing Asset:

```
$ dca tenant asset update --name Asset-Group-1 --op-code add
--port Leaf-1-ip[0/33]
```

- Delete an existing port from an existing Asset:

```
$ dca tenant asset update --name Asset-Group-1 --op-code delete
--port Leaf-3-ip[0/11]
```

Update Tenant

Default values are shown in parentheses ().

CLI	dca tenant update --name <tenant-name>
Behavior	Attach or detach an asset from tenant by providing op-codes `asset_add/asset_delete`. Update the vni pool that the tenant owns using the option `vni_update`. If <code>--force</code> option is provided with `asset_delete` any attached networks/pos/eggs to the tenant are deleted.
Field Description	<p>--name string Name of the tenant</p> <p>--l2-vni-range string Range of L2 Virtual Network Identifiers(VNI) reserved for tenant.Valid values are <1-16777215>.</p> <p>--l3-vni-range string Range of L3 Virtual Network Identifiers(VNI) reserved for tenant.Valid values are <1-16777215></p> <p>--l2-vni-count string Total number of L2 Virtual Network Identifiers(VNI) reserved for tenant.Valid values are from <1> to <16777215></p> <p>--l3-vni-count string Total number of L3 Virtual Network Identifiers(VNI)s reserved for tenant.Valid values are from <1> to <16777215></p> <p>--op-code string Operation code. Valid values are asset_add asset_delete vni_update.</p> <p>--asset string Asset name reserved for this tenant.</p> <p>--port stringArray List of physical ports of devices which will be reserved for the asset. Example SW1_IP[0/1],SW2_IP[0/5]</p> <p>--force Force the asset deletion on the Tenant if the option is provided</p>

- Attach a new asset to the existing Tenant:

```
$ dca tenant update --name Tenant-Sales --asset Asset-Group-2 --
op-code asset_add
```

- Detach an asset from an existing Asset:

```
$ dca tenant asset update --name Tenant-Sales ---Asset-Group-2 --  
op-code delete
```

Get operations

Get Tenant network (workflow CLI)

CLI	dca tenant workflow network show
Behavior	Displays the summary of all the Tenant networks when --name option is not provided. Displays the summary of a given Tenant network when --name and --detailed options are provided.
Field Description	--name string : Name of the Tenant network. --detailed : Displays the Tenant network details such as EPGs, network properties, and port-level configurations --tenant string :tenant name --all : Show Networks for all Tenants

Example:

```
$ dca tenant workflow network show --name net1 --detailed
```

```
-----+-----+
| Network Name | net1      |
| BD Enabled   | false    |
| Config State | READY    |
| VRF Name     | vrf1     |
| Anycast IP   | 10.30.33.4/22 |
+-----+-----+
```

Network Details

```
-----+-----+
| EPG Name     | net1-epg-333 |
| CTAG         | 333          |
| Mode         | trunk        |
| End Point Tracking | false      |
| Ports        | 10.24.51.135[0/33] |
| po101       | 10.24.51.131[0/33] |
|              | 10.25.225.58[0/33] |
+-----+-----+
```

EPG Details

```
-----+-----+-----+-----+-----+-----+-----+
| EPG NAME | DEVICE | PORT/PORTCHANNEL | SWITCHPORT MODE | CTAG | VE | PORT STATE |
+-----+-----+-----+-----+-----+-----+-----+
| net1-epg-333 | 10.24.51.135 | 0/33 | trunk | 333 | 333 | Configured |
| net1-epg-333 | 10.24.51.131 | 1 | trunk | 333 | 333 | Configured |
| net1-epg-333 | 10.25.225.58 | 1 | trunk | 333 | 1 | Configured |
+-----+-----+-----+-----+-----+-----+-----+
```

Get Tenant network (Advanced CLI)

CLI	dca tenant network show
Behavior	Displays the summary of all the Tenant networks when --all option is provided. Displays the summary of Tenant network when --name and --config options are provided owned by default tenant. If --tenant is provided, networks owned by that tenant will be shown.
Field Description	--name string : Name of the Tenant network --config : Displays the Tenant network details like EPGs, network properties, and port-level configurations. --all : Show Networks for all Tenants --tenant string :tenant name

Example:

```
$ dca tenant network show --name net1 --config
```

```
Tenant Network Info:
-----
Network Name      |net1
Endpoint Groups  |net1-epg-333
BD Enabled        |false
Vrf               |vrf1
Tenant Name      |default-tenant
-----
Device            |10.24.51.135
Vlan              |333
Vrf               |vrf1
  RD              |172.31.254.36:100
  RT              |100:100
IrbBd             |1023
IrbVe            |1023
Ve                |333
  Anycast         |10.30.33.4/22
Port              |ethernet-0/33
Vlan              |333
SwitchportMode   |trunk
-----
Device            |10.24.51.131
Vlan              |333
Vrf               |vrf1
  RD              |172.31.254.38:100
  RT              |100:100
IrbBd             |1023
IrbVe            |1023
Ve                |333
  Anycast         |10.30.33.4/22
Port              |portchannel-1
Vlan              |333
SwitchportMode   |trunk
-----
Device            |10.25.225.58
Vlan              |333
Vrf               |vrf1
  RD              |172.31.254.40:100
  RT              |100:100
IrbBd             |1023
IrbVe            |1023
Ve                |1
  Anycast         |10.30.33.4/22
Port              |portchannel-1
Vlan              |333
SwitchportMode   |trunk
-----
--- Time Elapsed: 17.5742ms ---
```

Get endpoint group

CLI	dca tenant epg show
Behavior	Displays the summary of all the EPGs when --all option is provided. Displays the summary of a given end point group when --name option is provided along with -tenant option.
Field Description	--name string : Name of the EPG. --all : Show EPGs for all Tenants --tenant string : tenant name

Example:

```
$ dca tenant epg show --name net1-epg-333
```

Name	net1-epg-333
Vlan	333
Tag Type	tagged
Switchport Mode	trunk
EndpointTracking	Disabled
Port-channel	po101
Device IP	10.24.51.135
Ports	ethernet : 0/33

```
--- Time Elapsed: 11.718ms ---
```

Get port-channel group

CLI	dca Tenant po show
Behavior	Displays the summary of all port-channels when the --all option is provided. Displays the summary of a given port-channel group when the --name option is provided along with -tenant option.
Field Description	--name string : Name of the port-channel. --all : Show port channels for all Tenants --tenant string : tenant name

Example:

```
$ dca tenant po show --name po101
```

Name	po101
Portchannel Interface Number	1
Speed	10Gbps
Negotiation	active
Device IP	10.24.51.131
Ports	ethernet : 0/33
Device IP	10.25.225.58
Ports	ethernet : 0/33

```
--- Time Elapsed: 9.7619ms ---
```


Get VRF

CLI	dca tenant vrf show
Behavior	Displays the summary of all the VRFs.
Field Description	--all : Show vrfs for all Tenants --tenant string : Tenant name

Example:

```
$ dca tenant vrf show
+-----+-----+-----+-----+-----+
| Name   | State | RouteTarget Type | Route Target NN:NN | IRB BD |
+-----+-----+-----+-----+-----+
| vrf_red | READY |                   |                   |        |
+-----+-----+-----+-----+-----+
| vrf11  | READY | both              | 101:101           |        |
+-----+-----+-----+-----+-----+
| vrf1   | READY | both              | 100:100           |        |
+-----+-----+-----+-----+-----+
Router Details
--- Time Elapsed: 26.3607ms ---
```


Get Tenant

CLI	dca tenant show
Behavior	Displays the summary of all the tenants.
Field Description	--name string : Name of the Tenant.

Get Asset

CLI	dca tenant asset show
Behavior	Displays the summary of all the assets
Field Description	--name string : Name of the Asset.

Delete operations

Delete Tenant network (workflow CLI)

CLI	dca tenant workflow network delete
Behavior	Deletes the Tenant network, including the configurations from the device, associated EPGs, and the port-channels if --name is provided. Deletes the Tenant network, including the configurations from the device if --name and --force options are provided. (Ignores any device errors while deleting the configurations from the devices)
Field Description	--name string : Comma-separated list of Tenant network names. --force : Force delete the configurations from devices. --tenant string : Name of the tenant (default "default-tenant")

Example:

```
$ dca tenant workflow network delete --name net1
```

Delete Tenant network (Advanced CLI)

CLI	dca tenant network delete
Behavior	Deletes the Tenant network, including the configurations from the device if the --name option is provided. Associated EPGs and port-channels will not be deleted. Deletes the Tenant network, including the configurations from the device if --name and --force options are provided. (Ignores any device errors while deleting the configurations from the devices)
Field Description	--name string : Name of the Tenant network. --force : Force delete the configurations from devices. --tenant string :tenant name

Example:

```
$ dca tenant network delete --name net1
```

Delete end point group

CLI	dca tenant epg delete
Behavior	Delete an EPG and its Tenant network configurations from the device if --name and --force options are provided. Delete EPG only if it is not attached to any Tenant network if --name is provided.
Field Description	--name string : Name of the EPG --force : Force delete the EPGs and its associated Tenant networks. --tenant string : Tenant Name

Example:

```
$ dca tenant epg delete --name net1-epg-333
```

Delete port-channel

CLI	dca tenant po delete
Behavior	Delete a port-channel and its Tenant network configurations from the device (if the port-channel is attached to any EPG) if --name and --force options are provided. Delete a port-channel only if it is not attached to any Tenant network if --name is provided.
Field Description	--name string : Name of the port channel. --force : Force delete the port-channel and its associated Tenant networks. --tenant string : Tenant Name

Example:

```
$ dca tenant po delete --name po11
```

Delete VRF

CLI	dca tenant vrf delete
Behavior	Delete a VRF only if it is not attached to any Tenant network if --name is provided.
Field Description	--name string : Name of the VRF. --tenant string : Tenant Name

Example:

```
$ dca tenant vrf delete --name vrf101
```

Delete Asset

CLI	dca tenant asset delete
Behavior	Delete Asset will delete the asset from the database. Delete operation with "force" option will delete all the networks, port-channels and endpoint groups from the database and device configurations.
Field Description	--name string : Name of the asset. --force : Force delete the asset and its associated Tenant networks.

Delete Tenant

CLI	dca tenant delete
Behavior	Delete Tenant will delete the tenant from the database. Delete operation with "force" option will delete all the networks, port-channels and endpoint groups from the database and device configurations.
Field Description	--name string : Name of the tenant. --force : Force delete the tenant and its associated Tenant networks.

Bulk operations

Bulk creation or deletion of tenant networks on a collection of physical/port channel interfaces can be achieved by using the tenant network workflow. The number of networks created is based on the ctag range provided.

NOTE: Bulk creation/deletion is supported only for L2 networks

Bulk create

CLI	dca tenant workflow network bulk-create
Behavior	Creates multiple tenant networks for the given ctag range.
Field Description	--net-prefix string : Prefix of the network --tenant string : Tenant Name --po-name stringArray : Name of the port-channel --po-speed stringArray : Speed for the port-channel and its member ports. <100Mbps 1Gbps 10Gbps 25Gbps 40Gbps 100Gbps> --po-ports stringArray : Comma-separated list of ports forming a port-channel --ports stringArray : Comma-separated list of physical ports --ctags string : Comma-separated Ctag values appearing on all the ports and port-channels --mode string : Configures switchport mode on the interfaces. Valid values are trunk access trunk-no-default-native . (default is "trunk") --enable-ept : Enable endpoint tracking

Bulk delete

CLI	dca tenant workflow network bulk-delete
Behavior	Deletes the Tenant network, including the configurations from the device, associated EPGs, and the port-channels if --net-prefix string and ctags are provided. Deletes the Tenant network, including the configurations from the device if -- net-prefi , ctags , and --force options are provided (ignores any device errors while deleting the configurations from the devices).
Field Description	--net-prefix string : Prefix of the network --ctags string : Comma-separated Ctag values appearing on all the ports and port-channels --force : Force the deletion on the Tenant network if the option is provided

Example:

```
$ dca tenant workflow network bulk-delete --net-prefix N1 --  
ctags 100-105
```

The above command does the following:

- Deletes all the networks starting with the network prefix provided and the underlying EPGs and port-channels

Appendix D: Database Backup and Restore

Overview

This section provides the steps for customers/developers to restore the DCAApp services, such as goInventory-service, goFabric-service, and goTenant-service, in case the DCAApp database becomes corrupted or the user wants to move back to the previously saved configurations. 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 **dca supportsave** command to back up the databases.

Example

```
$ dca supportsave
```

```
Version : 2.0.0  
Time Stamp: 19-01-25:17:35:32  
Support Save File: /var/log/dcapp/dcapp_1548895361.logs.zip  
--- Time Elapsed: 1.146039397s ---
```

Restore the database

Do the following to restore the database.

1. Unzip the supportSave log to get the backup databases
NOTE: If multiple supportSaves were captured, select latest database file for restoration.

```
$ unzip dcapp_1548895361.logs.zip
```

```

Archive: dcapp_1548895361.logs.zip
  creating: inventory/
  inflating: inventory/inventory-server-2019-01-26T02-44-09.796.log
  inflating: inventory/inventory-server-2019-01-26T02-55-55.126.log
  inflating: inventory/inventory-server-2019-01-26T03-08-41.174.log
  inflating: inventory/inventory-server-2019-01-28T23-43-48.418.log
  inflating: inventory/inventory-server.log
  inflating: inventory/inventory_database_dump_1548268980.log
  inflating: inventory/inventory_database_dump_1548471002.log
  inflating: inventory/inventory_database_dump_1548472270.log
  inflating: inventory/inventory_database_dump_1548721760.log
  inflating: inventory/inventory_database_dump_1548895360.log
  creating: fabric/
  inflating: fabric/.fabric-2019-01-28T21-56-25.675.log.swp
  inflating: fabric/fabric-2019-01-26T03-04-29.629.log
  inflating: fabric/fabric-2019-01-26T03-05-32.721.log
  inflating: fabric/fabric-2019-01-26T03-06-43.705.log
  inflating: fabric/fabric-2019-01-26T13-36-20.970.log
  inflating: fabric/fabric-2019-01-27T20-10-35.941.log
  inflating: fabric/fabric-2019-01-28T20-33-00.249.log
  inflating: fabric/fabric-2019-01-28T21-56-25.675.log
  inflating: fabric/fabric-2019-01-28T23-43-52.035.log
  inflating: fabric/fabric-2019-01-29T21-40-36.067.log
  inflating: fabric/fabric.log
  inflating: fabric/fabric_database_dump_1548268980.log
  inflating: fabric/fabric_database_dump_1548471002.log
  inflating: fabric/fabric_database_dump_1548472270.log
  inflating: fabric/fabric_database_dump_1548721760.log
  inflating: fabric/fabric_database_dump_1548895360.log
  creating: ts/
  inflating: ts/tenant_database_dump_1548268980.log
  inflating: ts/tenant_database_dump_1548471002.log
  inflating: ts/tenant_database_dump_1548472270.log
  inflating: ts/tenant_database_dump_1548721760.log
  inflating: ts/tenant_database_dump_1548895360.log
  inflating: ts/ts.log

```

2. Stop only the services whose databases you want to restore, or all three Inventory, Fabric, and Tenant Service containers.

NOTE: Postgres-DB services must be running.

```

$ docker ps
$ docker stop goinventory-service-v2.0.0
$ docker stop goFabric-service-v2.0.0
$ docker stop goTenant-service-v2.0.0

```

3. Delete the databases you want to restore, or all three Inventory, Fabric, and Tenant Service databases, and recreate the deleted empty databases.

- a. Connect to the database.

```
$ psql -U postgres -p 5432 -h localhost --W
```

- b. Delete the database.

```
postgres=# DROP DATABASE dcapp_Fabric;  
DROP DATABASE
```

```
postgres=# DROP DATABASE dcapp_Asset;  
DROP DATABASE
```

```
postgres=# DROP DATABASE dcapp_Tenant;  
DROP DATABASE
```

- c. Recreate the empty database.

```
postgres=# CREATE DATABASE dcapp_Asset OWNER Asset;  
CREATE DATABASE
```

```
postgres=# CREATE DATABASE dcapp_Fabric OWNER Fabric;  
CREATE DATABASE
```

```
postgres=# CREATE DATABASE dcapp_Tenant OWNER Tenant;  
CREATE DATABASE
```

- d. Create pgcrypto extension

```
$ psql -U postgres -p 5432 -h localhost -W -d dcapp_asset
```

```
dcapp_asset=#CREATE EXTENSION pgcrypto;
```

4. Restore the databases from the backup DBs.

- a. Restore the Inventory DB.

```
$ pg_restore -Fc -d dcapp_Asset -U Asset -h localhost -p  
5432 -v inventory/inventory_database_dump_1548895360.log  
pg_restore: connecting to database for restore  
Password:  
pg_restore: creating SCHEMA "public"  
pg_restore: creating COMMENT "SCHEMA public"  
pg_restore: creating EXTENSION "plpgsql"  
pg_restore: creating COMMENT "EXTENSION plpgsql"
```


- b. Restore the Fabric DB.

```
$ pg_restore -Fc -d dcapp_Fabric -U Fabric -h localhost -p 5432 -v  
Fabric/Fabric_database_dump_1548895360.log
```

```
pg_restore: connecting to database for restore
```

```
Password:
```

```
pg_restore: creating SCHEMA "public"
```

```
pg_restore: creating COMMENT "SCHEMA public"
```

```
pg_restore: creating EXTENSION "plpgsql"
```

```
pg_restore: creating COMMENT "EXTENSION plpgsql"
```

- c. Restore the Tenant DB.

```
$ pg_restore -Fc -d dcapp_Tenant -U Tenant -h localhost -p 5432 -v  
ts/Tenant_database_dump_1548895360.log
```

```
pg_restore: connecting to database for restore
```

```
Password:
```

```
pg_restore: creating SCHEMA "public"
```

```
pg_restore: creating COMMENT "SCHEMA public"
```

```
pg_restore: creating EXTENSION "plpgsql"
```

```
pg_restore: creating COMMENT "EXTENSION plpgsql"
```

5. Verify that all the databases are restored by connecting to the databases.

Restart the stopped containers and use **dca** CLI to confirm that all three service DBs are restored.

```
$ docker ps
```

```
$ docker start goinventory-service-v2.0.0
```

```
$ docker start goFabric-service-v2.0.0
```

```
$ docker start goTenant-service-v2.0.0
```

Appendix E: SLX-OS Device Firmware Update

Overview

This section guides the user to upgrade SLX-OS firmware after successfully deploying DCA. This procedure involves updating both DCA and the SLX-OS firmware. The basic steps are as follows:

1. Persist the configuration.
2. Back up the databases.
3. Update device SLX-OS firmware.
4. Update the devices in Asset Service (refer to [Device Update](#)).
5. Verify the update.

Persist the configuration

The user must persist the Fabric configuration before updating SLX-OS firmware. The switch is rebooted as part of a firmware update, which erases all unsaved configurations from the switch.

```
$ dca inventory device running-config persist --fabric stage3
```

Back up the DCA databases

In case anything goes wrong as part of the firmware update, DCA can be restored from the backup database.

```
$ dca supportsave
```

SLX-OS firmware update on the device

Refer to the SLX-OS software upgrade guide at

<https://documentation.extremenetworks.com/slxos/SW/18rx/slxr-18.1.00-upgradeguide.pdf>

for details specific to device type and release.

Update the inventory

Update the inventory Fabric level or device level. This updates the inventory with updated firmware version.

```
$ dca inventory device update --fabric stage3
```

Verify the firmware update

The new firmware version should be updated in the inventory and Fabric should remain in the same state as it was before the firmware update.

```
$ dca fabric show
```

```
$ dca inventory device list
```


Appendix F: Device Replacement and Compare

As part of the RMA process, the user must be able to replace a device in the application and confirm that there is no impact to the network or configurations. This section addresses these issues.

There are two possible scenarios for device replacement:

1. The device being replaced has the same configuration as the RMAed device.
2. The device being replaced has a configuration that is different from the device that was RMAed

Both options are addressed below.

Device replacement with the same configuration

The prerequisite is that you must ensure that the configuration that existed on the device is copied over to the new device.

For MCT leaf nodes assign node id 1 to device with least IP address and node id 2 for the other device. Please refer switch admin guide for configuring MCT node id.

You must also ensure that the device information that is maintained in DCA is current, by executing the following command.

dca inventory device update -ip <IP address of the device being replaced>

This ensures that the Asset Service has the latest information. If the “older” device is not reachable or responding and the configuration being replayed on the replacement device does not match the details in the Asset Service, you should treat this as option (2) above. – Even if the update operation were automated as part of the replacement, there is no guarantee that the device is reachable and in the correct state.

NOTE: If all the configuration changes to the device are done through DCA, then the above step is not necessary. Execute the following command.

\$ dca inventory device replace -ip=10.24.80.135

```
+-----+-----+-----+-----+-----+-----+-----+-----+
| ID | IP Address | Host Name | Model | Chassis Name | Firmware | ASN | Role | Fabric |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 7 | 10.24.80.135 | Fre-135 | 3001 | BR-SLX9140 | 18s.1.01a | 65000 | | |
+-----+-----+-----+-----+-----+-----+-----+-----+
Device Details
--- Time Elapsed: 15.516360256s ---
```

Any failures due to configuration mismatch would be indicated as follows:

\$ dca inventory device replace --ip 10.24.80.135

Device replacement failure

Key	Reason
Interface IPs Updated	Interface ethernet 0/5 has IP 1.1.1.1/31 but No previous IP was assigned for Device 10.24.80.135
VRF Interface Mapping Added	true

Note the following additional examples of failures.

\$ dca inventory device replace --ip 10.24.80.136

[13:25:41]

Device Replacement Failure

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 10.24.80.136 Interface ethernet 0/25:1 has IP 10.10.10.85/31 but No previous IP was assigned for Device 10.24.80.136 Interface ethernet 0/25:2 has IP 10.10.10.89/31 but No previous IP was assigned for Device 10.24.80.136 Interface ethernet 0/25:3 has IP 10.10.10.13/31 but No previous IP was assigned for Device 10.24.80.136 Interface ethernet 0/26:3 has IP 10.10.10.15/31 but No previous IP was assigned for Device 10.24.80.136 Interface ethernet 0/27 has IP 10.10.10.66/31 but No previous IP was assigned for Device 10.24.80.136 Interface ethernet 0/26:1 has IP 10.10.10.87/31 but No previous IP was assigned for Device 10.24.80.136 Interface ethernet 0/26:2 has IP 10.10.10.91/31 but No previous IP was assigned for Device 10.24.80.136 Interface ethernet 0/32 has IP 10.10.10.3/31 but No previous IP was assigned for Device 10.24.80.136 Interface ethernet 0/31 has IP 10.10.10.1/31 but No previous IP was assigned for Device 10.24.80.136
Added Interfaces	loopback 1
BGP Global Added	64512
VRF Updated	true
VRF Interface Mapping Added	true

Error Details

--- Time Elapsed: 12.232735111s ---

Device replacement with different configuration

Device replacement when the configuration of the device being replaced is not the same as that on the new device is handled as follows.

- 1) Remove the device from DCA, by using the following command.

dca fabric device remove --name <Fabric name> --ip <IP address of the device>

The above command ensures that the device is removed (decommissioned from the Fabric) and all the relevant neighbors are also cleaned up.

NOTE: - This also cleans up the subconfigurations from the MCT and BGP configurations.

- 2) Add the new device to DCA, by using the following command.

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) Once this is done, you must ensure that the device shows up in the Fabric, by means of the following command.

dca fabric show

```
Fabric Name: default, Fabric Description: Default Fabric, Fabric Stage: 3
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| IP ADDRESS | POD | HOST NAME | ASN | ROLE | DEVICE PROV STATE | APP CONFIG STATE | CONFIG GEN REASON | PENDING CONFIGS | VTLB ID | LB ID |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 10.24.80.136 | | Fre-136 | 64512 | spine | not provisioned | cfg ready | DA | SYSP-U,BGP-C,INTIP-C | NA | 1 |
| 10.24.80.137 | | SLX | 64512 | spine | not provisioned | cfg ready | DA | SYSP-U,BGP-C,INTIP-C | NA | 1 |
| 10.24.80.134 | | Fre-134 | 65000 | leaf | not provisioned | cfg ready | DA | SYSP-U,MCT-C,MCT-PA,BGP-C,INTIP-C,EVPN-C,OVG-C | 2 | 1 |
| 10.24.80.135 | | Fre-135 | 65000 | leaf | not provisioned | cfg ready | DA | SYSP-U,MCT-C,MCT-PA,BGP-C,INTIP-C,EVPN-C,OVG-C | 2 | 1 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

CONFIG GEN REASON:
LD - Link Delete, LA - Link Add, IU - Interface Update
MD - MCT Delete, OD - Overlay Gateway Delete, OU - Overlay Gateway Update, ED - Evpn Delete
DD - Dependent Device Update, DA - Device Add, DR - Device ReAdd, ASN - Asn Update, HN - HostName Update, NA - Not Applicable

PENDING CONFIGS:
MCT - MCT Cluster, OVG - Overlay Gateway, SYSP - System Properties, INTIP - Interface IP
C/D/U - Create/Delete/Update, PA/PD - Port Add/Port Delete
```

To see the reason for an application or device error, execute the **dca fabric error show** command..

To see the reason for the configuration refresh, execute the **dca fabric debug config-gen-reason** command.

- 4) Configure the Fabric by using the following command.

dca fabric configure --name <name of the Fabric>

NOTE: You must rerun all the Tenant configurations after the device is added to the Fabric and configured.

Device compare

This feature allows users to get a preview of the configurations on the device that are out of sync with what is in the Asset Service. This is a helper utility that displays a summary of the information to be updated in the Asset database. Execute the following command.

dca inventory device compare --ip <IP address of the device>

```
$ dca inventory device compare --ip 10.24.80.134 |
Device 10.24.80.134 configuration is current in the application.
--- Time Elapsed: 14.996360091s ---
```

If there are configuration differences, the output is as follows

```
$ dca inventory device compare --ip 10.24.80.135
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Key | Reason |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Interface IPs Updated | Interface ethernet 0/5 has IP 1.1.1.1/31 but No previous IP |
| | was assigned for Device 10.24.80.135 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| VRF Interface Mapping Added | true |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

Execute-cli

This feature allows users to execute CLIs on the device(s) that are registered in the Asset Service. This is a convenient helper utility to troubleshoot issues on the device(s).

The --command input parameter can be multiple CLIs delimited by comma or semicolon. The comma delimiter will keep the current sub-mode context of the device whereas the semicolon will start at the top level device context.

Execute the following command.

dca inventory device execute-cli --command <"CLI to be executed on the device(s)"> [--config] --ip <IP address(es) of the device(s)> | --fabric <name of fabric>

Exec-mode command example:

```

$ dca inventory device execute-cli --command 'show cluster management' --ip 10.24.85.74,10.24.85.76
Execute CLI[success]
-----
| IP Address | Host Name | Fabric | Command | Status | Reason | Output |
-----
| 10.24.85.74 | SLX | stage3 | show cluster management | Success | | SLX# show cluster management
| | | | | | | Total Number of Nodes in Cluster : 2
| | | | | | | Node-Id Switch MAC IP Address Status
| | | | | | | -----
| | | | | | | 1 >00:04:96:A2:D8:2E* 10.20.20.4 Co-ordinator
| | | | | | | 2 00:04:96:9F:5C:34 10.20.20.5 Connected
| | | | | | | '*' indicates current node of the management cluster.
| | | | | | | '>' indicates principal node of the management cluster.
-----
| 10.24.85.76 | SLX | stage3 | show cluster management | Success | | SLX# show cluster management
| | | | | | | Total Number of Nodes in Cluster : 2
| | | | | | | Node-Id Switch MAC IP Address Status
| | | | | | | -----
| | | | | | | 1 >00:04:96:A2:D8:2E 10.20.20.4 Co-ordinator
| | | | | | | 2 00:04:96:9F:5C:34* 10.20.20.5 Connected
| | | | | | | '*' indicates current node of the management cluster.
| | | | | | | '>' indicates principal node of the management cluster.
-----
Execute CLI Details
--- Time Elapsed: 3.049923883s ---

```

Config-mode command example:

```
$ dca inventory device execute-cli --command "int eth 0/10-12,no shut;do show run int eth 0/10-12" --config --ip 10.24.85.74,10.24.85.76
Execute CLI[success]
```

IP Address	Host Name	Fabric	Command	Status	Reason	Output
10.24.85.74	SLX	stage3	int eth 0/10-12 no shut	Success		SLX(config)# int eth 0/10-12 SLX(conf-if-eth-0/10-12)# no shut
10.24.85.74	SLX	stage3	do show run int eth 0/10-12	Success		SLX(config)# do show run int eth 0/10-12 interface Ethernet 0/10 no shutdown ! interface Ethernet 0/11 no shutdown ! interface Ethernet 0/12 no shutdown !
10.24.85.76	SLX	stage3	int eth 0/10-12 no shut	Success		SLX(config)# int eth 0/10-12 SLX(conf-if-eth-0/10-12)# no shut
10.24.85.76	SLX	stage3	do show run int eth 0/10-12	Success		SLX(config)# do show run int eth 0/10-12 interface Ethernet 0/10 no shutdown ! interface Ethernet 0/11 no shutdown ! interface Ethernet 0/12 no shutdown !

```
Execute CLI Details
--- Time Elapsed: 5.848328414s ---
```