

# Juniper Apstra 4.0 User Guide

Published  
2023-04-27

RELEASE

# Table of Contents

## Get Started

[Install Apstra Software | 1](#)

[Devices | 1](#)

[Design | 2](#)

[Resources | 2](#)

[Blueprints | 2](#)

[Next Steps | 3](#)

## Apstra Server

[Apstra Server Installation and Configuration | 4](#)

[Install on ESXi | 4](#)

[Install on KVM | 8](#)

[Install on KVM with Virtual Machine Manager | 8](#)

[Install on KVM with CLI | 13](#)

[Install on Hyper-V | 15](#)

[Install on VirtualBox | 19](#)

[Initial Configuration | 21](#)

[Configure Static Management IP Address \(Apstra Server\) | 22](#)

[Change Hostname \(Apstra Server\) | 23](#)

[Configure Docker Subnets | 24](#)

[Configure New SSH Keys | 25](#)

[Apstra GUI Overview | 26](#)

[Replace SSL Certificate with Signed One | 27](#)

[Replace SSL Certificate with Self-Signed One | 29](#)

[Check GUI Version | 30](#)

Update GUI Version | 30

Restore GUI Version | 31

Reset GUI Admin Password | 31

## **Apstra Server Management | 32**

Monitor Apstra Server via CLI | 32

Restart Apstra Server | 33

Reset Apstra Server VM Password | 33

Reinstall Apstra Server | 38

Apstra Server Database Overview | 39

Back up Database | 40

Restore Database | 41

Reset Database | 45

Migrate Database | 46

## **Apstra Server Upgrade | 50**

Upgrade VM-to-VM (Recommended) | 52

Pre-Upgrade Validation (VM-VM) | 52

Deploy New Apstra Server (VM-VM) | 54

Import State (VM-VM) | 54

Keep Old VM's IP Address (Optional) | 58

Change Operation Mode to Normal (VM-VM) | 58

Upgrade On-box Agents (VM-VM) | 60

Shut Down Old Apstra Server (VM-VM) | 61

Upgrade In-Place | 61

Pre-Upgrade Validation (In-Place) | 62

Deploy New Apstra Server (In-Place) | 63

Change Operation Mode to Normal (In-place) | 66

Upgrade On-box Agents (In-Place) | 68

Upgrade Worker Nodes (Apstra Cluster Only) | 69

## **Reference (Apstra Server) | 70**

Supported Platforms	71
Required Server Resources	71
Required Communication Ports	72
Additional Network Protocols	73
Network Client Services	73
Apstra Server Upgrade Paths	74
Apstra Server Configuration File	75

## Design

### Logical Devices (Design) | 84

Logical Device Overview	84
Create Logical Device	87
Edit Logical Device	90
Delete Logical Device	91

### Interface Maps (Design) | 91

Interface Map Overview	91
Create Interface Map	93
Example: Create Interface Map with Breakout Ports	94
Example: Inter Port Constraints - Disabled Ports	97
Edit Interface Map	100
Delete Interface Map (Design)	101

### Rack Types (Design) | 101

Rack Type Overview	101
Create Rack Type	106
Example: Create Rack Type	106
Edit Rack Type in Global Catalog	109
Edit Rack Type in Template	109

Edit Rack Type in Blueprint | 110

Delete Rack Type | 110

## Templates (Design) | 110

Template Overview | 110

Create Rack Based Template | 116

Create Pod Based Template | 117

Create Collapsed Template | 118

Edit Template | 118

Update Rack Type in Rack Based Template | 118

Delete Template | 118

## Configlets (Design) | 119

Configlet Overview | 119

Create Configlet | 124

Edit / Delete Configlet (Design) | 125

Edit Configlet | 125

Delete Configlet | 125

## Property Sets (Design) | 126

Property Set Overview | 126

Create Property Set | 127

Edit Property Set | 127

Delete Property Set (Design) | 127

## TCP/UDP Port Aliases (Design) | 128

TCP/UDP Port Alias Overview | 128

Create TCP/UDP Port Alias | 128

Edit TCP/UDP Port Alias | 129

Delete TCP/UDP Port Alias | 129

## Tags (Design) | 129

Tags Overview	129
Create Tag (Design)	131
Edit Tag (Design)	131
Delete Tag (Design)	131

## Devices

### Managed Devices (Devices) | 132

Managed Devices Overview	133
Acknowledge Device(s)	134
Set Admin State on Devices	134
Update User Config	135
Edit System	135
Modular Devices and Device Profiles	137
Edit Pristine Config	138
Update Pristine Config from Device	139
Delete System(s)	139
Managed Device Telemetry	140

### Device Configuration Lifecycle | 140

Terminology	140
Configuration Stages: Overview	141
Configuration Stages: Detail	143
Configuration Deviations	147
Device Offline (Unavailable)	148
Manually Apply Full Config	148
Deploy Modes	148

### Add Device | 150

### Deploy Device | 150

**Drain Device Traffic | 152**

- Upgrade MLAG Pair | 154

**Upgrade Device NOS | 158**

- NOS Upgrade Overview | 158
- Update User-defined Device Profiles | 160
- Upgrade OS | 162

**Remove Device | 166**

- Remove Device from Blueprint | 166
- Remove Device from Apstra Management | 168
- Replace Device | 168

**Reuse Device | 168****Device AAA Support | 169**

- Device AAA Overview | 169
- Supported Platforms | 170

**Telemetry (Devices) | 171**

- Services | 172
- Service Registry | 174
  - Service Registry Overview | 174
  - Import Service Schemas | 175
  - Delete Service Registry | 176
- Telemetry Collection Statistics | 176
- Telemetry Streaming | 177
- Route Anomalies for a Host - Example | 178
- Telemetry Command Reference | 181
- Cisco Telemetry | 181
- Arista Telemetry Commands | 182
- Cumulus | 182

Linux Servers | **183**

Debugging Telemetry | **183**

## **Agents (Devices) | 184**

Create On-box Agent | **189**

Create Off-box Agent | **190**

Uninstall Agent | **193**

Edit / Delete Agent | **194**

    Edit Agent | **194**

    Delete Agent | **194**

Juniper Device Agent | **195**

    Juniper ZTP | **195**

    Disable ZTP | **195**

    Apply Initial Juniper Junos Configuration | **196**

    Configure super-user User | **197**

    Configure IP address and Management VRF | **198**

    Configure SSH and NETCONF | **199**

    Add Junos License Configuration | **199**

SONiC Device Agent | **199**

    SONiC Device Agent Overview | **200**

    Configure Management IP Manually (SONiC) | **200**

    Install Agent Manually (SONiC) | **202**

    Uninstall Agent Manually (SONiC) | **206**

Cisco Device Agent | **207**

    Cisco NX-OS Device Agent Overview | **207**

    Device Configuration Requirements | **208**

    Resize and Enable Guestshell | **208**

    Download Agent Installer | **209**

    Install Cisco Device Agent | **210**

    Device Agent Configuration File | **210**

    Activate Apstra Devices on the Apstra Server | **211**

    Deploy Device | **211**

    Reset Apstra Device Agent | **211**

- Uninstall Apstra Device Agent | 211
- Remove Apstra EEM Scripts | 212
- Cisco Agent Troubleshooting | 212

#### Arista Device Agent | 219

- Initial Arista EOS Configuration | 220
- Decommission Device | 222
- Remove Apstra Package from Device | 223
- Restart System | 224
- Manually Install Arista Device Agent | 225
- Device Agent Configuration File | 227
- Arista Agent Troubleshooting | 227

#### Cumulus Device Agent | 239

- Quick start | 240
- Cumulus Initial Configuration | 242
- Download Agent Installer | 245
- Install Cumulus Device Agent | 246
- Device Agent Configuration File | 247
- Device Agent Management | 248
- Deploy Device | 249
- Uninstall Apstra Device Agent | 249
- Cumulus Agent Troubleshooting | 251

### Agent Profiles (Devices) | 260

- Create Agent Profile | 261
- Edit / Delete Agent Profile | 262
  - Edit Agent Profile | 262
  - Delete Agent Profile | 262

### Packages (Devices) | 262

- Packages Overview | 263
- Upload Packages | 263

### OS Images (Devices) | 264

- Register OS Image | 264
- Method One: Upload Image | 265

Method Two: Provide Image URL | 265

Add Checksum (Optional) | 266

Edit OS Image | 267

Delete OS Image | 267

## **Apstra ZTP (Devices) | 267**

Apstra ZTP Overview | 268

Download and Deploy Apstra ZTP VM | 273

Configure Static Management IP Address (Apstra ZTP) | 274

Configure ZTP User | 275

Configure DHCP Server | 276

Configure Controller IP Address for ZTP | 279

Edit Apstra ZTP Configuration File | 279

Apstra ZTP - Juniper Junos | 286

Juniper Junos and ZTP Disk Space | 286

Example: Juniper Junos ztp.json | 287

Juniper Junos Bootstrap File | 287

Juniper Junos Custom Config File | 287

Restart Juniper Junos ZTP | 288

Troubleshoot Juniper Junos ZTP | 289

Apstra ZTP - SONiC | 289

Enterprise SONiC and ZTP Overview | 289

Example: Enterprise SONiC ztp.json | 290

Enterprise SONiC Custom Config File | 290

Restart Enterprise SONiC ZTP | 291

| 291

Apstra ZTP - Cisco | 291

Cisco NX-OS and ZTP Disk Space | 291

Example: Cisco NX-OS ztp.json | 292

Cisco NX-OS Custom Config File | 292

Cisco NX-OS Off-box Agent Custom Config File | 293

- Restart Cisco NX-OS ZTP | 293

## Apstra ZTP - Arista | 294

- Arista EOS | 294
- Example: Arista EOS ztp.json | 295
- Arista EOS Custom Config File | 295
- Restart Arista EOS ZTP | 296

## Apstra ZTP - Cumulus | 297

- Cumulus Linux | 297
- Example: Cumulus Linux ztp.json | 298
- Cumulus Linux Custom Config File | 299
- Restart Cumulus Linux ZTP | 299
- Troubleshoot Cumulus Linux ZTP | 300

## Device Profiles (Devices) | 300

- Device Profile Overview | 300

- Create Device Profile | 308

- Edit Device Profile | 309

- Edit Device Profile | 309
- Edit Device Profile - Example | 309

- Delete Device Profile | 312

- Juniper Device Profiles | 312

- SONiC Device Profile | 314

- Background | 315
- Problem Statement | 315
- Solution | 315
- User Interface | 315
- Selector information | 316
- Capabilities | 316
- Interface naming conventions | 317
- Troubleshooting | 317
- Example: DP and port\_config.ini | 318

- Cumulus Device Profile | 359

Background	360
Problem Statement	360
Solution	360
User Interface	361
DP Data Model	361
Get selector information from the device	368
Model and manufacturer/vendor	370
OS Version	371
OS Family	371
Common default Values	372
Capabilities	373
Port-specific semantics	374
Interface Name semantics on Cumulus	375
Port Setting Schema on Cumulus	375
Edit Interface Setting per Interface on UI	376
Auto-negotiation for 10GBaseT ports	380
Inter port constraints	381
Dell Z9100-ON	382
Dell 4128F	383
Dell 4148T	384
Dell 4148F	385
Edgecore 6812_32x_O	385
Mellanox 2700	386
Troubleshooting	390
Appendix	390
References	390

## Resources

### ASN Pools (Resources) | 391

ASN Pool Overview	391
Create ASN Pool	392
Edit ASN Pool	392
Delete ASN Pool	393

### VNI Pools (Resources) | 393

VNI Pool Overview | 393

Create VNI Pool | 394

Edit VNI Pool | 394

Delete VNI Pool | 395

## **IP Pools (Resources) | 395**

IP Pool Overview | 395

Create IPv4 Pool | 396

Edit IPv4 Pool | 397

Delete IPv4 Pool | 397

## **IPv6 Pools (Resources) | 397**

IPv6 Pool Overview | 398

Create IPv6 Pool | 399

Edit IPv6 Pool | 399

Delete IPv6 Pool | 399

## **Blueprints**

Create Blueprint | 400

Blueprint Dashboard | 401

Delete Blueprint | 402

## **Analytics**

Analytics Overview | 403

Analytics Dashboard | 404

Configure Auto-Enabled Dashboards | 405

Instantiate Predefined Dashboard | 405

Create Analytics Dashboard | 406

Edit / Delete Dashboard | 406

Edit Dashboard | 406

- [Delete Dashboard | 407](#)

**Anomalies (Analytics) | 407**

**Widgets Overview | 407**

**Create Anomaly Heat Map Widget | 408**

**Create Stage Widget | 409**

- [Create Stage Widget from Widgets View | 409](#)

- [Create Stage Widget from Probes View | 409](#)

**Edit / Delete Widget | 410**

- [Edit Widget | 410](#)

- [Delete Widget | 410](#)

**Probes | 410**

- [IBA Probes Overview | 411](#)

**Instantiate Predefined Probe | 416**

**Create Probe | 417**

**Import / Export Probe | 417**

- [Import Probe | 417](#)

- [Export Probe | 417](#)

**Edit / Delete Probe | 418**

- [Edit Probe | 418](#)

- [Delete Probe | 418](#)

**Staged (Blueprints)**

**Build (Physical) | 419**

- [Stage Physical Resources | 420](#)

- [Update Physical Resource Assignments | 420](#)

- [Reset Physical Resource Group Overrides | 421](#)

- [Stage Device Profiles | 422](#)

- [Stage Devices | 423](#)

- Stage Device Overview | 423
- Assign (Multiple) System IDs and Deploy Modes | 424
- Assign (Single) System ID | 427
- Set Deploy Mode (Single Device) | 428
- Change Hostname - One Device | 429

Stage Configlets | 429

## Topology (Staged) | 430

- 2D Topology View (Staged) | 430
- 3D Topology View (Staged) | 432
- Neighbors View (Staged) | 434
- Links View (Staged Topology) | 436
- Virtual Networks Endpoints (Staged) | 437
- Add Links | 437
- Add Leaf Peer Links | 441
- Update Logical Links and LAG Mode | 442
- Change Link Speeds | 444
- Delete Link | 445

## Nodes (Staged) | 446

- Add Generic System | 448
  - Add Generic System (from Topology View) | 449
  - Copy Existing Generic (from Topology View) | 453
- Add External Generic System | 455
  - Add External Generic (from Topology View) | 456
  - Add External Generic (from Nodes View) | 459
- Edit Server Names and Hostnames | 463
- Edit Port Channel ID Min Max | 463
- Set Deploy Mode (Multiple Devices) | 464
- Edit Device Details | 464

Edit Device Properties | 465

Add/Remove Tags (Single Node) | 466

Add/Remove Tags (Multiple Nodes) | 466

View Node's Static Routes | 467

## **Links (Staged) | 468**

Import Cabling Map | 470

Export Cabling Map | 470

Override Cabling (GUI) | 471

Override Cabling (JSON) | 472

Change Link Speeds | 473

Fetch Discovered LLDP Data | 473

Edit Link Properties | 474

Add/Remove Tags on One Link | 475

Add/Remove Tags (Multiple Links) | 476

## **Racks (Staged) | 477**

Change Rack Name | 479

Add Rack | 479

Export Rack Type | 479

Edit Rack | 480

Delete Rack | 480

## **Pods (Staged) | 481**

Change Pod Name | 483

Add Pod | 483

Delete Pod | 484

## **Planes (Staged) | 485**

Planes Overview | 486

| Add Superspines per Plane | 486

## Virtual | 487

Stage Virtual Resources | 487

| Update Virtual Resources Assignments | 487

| Reset Virtual Resource Group Overrides | 488

Virtual Networks | 488

| Virtual Networks Overview | 489

| Create Virtual Network | 494

| Edit Virtual Network | 496

| Delete Virtual Network | 496

Routing Zones (Virtual) | 496

| Routing Zone Overview | 496

| Create Routing Zone | 498

| Assign Resources to Routing Zone | 499

| Assign DHCP to Routing Zone | 500

| Edit Routing Zone | 500

| Delete Routing Zone | 501

Protocol Sessions (Virtual) | 501

Data Center Interconnect (DCI) / Remote EVPN Gateways (Virtual) | 502

| DCI / EVPN Gateway Overview | 503

| DCI Deployment Options | 504

| Implementation | 506

| Apstra Workflow | 509

Virtual Infra (Virtual) | 514

| VMware vCenter/vSphere Virtual Infra | 515

| NSX-T Integration | 521

| NSX-T 3.0 Edge and Connectivity Templates | 531

| VMware NSX-T Inventory Mapping to Apstra Virtual Infrastructure | 542

Endpoints Overview (Virtual) | 578

| Internal Endpoints (Virtual) | 579

| External Endpoints (Virtual) | 580

| Enforcement Points (Virtual) | 581

| Endpoint Groups (Virtual) | 581

## **Policies | 583**

Security Policies | 583

| Security Policy Overview | 583  
| Security Policy Parameters | 585  
| Create Security Policy | 587  
| Policy Errors | 588  
| Edit Security Policy | 589  
| Delete Security Policy | 589  
| Security Policy Search | 589  
| Security Policy Conflicts | 590  
| Security Policy Settings | 591

Interface Policies | 591

Routing Policies | 599

| Routing Policy Overview | 599  
| Create Routing Policy | 602  
| Edit Routing Policy | 602  
| Delete Routing Policy | 602

Fabric Addressing Policy | 603

| Enable IPv6 Applications | 603  
| ESI MAC MSB | 603

Virtual Network Policy | 604

| Virtual Network Policy Overview | 604  
| Modify Virtual Network Policy | 606

Anti-Affinity Policy | 606

| Anti-Affinity Policy Overview | 607  
| Enable/Disable Anti-Affinity Policy | 608

Modify SVI IP Validation Policy | 608

## **Catalog | 609**

Logical Devices (Blueprint Catalog) | 609

| Logical Devices Overview (Blueprint Catalog) | 609

Export Logical Device | 610

#### Interface Maps (Blueprint Catalog) | 610

Interface Maps Overview (Blueprint) | 611

Import Interface Map | 611

Delete Interface Map (Blueprint) | 611

#### Property Sets (Blueprint Catalog) | 612

Property Sets Overview (Blueprint) | 612

Import Property Set | 612

Re-import Property Set | 613

Delete Property Set (Blueprint) | 613

#### Configlets (Blueprint Catalog) | 614

Import Configlet | 615

Edit (Blueprint) Configlet | 617

Delete (Blueprint) Configlet | 618

#### AAA Servers (Blueprint Catalog) | 618

AAA Servers Overview | 619

Create AAA Server | 620

Edit AAA Server | 620

Delete AAA Server | 620

AAA RADIUS Server Configuration Tasks | 620

Client Supplicant Configuration Tasks | 621

#### Tags (Blueprint Catalog) | 622

Tags Overview (Blueprint) | 622

Search Tags (Blueprint) | 623

Create Tag (Blueprint) | 623

Import Tag | 624

Export Tag | 624

Edit Tag (Blueprint) | 624

Delete Tag (Blueprint) | 624

#### Tasks | 624

#### Connectivity Templates | 625

Primitives | 625

- Virtual Network (Single) | 627
- Virtual Network (Multiple) | 628
- IP Link | 628
- Static Route | 629
- Custom Static Route | 630
- BGP Peering (IP Endpoint) | 631
- BGP Peering (Generic System) | 632
- Dynamic BGP Peering | 634
- Routing Policy | 635
- User-defined | 636
- Pre-defined | 637

View Connectivity Templates | 637

Create Connectivity Template for Multiple VNs on Same Interface (Example) | 638

Create Connectivity Template for Layer 2 Connected External Router (Example) | 640

Assign Connectivity Template | 643

- Assign Connectivity Template Overview | 643
- Method 1 | 644
- Method 2 | 645
- Force Assign VN Templates | 646

Edit Connectivity Template | 647

Delete Connectivity Template | 647

Find by Tags | 647

## Uncommitted (Blueprints)

Uncommitted Overview | 648

Review Staged Changes | 650

Commit Staged Changes | 653

Revert Staged Changes | 653

## Active (Blueprints)

Status (Active) | 655

**Selection (Active) | 655****Topology (Active) | 656**

2D Topology View (Active) | 656

3D Topology View (Active) | 658

Neighbors View (Active) | 659

Links View (Active Topology) | 662

Virtual Networks Endpoints (Active) | 663

Headroom (Topology) | 663

**Nodes (Active) | 665**

Active Nodes Overview | 666

Apply Full Config | 666

**Links (Active) | 667**

Active Links Overview | 667

Export Cabling Map | 667

**Racks (Active) | 668**

Change Rack Name | 668

**Pods (Active) | 669****Query | 670****Anomalies (Service) | 671**

Discovery Anomalies | 672

Configuration Deviation | 676

**Root Causes | 679**

Root Cause Overview | 680

Enable Root Cause Analysis | 680

View Root Cause Analysis | 681

**Time Voyager (Blueprints)**

[Time Voyager Overview | 682](#)

[Jump to Previous Blueprint Revision | 684](#)

[Keep Saved Blueprint Revision | 685](#)

[Update Blueprint Revision Description | 685](#)

[Delete Kept Blueprint Revision | 685](#)

## **Providers (External Systems)**

### **LDAP Provider | 687**

[Create LDAP Provider | 687](#)

[Configure LDAP Provider | 689](#)

### **Active Directory Provider | 690**

[Create Active Directory Provider | 690](#)

### **TACACS+ Provider | 691**

[Create TACACS+ Provider | 692](#)

[Configure TACACS+ Provider | 692](#)

### **RADIUS Provider | 693**

[RADIUS Limitations | 693](#)

[Create RADIUS Provider | 694](#)

### **Edit / Delete Provider | 695**

[Edit Provider | 695](#)

[Delete Provider | 696](#)

### **Provider Role Map Overview | 696**

### **Create Provider Role Map | 697**

### **Edit / Delete Role Map | 697**

[Edit Role Map | 698](#)

[Delete Role Map | 698](#)

## **Platform**

**User/Role Management (Platform) | 699**

User Profile Management | 699

User Role Management | 700

User Profile Use Cases | 702

| Use Case Overview | 702

Create User Profile | 706

Change User Password | 706

Log Out User | 706

Edit / Delete User Profile | 706

| Edit User Profile | 706

| Delete User Profile | 707

User Role Use Cases | 707

| Use Cases Overview | 707

Create User Role | 712

Edit / Delete User Role | 712

| Edit User Role | 713

| Delete User Role | 713

**Security (Platform) | 713**

Allowed List | 714

| Allowed List Overview | 714

| Add IP/Subnet to Allowed List | 714

| Edit IP/Subnet to Allowed List | 715

| Delete IP/Subnet from Allowed List | 715

Banned List | 715

| Banned List Overview | 715

| Delete IP/Subnet from Banned List | 716

Rate Limit Configuration | 716

| Rate Limit Configuration Overview | 716

| Edit Rate Limit Configuration | 717

[Edit Password Complexity Requirements | 717](#)

## **Syslog Configuration (Platform) | 718**

[Syslog Configuration Overview | 718](#)

[Create Syslog Config | 720](#)

[Edit Syslog Config | 721](#)

[Delete Syslog Config | 721](#)

## **Receivers (Platform) | 721**

[Streaming Receivers Overview | 722](#)

[Create Receiver | 722](#)

[Delete Receiver | 723](#)

[Configure Receivers Using Telegraf Plugin | 723](#)

## **Global Statistics (Platform) | 724**

## **Event Log (Platform) | 725**

[Event Log Overview | 725](#)

[Export Event Log to CSV File | 727](#)

[Send Event Log to External System | 727](#)

## **Apstra Cluster (Platform) | 728**

[Cluster Nodes | 728](#)

[Nodes Overview | 728](#)

[Create Apstra Node | 731](#)

[Edit Apstra Node | 731](#)

[Delete Apstra Node | 731](#)

[Cluster Management | 732](#)

## **Developers (Platform) | 734**

[Resource Pools \(API\) | 735](#)

[Configlets \(API\) | 746](#)

[Property Sets \(API\) | 749](#)

Interface Descriptions (API) | 751

Probes (API) | 755

RCI Fault Model (API) | 769

Apstra Cluster (API) | 773

API From Python | 774

REST API Explorer | 776

## **Technical Support (Platform) | 777**

Show Tech: Apstra Controller and On-box Agents (GUI) | 778

Show Tech: Off-box Agents (CLI) | 781

Show Tech: Infra Off-box Agents (CLI) | 781

Show Tech: Apstra Server (CLI) | 782

Show Tech: On-box Agents (CLI) | 783

## **Favorites & User**

**Manage Favorites | 784**

**Change Your User Password | 786**

**Change Your User Name/Email | 786**

**Log Out | 786**

## **Guides**

### **Extensible Telemetry Guide | 787**

Extensible Telemetry Overview | 787

Set Up Development Environment | 788

Develop Collector | 789

Write Collector | 792

Unit Test Collector | 798

Package Collector | 800

Upload Packages | 800

Use Telemetry Collector | 801

## 5-Stage Clos Architecture | 802

5-Stage Clos Overview | 802

Create 5-Stage Clos Network | 804

Modify 5-stage Clos Network | 805

## Access Layer Switches | 805

## Juniper EVPN Support | 806

Overview | 807

EVPN multi-homing Terminology and Concepts | 807

Topology Specification | 808

EVPN Services | 810

Configuration Rendering | 811

## Intent-Based Analytics with AOS-CLI Utility | 814

IBA with AOS-CLI Overview | 814

Install AOS-CLI | 815

Install Packages | 815

Create Agent Profiles | 818

Create Agents | 819

Update Agents from AOS-CLI | 821

Install IBA Probes | 822

Apstra IBA Probes Examples | 824

## AOSOM-Streaming Guide | 828

AOSOM-Streaming Overview | 829

Configure Aosom-Streaming | 834

Reconfigure Aosom-streaming after Apstra Server Upgrade | 836

Build Aosom-Streaming VM (Optional) | 837

| Troubleshooting | 841

## Mixed Uplink Speeds between Leafs and Spines | 842

## Enable VXLAN Routing on Cumulus Tomahawk | 845

Overview | 845

Create Device Profile for Hyperloop | 845

Update Logical Device for Hyperloop | 847

Create Interface Map for Hyperloop | 849

Update Managed Devices for Hyperloop | 850

Assign Hyperloop Device Profile | 851

Verify Loopback Port | 851

## References

## Apstra Feature Matrix | 853

Apstra 4.0.2 Feature Matrix | 853

Fabric Roles | 854

Fabric Connectivity | 854

Device Management | 855

Connectivity (from Leaf Layer) | 856

Connectivity (from Access Layer) | 856

Routing Policies | 857

Miscellaneous | 857

Virtual Network CT Type | 858

IP Link CT Type | 858

Static Route CT Type | 859

Custom Static Route CT Type | 859

BGP to Generic CT Type | 860

BGP to IP Endpoint CT Type | 863

Dynamic BGP Peering CT Type | 864

Routing Policy CT Type | 865

BGP Attributes (common to all BGP CTs) | 865

Apstra 4.0.1 Feature Matrix | 866

Fabric Roles | 867

Fabric Connectivity	867
Device Management	868
Connectivity (from Leaf Layer)	868
Connectivity (from Access Layer)	869
Routing Policies	870
Miscellaneous	870
Virtual Network CT Type	870
IP Link CT Type	871
Static Route CT Type	871
Custom Static Route CT Type	872
BGP to Generic CT Type	872
BGP to IP Endpoint CT Type	876
Dynamic BGP Peering CT Type	878
Routing Policy CT Type	879
BGP Attributes (common to all BGP CTs)	880

#### Apstra 4.0.0 Feature Matrix | 880

Fabric Connectivity	881
Device Management	882
Connectivity (from Leaf Layer)	882
Routing Policies	883
Miscellaneous	884
Virtual Network CT Type	884
IP Link CT Type	885
Static Route CT Type	885
Custom Static Route CT Type	886
BGP to Generic CT Type	886
BGP to IP Endpoint CT Type	890
Dynamic BGP Peering CT Type	892
Routing Policy CT Type	893
BGP Attributes (common to all BGP CTs)	894

#### Qualified Device and NOS (Devices) | 894

Apstra Release 4.0.2	895
Apstra Release 4.0.1	897
Apstra Release 4.0.0	899

Juniper Junos OS Evolved on Apstra Versions 4.0.1 and 4.0.0 | 901

## Agent Configuration File (Devices) | 901

Controller Section | 902

Service Section | 903

Logrotate Section | 904

Device Info Section | 905

Device Profile Section | 905

## NOS Upgrade Paths (Devices) | 906

## Apstra EVPN Support Addendum | 909

Qualified Vendor and NOS | 910

Limitations | 911

Cumulus RN-766 Support | 912

TCAM Carving in NX-OS | 913

Arista EOS VxLAN Routing | 914

Graph Node VTEP Types | 915

## Predefined Dashboards (Analytics) | 919

Device Health Summary Dashboard | 919

Drain Validation Dashboard | 920

Throughput Health MLAG Dashboard | 920

Traffic Trends Dashboard | 920

Virtual Infra Fabric Health Check Dashboard | 920

Virtual Infra Redundancy Check Dashboard | 921

## Predefined Probes (Analytics) | 921

Bandwidth Utilization Probe | 923

Critical Services: Utilization, Trending, Alerting Probe (new in 4.0.1) | 925

Device System Health Probe | 926

Device Traffic Probe	928
Drain Traffic Anomaly Probe	932
ECMP Imbalance (External Interfaces) Probe	933
ECMP Imbalance (Fabric Interfaces) Probe	935
ECMP Imbalance (Spine to Superspine Interfaces) Probe	938
ESI Imbalance Probe	940
EVPN VXLAN Type-3 Route Validation Probe	942
EVPN VXLAN Type-5 Route Validation Probe	944
External Routes Probe	946
Hot/Cold Interface Counters (Fabric Interfaces) Probe	946
Hot/Cold Interface Counters (Specific Interfaces) Probe	951
Hot/Cold Interface Counters (Spine to Superspine Interfaces) Probe	953
Hypervisor & Fabric LAG Config Mismatch Probe (Virtual Infra)	955
Hypervisor & Fabric VLAN Config Mismatch Probe (Virtual Infra)	956
Hypervisor & Fabric VLAN Config Mismatch Probe Overview	957
Usage with NSX-T Integration	958
Usage with VCenter Integration	963
Hypervisor MTU Mismatch Probe (Virtual Infra)	963
Hypervisor MTU Threshold Check Probe (Virtual Infra)	964
Hypervisor Missing LLDP Config Probe (Virtual Infra)	965
Hypervisor Redundancy Checks Probe (Virtual Infra)	965
Interface Flapping (Fabric Interfaces) Probe	966
Interface Flapping (Specific Interfaces) Probe	968
Interface Flapping (Specific Interfaces) Probe	970
Interface Policy 802.1x Probe	972
LAG Imbalance Probe	973
Leafs Hosting Critical Services: Utilization, Trending, Alerting Probe (new in 4.0.1)	975

Link Fault Tolerance in Leaf and Access LAGs Probe | 976

MLAG Imbalance Probe | 978

Multiagent Detector Probe | 982

Packet Discard Percentage Probe | 983

Spine Fault Tolerance Probe | 985

Total East/West Traffic Probe | 986

VMs without Fabric Configured VLANs Probe (Virtual Infra) | 988

VXLAN Flood List Validation Probe | 991

### **Probe Processors (Analytics) | 993**

Processor: Accumulate | 994

Processor: Average | 998

Processor: Comparison | 999

Processor: EVPN Type 3 | 1000

Processor: EVPN Type 5 | 1001

Processor: Extensible Service Data Collector | 1002

Processor: Generic Graph Collector | 1005

Processor: Generic Service Data Collector | 1008

Processor: Interface Counters | 1011

Processor: Logical Operator | 1014

Processor: Match Count | 1015

Processor: Match Percentage | 1017

Processor: Match String | 1019

Processor: Max | 1022

Processor: Min | 1024

Processor: Periodic Average | 1026

Processor: Range | 1029

Processor: Ratio | 1032

Processor: Service Data Collector | 1034

Processor: Set Comparison | 1037

Processor: Set Count | 1039

Processor: Standard Deviation | 1040

Processor: State | 1042

Processor: Subtract | 1045

Processor: Sum | 1046

Processor: System Utilization | 1047

Processor: Time in State | 1048

Processor: Traffic Monitor | 1053

Processor: Union | 1055

Processor: VXLAN Floodlist | 1057

## **Configlet Examples (Design) | 1057**

### **Graph | 1066**

Graph Overview | 1066

Query Specification | 1067

Change Notification | 1069

Notification Processing | 1070

Putting It All Together | 1071

Convenience Functions | 1072

Apstra Graph Datastore | 1081

## **Juniper Apstra Technology Previews (Tech Previews) | 1082**

# Get Started

## IN THIS SECTION

- [Install Apstra Software](#) | 1
- [Devices](#) | 1
- [Design](#) | 2
- [Resources](#) | 2
- [Blueprints](#) | 2
- [Next Steps](#) | 3

Welcome! Juniper Apstra (formerly known as AOS) automates all aspects of the data center network design, build, deploy, and operation phases. It leverages advanced intent-based analytics to continually validate the network, thereby eliminating complexity, vulnerabilities, and outages resulting in a secure and resilient network. To get started with Juniper Apstra, you'll install and configure the software, replace the SSL certificate and default passwords, then start building the elements of your physical network. Depending on the complexity of your design, other tasks may be required in addition to the ones included in this general workflow.

## Install Apstra Software

1. ["Install and configure"](#) on page 4 the Apstra server.
2. ["Replace the SSL certificate"](#) on page 27 and ["change the default password"](#) on page 21 for the Apstra GUI.

## Devices

1. ["Device profiles"](#) on page 300 (Devices > Device Profiles) represent the physical devices in your network. Many device profiles are predefined for you. Check the list, and if one that you need is not included, you can create it.
2. Create and install ["device agents"](#) on page 184 (Devices > System Agents > Agents) for the devices to be managed in the Apstra environment. If you have many of the same devices using the same

configuration you might consider creating ["agent profiles" on page 260](#) (Device > Agent Profiles), which can streamline the task of creating many agents.

3. When agents are created, they appear in the managed devices list in the quarantined state. ["Acknowledge" on page 132](#) them (Devices > Managed Devices) to put them in the ready state which allows them to be managed by Apstra software. (If you have a ["modular device" on page 132](#) in your network, you may need to change the associated device profile. It's best to do this before acknowledging.)

## Design

1. ["Logical devices" on page 84](#) (Design > Logical Devices) are abstractions of physical devices. They allow you to specify device capabilities before selecting specific vendor hardware. Check the logical device design (global) catalog for ones that meet your requirements; create them if needed.
2. ["Interface maps" on page 91](#) (Design > Interface Maps) combine device profiles and logical devices. Check the interface map design (global) catalog for ones that meet your requirements; create them if needed.
3. ["Rack types" on page 101](#) (Design > Rack Types) are logical representations of racks. Check the rack type design (global) catalog for ones that meet your requirements; create them if needed.
4. ["Templates" on page 110](#) (Design > Templates) are used to build rack designs (blueprints). Check the template design (global) catalog for one that meets your requirements; create it if needed.

## Resources

Create resource pools (["ASNs" on page 391](#), ["IPv4 addresses" on page 395](#), and ["IPv6 addresses" on page 397](#) if needed) for your network. When you're ready to assign resources to your blueprint, you'll specify a resource pool, then the resources will automatically be assigned from that pool.

## Blueprints

1. Create a ["blueprint" on page 400](#) from one of the templates in the design section.
2. Build the network (Blueprints > <your\_blueprint\_name> > Staged > Physical > Build) by assigning ["resources" on page 420](#), ["device profiles" on page 422](#), and ["devices" on page 423](#) (S/Ns).
3. Review the calculated ["cabling map" on page 468](#) (Blueprints > <your\_blueprint\_name> > Staged > Physical > Links), then cable up the physical devices according to the map. If you have a set of pre-

cabled switches, ensure that you have configured interface maps according to the actual cabling so that calculated cabling matches actual cabling.

4. When you're finished with assignments and the blueprint is error-free, ["commit" on page 648](#) the blueprint (Blueprints > <your\_blueprint\_name> > Uncommitted). Committing a blueprint initiates work on the intent and realizes it on the network by pushing configuration changes on assigned devices.
5. Review the ["blueprint dashboard" on page 400](#) (Blueprints > Dashboard) for ["anomalies" on page 671](#). If you have cabling anomalies, the likely reason is a mismatch in calculated cabling and actual cabling. Either re-cable the switches, recreate the blueprint with appropriate interface maps or use AOS CLI to override the cabling in the blueprint with discovered cabling.

## Next Steps

After your deployment is running, you can proceed to ["build" on page 487](#) the virtual environment with ["virtual networks" on page 488](#) and ["routing zones" on page 496](#), as needed. You can also refer to the guides in the Reference section to learn about other Apstra capabilities.

# Apstra Server

## IN THIS SECTION

- [Apstra Server Installation and Configuration | 4](#)
- [Apstra GUI Overview | 26](#)
- [Apstra Server Management | 32](#)
- [Apstra Server Upgrade | 50](#)
- [Reference \(Apstra Server\) | 70](#)

## Apstra Server Installation and Configuration

### IN THIS SECTION

- [Install on ESXi | 4](#)
- [Install on KVM | 8](#)
- [Install on Hyper-V | 15](#)
- [Install on VirtualBox | 19](#)
- [Initial Configuration | 21](#)
- [Configure Static Management IP Address \(Apstra Server\) | 22](#)
- [Change Hostname \(Apstra Server\) | 23](#)
- [Configure Docker Subnets | 24](#)
- [Configure New SSH Keys | 25](#)

Apstra software is delivered pre-installed on a single virtual machine (VM). You can install and configure the Apstra VM image on various hypervisor platforms as described in the next sections.

### Install on ESXi

The instructions below are for *installing Apstra software* on an ESXi hypervisor. For information specific to using ESXi, refer to VMware's [ESXi documentation](#).

1. Confirm that you are running a ["supported ESXi version" on page 71](#) and that the platform has enough ["resources" on page 71](#) for the Apstra server.

2. Apstra software is delivered pre-installed on a single virtual machine (VM). As a registered support user, [download the OVA Apstra VM image from Juniper Support Downloads](#).



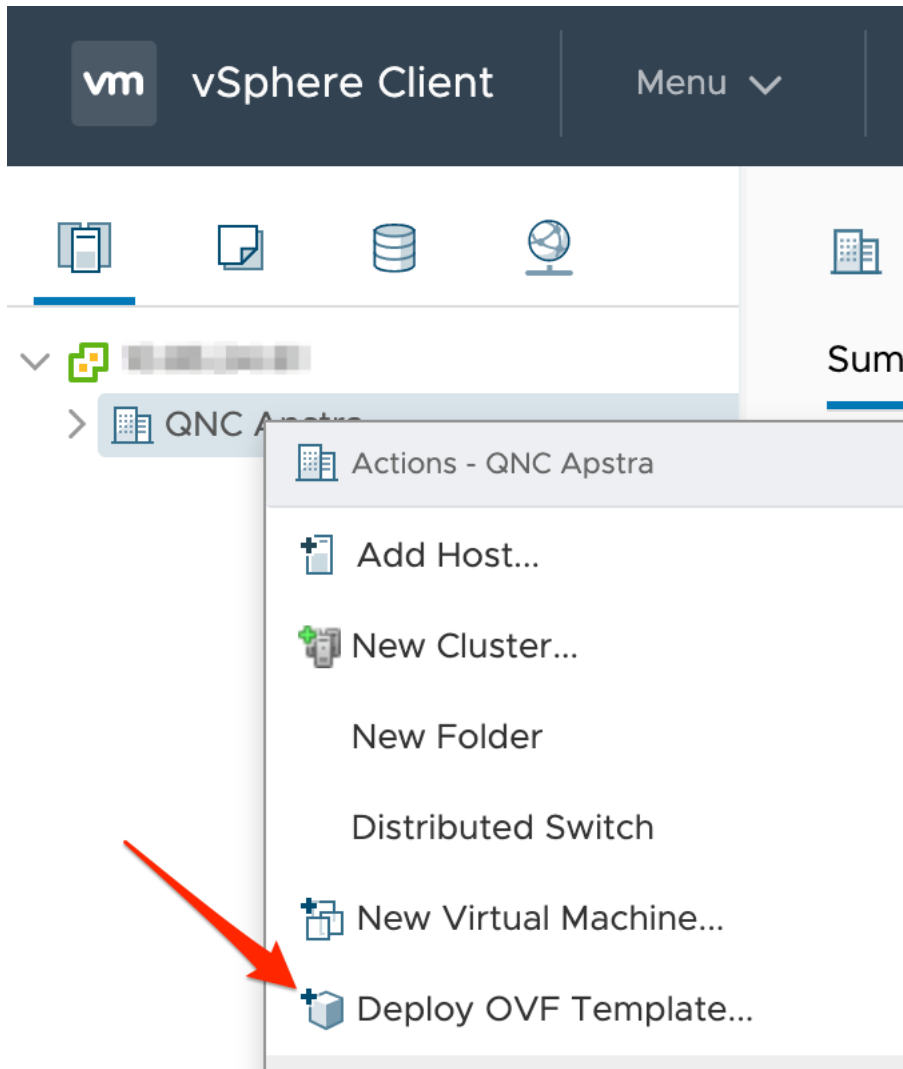
To download the image on your localhost, [CLICK HERE](#)

To download the image directly on your device, use the following URL:

[https://cdn.juniper.net/software/jafc/4.0.2/aos\\_server\\_4.0.2-142.ova?](https://cdn.juniper.net/software/jafc/4.0.2/aos_server_4.0.2-142.ova?)

copy

- Log into vCenter, right-click your target deployment environment and click **Deploy OVF Template**.



- Specify the URL or local file location for the OVA file you downloaded and click **Next**.

## Deploy OVF Template

### 1 Select an OVF template

#### 2 Select a name and folder

#### 3 Select a compute resource

#### 4 Review details

#### 5 Select storage

#### 6 Ready to complete

### Select an OVF template

Select an OVF template from remote URL or local file system

Enter a URL to download and install the OVF package from the Internet, or browse to a location accessible from your computer, such as a local hard drive, a network share, or a CD/DVD drive.

☐ URL

☒ Local file

aos\_server\_4.0.2-142.ova

5. Specify a unique name and target location for the VM and click **Next**.

### Deploy OVF Template

✓ 1 Select an OVF template

2 Select a name and folder

3 Select a compute resource

4 Review details

5 Select storage

6 Ready to complete

Select a name and folder

Specify a unique name and target location

Virtual machine name: aos\_server4.0.2-142

Select a location for the virtual machine.

▼                                

- Map the Apstra Management network to enable it to reach the virtual networks that the Apstra server will manage on ESXi, then click **Next**.

## Deploy OVF Template

- ✓ 1 Select an OVF template
- ✓ 2 Select a name and folder
- ✓ 3 Select a compute resource
- ✓ 4 Review details
- ✓ 5 Select storage
- 6 Select networks**
- 7 Ready to complete

### Select networks

Select a destination network for each source network.

Source Network	Destination Network
VM Network	topology1
	1 items

### IP Allocation Settings

IP allocation: Static - Manual

IP protocol: IPv4

- Review your specifications and click **Finish**.

Start the VM (if not already running) and ["configure" on page 21](#) the Apstra server.

## Install on KVM

### SUMMARY

You can install KVM with **Virtual Machine Manager** or with the CLI.

### IN THIS SECTION


- [Install on KVM with Virtual Machine Manager | 8](#)
- [Install on KVM with CLI | 13](#)

The instructions below are for *installing Apstra software* on a KVM hypervisor. For information specific to using KVM, refer to Linux [KVM documentation](#).

### Install on KVM with Virtual Machine Manager

- Confirm that you are running a ["supported KVM version" on page 71](#) and that the platform has enough ["resources" on page 71](#) for the Apstra server.

2. Apstra software is delivered pre-installed on a single virtual machine (VM). As a registered support user, [download the QCOW2 Apstra VM image from Juniper Support Downloads](#).



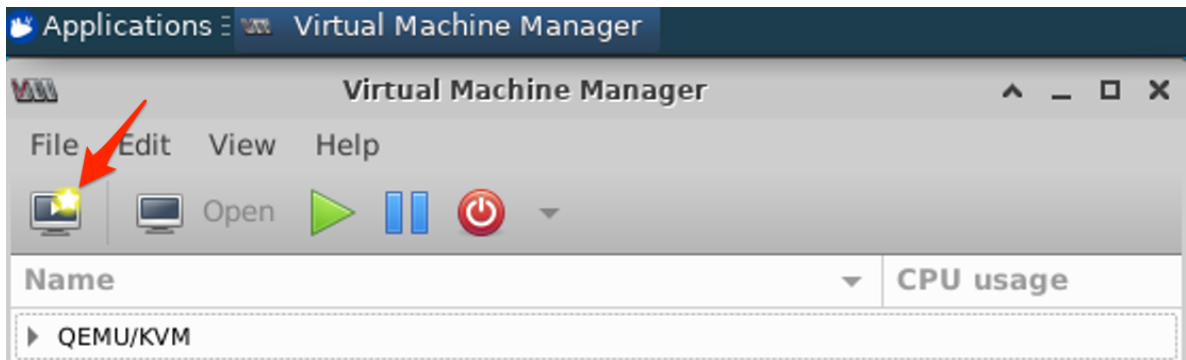
To download the image on your localhost, [CLICK HERE](#)

To download the image directly on your device, use the following URL:

`https://cdn.juniper.net/software/jafc/4.0.2/aos_server_4.0.2-142.qcow2.gz?`  
`[blurred text]`

copy

3. Uncompress and move the disk image to where it will run.
4. Start **Virtual Machine Manager** and click the **Create a new virtual machine** button.

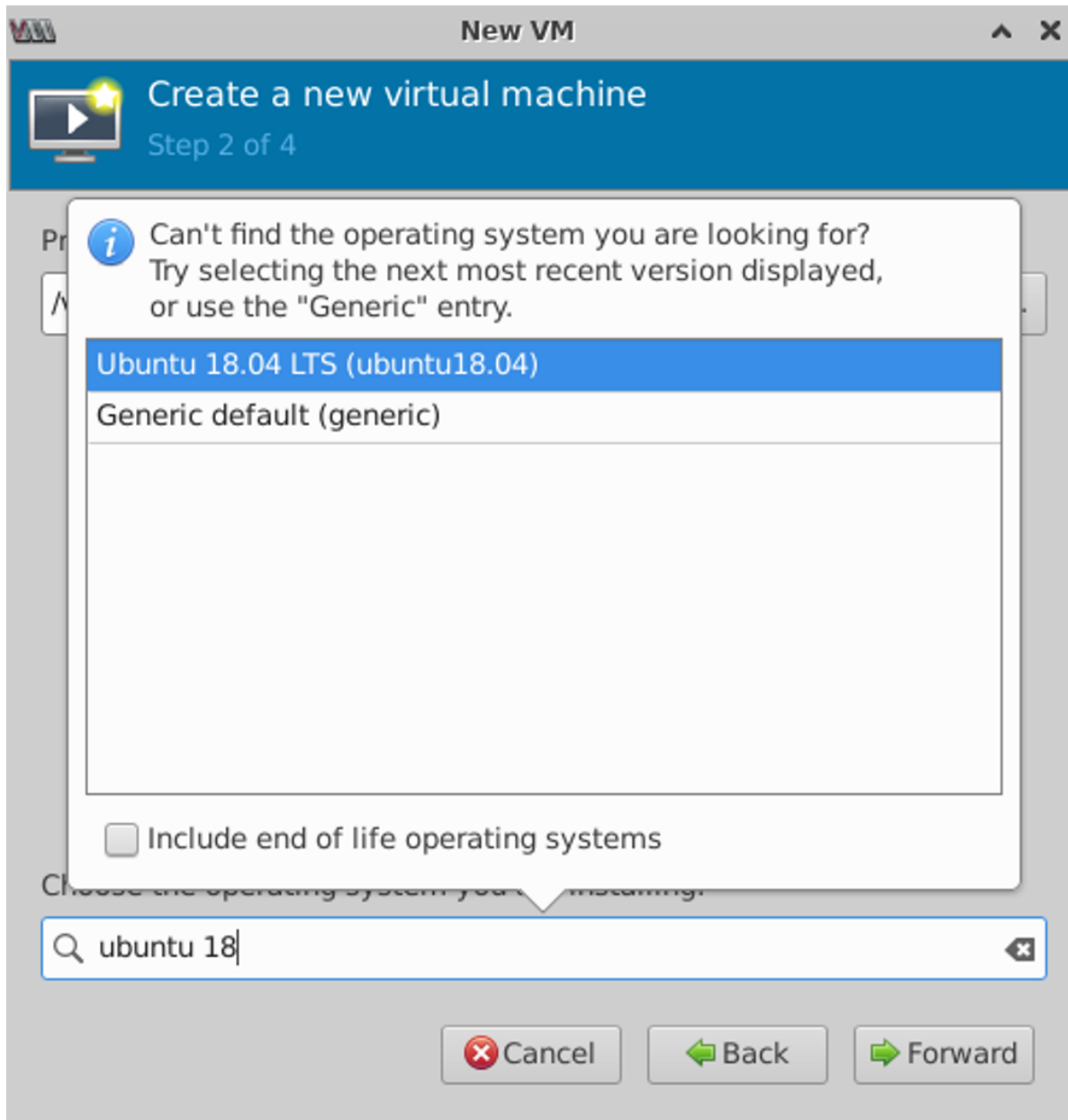


5. Select **Import existing disk image** and click **Forward**.

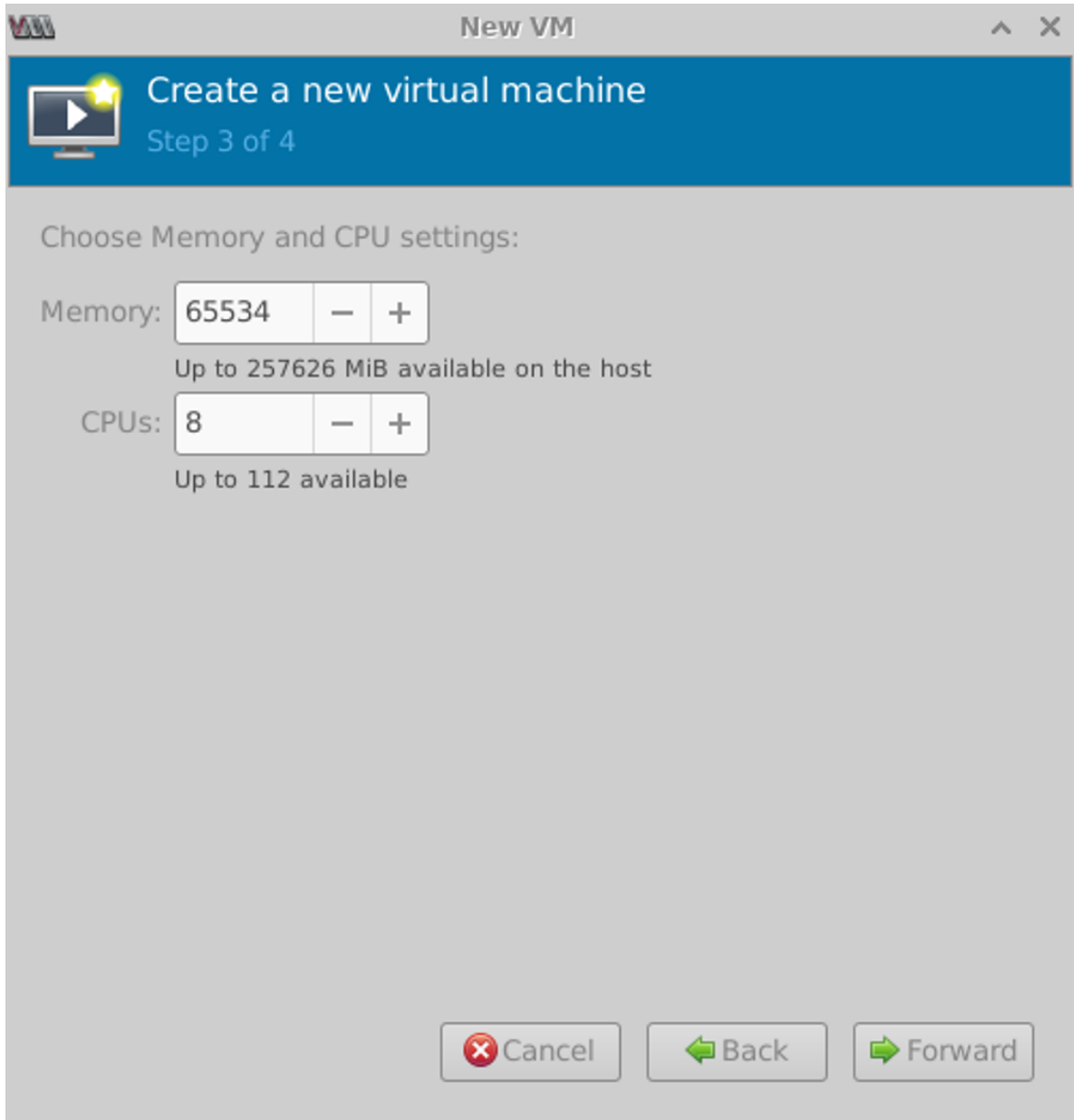


6. Browse to where you moved the QCOW2 image and click **Choose Volume**.

7. Select **Ubuntu 18.04 LTS** operating system and click **Forward**.



8. Specify memory and CPU requirements based on your environment. (See "[Required Server Resources](#)" on page 71.)



New VM

## Create a new virtual machine

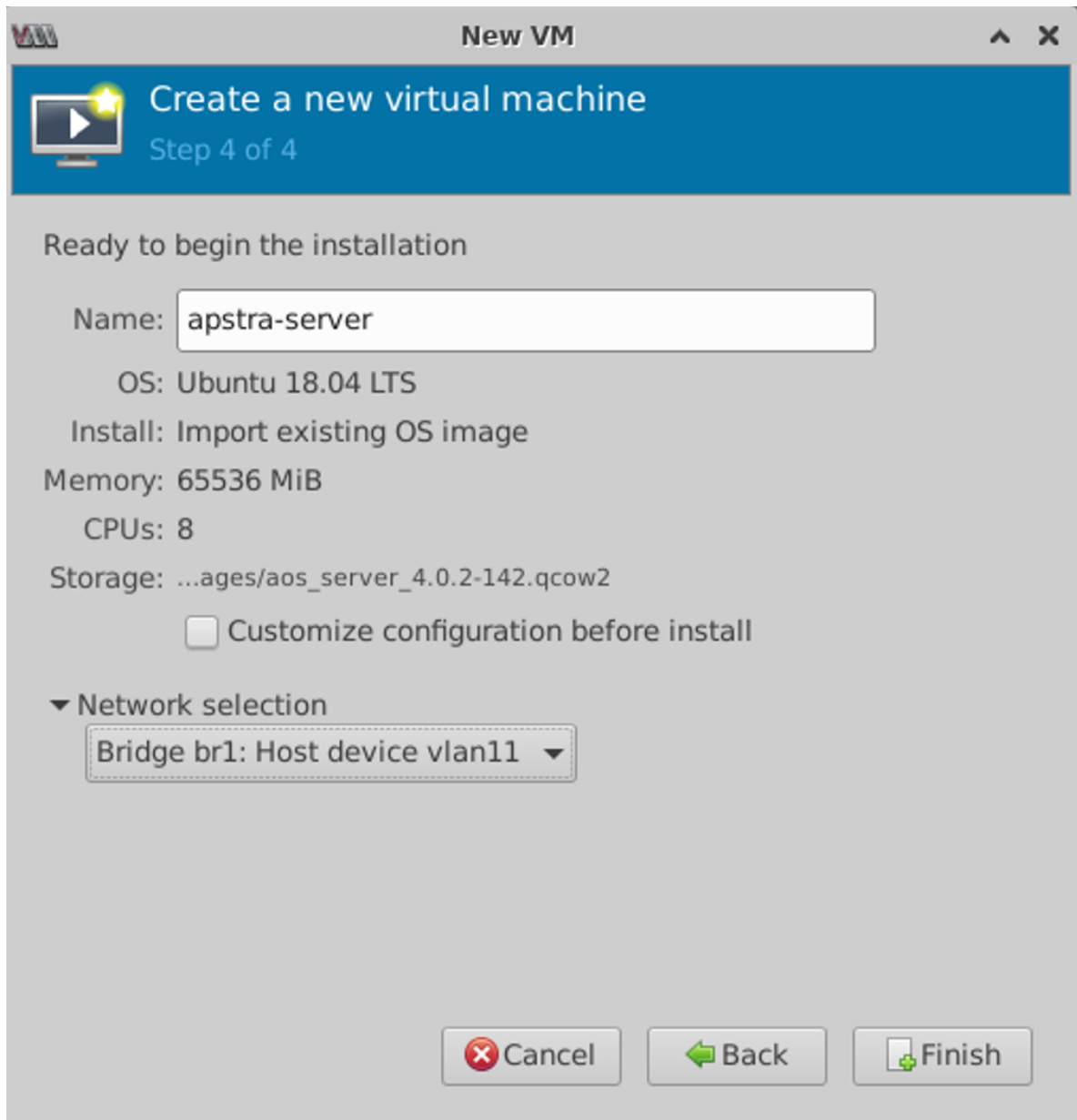
Step 3 of 4

Choose Memory and CPU settings:

Memory:  - +  
Up to 257626 MiB available on the host

CPUs:  - +  
Up to 112 available

9. Change the default name (optional), select the VM network that you want the VM to connect to, then click **Finish**. It may take a few minutes for the VM to be created.



Start the VM (if not already running) and ["configure" on page 21](#) the Apstra server.

#### Install on KVM with CLI

1. Confirm that you are running a ["supported KVM version" on page 71](#) and that the platform has enough ["resources" on page 71](#) for the Apstra server.
2. Ensure that the QEMU environment and bridge networking are installed and configured. For examples of installing and configuring QEMU, refer to the following documents:

- Ubuntu - <https://help.ubuntu.com/community/KVM/Installation>
  - RHEL - [https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/7/html/virtualization\\_deployment\\_and\\_administration\\_guide/index](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/virtualization_deployment_and_administration_guide/index)
3. You must use `e1000` or `virtio` Linux KVM network drivers. Confirm which network drivers you're using by running the command `ethtool -i eth0` from the Apstra server.

```
admin@aos-server:~$ ethtool -i eth0
driver: virtio_net
version: 1.0.0
firmware-version:
expansion-rom-version:
bus-info: 0000:00:03.0
supports-statistics: no
supports-test: no
supports-eeprom-access: no
supports-register-dump: no
supports-priv-flags: no
admin@aos-server:~$
```



**CAUTION:** Using other drivers such as `rtl8139` may result in high CPU utilization for the `ksoftirqd` process.

4. As a registered support user, [download the QCOW2 Apstra VM image from Juniper Support Downloads](#).
5. Uncompress (with **gunzip**) and move the disk image to where it will run.

```
ubuntu@ubuntu:~$ ls -l
total 1873748
-rw-r--r-- 1 ubuntu ubuntu 1918712115 Feb  4 22:28 aos_server_4.0.2-142.qcow2.gz
ubuntu@ubuntu:~$ gunzip aos_server_4.0.2-142.qcow2.gz
ubuntu@ubuntu:~$ ls -l
total 1905684
-rw-r--r-- 1 ubuntu ubuntu 1951413760 Feb  4 22:28 aos_server_4.0.2-142.qcow2.gz
ubuntu@ubuntu:~$
```

6. Create a VM with the `virt-install` command line tool. For example, to install the `aos_server_4.0.2-142.qcow2.gz` image using the existing bridge network (named `br0`), use the following command:

```
ubuntu@ubuntu:~$ sudo virt-install --name=aos-server --disk=aos_server_4.0.2-142.qcow2 --os-
type=linux --os-variant ubuntu18.04 --import --noautoconsole --vcpu=8 --ram=65536 --network
bridge=br0,model=virtio

Starting install...
Domain creation completed.
ubuntu@ubuntu:~$ sudo virsh list
  Id    Name                State
  ----  -
   4    aos-server          running

ubuntu@ubuntu:~$
```

7. Connect to the VM console.

```
ubuntu@ubuntu:~$ sudo virsh console aos-server
Connected to domain aos-server
Escape character is ^]

Apstra Operating System (AOS)

aos-server login:
```

Start the VM (if not already running) and ["configure" on page 21](#) the Apstra server..

## Install on Hyper-V

The instructions below are for *installing Apstra software* on a Microsoft Hyper-V hypervisor using Hyper-V Manager on a Windows Server 2016 Datacenter Edition. For information specific to using Hyper-V, refer to Microsoft's [Hyper-V documentation](#).

1. Confirm that you are running a ["supported Hyper-V version" on page 71](#) and that the platform has enough ["resources" on page 71](#) for the Apstra server.

2. Apstra software is delivered pre-installed on a single virtual machine (VM). As a registered support user, [download the VHDX Apstra VM image from Juniper Support Downloads](#).



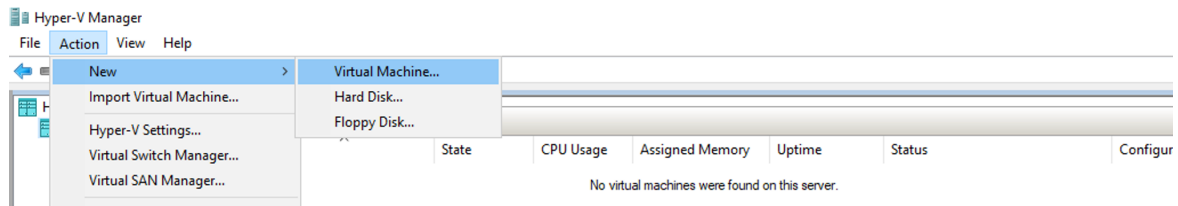
To download the image on your localhost, [CLICK HERE](#)

To download the image directly on your device, use the following URL:

[https://cdn.juniper.net/software/jafc/4.0.1/aos\\_server\\_4.0.1-1045.vhdx.gz?](https://cdn.juniper.net/software/jafc/4.0.1/aos_server_4.0.1-1045.vhdx.gz?)

copy

3. Uncompress and move the disk image to where it will run.
4. Start Hyper-V Manager, select the server for the VM and navigate to **Actions > New > Virtual Machine**. The **New Virtual Machine Wizard** opens.



5. Specify a VM name and location, then click **Next**.

New Virtual Machine Wizard ×

**Specify Name and Location**

Before You Begin

**Specify Name and Location**

Specify Generation

Assign Memory

Configure Networking

Connect Virtual Hard Disk

Installation Options

Summary

Choose a name and location for this virtual machine.


The name is displayed in Hyper-V Manager. We recommend that you use a name that helps you easily identify this virtual machine, such as the name of the guest operating system or workload.

Name:

You can create a folder or use an existing folder to store the virtual machine. If you don't select a folder, the virtual machine is stored in the default folder configured for this server.

☐ Store the virtual machine in a different location

Location:  Browse...

 If you plan to take checkpoints of this virtual machine, select a location that has enough free space. Checkpoints include virtual machine data and may require a large amount of space.

< Previous
Next >
Finish
Cancel

6. Specify **Generation 1** and click **Next**.

New Virtual Machine Wizard ×

**Specify Generation**

Before You Begin

Specify Name and Location

**Specify Generation**

Assign Memory

Configure Networking

Connect Virtual Hard Disk

Installation Options

Summary


Choose the generation of this virtual machine.

☒ **Generation 1**

This virtual machine generation supports 32-bit and 64-bit guest operating systems and provides virtual hardware which has been available in all previous versions of Hyper-V.

☐ **Generation 2**

This virtual machine generation provides support for newer virtualization features, has UEFI-based firmware, and requires a supported 64-bit guest operating system.

 Once a virtual machine has been created, you cannot change its generation.

- Specify memory based on your environment (see ["Required Server Resources" on page 71](#)) and click **Next**.

New Virtual Machine Wizard

**Assign Memory**

Before You Begin  
Specify Name and Location  
Specify Generation  
**Assign Memory**  
Configure Networking  
Connect Virtual Hard Disk  
Installation Options

Specify the amount of memory to allocate to this virtual machine. You can specify an amount from 32 MB through 12582912 MB. To improve performance, specify more than the minimum amount recommended for the operating system.

Startup memory:  MB

☐ Use Dynamic Memory for this virtual machine.

**i** When you decide how much memory to assign to a virtual machine, consider how you intend to use the virtual machine and the operating system that it will run.

- Configure the virtual switch as required for your deployment environment, then click **Next**.

New Virtual Machine Wizard

**Configure Networking**

Before You Begin  
Specify Name and Location  
Specify Generation  
Assign Memory  
**Configure Networking**

Each new virtual machine includes a network adapter. You can configure the network adapter to use a virtual switch, or it can remain disconnected.

Connection:

- Select **Use an existing virtual hard disk** and browse to the extracted file, then click **Finish**.

New Virtual Machine Wizard

**Connect Virtual Hard Disk**

Before You Begin  
Specify Name and Location  
Specify Generation  
Assign Memory  
Configure Networking  
**Connect Virtual Hard Disk**  
Summary

A virtual machine requires storage so that you can install an operating system. You can specify the storage now or configure it later by modifying the virtual machine's properties.

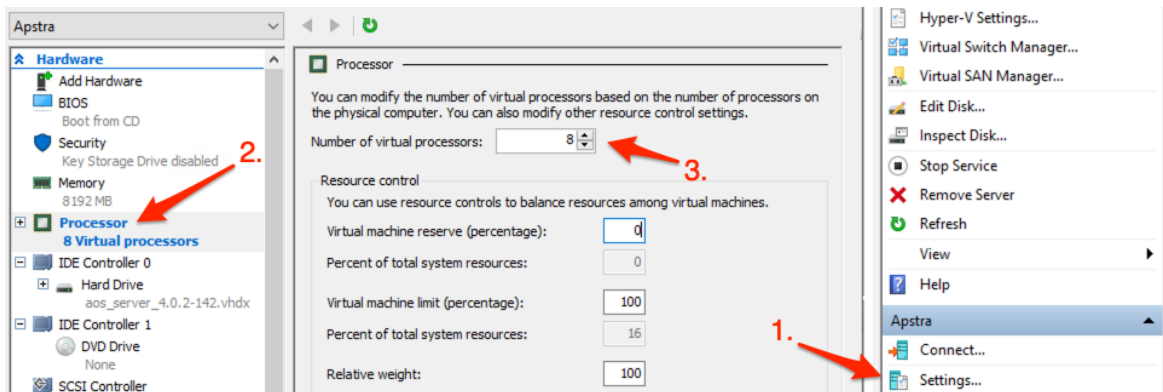
☐ Create a virtual hard disk  
Use this option to create a VHDX dynamically expanding virtual hard disk.

Name:   
Location:    
Size:  GB (Maximum: 64 TB)

☒ Use an existing virtual hard disk  
Use this option to attach an existing virtual hard disk, either VHD or VHDX format.

Location:

- Click **Settings** (right panel), click **Processor** (left panel), then specify the number of virtual processors based on your environment (see "[Required Server Resources](#)" on page 71) and click **OK**.



Start the VM (if not already running) and "[configure](#)" on page 21 the Apstra server. (When the Apstra server is configured, the Docker daemon runs properly.)

## Install on VirtualBox

VirtualBox is for demonstration and lab purposes only. Production environments require a proper enterprise-scale virtualization solution (See "[supported platforms](#)" on page 71). The instructions below are for *installing Apstra software* on a VirtualBox hypervisor. For information specific to using VirtualBox, refer to Oracle's [VirtualBox documentation](#) or the open-source community.

- Apstra software is delivered pre-installed on a single virtual machine (VM). As a registered support user, [download the OVA Apstra VM image from Juniper Support Downloads](#) to your local workstation.



To download the image on your localhost, [CLICK HERE](#)

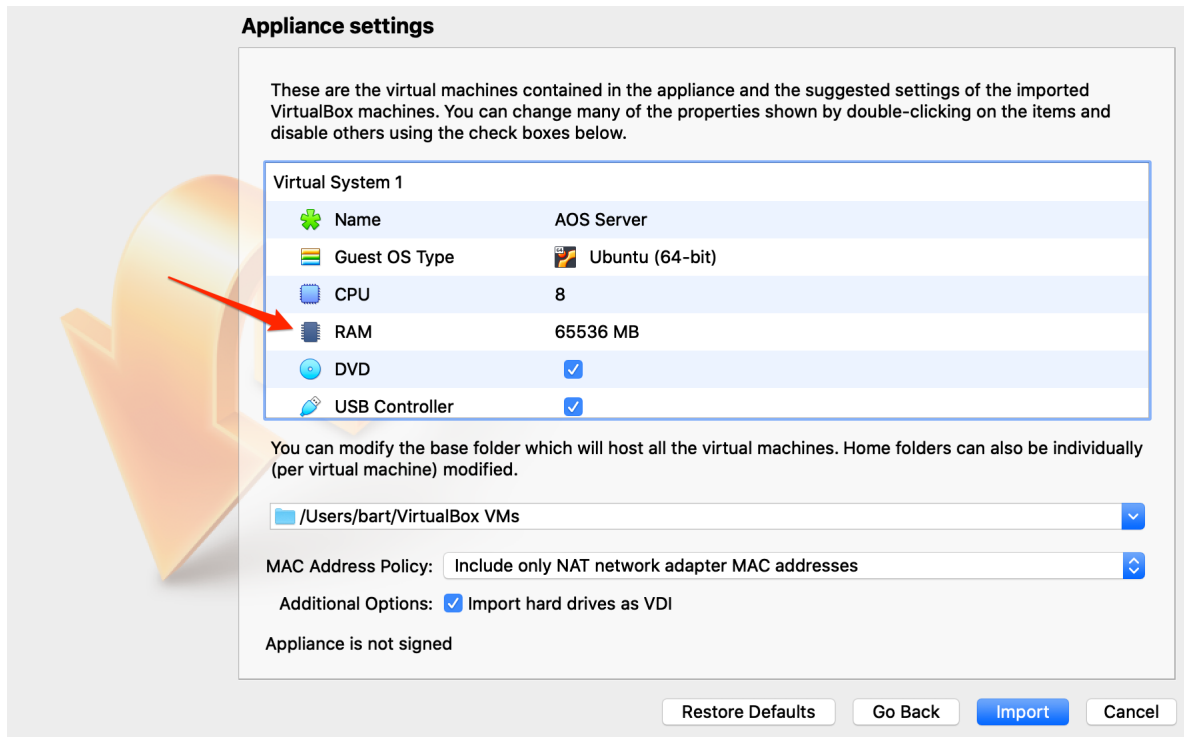
To download the image directly on your device, use the following URL:

[https://cdn.juniper.net/software/jafc/4.0.2/aos\\_server\\_4.0.2-142.ova?](https://cdn.juniper.net/software/jafc/4.0.2/aos_server_4.0.2-142.ova?)

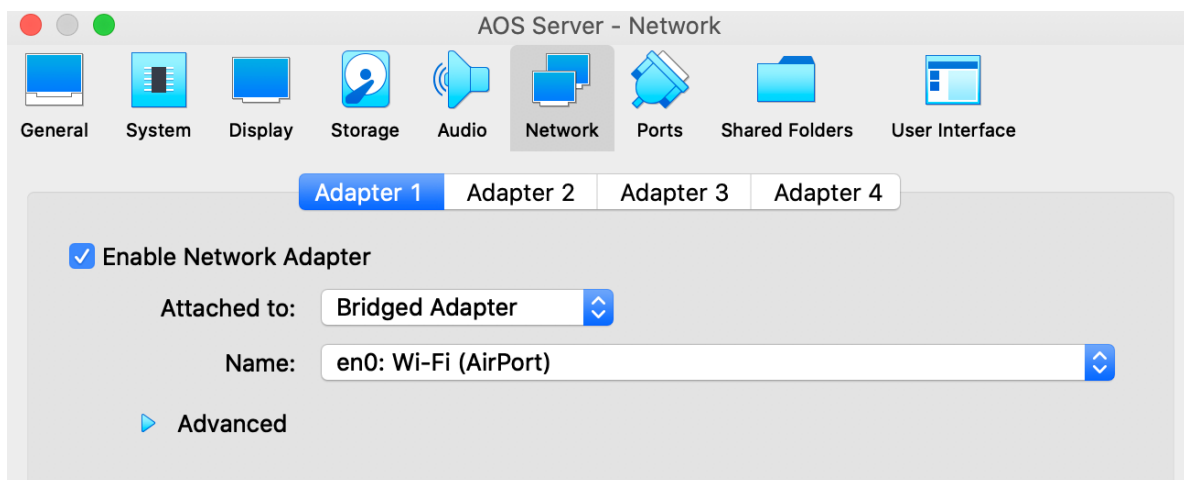
copy

- Start VirtualBox, select **File > Import Appliance**, navigate to the OVA file, select it and click **Continue**.

3. Change **RAM** to 8 GB. 8 GB is sufficient for lab and testing purposes.



4. Click **Import** to start the import process.
5. When the import is complete, start the server VM, click **Settings** and confirm that the VM meets requirements. In particular, check network settings for the adapter that's attached to your management network. If this value is not set correctly, the Apstra server does not receive an IP address. Since VirtualBox has one network adapter attached to the bridged adapter using active networking, full connectivity from your workstation to the VM (HTTP, SSH) is expected by default.



6. "Configure" on page 21 the Apstra server.
7. Confirm connectivity. (Run `ifconfig -a` on the VM to get the IP address.)
  - SSH from your workstation to the VM's active network adapter IP address.

- Point a web browser to the VM's active network adapter IP address.

## Initial Configuration

The first time you boot the Apstra server VM, a configuration tool opens to assist you. (You can open this tool at any time by running the command `aos_config`.)

1. The default credentials for the Apstra console are user=admin and password=admin. SSH into the Apstra server (`ssh admin@apstra-server-ip` where `apstra-server-ip` is the IP address of the Apstra server.) If you haven't changed the default password for user **admin** you are prompted to change it.

It looks like you haven't changed the default password with a strong one. Do you want to do it now?

IMPORTANT: if you select No, you are exposing AOS and your network to potential risks: Apstra highly recommends to change the default password as soon as possible.

<Yes>

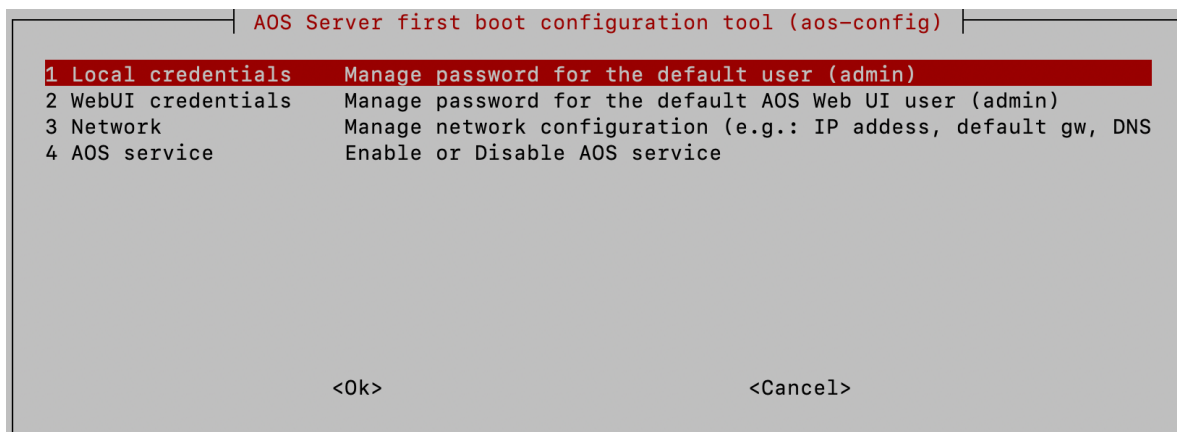
<No>



**CAUTION:** We highly recommend that you change default passwords. User **admin** has full root access. Juniper is not responsible for security-related incidents due to failing to change default passwords.

2. Select **Yes**, and follow the prompts to enter a strong password that doesn't contain the current username in any form and that has a minimum of fourteen characters, one uppercase character, and one digit.
3. After you've changed the password you are prompted to start Apstra service. Select **Yes**. When service is up and running click **OK**. The main menu appears.

4. You updated the default local credentials in the previous step. To change the password again at any time, select **Local credentials** and follow the prompts.



5. Select **WebUI credentials** and change the default password for the GUI user **admin**. (Service must be up and running to change the GUI password. If service is stopped, proceed to step 7 and start service.)
6. The network is configured to use DHCP by default. To assign static IP addresses instead, select **Network**, change it to **Manual**, and provide the following:
- (Static Management) IP address in CIDR format (for example, 192.168.0.10/24)
  - Gateway IP address
  - Primary DNS
  - Secondary DNS (optional)
  - Domain
7. Apstra service is stopped by default. To start and stop Apstra service, select **AOS service** and select **Start** or **Stop**, as appropriate. Starting service from this configuration tool invokes `/etc/init.d/aos`, which is the equivalent of running the command `service aos start`.
8. To exit the configuration tool and return to the CLI, select **Cancel** from the main menu.

Access the ["Apstra server GUI" on page 26](#) and replace the default SSL certificate with a signed one.



**CAUTION:** We recommend that you ["back up" on page 40](#) the Apstra server on a regular basis (since HA is not available). For information about setting up automated backup collection see the [Juniper Support Knowledge Base article KB37808](#).

## Configure Static Management IP Address (Apstra Server)

If you're not using DHCP, you can use the ["configuration tool" on page 21](#) to enter a static management IP address, or you can configure it as follows:

1. SSH into the Apstra server as user **admin**. (ssh admin@<apstra-server-ip> where <apstra-server-ip> is the IP address of the Astra server.)
2. Edit the /etc/netplan/01-netcfg.yaml file (per standard Ubuntu 18.04 practice) to configure the static management IP address. See example below. (For more information about using netplan, see <https://netplan.io/examples>.)

```
admin@aos-server:~$ sudo vi /etc/netplan/01-netcfg.yaml
[sudo] password for admin:

# This file describes the network interfaces available on your system
# For more information, see netplan(5).
network:
  version: 2
  renderer: networkd
  ethernets:
    eth0:
      dhcp4: no
      addresses: [192.168.59.250/24]
      gateway4: 192.168.59.1
      nameservers:
        search: [example.com, example.net]
        addresses: [69.16.169.11, 69.16.170.11]
```

3. Apply the change with one of the following methods:
  - Method 1 - Reboot the Apstra server with the command `sudo reboot`.
  - Method 2 - Run the command `sudo netplan apply`, then SSH to the new IP address and run the command `sudo service aos restart`.
4. If you already have on-box agents, you must reconfigure each device agent (/mnt/flash/aos-config, /etc/aos/aos.conf) to point to the new Apstra server IP address.

## Change Hostname (Apstra Server)



**CAUTION:** To avoid issues with the Apstra container's binding, don't change the /etc/hostname file directly with any Linux CLI command or other command than the one below.

1. SSH into the Apstra server as user **admin**. (ssh admin@<apstra-server-ip> where <apstra-server-ip> is the IP address of the Astra server.)

2. With root privileges, run the command `/#aos_hostname <hostname>` where `<hostname>` is the new hostname of the Apstra server. This command modifies the hostname in the `/etc/hostname` file and performs necessary backend configuration.
3. For the change to take effect, reboot the Apstra server, preferably during a maintenance window. The Apstra server is temporarily unavailable during a reboot, though it most likely won't be service-impacting.

## Configure Docker Subnets

### IN THIS SECTION

- [docker0 | 24](#)
- [Apstra In-Place Upgrade Docker Network | 25](#)

The Apstra server Docker containers require one network for internal connectivity, which is automatically configured with the following subnets:

- `docker0`: inet 172.17.0.1/16
- Apstra in-place upgrade docker network: inet 172.18.0.1/16

If these subnets must be used elsewhere, you can avoid conflicts by changing the Docker network as follows:

### `docker0`

Update `bip` with the new subnet. If the `/etc/docker/daemon.json` file doesn't already exist, create one with the following format (Replace 172.26.0.1/16 in the example below with your own subnet.):

```
$ sudo vi /etc/docker/daemon.json

{
  "bip": "172.26.0.1/16"
}

$ sudo service docker restart
$ sudo service aos restart
```

## Apstra In-Place Upgrade Docker Network

If you're upgrading your Apstra server ["in-place" on page 61](#), the Apstra upgrade creates an additional Docker network. By default in Docker, this network is 172.18.0.1/16. If you are using this network elsewhere on your network, the conflict may cause the Apstra upgrade to fail.

To use a different subnet, create or edit the `/etc/docker/daemon.json` file with the following format (Replace 172.27.0.0/16 in the example with your own subnet).

```
$ sudo vi /etc/docker/daemon.json

{
  "default-address-pools":
  [
    {
      "base": "172.27.0.0/16",
      "size": 24
    }
  ]
}

$ sudo service docker restart
$ sudo service aos restart
```

## Configure New SSH Keys

You can replace SSH host keys on new or existing Apstra server VMs.

1. SSH into the Apstra server as user **admin**. (`ssh admin@<apstra-server-ip>` where `<apstra-server-ip>` is the IP address of the Astra server.)
2. Run the command `sudo rm /etc/ssh/ssh_host*` to remove SSH host keys.
3. Run the command `sudo dpkg-reconfigure openssh-server` to configure new SSH host keys.

```
admin@aos-server:~$ sudo dpkg-reconfigure openssh-server
Creating SSH2 RSA key; this may take some time ...
2048 SHA256:EWRFcS4V6Bm0ILR3T2Psnxng1uE0qXQ/z9IKkXrnLpJs root@aos-server (RSA)
Creating SSH2 ECDSA key; this may take some time ...
256 SHA256:THaXEia8VW6Jfw60BXFegu1Cav0zcGSV0y9RkN0Pxf4 root@aos-server (ECDSA)
Creating SSH2 ED25519 key; this may take some time ...
256 SHA256:0H0n0nnF+7oRaF5HggI4vWeyxT+UNsHcbvNpBJdaKhQ root@aos-server (ED25519)
admin@aos-server:~$ sudo dpkg-reconfigure openssh-server
```

4. To restart the SSH server process, run the command `sudo systemctl restart ssh`.

## Apstra GUI Overview

### IN THIS SECTION

- [Replace SSL Certificate with Signed One | 27](#)
- [Replace SSL Certificate with Self-Signed One | 29](#)
- [Check GUI Version | 30](#)
- [Update GUI Version | 30](#)
- [Restore GUI Version | 31](#)
- [Reset GUI Admin Password | 31](#)

After you've ["installed and configured" on page 4](#) the Apstra server, you can design, build, deploy, operate and validate your network from the Apstra server GUI.

1. From the latest web browser version of Google Chrome or Mozilla FireFox, enter the URL `https://<apstra_server_ip>` where `<apstra_server_ip>` is the IP address of the Apstra server (or a DNS name that resolves to the IP address of the Apstra server).
2. If a security warning appears, click **Advanced** and **Proceed to ...** the site. The warning occurs because the SSL certificate that was generated during installation is self-signed.



**CAUTION:** For security, replace the default self-signed ["SSL certificate" on page 27](#) with one from your own certificate authority. Web server certificate management is the responsibility of the end user. Juniper support is best effort only.

3. From the login page, enter username **admin** and password **admin** to go to the main screen. (Entering the password incorrectly too many times locks you out for a few minutes depending on how password requirements have been configured.)



**CAUTION:** For security, ["change the default password" on page 706](#). We recommend changing both the GUI password and OS password. You can use the ["configuration tool" on page 21](#) to change local and webUI credentials from the CLI.

## Replace SSL Certificate with Signed One

When you boot up the Apstra server for the first time, a unique self-signed certificate is automatically generated and stored on the Apstra server at `/etc/aos/nginx.conf.d` (`nginx.crt` is the public key for the webserver and `nginx.key` is the private key.) The certificate is used for encrypting the Apstra server and REST API, not for any internal device-server connectivity. When you perform system backups you must manually back up the `etc/aos` folder, since the HTTPS certificate is not retained. We recommend replacing the default SSL certificate. Web server certificate management is the responsibility of the end user. Juniper support is best effort only.

1. Back up the existing OpenSSL keys.

```
admin@aos-server:/$ sudo -s
[sudo] password for admin:

root@aos-server:/# cd /etc/aos/nginx.conf.d
root@aos-server:/etc/aos/nginx.conf.d# cp nginx.crt nginx.crt.old
root@aos-server:/etc/aos/nginx.conf.d# cp nginx.key nginx.key.old
```

2. Create a new OpenSSL private key with the built-in openssl command.

```
root@aos-server:/etc/aos/nginx.conf.d# openssl genrsa -out nginx.key 2048
Generating RSA private key, 2048 bit long modulus
.....+++
.....+++
e is 65537 (0x10001)
```



**CAUTION:** Do not attempt to modify the default `nginx.crt` or `nginx.key` filenames. These values are referenced from nginx's configuration file. These files could be replaced as part of a subsequent service upgrade, so the filenames must be predictable. Moreover, do not make configuration changes to `nginx.conf`, as this file may be replaced during Apstra server upgrade.

3. Create a certificate signing request. If you want to create a signed SSL certificate with a Subjective Alternative Name (SAN) for your Apstra server HTTPS service, you must manually create an OpenSSL template. For details, see [Juniper Support Knowledge Base article KB37299](#).



**CAUTION:** If you have created custom OpenSSL configuration files for advanced certificate requests, do not leave them in the nginx configuration folder, as nginx will attempt to load them (\*.conf) on service startup, causing a service failure.

```
root@aos-server:/etc/aos/nginx.conf.d# openssl req -new -sha256 -key nginx.key -out nginx.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:California
Locality Name (eg, city) []:Menlo Park
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Apstra, Inc
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:aos-server.apstra.com
Email Address []:support@apstra.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

4. Submit your Certificate Signing Request (nginx.csr) to your Certificate Authority. The required steps are outside the scope of this document - CA instructions differ per implementation. Any valid SSL certificate will work. The example below is of self-signing the certificate.

```
root@aos-server:/etc/aos/nginx.conf.d# openssl req -x509 -sha256 -days 3650 -key nginx.key -
in nginx.csr -out nginx.crt
root@aos-server:/etc/aos/nginx.conf.d#
```

5. Verify that the SSL certificates match: private key, public key, and CSR.

```
root@aos-server:/etc/aos/nginx.conf.d# openssl rsa -noout -modulus -in nginx.key | openssl md5
(stdin)= 60ac4532a708c98d70fee0dbcaab1e75

root@aos-server:/etc/aos/nginx.conf.d# openssl req -noout -modulus -in nginx.csr | openssl md5
```

```
(stdin)= 60ac4532a708c98d70fee0dbcaab1e75

root@aos-server:/etc/aos/nginx.conf.d# openssl x509 -noout -modulus -in nginx.crt | openssl
md5
(stdin)= 60ac4532a708c98d70fee0dbcaab1e75
```

6. To load the new certificate, restart the nginx container.

```
root@aos-server:/etc/aos/nginx.conf.d# docker restart aos_nginx_1
aos_nginx_1
root@aos-server:/etc/aos/nginx.conf.d
```

7. Confirm that the new certificate is in your web browser and that the new certificate common name matches 'aos-server.apstra.com'.

## Replace SSL Certificate with Self-Signed One

When you boot up the Apstra server for the first time, a unique self-signed certificate is automatically generated and stored on the Apstra server at `/etc/aos/nginx.conf.d` (`nginx.crt` is the public key for the webserver and `nginx.key` is the private key.) The certificate is used for encrypting the Apstra server and REST API, not for any internal device-server connectivity. When you perform system backups you must manually back up the `etc/aos` folder, since the HTTPS certificate is not retained. We support and recommend replacing the default SSL certificate.

1. Back up the existing OpenSSL keys.

```
admin@aos-server:/$ sudo -s
[sudo] password for admin:

root@aos-server:/# cd /etc/aos/nginx.conf.d
root@aos-server:/etc/aos/nginx.conf.d# cp nginx.crt nginx.crt.old
root@aos-server:/etc/aos/nginx.conf.d# cp nginx.key nginx.key.old
```

2. If a Random Number Generator seed file `.rnd` doesn't exist in `/home/admin`, create one.

```
root@aos-server:~# touch /home/admin/.rnd
root@aos-server:~#
```

### 3. Generate a new OpenSSL private key and self-signed certificate.

```

root@aos-server:/etc/aos/nginx.conf.d# openssl req -newkey rsa:2048 -nodes -keyout nginx.key -
x509 -days 824 -out nginx.crt -addext extendedKeyUsage=serverAuth -addext
subjectAltName=DNS:apstra.com
Generating a RSA private key
.....+++++
.....+++++
writing new private key to 'nginx.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:California
Locality Name (eg, city) []:Menlo Park
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Apstra, Inc
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:aos-server.apstra.com
Email Address []:support@apstra.com
root@aos-server:/etc/aos/nginx.conf.d#

```

### 4. To load the new certificate, restart the nginx container.

```

root@aos-server:/etc/aos/nginx.conf.d# docker restart aos_nginx_1
aos_nginx_1
root@aos-server:/etc/aos/nginx.conf.d#

```

## Check GUI Version

From the Apstra GUI, from the left navigation menu, navigate to **Platform > About** to see the **Juniper Apstra UI version**.

## Update GUI Version

When it's available, you can install an optional Apstra server UI update to add web interface functionality. This is independent of the Apstra server backend and does not affect the state of the Apstra server or the established configuration.

1. Download the **web UI** run file from [Juniper Support Downloads](#).
2. Upload the file to the Apstra server. For this example, the file is named **aos-web-ui\_2.2.0-67.run**.
3. From the Apstra server CLI, as root user, run the file as shown in the example below.

```
admin@aos-server:~$ sudo -s
[sudo] password for admin:
root@aos-server:~# bash aos-web-ui_2.2.0-67.run
Verifying archive integrity... All good.
Uncompressing AOS WebUI installer 100%
### Backing up existing AOS WebUI into /opt/aos/frontend/snapshot/2018-02-25_20-34-15 ...
### Copying AOS WebUI file into aos_controller_1 ...
### Initializing new AOS WebUI ...
### Done!
root@aos-server:~#
```

The current UI version is copied to the `/opt/aos/frontend/snapshot/` directory.

4. From the Apstra GUI, from the left navigation menu, navigate to **Platform > About** to confirm that the **Juniper Apstra UI version** has been updated.

## Restore GUI Version

You can restore a previous GUI version at any time without affecting the state of the Apstra server.

1. From the Apstra server CLI, navigate to the snapshot directory and run the command `webui_restore` as shown in the example below.

```
root@aos-server:~# cd /opt/aos/frontend/snapshot/2018-02-25_20-34-15
root@aos-server:/opt/aos/frontend/snapshot/2018-02-25_20-34-15# ls
aos-web-ui.zip  webui_restore
root@aos-server:/opt/aos/frontend/snapshot/2018-02-25_20-34-15# ./webui_restore
### Copying AOS WebUI file into aos_controller_1...
### Initializing AOS WebUI...
### Done!
root@aos-server:/opt/aos/frontend/snapshot/2018-02-25_20-34-15#
```

2. From the Apstra GUI, from the left navigation menu, navigate to **Platform > About** to confirm that the **Juniper Apstra UI version** has been restored.

## Reset GUI Admin Password

You can reset a lost admin GUI password.

1. SSH into the Apstra server CLI as the default admin user.

2. Run the command `aos_reset_admin_password` as shown in the example below.

```
admin@aos-server:~$ aos_reset_admin_password
Resetting UI "admin" user password to default "admin"
Successfully reset admin's password
admin@aos-server:~$
```

You can now access the GUI with the default password `admin`.



**CAUTION:** For security, ["change the admin password" on page 706](#) after resetting it to the default.

## Apstra Server Management

### IN THIS SECTION

- [Monitor Apstra Server via CLI | 32](#)
- [Restart Apstra Server | 33](#)
- [Reset Apstra Server VM Password | 33](#)
- [Reinstall Apstra Server | 38](#)
- [Apstra Server Database Overview | 39](#)
- [Back up Database | 40](#)
- [Restore Database | 41](#)
- [Reset Database | 45](#)
- [Migrate Database | 46](#)

### Monitor Apstra Server via CLI

1. To check general status from the Apstra server CLI, run the command `sudo service aos status`.

```
admin@aos-server:~$ sudo service aos status
* aos.service - LSB: Start AOS management system
```

```

Loaded: loaded (/etc/init.d/aos; generated)
Active: active (exited) since Tue 2020-07-28 00:35:38 UTC; 2h 13min ago
   Docs: man:systemd-sysv-generator(8)
  Tasks: 0 (limit: 4915)
 CGroup: /aos.service

Jul 28 00:35:35 aos-server systemd[1]: Starting LSB: Start AOS management system...
Jul 28 00:35:36 aos-server aos[1040]: net.core.wmem_max = 33554432
Jul 28 00:35:37 aos-server aos[1040]: Creating aos_sysdb_1 ...
Jul 28 00:35:37 aos-server aos[1040]: Creating aos_nginx_1 ...
Jul 28 00:35:37 aos-server aos[1040]: Creating aos_auth_1 ...
Jul 28 00:35:37 aos-server aos[1040]: Creating aos_controller_1 ...
Jul 28 00:35:37 aos-server aos[1040]: Creating aos_metadb_1 ...
Jul 28 00:35:38 aos-server aos[1040]: [240B blob data]
Jul 28 00:35:38 aos-server systemd[1]: Started LSB: Start AOS management system.
admin@aos-server:~$

```

2. To troubleshoot, run the `aos_controller_health_check` script. It searches for known error signatures in the Apstra server logs (such as agent crashes) and returns the output. If no errors are found, no output is returned. See below for sample command.

```

admin@aos-server:~$ docker exec aos_controller_1 aos_controller_health_check
admin@aos-server:~$

```

## Restart Apstra Server

To restart the Apstra server you can reboot the VM or run the following commands.

1. Run the command `sudo service aos stop`.

When the Apstra server is down, device agents may temporarily log "liveness" telemetry alarms.

2. Run the command `sudo service aos start`.

```

admin@aos-server:~$ sudo service aos stop
admin@aos-server:~$ sudo service aos start
admin@aos-server:~$

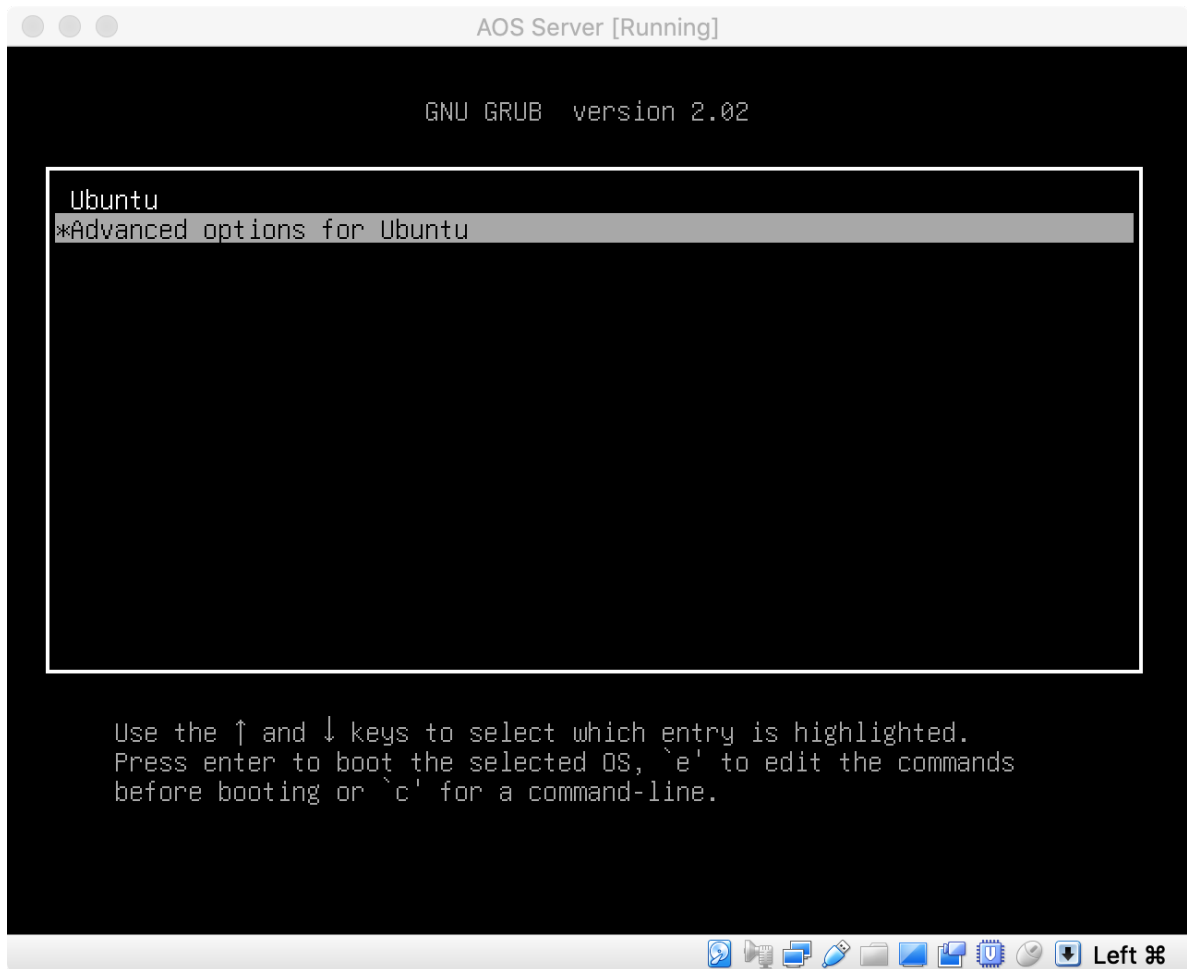
```

After services are restored (in a minute or two) the "liveness" telemetry alarm resets.

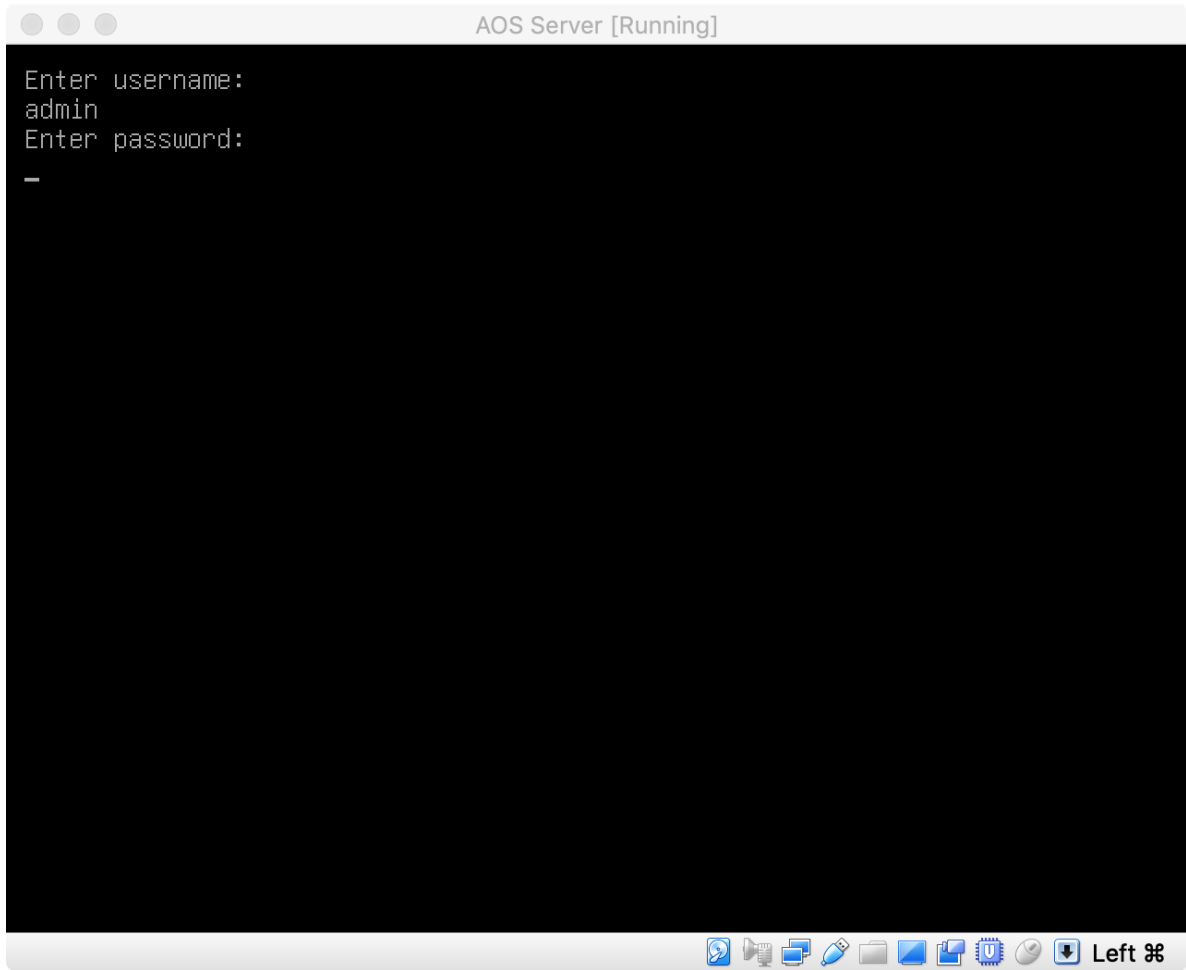
## Reset Apstra Server VM Password

If you lose your **admin** password for the Apstra server VM, and *you still have console access to the Apstra server VM*, you can reset your password.

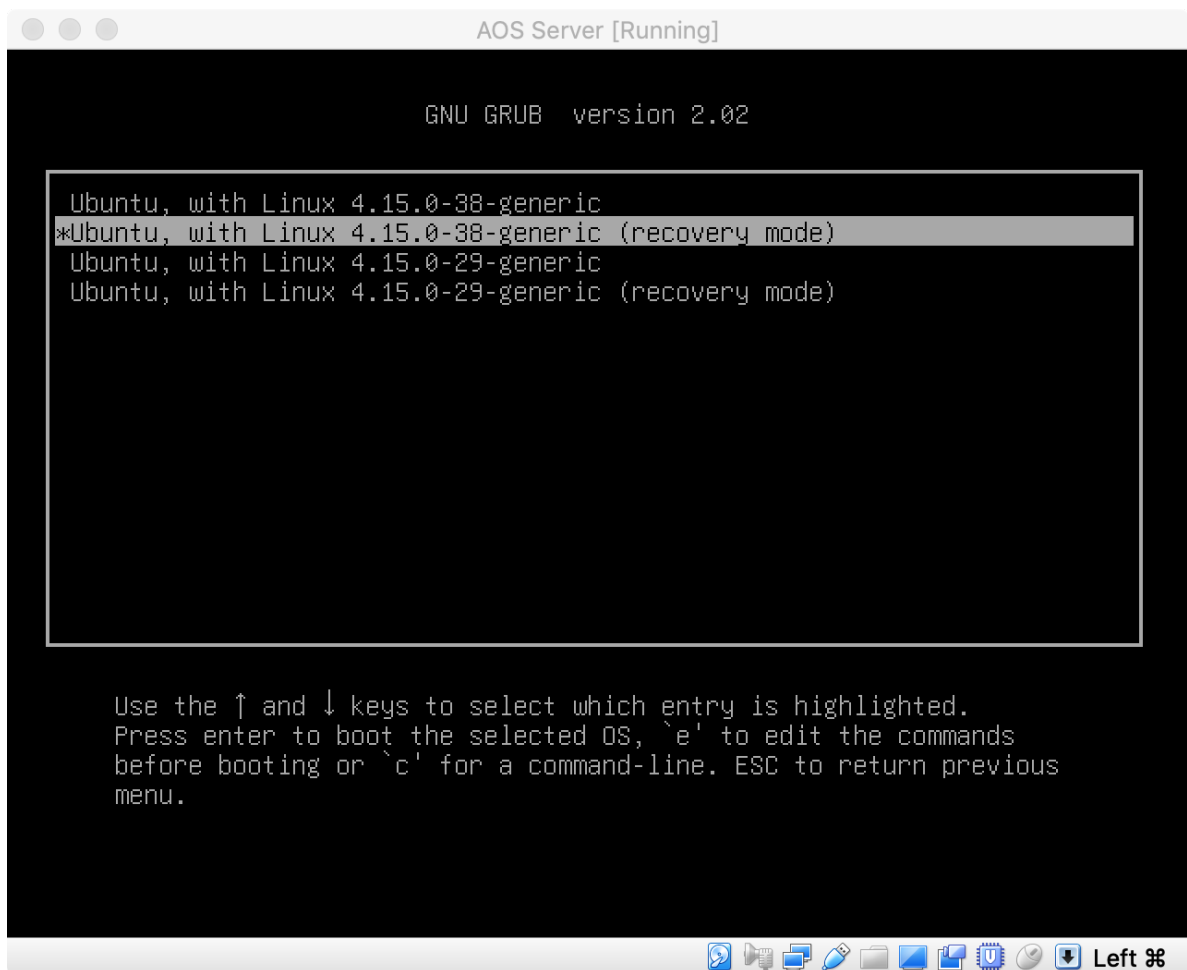
1. Attach to the Apstra server console and send a "reset" signal to the VM. To access the GRUB menu, immediately press the **esc** or **shift** key in the console on reboot.
2. Select **Advanced options for Ubuntu**.



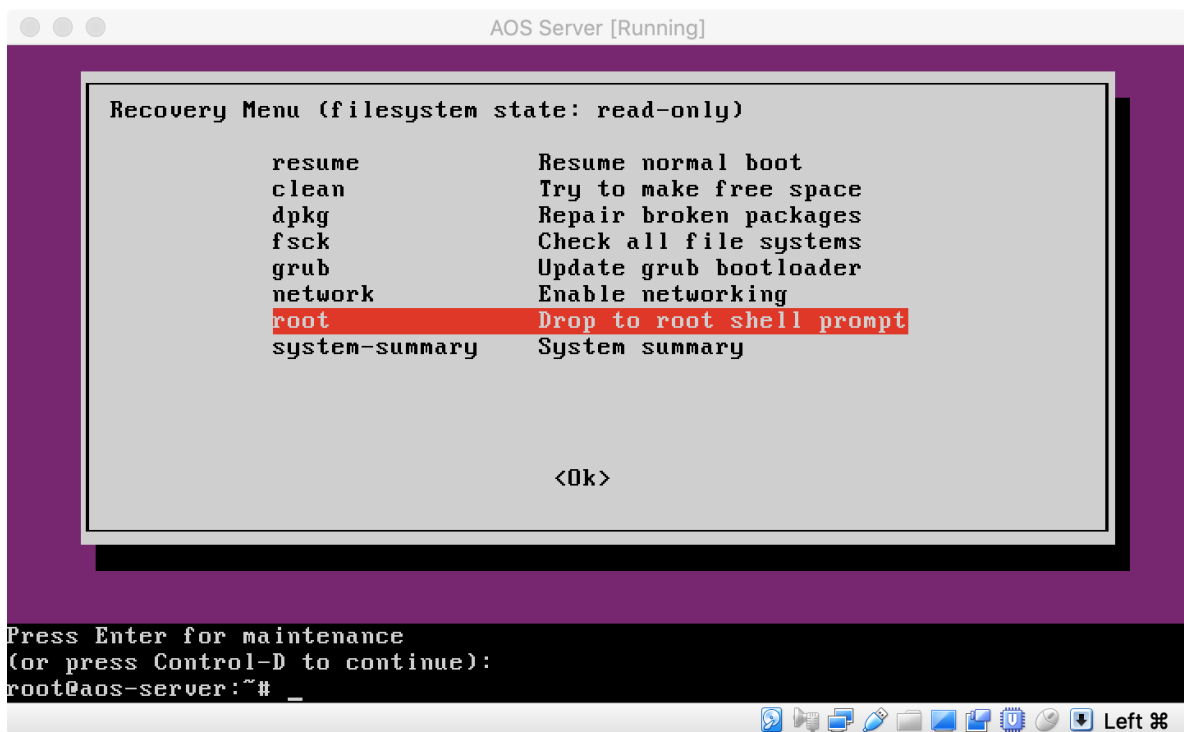
3. Enter username **admin** and password **apstra**.



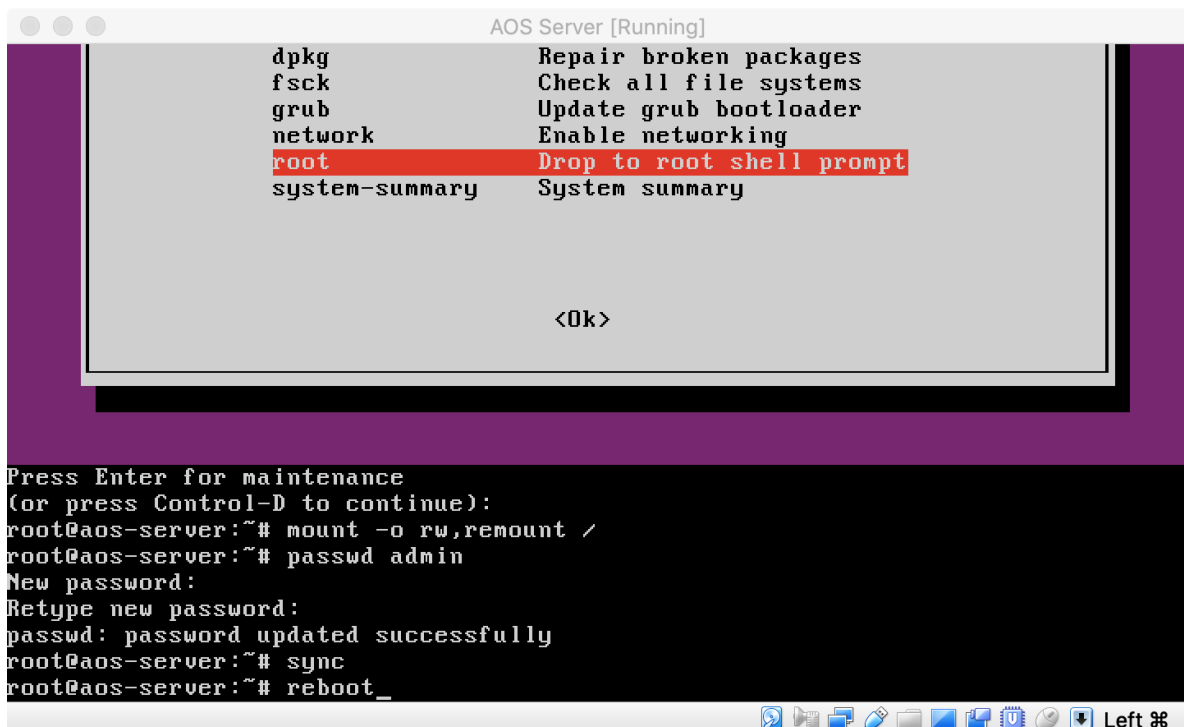
4. At the next GRUB menu, select the first **(recovery mode)** option.



- From the **Recovery Menu**, select **root**, then press **Enter** to enter a root shell prompt.



- At the root shell prompt run the command `mount -o rw,remount /`.
- Run the command `passwd admin` to reset the default CLI password for **admin**.
- Run the command `sync`.
- Run the command `reboot` to reboot the Apstra server VM. (Your deployed fabric is not affected.)



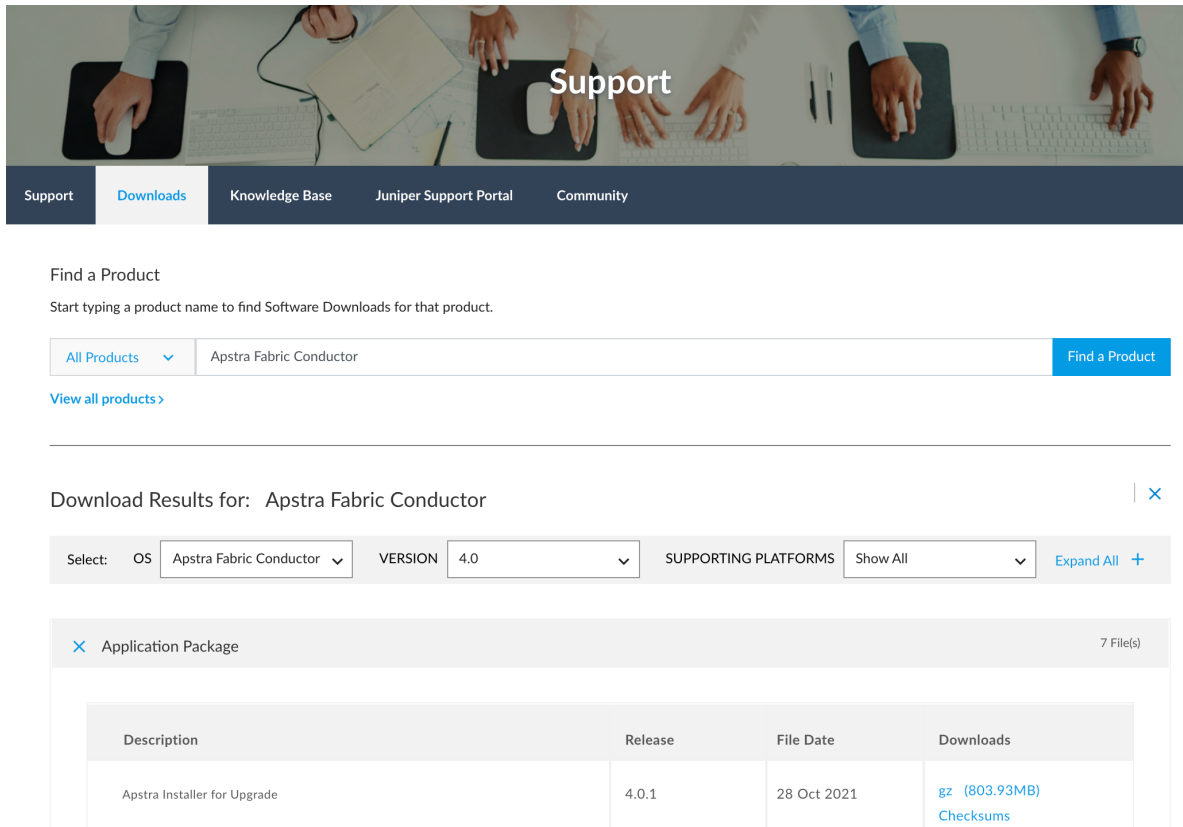
After reboot, you can log into the Apstra server VM Linux CLI as user **admin** with the new password.

## Reinstall Apstra Server



**CAUTION:** Reinstalling the Apstra server removes ALL Apstra data from the Apstra server VM and reinstalls a fresh version. Use with care. This is mostly helpful for *proof of concepts* or demo installs. If you have problems that require you to reinstall the software, contact "[Juniper Support](#)" on page 777.

1. Download the "Installer" .run file from [Juniper Support Downloads](#).



Support

Support Downloads Knowledge Base Juniper Support Portal Community

Find a Product

Start typing a product name to find Software Downloads for that product.

All Products ▾ Apstra Fabric Conductor Find a Product

[View all products >](#)

Download Results for: Apstra Fabric Conductor | X

Select: OS Apstra Fabric Conductor ▾ VERSION 4.0 ▾ SUPPORTING PLATFORMS Show All ▾ Expand All +

X Application Package 7 File(s)

Description	Release	File Date	Downloads
Apstra Installer for Upgrade	4.0.1	28 Oct 2021	<a href="#">gz (803.93MB)</a> <a href="#">Checksums</a>

2. Run the command `service aos stop` to stop Apstra service, if possible.

```
admin@aos-server:~$ sudo service aos stop
admin@aos-server:~$
```

### 3. Delete the Apstra server database.

```
admin@aos-server:~$ sudo rm -rf /var/lib/aos/db/*
admin@aos-server:~$
```

### 4. Remove the aos-compose package.

```
admin@aos-server:~$ sudo dpkg -r aos-compose
(Reading database ... 110457 files and directories currently installed.)
Removing aos-compose (3.3.0-660) ...
Processing triggers for ureadahead (0.100.0-21) ...
Processing triggers for systemd (237-3ubuntu10.41) ...
admin@aos-server:~$
```

### 5. Reinstall the software from the .run file.

```
admin@aos-server:~$ sudo bash aos_3.3.0-662.run
Verifying archive integrity... All good.
Uncompressing AOS installer 100%
610bd1ae69b7: Loading layer [=====>] 52.44MB/
52.44MB
87db235c4ff8: Loading layer [=====>] 211.3MB/
211.3MB
668b88b6cd3d: Loading layer [=====>] 117.3MB/
117.3MB
b1dd55ca7fd9: Loading layer [=====>] 20.63MB/
20.63MB
3f8ebc7f1fae: Loading layer [=====>] 4.608kB/
4.608kB
Loaded image: aos:3.3.0-662
AOS[2020-07-28_02:58:36]: Installing AOS 3.3.0-662 package
admin@aos-server:~$
```

You can now ["restore" on page 41](#) a backup or build a new blueprint.

## Apstra Server Database Overview

The Apstra server container and related database containers are run under Docker. The database is stored in a single folder in the Apstra server at `/var/lib/aos/db`. You can copy the database between Apstra servers.

Source and Target database versions must be the same version. If versions are different, contact ["Juniper Support" on page 777](#) for assistance before proceeding.

To ensure that device agents can 'call home' properly after database restoration, Source and Target must have the same IP address when starting the Apstra server. You can restore the software to a different IP address, but then you must reconfigure each device agent (`/mnt/flash/aos-config`, `/etc/aos/aos.conf`) to point to the new Apstra server IP address.



**CAUTION:** Any changes you make *within* the Apstra server are *not* stored in the backup.

## Back up Database

You can back up the database while the Apstra server is running. Device/OS image information is not included in backups. When restoring a database, any device/OS image information is discarded.

Disable any active IBA probes and wait until any DB "write" tasks have completed before backing up your database.

1. Run the command `aos_backup` to back up the database. Backups are saved as dated snapshots (`/var/lib/aos/snapshot/<date>/aos.data.tar.gz`) in the Apstra server.

If all IBA probes have been disabled and all "write" tasks have completed, the following message appears.

```
admin@aos-server:~$ sudo aos_backup
=====
Backup operation completed successfully.
=====
New AOS snapshot: 2021-07-28_20-56-26
admin@aos-server:~$
```

If many IBA probes are enabled or if any other DB "write" tasks are in progress, they may not be included in the backup, and the following message appears.

```
admin@aos-server:~$ sudo aos_backup
=====
Warning:
Backup operation has been completed successfully. However AOS state
has been changed while this script was running, which means some
changes might not have been captured in the snapshot created in this
backup. You may choose to invoke aos_backup script again if you wish
to capture these changes right now instead of waiting for the next
backup operation.
```

```
=====
New AOS snapshot: 2021-12-06_16-15-57
admin@aos-server:~$
```

If this message appears, disable your IBA probes and run the `aos_backup` command again.

2. Backups are stored on the Apstra server itself. If the server needs to be restored or if its disk image becomes corrupt, any backups/restores are lost along with the Apstra server. We recommend that you periodically move backups/restores off of the Apstra server to a secure location. Also, if you've scheduled [cron jobs](#) to periodically backup the database, make sure to rotate those files off of the Apstra server to keep the Apstra server VM disk from becoming full. Copy the contents of the snapshot directory to your backup infrastructure.

```
admin@aos-server:~$ sudo ls -lah /var/lib/aos/snapshot/
total 20K
drwx----- 5 root root 4.0K Jul 28 20:58 .
drwxr-xr-x 7 root root 4.0K Jul 28 02:43 ..
drwx----- 2 root root 4.0K Jul 28 02:43 2021-07-28_02-43-12
drwx----- 2 root root 4.0K Jul 28 20:56 2021-07-28_20-56-26
drwx----- 2 root root 4.0K Jul 28 20:58 2021-07-28_20-58-54
admin@aos-server:~$
```

## Restore Database

**NOTE:** If you're restoring a backup to a new Apstra server that uses a different network interface for access (eth1 vs eth0 for example), you must update the `metadb` variable in the `[controller]` section of the `/etc/aos/aos.conf` configuration file, then restart the Apstra server.

1. Verify that the contents of the snapshot folder are on the filesystem. In the example below, we have copied the restoration data to `/tmp/aos_test_restore`.

```
admin@aos-server:~$ sudo ls -lah /var/lib/aos/snapshot/2021-07-28_20-56-26/
total 21M
drwx----- 2 root root 4.0K Jul 28 20:56 .
drwx----- 5 root root 4.0K Jul 28 20:58 ..
-rw----- 1 root root 21M Jul 28 20:56 aos.data.tar.gz
-rwxr-xr-x 1 root root 1.3K Jul 28 20:56 aos_restore
-rw----- 1 root root 1 Jul 28 20:56 comment.txt
admin@aos-server:~$
```

2. Run the `aos_restore` command as illustrated below. The restore process first backs up the current database.

```
admin@aos-server:~$ sudo bash /var/lib/aos/snapshot/2021-07-28_20-56-26/aos_restore
=====
Backup operation completed successfully.
=====
New AOS snapshot: 2020-07-28_20-58-54
(Reading database ... 110457 files and directories currently installed.)
Removing aos-compose (3.3.0-660) ...
Processing triggers for ureadahead (0.100.0-21) ...
Processing triggers for systemd (237-3ubuntu10.41) ...
tar: Removing leading '/' from member names
/etc/aos/aos.conf
/etc/aos-credential/secret_key
/var/lib/aos/db/
/var/lib/aos/db/_Main-000000005f1f7314-000dd7b1-checkpoint
/var/lib/aos/db/_AosController-000000005f1f7314-00035dd2-checkpoint-valid
/var/lib/aos/db/_Central-000000005f1f7313-000ab5f7-checkpoint-valid
/var/lib/aos/db/_Metadb-000000005f1f7312-000a27e1-checkpoint
/var/lib/aos/db/_AosSysdb-000000005f1f7312-000a31be-checkpoint-valid
/var/lib/aos/db/_Credential-000000005f1f7312-000ea6ba-log
/var/lib/aos/db/_Central-000000005f1f7313-000ab5f7-log
/var/lib/aos/db/_Auth-000000005f1f7313-0001e8cf-log-valid
/var/lib/aos/db/_Metadb-000000005f1f7312-000a27e1-log-valid
/var/lib/aos/db/_Auth-000000005f1f7313-0001e8cf-checkpoint-valid
/var/lib/aos/db/_Metadb-000000005f1f7312-000a27e1-checkpoint-valid
/var/lib/aos/db/_Main-000000005f1f7314-000dd7b1-log
/var/lib/aos/db/_Auth-000000005f1f7313-0001e8cf-log
/var/lib/aos/db/_Metadb-000000005f1f7312-000a27e1-log
/var/lib/aos/db/_AosSysdb-000000005f1f7312-000a31be-log-valid
/var/lib/aos/db/_AosAuth-000000005f1f7312-000a0e46-log-valid
/var/lib/aos/db/_AosSysdb-000000005f1f7312-000a31be-log
/var/lib/aos/db/blueprint_backups/
/var/lib/aos/db/blueprint_backups/37321b9c-25b1-4111-849b-522a3852949d/
/var/lib/aos/db/blueprint_backups/37321b9c-25b1-4111-849b-522a3852949d/48/
/var/lib/aos/db/blueprint_backups/37321b9c-25b1-4111-849b-522a3852949d/48/graph.json.zip
/var/lib/aos/db/blueprint_backups/37321b9c-25b1-4111-849b-522a3852949d/48/graph.md5sum
/var/lib/aos/db/_Credential-000000005f1f7312-000ea6ba-log-valid
/var/lib/aos/db/_Credential-000000005f1f7312-000ea6ba-checkpoint
/var/lib/aos/db/_Main-000000005f1f7314-000dd7b1-checkpoint-valid
/var/lib/aos/db/_Central-000000005f1f7313-000ab5f7-log-valid
```

```

/var/lib/aos/db/_AosController-000000005f1f7314-00035dd2-log-valid
/var/lib/aos/db/_AosAuth-000000005f1f7312-000a0e46-checkpoint-valid
/var/lib/aos/db/_AosSysdb-000000005f1f7312-000a31be-checkpoint
/var/lib/aos/db/_AosController-000000005f1f7314-00035dd2-checkpoint
/var/lib/aos/db/_AosAuth-000000005f1f7312-000a0e46-checkpoint
/var/lib/aos/db/.devpi/
/var/lib/aos/db/.devpi/server/
/var/lib/aos/db/.devpi/server/.nodeinfo
/var/lib/aos/db/.devpi/server/.secret
/var/lib/aos/db/.devpi/server/.sqlite
/var/lib/aos/db/.devpi/server/.serverversion
/var/lib/aos/db/.devpi/server/.event_serial
/var/lib/aos/db/_AosController-000000005f1f7314-00035dd2-log
/var/lib/aos/db/_Main-000000005f1f7314-000dd7b1-log-valid
/var/lib/aos/db/_Central-000000005f1f7313-000ab5f7-checkpoint
/var/lib/aos/db/_Auth-000000005f1f7313-0001e8cf-checkpoint
/var/lib/aos/db/_Credential-000000005f1f7312-000ea6ba-checkpoint-valid
/var/lib/aos/db/_AosAuth-000000005f1f7312-000a0e46-log
/var/lib/aos/anomaly/
/var/lib/aos/anomaly/_Anomaly-000000005f1f7313-00060aba-log
/var/lib/aos/anomaly/_Anomaly-000000005f1f7313-00060aba-checkpoint-valid
/var/lib/aos/anomaly/_Anomaly-000000005f1f7313-00060aba-log-valid
/var/lib/aos/anomaly/_Anomaly-000000005f1f7313-00060aba-checkpoint
/opt/aos/aos-compose.deb
/opt/aos/frontend_images/
/opt/aos/frontend_images/aos-web-ui.zip
Selecting previously unselected package aos-compose.
(Reading database ... 110440 files and directories currently installed.)
Preparing to unpack /opt/aos/aos-compose.deb ...
Unpacking aos-compose (3.3.0-660) ...
Setting up aos-compose (3.3.0-660) ...
Processing triggers for ureadahead (0.100.0-21) ...
Processing triggers for systemd (237-3ubuntu10.41) ...
Starting aos_sysdb_1      ... done
Starting aos_controller_1 ... done
Starting aos_nginx_1     ... done
Starting aos_auth_1      ... done
Starting aos_metadb_1    ... done
admin@aos-server:~$

```

- When the database has been restored and migrated to a new server, the entire system state has been copied from the backed up installation to the new target. Run the command `service aos status` to validate the restoration.

```
admin@aos-server:~$ sudo service aos status
* aos.service - LSB: Start AOS management system
   Loaded: loaded (/etc/init.d/aos; generated)
   Active: inactive (dead)
     Docs: man:systemd-sysv-generator(8)

Jul 28 00:36:32 aos-server aos[1078]: [240B blob data]
Jul 28 00:36:32 aos-server systemd[1]: Started LSB: Start AOS management system.
Jul 28 02:45:45 aos-server systemd[1]: Stopping LSB: Start AOS management system...
Jul 28 02:45:46 aos-server aos[4968]: Stopping aos_controller_1 ...
Jul 28 02:45:46 aos-server aos[4968]: Stopping aos_metadb_1 ...
Jul 28 02:45:46 aos-server aos[4968]: Stopping aos_auth_1 ...
Jul 28 02:45:46 aos-server aos[4968]: Stopping aos_sysdb_1 ...
Jul 28 02:45:46 aos-server aos[4968]: Stopping aos_nginx_1 ...
Jul 28 02:45:58 aos-server aos[4968]: [240B blob data]
Jul 28 02:45:58 aos-server systemd[1]: Stopped LSB: Start AOS management system.
admin@aos-server:~$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED
8bc9c1dd7a3a	aos:3.3.0-660	"/usr/bin/aos_launch..."	About a minute ago
Up About a minute		iba141638ea	
b0191320d2bd	aos:3.3.0-660	"/usr/sbin/aos_launc..."	About a minute ago
Up About a minute		aos-offbox-172_20_34_8-f	
136736759f45	aos:3.3.0-660	"/usr/sbin/aos_launc..."	About a minute ago
Up About a minute		aos-offbox-172_20_34_10-f	
00a12eb03ae5	aos:3.3.0-660	"/usr/sbin/aos_launc..."	About a minute ago
Up About a minute		aos-offbox-172_20_34_11-f	
c9b18cd4f55a	aos:3.3.0-660	"/usr/sbin/aos_launc..."	About a minute ago
Up About a minute		aos-offbox-172_20_34_7-f	
90f35781d2a0	aos:3.3.0-660	"/usr/sbin/aos_launc..."	About a minute ago
Up About a minute		aos-offbox-172_20_34_12-f	
f5c2d249176b	aos:3.3.0-660	"/usr/sbin/aos_launc..."	About a minute ago
Up About a minute		aos-offbox-172_20_34_9-f	
ab6c532a37ad	aos:3.3.0-660	"/usr/bin/aos_launch..."	20 hours ago
Up 2 minutes		aos_controller_1	
8e6fd8ae8f08	aos:3.3.0-660	"/usr/bin/aos_launch..."	20 hours ago
Up 2 minutes		aos_metadb_1	

```

5b6359e21386      aos:3.3.0-660      "/usr/bin/aos_launch..." 20 hours ago
Up 2 minutes      aos_auth_1
f665ce206f46      aos:3.3.0-660      "/usr/bin/aos_launch..." 20 hours ago
Up 2 minutes      aos_sysdb_1
335dec5fba44      nginx:1.14.2-upload-echo "nginx -g 'daemon of..." 20 hours ago
Up 2 minutes      aos_nginx_1
admin@aos-server:~$

```

4. The database is stored on the Apstra server itself. If the server needs to be restored or if its disk image becomes corrupt, any backups/restores are lost along with the Apstra server. We recommend that you periodically move backups/restores off of the Apstra server to a secure location. Also, if you've scheduled [cron jobs](#) to periodically backup the database, make sure to rotate those files off of the Apstra server to keep the Apstra server VM disk from becoming full. Copy the contents of the snapshot directory to your backup infrastructure.

```

admin@aos-server:~$ sudo ls -lah /var/lib/aos/snapshot/
total 20K
drwx----- 5 root root 4.0K Jul 28 20:58 .
drwxr-xr-x 7 root root 4.0K Jul 28 02:43 ..
drwx----- 2 root root 4.0K Jul 28 02:43 2021-07-28_02-43-12
drwx----- 2 root root 4.0K Jul 28 20:56 2021-07-28_20-56-26
drwx----- 2 root root 4.0K Jul 28 20:58 2021-07-28_20-58-54
admin@aos-server:~$

```

## Reset Database

The commands below delete *a//* data on the Apstra server to a fresh state.

1. Run the command `service aos stop`.
2. Run the command `rm -rf /var/lib/aos/db/*`.
3. Run the command `service aos start`.

```

admin@aos-server:~$ sudo service aos stop
admin@aos-server:~$ sudo rm -rf /var/lib/aos/db/*
admin@aos-server:~$ sudo service aos start
admin@aos-server:~$

```

## Migrate Database



**CAUTION:** If you bring up a new Apstra server with the same IP address as your old Apstra server without any configuration, when the device agents re-register with the new Apstra server they will revert to an unconfigured "Quarantined" state. You must isolate the new Apstra server from the network while you change its IP address, restore the database and restart the Apstra server.

If you want to **maintain the same IP address** on the new Apstra server, then bring up a new Apstra server VM (with the same version as the original Apstra server) with a temporary IP address. After migrating an `aos_backup` to the new Apstra server, the original Apstra server will be shut down and the IP address will be changed to the original IP address on the new server. We recommend this process if you're using on-box device system agents.

If you want to **use a new IP address** on the new Apstra server, you must manually reconfigure the `aos.conf` file for each on-box device system agent. This is not required for off-box device system agents.

To migrate an active instance from one server to another:

1. Run the command `sudo aos_backup` to back up the original Apstra server.

```
admin@aos-server:~$ sudo aos_backup
=====
Backup operation completed successfully.
=====
New AOS snapshot: 2020-07-27_22-49-34
admin@aos-server:~$
```

2. Copy the snapshot to the new server using a temporary IP address on the new Apstra server.
3. Compress and move the snapshot directory to the new Apstra server. This example uses the `scp` command to copy the file to the new Apstra server using a different IP address.

```
admin@aos-server:~$ sudo tar zcvf aos_backup.tar.gz /var/lib/aos/snapshot/2020-07-27_22-49-3
2020-07-27_22-49-34/
2020-07-27_22-49-34/comment.txt
2020-07-27_22-49-34/aos_restore
2020-07-27_22-49-34/aos.data.tar.gz
admin@aos-server:~$ sudo chown admin:admin aos_backup.tar.gz
admin@aos-server:~$ scp aos_backup.tar.gz admin@172.20.203.4:
Apstra Operating System (AOS) Virtual Appliance

Password:
```

```
aos_backup.tar.gz                100%  20MB 140.9MB/s   00:00
admin@aos-server:~$
```

4. After the snapshot has been removed from the old Apstra server, disconnect the old Apstra server by stopping service or completely shutting down the Apstra server VM.

```
admin@aos-server:~$ sudo service aos stop
admin@aos-server:~$
```

5. If you want to use the same IP address, you must manually reconfigure the eth0 interface on the new Apstra server to the IP address of the old Apstra server. For more information, see ["Configuration" on page 21](#).
6. On the new Apstra server, uncompress the tar.gz file.

```
admin@aos-server:~$ tar zxvf aos_backup.tar.gz
2020-07-27_22-49-34/
2020-07-27_22-49-34/comment.txt
2020-07-27_22-49-34/aos_restore
2020-07-27_22-49-34/aos.data.tar.gz
admin@aos-server:~$
```

7. Run the command `aos_restore` to restore the database on the new Apstra server. This command automatically starts the service after restoring the database.

```
admin@aos-server:~$ cd 2020-07-27_22-49-34
admin@aos-server:~/2020-07-27_22-49-34$ sudo bash aos_restore
[sudo] password for admin:
=====
Backup operation completed successfully.
=====
New AOS snapshot: 2020-07-27_23-07-13
Stopping aos_sysdb_1      ... done
Stopping aos_auth_1      ... done
Stopping aos_controller_1 ... done
Stopping aos_nginx_1     ... done
Stopping aos_metadb_1     ... done
(Reading database ... 110457 files and directories currently installed.)
Removing aos-compose (3.3.0-658) ...
Processing triggers for ureadahead (0.100.0-21) ...
Processing triggers for systemd (237-3ubuntu10.41) ...
tar: Removing leading '/' from member names
```

```

/etc/aos/aos.conf
/etc/aos-credential/secret_key
/var/lib/aos/db/
/var/lib/aos/db/_AosController-000000005f1f376f-0003998b-checkpoint
/var/lib/aos/db/_AosSysdb-000000005f1f376d-000a90ba-log-valid
/var/lib/aos/db/_Main-000000005f1f376f-000569a8-checkpoint
/var/lib/aos/db/_Central-000000005f1f376e-000da3de-checkpoint-valid
/var/lib/aos/db/_Central-000000005f1f376e-000da3de-log
/var/lib/aos/db/_Main-000000005f1f376f-000569a8-log-valid
/var/lib/aos/db/_AosAuth-000000005f1f376d-000a40ff-log
/var/lib/aos/db/_Auth-000000005f1f376e-000f2d35-log-valid
/var/lib/aos/db/_Auth-000000005f1f376e-000f2d35-checkpoint-valid
/var/lib/aos/db/_Metadb-000000005f1f376d-000cb9a9-checkpoint-valid
/var/lib/aos/db/_Central-000000005f1f376e-000da3de-checkpoint
/var/lib/aos/db/_Metadb-000000005f1f376d-000cb9a9-log
/var/lib/aos/db/_Credential-000000005f1f376e-000d740e-log-valid
/var/lib/aos/db/_AosAuth-000000005f1f376d-000a40ff-checkpoint-valid
/var/lib/aos/db/_Metadb-000000005f1f376d-000cb9a9-checkpoint
/var/lib/aos/db/_Main-000000005f1f376f-000569a8-log
/var/lib/aos/db/_AosSysdb-000000005f1f376d-000a90ba-checkpoint-valid
/var/lib/aos/db/_AosController-000000005f1f376f-0003998b-log-valid
/var/lib/aos/db/_Auth-000000005f1f376e-000f2d35-checkpoint
/var/lib/aos/db/_AosSysdb-000000005f1f376d-000a90ba-log
/var/lib/aos/db/_AosSysdb-000000005f1f376d-000a90ba-checkpoint
/var/lib/aos/db/_AosAuth-000000005f1f376d-000a40ff-log-valid
/var/lib/aos/db/blueprint_backups/
/var/lib/aos/db/blueprint_backups/6b90ccfd-a1e0-4473-83e7-d62bce24635f/
/var/lib/aos/db/blueprint_backups/6b90ccfd-a1e0-4473-83e7-d62bce24635f/47/
/var/lib/aos/db/blueprint_backups/6b90ccfd-a1e0-4473-83e7-d62bce24635f/47/graph.json.zip
/var/lib/aos/db/blueprint_backups/6b90ccfd-a1e0-4473-83e7-d62bce24635f/47/graph.md5sum
/var/lib/aos/db/_Central-000000005f1f376e-000da3de-log-valid
/var/lib/aos/db/_Auth-000000005f1f376e-000f2d35-log
/var/lib/aos/db/_Credential-000000005f1f376e-000d740e-log
/var/lib/aos/db/_Credential-000000005f1f376e-000d740e-checkpoint
/var/lib/aos/db/_Credential-000000005f1f376e-000d740e-checkpoint-valid
/var/lib/aos/db/.devpi/
/var/lib/aos/db/.devpi/server/
/var/lib/aos/db/.devpi/server/.nodeinfo
/var/lib/aos/db/.devpi/server/.secret
/var/lib/aos/db/.devpi/server/.sqlite
/var/lib/aos/db/.devpi/server/.serverversion
/var/lib/aos/db/.devpi/server/.event_serial
/var/lib/aos/db/_AosController-000000005f1f376f-0003998b-log

```

```

/var/lib/aos/db/_Main-000000005f1f376f-000569a8-checkpoint-valid
/var/lib/aos/db/_Metadb-000000005f1f376d-000cb9a9-log-valid
/var/lib/aos/db/_AosAuth-000000005f1f376d-000a40ff-checkpoint
/var/lib/aos/db/_AosController-000000005f1f376f-0003998b-checkpoint-valid
/var/lib/aos/anomaly/
/var/lib/aos/anomaly/_Anomaly-000000005f1f36a4-000aaa68-checkpoint-valid
/var/lib/aos/anomaly/_Anomaly-000000005f1f331b-0000e8eb-checkpoint
/var/lib/aos/anomaly/_Anomaly-000000005f1f376f-00002176-checkpoint
/var/lib/aos/anomaly/_Anomaly-000000005f1f376f-00002176-log
/var/lib/aos/anomaly/_Anomaly-000000005f1f331b-0000e8eb-log
/var/lib/aos/anomaly/_Anomaly-000000005f1f2abc-0000a867-log
/var/lib/aos/anomaly/_Anomaly-000000005f1f331b-0000e8eb-checkpoint-valid
/var/lib/aos/anomaly/_Anomaly-000000005f1f2abc-0000a867-checkpoint
/var/lib/aos/anomaly/_Anomaly-000000005f1f36a4-000aaa68-checkpoint
/var/lib/aos/anomaly/_Anomaly-000000005f1f376f-00002176-log-valid
/var/lib/aos/anomaly/_Anomaly-000000005f1f36a4-000aaa68-log
/var/lib/aos/anomaly/_Anomaly-000000005f1f331b-0000e8eb-log-valid
/var/lib/aos/anomaly/_Anomaly-000000005f1f2abc-0000a867-checkpoint-valid
/var/lib/aos/anomaly/_Anomaly-000000005f1f2abc-0000a867-log-valid
/var/lib/aos/anomaly/_Anomaly-000000005f1f36a4-000aaa68-log-valid
/var/lib/aos/anomaly/_Anomaly-000000005f1f376f-00002176-checkpoint-valid
/opt/aos/aos-compose.deb
/opt/aos/frontend_images/
/opt/aos/frontend_images/aos-web-ui.zip
Selecting previously unselected package aos-compose.
(Reading database ... 110440 files and directories currently installed.)
Preparing to unpack /opt/aos/aos-compose.deb ...
Unpacking aos-compose (3.3.0-658) ...
Setting up aos-compose (3.3.0-658) ...
Processing triggers for ureadahead (0.100.0-21) ...
Processing triggers for systemd (237-3ubuntu10.41) ...
Starting aos_nginx_1      ... done
Starting aos_sysdb_1     ... done
Starting aos_controller_1 ... done
Starting aos_metadb_1    ... done
Starting aos_auth_1      ... done
admin@aos-server:~/2020-07-27_22-49-34$

```

**8. Run the command `service aos status` and verify that the Apstra server is running.**

```

admin@aos-server:~/2020-07-27_22-49-34$ service aos status
* aos.service - LSB: Start AOS management system

```

```

Loaded: loaded (/etc/init.d/aos; generated)
Active: active (exited) since Mon 2020-07-27 20:23:09 UTC; 2h 45min ago
Docs: man:systemd-sysv-generator(8)
Tasks: 0 (limit: 4915)
CGroup: /aos.service
admin@aos-server: ~/2020-07-27_22-49-34$

```

9. From the Apstra GUI, from the left navigation menu, navigate to **Devices > Managed Devices** to verify that your devices are online in the "Active" state.

Query: All

all selected only unselected only

Key	Device Profile	Operation Mode	Management IP	Apstra Version	Hostname	OS	Acknowledged?	State
74185F	Cisco NXOSv	FULL CONTROL	10.29.29.13	AOS_4.0.1_OB.1045	leaf5	NXOS 9.3(7)	✓	IS-ACTIVE
36FC3A	Cisco NXOSv	FULL CONTROL	10.29.29.14	AOS_4.0.1_OB.1045	sspine2	NXOS 9.3(7)	✓	IS-ACTIVE
FA3BB6	Cisco NXOSv	FULL CONTROL	10.29.29.15	AOS_4.0.1_OB.1045	sspine1	NXOS 9.3(7)	✓	IS-ACTIVE
535370	Cisco NXOSv	FULL CONTROL	10.29.29.16	AOS_4.0.1_OB.1045	spine1	NXOS 9.3(7)	✓	IS-ACTIVE
5D4E6	Cisco NXOSv	FULL CONTROL	10.29.29.17	AOS_4.0.1_OB.1045	leaf4	NXOS 9.3(7)	✓	IS-ACTIVE
780E5D	Cisco NXOSv	FULL CONTROL	10.29.29.18	AOS_4.0.1_OB.1045	spine3	NXOS 9.3(7)	✓	IS-ACTIVE
BF42C7	Cisco NXOSv	FULL CONTROL	10.29.29.19	AOS_4.0.1_OB.1045	spine2	NXOS 9.3(7)	✓	IS-ACTIVE

## Apstra Server Upgrade

### IN THIS SECTION

- [Upgrade VM-to-VM \(Recommended\) | 52](#)
- [Upgrade In-Place | 61](#)

You can upgrade the Apstra server either on a new VM (recommended) or on the same VM as the current version. See the following table for the general workflow and the sections below for details.

Table 1: Apstra Upgrade Workflow

Stage	Description
Pre-Upgrade Validation	<ul style="list-style-type: none"> <li>• Verify upgrade path is supported</li> <li>• Verify sufficient VM memory for new version</li> <li>• Review blueprints and address issues</li> <li>• Verify device OS versions are supported</li> <li>• Remove any device AAA configuration</li> <li>• Remove any configlets used to configure firewalls</li> <li>• Back up current Apstra environment</li> </ul>
Upgrade Apstra Server	<ul style="list-style-type: none"> <li>• Download Apstra VM image</li> <li>• Install software on controller VM (Check configlets for conflicts.)</li> <li>• Install software on worker VMs (Apstra Cluster VM-VM only)</li> <li>• Verify connections to new server</li> <li>• Import SysDB database (VM-VM only)</li> <li>• Log into new server</li> <li>• Change operation mode to normal</li> </ul>
Upgrade Agents	<ul style="list-style-type: none"> <li>• From the Apstra GUI</li> </ul>
Upgrade Worker Nodes (Apstra Cluster only)	<ul style="list-style-type: none"> <li>• If re-using VMs for worker nodes - import state</li> <li>• If using new VMs for worker nodes - install software on worker VM(s)</li> </ul>

## Upgrade VM-to-VM (Recommended)

### IN THIS SECTION

- [Pre-Upgrade Validation \(VM-VM\) | 52](#)
- [Deploy New Apstra Server \(VM-VM\) | 54](#)
- [Import State \(VM-VM\) | 54](#)
- [Keep Old VM's IP Address \(Optional\) | 58](#)
- [Change Operation Mode to Normal \(VM-VM\) | 58](#)
- [Upgrade On-box Agents \(VM-VM\) | 60](#)
- [Shut Down Old Apstra Server \(VM-VM\) | 61](#)

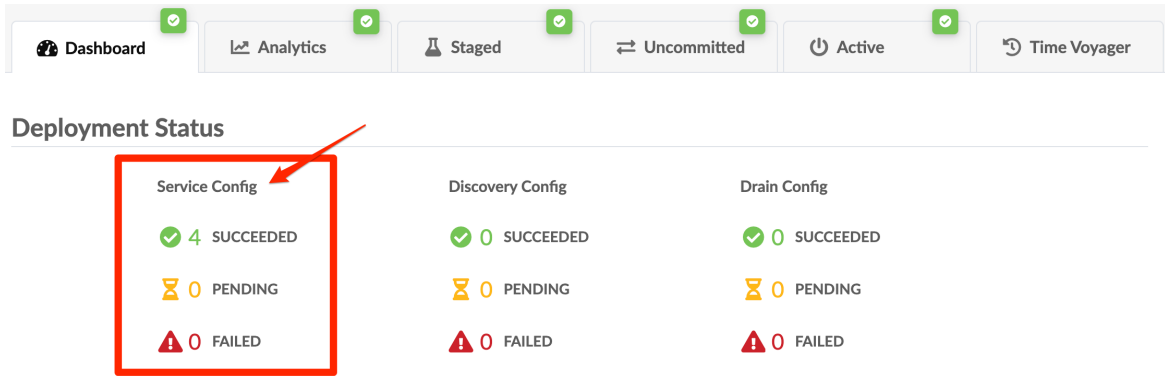
We recommend upgrading on a new VM (instead of in-place on the same VM) so you'll receive Ubuntu Linux OS fixes including security vulnerability updates. To upgrade the Apstra server you need Apstra OS admin user privileges and Apstra admin user group permissions.

### Pre-Upgrade Validation (VM-VM)

1. Confirm that you are upgrading to a ["supported version" on page 74](#).
2. Log into the Apstra server as **admin** (For example, if your Apstra server IP address were 10.28.105.3, the command would be `ssh admin@10.28.105.3`).
3. Check that the server is active and has no issues by running the command `service aos status`.

```
admin@aos-server:~$ service aos status
* aos.service - LSB: Start AOS management system
   Loaded: loaded (/etc/init.d/aos; generated)
   Active: active (exited) since Thu 2021-10-28 17:11:27 UTC; 24h ago
     Docs: man:systemd-sysv-generator(8)
   Process: 1157 ExecStart=/etc/init.d/aos start (code=exited, status=0/SUCCESS)
```

- Review each blueprint to confirm that all **Service Config** has succeeded. If necessary, undeploy and remove devices from the blueprint to resolve any pending or failed service config.



Service Config	Discovery Config	Drain Config
✓ 4 SUCCEEDED	✓ 0 SUCCEEDED	✓ 0 SUCCEEDED
⌚ 0 PENDING	⌚ 0 PENDING	⌚ 0 PENDING
⚠ 0 FAILED	⚠ 0 FAILED	⚠ 0 FAILED

- Review each blueprint for probe anomalies, and resolve them as much as possible. Take notes of any remaining anomalies.
- Verify that your devices' ["NOS versions" on page 894](#) are qualified on the new Apstra version. Upgrade or downgrade, as needed, to one of the supported versions.
- Remove any ["device AAA" on page 169](#) configuration. During device upgrade, configured device agent credentials are required for SSH access.
- Remove any configlets used to configure firewalls. If you use FW's Routing Engine filters on devices, you'll need to update them to include the IP address of the new controller and worker VMs.
- Run an Apstra System "Check" job for all devices (Devices > Agents) and verify that all job states are SUCCESS.

**NOTE:** To upgrade device system agents, Apstra software must SSH to all devices using the configured credentials that were used when creating the device system agent. To verify SSH using these credentials, we recommend running an Apstra System "Check" job for all devices. If any check job fails, resolve the issue before proceeding with the Apstra upgrade.

- As **root** user, back up the Apstra server by running the command `sudo aos_backup`.

```
admin@aos-server:~$ sudo aos_backup
[sudo] password for admin:
=====
Backup operation completed successfully.
=====
New AOS snapshot: 2021-10-29_18-58-56
admin@aos-server:~$
```

- Copy the backup files from `/var/lib/aos/snapshot/<shapshot_name>` to an external location.
- Make sure that the new VM has sufficient ["resources" on page 71](#) for the Apstra server.

## Deploy New Apstra Server (VM-VM)

**NOTE:** If you customized the `/etc/aos/aos.conf` file in the old Apstra server (for example, if you updated the `metadb` field to use a different network interface), you must re-apply the changes to the same file in the new Apstra server VM. It's not migrated automatically.

1. As a registered support user, [download the Apstra VM image from Juniper Support Downloads](#) (`aos_server_4.0.2-142.ova` for example) and transfer it to the new Apstra server.
2. ["Install and configure" on page 4](#) the new Apstra VM image with the new IP address (same or new FQDN may be used).
3. If you're using an Apstra cluster (off-box agents, IBA probes) and you want to put your worker nodes on new VMs, download and deploy a new VM for each worker node. (If you're going to re-use your worker VMs, skip this step.) The upgrade process automatically creates the cluster.

**NOTE:** Example of replacing all VMs: if you had a controller and 2 worker nodes and you wanted to upgrade all of them to new VMs, you would create 3 VMs with the new Apstra version and designate one of them to be the controller.

4. Verify that the new Apstra server has SSH access to the old Apstra server.
5. Verify that the new Apstra server can reach system agents. (See ["Required Communication Ports" on page 72.](#))
6. Verify that the new Apstra server can reach applicable external systems (such as NTP, DNS, vSphere server, LDAP/TACACs+ server and so on).

## Import State (VM-VM)



**CAUTION:** If you perform any API/GUI write operations to the old Apstra server after you've started importing the new VM, those changes won't be copied to the new Apstra server.



**CAUTION:** The Apstra Reference Design in the new Apstra release may have changed in a way that invalidates configlets. To avoid unexpected outcomes, verify that your conflicts don't conflict with the newly rendered config. You can see the newly rendered config in the interactive menu described in this section. If you need to update your configlets, quit the upgrade, update your configlets, then restart the import state procedure.

1. Log into the new Apstra server as user **admin**.
2. Run a command with the following elements to import SysDB from the old server, apply necessary translations, and import configuration:

- `sudo aos_import_state`
- `--ip-address <old-apstra-server-ip>`
- `--username <admin-username>`
- For Apstra clusters with new worker node IP addresses, include the following:

```
--cluster-node-address-mapping <old-node-ip> <new-node-ip>
```

Example command: Single VM or Apstra Cluster with Same Worker Nodes

```
admin@aos-server:~$ sudo aos_import_state --ip-address 10.28.105.3 --username admin
```

Example Command: Apstra Cluster with New Worker Nodes

```
admin@aos-server:~$ sudo aos_import_state --ip-address 10.28.105.3 --username admin --cluster-
node-address-mapping 10.28.105.4 10.28.105.6 --cluster-node-address-mapping 10.28.105.7
10.28.105.8
```

In the example above, 10.28.105.4 and 10.28.105.7 are old worker node IP addresses; 10.28.105.6 and 10.28.105.8 are new worker node IP addresses.

```
admin@aos-server:~$ sudo aos_import_state --ip-address 10.28.105.3 --username admin --cluster-
node-address-mapping 10.28.105.4 10.28.105.6
[sudo] password for admin:
AOS[2021-10-27_20:17:23]: Initiating docker library import DONE
SSH password for remote AOS VM:
Root password for remote AOS VM:
AOS[2021-10-27_20:17:50]: Preparing to retrieve data from remote AOS Server. DONE
AOS[2021-10-27_20:18:29]: Retrieving data from remote AOS Server. This step can take up to 10
minutes DONE
AOS[2021-10-27_20:21:44]: Importing retrieved state to AOS. This step can take up to 30
minutes DONE
AOS[2021-10-27_20:21:48]: Waiting for blueprint <3db44826-807f-4ab9-8ca0-e25040af7ef6>
```

```
processing to finish. DONE
AOS[2021-10-27_20:21:55]: Waiting for blueprint <964211f7-7f3c-4b0a-b6b7-137790c461f5>
processing to finish. DONE
Summary saved to /tmp/aos-upgrade-config-summary-2021.10.27-202203
```

Root is required for importing the database, so you'll be asked for the SSH password and root password for the remote Apstra VM.

**NOTE:** As of Apstra 4.0.2, when you upgrade Apstra cluster, SSH password for old controller, old worker and new worker must be identical, otherwise the upgrade fails with authentication failure. In the above example, the password you enter for 'SSH password for remote AOS VM' is used for remote controller, old worker, and new worker VMs. (AOS-27351)

If you change the worker VMs' SSH password after the upgrade, then you also need to update the worker's password in the GUI (Platform > Apstra Cluster > Nodes).

You're shown a summary of configuration changes that will be pushed to devices as part of the upgrade.

#### Apstra Upgrade Summary

```
=====
This is a summary of configuration pushed to devices logically grouped
into sections. Use 'q' to exit this view. For more device specific
configurations, use the menu after quitting this view
```

```
BLUEPRINT: 3db44826-807f-4ab9-8ca0-e25040af7ef6
(BP2)
```

```
BLUEPRINT: 964211f7-7f3c-4b0a-b6b7-137790c461f5
(BP1)
```

```
Section: FULL_CONFIG
```

```
~~~~~
```

```
Full configuration apply.
```

```
Configuration      Role                      Systems
```

```
=====
```

```
Spine              spine2 [525400E3EF4A, 10.28.105.10]
                   spine1 [52540006D434, 10.28.105.9]
```

```
-----
```

```
Leaf              12-virtual-ext-001-leaf1 [5254006260B2, 10.28.105.11]
```

```
l2-virtual-ext-002-leaf1 [5254009D09D6, 10.28.105.12]
```

Warnings: Template '\_L2 Virtual EVPN' (id: '7bc432ed-c219-4e77-b08d-889ccf939add') has external connectivity settings in RackTypes: ('L2 Virtual Ext' (id: 'L2\_Virtual\_External')) which will be removed during upgrade.

As of Apstra version 4.0.1, the **Apstra Upgrade Summary** shows information separated by device roles (superspine, spine, leaf, leaf pair, and access switch for example). If an incremental config was applied instead of a full config, more details are displayed about the changes.

3. After you've reviewed the summary, enter `q` to exit the summary. The **AOS Upgrade: Interactive Menu** appears where you can access additional information, and continue or quit the upgrade.

```
AOS Upgrade: Interactive Menu
=====
<Device SN> - display config changes using a
               specific device serial number
(s)ummary   - display config change summary
(l)ist      - list all devices with config changes
(d)ump      - dump all config changes to a file
(c)ontinue  - continue with AOS upgrade
(q)uit      - quit AOS upgrade

aos-upgrade (h for help)#
```



**CAUTION:** The Apstra Reference Design in the new Apstra release may have changed in a way that invalidates configlets. To avoid unexpected outcomes, verify that your configlets don't conflict with the newly rendered config. If you need to update your configlets, quit the upgrade, update your configlets, then run the upgrade again.

4. If you want to continue with the upgrade after reviewing pending changes, enter `c`.
5. If you want to stop the upgrade, enter `q` to abort the process. If you quit at this point and later decide to upgrade, you must start the process from the beginning.

**NOTE:** If the Apstra upgrade fails (or in the case of some other malfunction) you can gracefully shut down the new Apstra server and re-start the old Apstra server to continue operations.

### Keep Old VM's IP Address (Optional)

If you want to keep the old VM's IP address you must perform the following extra steps before changing the Operation Mode and upgrading the devices' agent.

1. Shutdown the old VM or change its IP address to a different address to release the IP. This is required to avoid any duplicated IP Address issue.
2. Go to the new VM's Apstra interactive menu from the CLI.

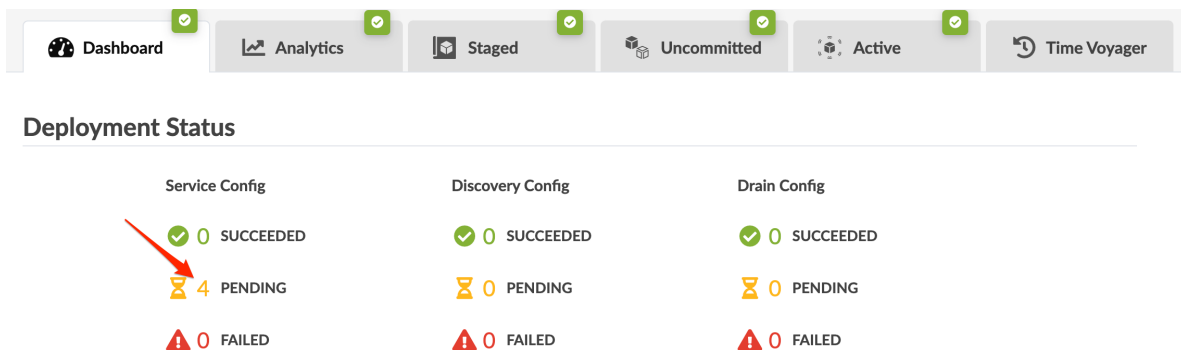
```
admin@aos-server:~$ sudo aos_config
```

3. Click **Network** to update the IP address and confirm the other parameters.
4. For the new IP to take effect, restart the network service, either from the same menu before exiting or from the CLI after leaving the menu.

### Change Operation Mode to Normal (VM-VM)

When you initiate an Apstra server upgrade, the operation mode changes from **Normal** to **Maintenance** automatically. Maintenance mode prevents any off-box agents from coming online prematurely. No configuration is pushed and no telemetry is pulled. At this point, if you can decide to continue using the previous Apstra version instead of upgrading, you could just shut down the new Apstra server. If you've decided to complete the upgrade, change the mode back to **Normal**.

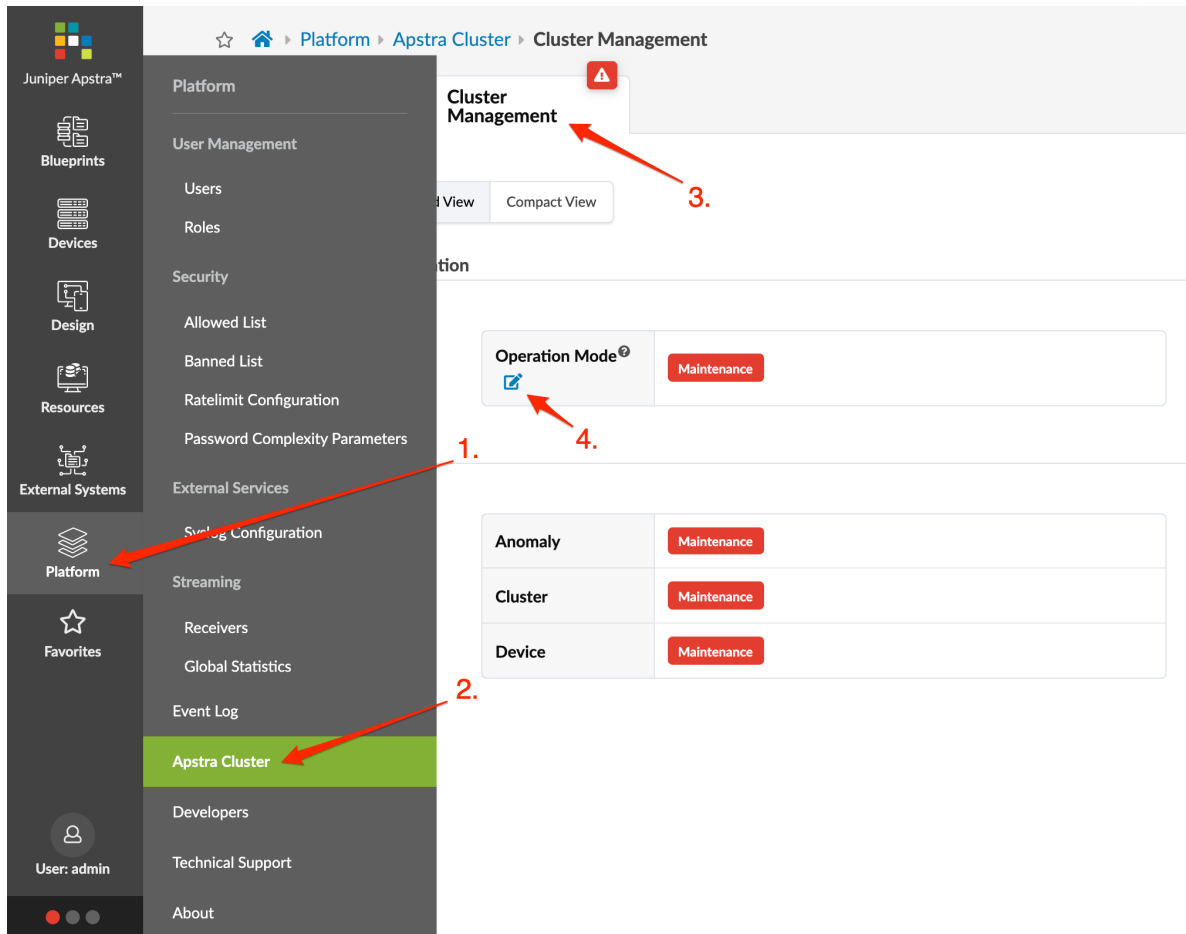
1. Log into the Apstra GUI.
2. If you'd like to view pending service configuration changes, navigate to the dashboard of the blueprint and click **PENDING** to see the affected devices.



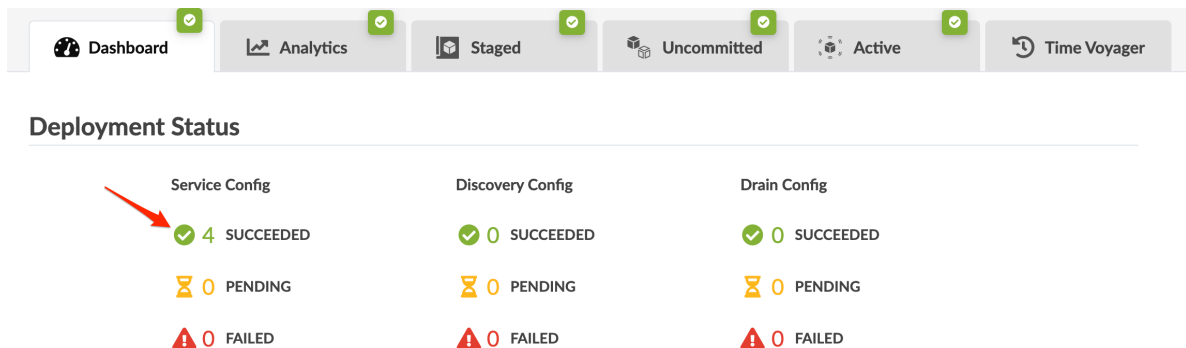
The screenshot shows the Apstra GUI Dashboard with a navigation bar at the top containing: Dashboard, Analytics, Staged, Uncommitted, Active, and Time Voyager. Below the navigation bar is the 'Deployment Status' section, which is divided into three columns: Service Config, Discovery Config, and Drain Config. Each column displays three status indicators: SUCCEEDED (green checkmark), PENDING (yellow hourglass), and FAILED (red exclamation mark). A red arrow points to the 'PENDING' status in the Service Config column.

Service Config	Discovery Config	Drain Config
✓ 0 SUCCEEDED	✓ 0 SUCCEEDED	✓ 0 SUCCEEDED
⌚ 4 PENDING	⌚ 0 PENDING	⌚ 0 PENDING
! 0 FAILED	! 0 FAILED	! 0 FAILED

3. From the left navigation menu, navigate to **Platform > Apstra Cluster > Cluster Management**.



4. Click the **Change Operation Mode** button, select **Normal**, then click **Update**. Any off-box agents, whether they're on the controller or worker VMs automatically come online and reconnect devices and push any pending configuration changes. After a few moments the temporary anomalies on the dashboard resolve and the service configuration section shows that the operation has **SUCCEEDED**.



You can also access the **Cluster Management** page from the lower left section of any page. You have continuous platform health visibility from here as well, based on the colors.

**NOTE:** This feature has been classified as a Juniper Apstra Technology Preview feature. These features are "as is" and voluntary use. Juniper Support will attempt to resolve any issues that customers experience when using these features and create bug reports on behalf of support cases. However, Juniper may not provide comprehensive support services to Tech Preview features.

For additional information, refer to the Juniper Apstra Technology Previews <tech\_previews> page or contact Juniper Support <support>.

From the bottom of the left navigation menu, click one of the dots, then click **Operation Mode** to go to **Cluster Management**. Click the **Change Operation Mode** button, select **Normal**, then click **Update**.

The screenshot shows the Juniper Apstra web interface. The left navigation menu includes options like Blueprints, Devices, Design, Resources, External Systems, Platform, Favorites, and a bottom section with a red dot labeled '1.' and an arrow pointing to it. The main content area is titled 'Cluster Management' and shows a 'Change Operation Mode' button labeled '2. Change operation mode' with an arrow pointing to it. Below this is a table with the following data:

Configuration	
Operation Mode®	Maintenance
Status	
Anomaly	Maintenance
Cluster	Maintenance
Device	Maintenance

### Upgrade On-box Agents (VM-VM)

The Apstra server and on-box agents must be running the same Apstra version. If versions are different the agents won't connect to the Apstra server.

If you're running a multi-state blueprint, especially 5-stage, we recommend that you upgrade agents in stages: first upgrade superspines, then spines, then leafs. We recommend this order because of path hunting. Instead of routing everything up to a spine, or from a spine to a superspine, it's possible for routing to temporarily go from leaf to spine back down to another leaf and back up to another spine. To minimize the chances of this happening, we recommend upgrading devices in stages.

1. Log into the Apstra GUI as user **admin**.
2. From the left navigation menu, navigate to **Devices > System Agents > Agents** and select the device(s) to upgrade (up to 100 devices at a time as of Apstra version 4.0.1). You can upgrade multiple on-box agents at the same time, but the order of device upgrade is important.
  - Upgrade agents for superspines first.
  - Upgrade agents for spines second.
  - Upgrade agents for leafs third.
3. Click the **Install** button to initiate the install process. The job state changes to **IN PROGRESS**. If agents are using a previous version of the Apstra software, they are automatically upgraded to the new version. Then they connect to the server and push any pending configuration changes to the devices. Telemetry also resumes, and the job states change to **SUCCESS**.
4. In the **Liveness** section on the ["blueprint dashboard" on page 400](#) confirm that you don't have any device anomalies.

**NOTE:** If you need to roll back to the previous Apstra version after initiating agent upgrade, you must build a new VM with the previous Apstra version and restore the configuration to that VM. For assistance, contact ["Juniper Support" on page 777](#).

### Shut Down Old Apstra Server (VM-VM)

1. Update any DNS entries to use the new Apstra server IP/FQDN based on your configuration.
2. If you're using a proxy for the Apstra server, make sure it points to the new Apstra server.
3. Gracefully shut down the old Apstra server.
4. If you're upgrading an Apstra cluster and you replaced your worker nodes with new VMs, shut down the old worker VMs as well.

## Upgrade In-Place

### IN THIS SECTION

- [Pre-Upgrade Validation \(In-Place\) | 62](#)
- [Deploy New Apstra Server \(In-Place\) | 63](#)
- [Change Operation Mode to Normal \(In-place\) | 66](#)
- [Upgrade On-box Agents \(In-Place\) | 68](#)
- [Upgrade Worker Nodes \(Apstra Cluster Only\) | 69](#)

To upgrade the Apstra server you need Apstra OS admin user privileges and Apstra admin user group permissions.

### Pre-Upgrade Validation (In-Place)

1. Confirm that you are upgrading to a ["supported version" on page 74](#).
2. Log into the Apstra server as **admin** (For example, if your Apstra server IP address were 10.28.105.3, the command would be `ssh admin@10.28.105.3`).
3. Check that memory utilization is less than 50% to confirm that the VM has enough memory to hold two versions of the Apstra software at the same time. Check resources by running the command `free -h`.

```
admin@aos-server:~$ free -h
```

	total	used	free	shared	buff/cache	available
Mem:	15G	5.1G	8.8G	7.8M	1.8G	10G
Swap:	3.8G	0B	3.8G			

4. If utilization is > 50%, gracefully shutdown the Apstra server, add resources, then restart the Apstra server.
5. Check that the server is active and has no issues by running the command `service aos status`.

```
admin@aos-server:~$ service aos status
* aos.service - LSB: Start AOS management system
   Loaded: loaded (/etc/init.d/aos; generated)
   Active: active (exited) since Thu 2021-10-28 17:11:27 UTC; 24h ago
     Docs: man:systemd-sysv-generator(8)
  Process: 1157 ExecStart=/etc/init.d/aos start (code=exited, status=0/SUCCESS)
```

6. Review each blueprint to confirm that all **Service Config** has succeeded. If necessary, undeploy and remove devices from the blueprint to resolve any pending or failed service config.

The screenshot shows the Apstra dashboard with a navigation bar at the top containing links to Dashboard, Analytics, Staged, Uncommitted, Active, and Time Voyager. Below the navigation bar is the 'Deployment Status' section. This section contains three columns of status information:

Service Config	Discovery Config	Drain Config
✓ 4 SUCCEEDED	✓ 0 SUCCEEDED	✓ 0 SUCCEEDED
⌚ 0 PENDING	⌚ 0 PENDING	⌚ 0 PENDING
⚠ 0 FAILED	⚠ 0 FAILED	⚠ 0 FAILED

A red box highlights the 'Service Config' column, and a red arrow points to the '4 SUCCEEDED' status.

7. Review each blueprint for probe anomalies, and resolve them as much as possible. Take notes of any remaining anomalies.
8. Refer to ["Qualified Device and NOS" on page 894](#) to verify that the devices' NOS versions are qualified on the new Apstra version. Upgrade or downgrade as needed, to one of the supported versions.
9. Remove any ["device AAA" on page 169](#) configuration. During device upgrade, configured device agent credentials are required for SSH access.
10. Remove any configlets used to configure firewalls. If you use FW's Routing Engine filters on devices, you'll need to update them to include the IP address of the new controller and worker VMs.
11. Run an Apstra System "Check" job for all devices (Devices > Agents) and verify that all job states are SUCCESS.

**NOTE:** To upgrade device system agents, Apstra software must SSH to all devices using the configured credentials that were used when creating the device system agent. To verify SSH using these credentials, we recommend running an Apstra System "Check" job for all devices. If any check job fails, resolve the issue before proceeding with the Apstra upgrade.

12. As **root** user, back up the Apstra server by running the command `sudo aos_backup`.

**NOTE:**

```
admin@aos-server:~$ sudo aos_backup
[sudo] password for admin:
=====
Backup operation completed successfully.
=====
New AOS snapshot: 2021-10-29_18-58-56
```

13. Copy the backup files from `/var/lib/aos/snapshot/<snapshot_name>` to an external location.

### Deploy New Apstra Server (In-Place)

1. Download the Apstra installer package for your platform from [Juniper Support Downloads](#) (aos\_4.0.1-1045.run.gz for example) and transfer it to the Apstra server.

```
admin@aos-server:~$ ls -l
total 823228
-rw----- 1 admin admin 842984302 Oct 26 00:44 aos_4.0.1-1045.run.gz
```

2. Unzip the Apstra installer package.

```
admin@aos-server:~$ gunzip aos_4.0.1-1045.run.gz
admin@aos-server:~$ ls -l
total 823108
-rw----- 1 admin admin 842860338 Oct 26 00:44 aos_4.0.1-1045.run
```

3. If you're using an Apstra cluster (off-box agents, IBA probes), download the installer package to the worker nodes as well. You'll upgrade the worker nodes in a later step.
4. Log into the Apstra server as **admin**.
5. Run the `sudo bash aos_<aos_version>.run` command, where `<aos_version>` is the version of the run file. For example, if the version is 4.0.1-1045 the command would be `sudo bash aos_4.0.1-1045.run` as shown below.

```
admin@aos-server:~$ sudo bash aos_4.0.1-1045.run
[sudo] password for admin:
Verifying archive integrity... All good.
Uncompressing AOS installer 100%
=====
Backup operation completed successfully.
=====
AOS[2021-10-28_01:28:52]: Loading AOS 4.0.1-1045 image
AOS[2021-10-28_01:29:44]: Initiating upgrade pre-checker
AOS[2021-10-28_01:29:45]: Initiating docker library import DONE
AOS[2021-10-28_01:30:59]: Preparing to retrieve data from running AOS Server. DONE
AOS[2021-10-28_01:31:11]: Retrieving data from running AOS Server. This step can take up to
10 minutes DONE
AOS[2021-10-28_01:34:30]: Importing retrieved state to AOS pre-checker. This step can take up
to 20 minutes DONE
AOS[2021-10-28_01:34:35]: Waiting for blueprint <3db44826-807f-4ab9-8ca0-e25040af7ef6>
processing to finish. DONE
AOS[2021-10-28_01:34:42]: Waiting for blueprint <964211f7-7f3c-4b0a-b6b7-137790c461f5>
processing to finish. DONE
Summary saved to /tmp/aos-upgrade-config-summary-2021.10.28-013449
```

When you run this command, if any previous Apstra versions are detected, the script enters upgrade mode instead of new installation mode. The new Docker container installs next to the Docker containers from the previous version. The script imports the data from the previous version and migrates it to the Apstra SysDB on the new version.

You'll be shown a summary of configuration changes that will be pushed to devices as part of the upgrade

```

Apstra Upgrade Summary
=====
This is a summary of configuration pushed to devices logically grouped
into sections. Use 'q' to exit this view. For more device specific
configurations, use the menu after quitting this view

BLUEPRINT: 3db44826-807f-4ab9-8ca0-e25040af7ef6
(BP2)

BLUEPRINT: 964211f7-7f3c-4b0a-b6b7-137790c461f5
(BP1)
Section: FULL_CONFIG
~~~~~
Full configuration apply.
Configuration      Role                      Systems
=====
                Spine      spine2 [525400E3EF4A, 10.28.105.10]
                        spine1 [52540006D434, 10.28.105.9]

-----
                Leaf        l2-virtual-ext-001-leaf1 [5254006260B2, 10.28.105.11]
                        l2-virtual-ext-002-leaf1 [5254009D09D6, 10.28.105.12]

```

As of Apstra version 4.0.1, the **Apstra Upgrade Summary** shows information separated by device roles (superspine, spine, leaf, leaf pair, and access switch for example). If an incremental config was applied instead of a full config, more details are displayed about the changes.

6. After you've reviewed the summary, enter q to exit the summary. The **AOS Upgrade: Interactive Menu** appears where you can access additional information, and continue or quit the upgrade.

```

AOS Upgrade: Interactive Menu
=====
<Device SN> - display config changes using a
              specific device serial number
(s)ummary   - display config change summary
(l)ist      - list all devices with config changes
(d)ump      - dump all config changes to a file
(c)ontinue  - continue with AOS upgrade

```

```
(q)uit      - quit AOS upgrade
```

```
aos-upgrade (h for help)#
```



**CAUTION:** The Apstra Reference Design in the new Apstra release may have changed in a way that invalidates configlets. To avoid unexpected outcomes, verify that your configlets don't conflict with the newly rendered config. If you need to update your configlets, quit the upgrade, update your configlets, then run the upgrade again.

7. If you want to continue with the upgrade after reviewing pending changes, enter **c**. The older Apstra version is deleted and the new Apstra version is activated on the server. When the upgrade is complete, you can check the version by navigating to **Platform > About** in the Apstra GUI.



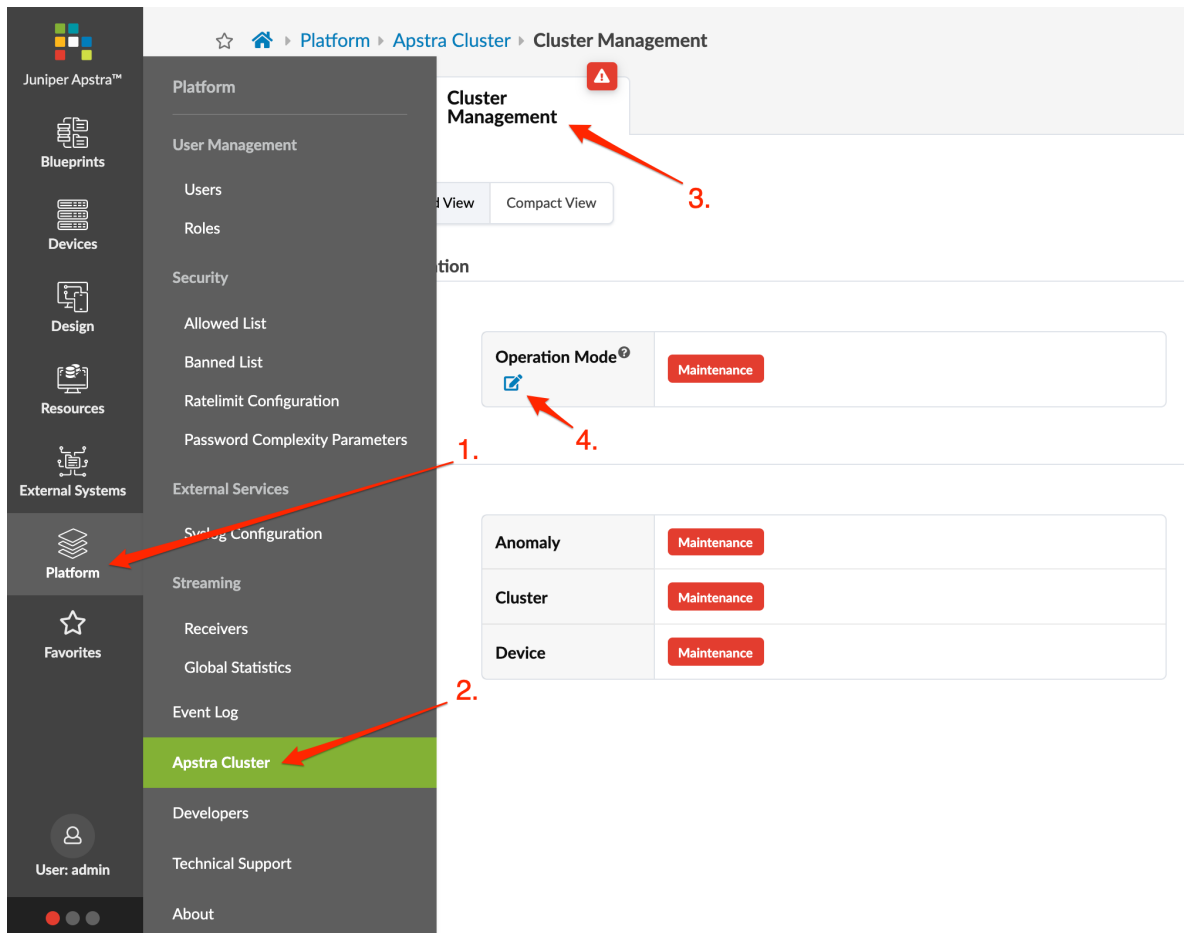
**CAUTION:** Upgrading the Apstra server is a disruptive process. When you upgrade in-place (same VM) and continue with the upgrade from this point, you cannot roll back the upgrade. The only way to return to the previous version is to reinstall a new VM with the previous version and restore the database from the backup that you previously made.

8. If you want to stop the upgrade, enter **q** to abort the process. If you quit at this point and later decide to upgrade, you must start the process from the beginning.
9. If you're using an Apstra cluster, the worker nodes disconnect from the Apstra controller and change to the **FAILED** state. This state means that off-box agents and the IBA probe containers that are on the worker nodes are not available; devices that are managed by the off-box agents do remain in service though. After you upgrade the agents in a later step, you'll upgrade the worker nodes in your Apstra cluster and the agents and/or probes will become available.

### Change Operation Mode to Normal (In-place)

When you initiate an Apstra server upgrade, the operation mode changes from **Normal** to **Maintenance** automatically. After you've completed the upgrade you must manually change the mode back to **Normal**.

1. From the left navigation menu in the Apstra GUI, navigate to **Platform > Apstra Cluster > Cluster Management**.



2. Click the **Change Operation Mode** button, select **Normal**, then click **Update**. When you change the mode to **Normal**, any configured off-box agents are activated, but you must initiate the upgrade of any on-box agents (in the next section).

You can also access the **Cluster Management** page from the lower left section of any page. You have continuous platform health visibility from here as well, based on the colors.

**NOTE:** This feature has been classified as a Juniper Apstra Technology Preview feature. These features are "as is" and voluntary use. Juniper Support will attempt to resolve any issues that customers experience when using these features and create bug reports on behalf of support cases. However, Juniper may not provide comprehensive support services to Tech Preview features.

For additional information, refer to the ["Juniper Apstra Technology Previews" on page 1082](#) page or contact ["Juniper Support" on page 777](#).

From the bottom of the left navigation menu, click one of the dots, then click **Operation Mode** to go to **Cluster Management**. Click the **Change Operation Mode** button, select **Normal**, then click **Update**.

Juniper Apstra™

Blueprints

Devices

Design

Resources

External Systems

Platform

Favorites

Operation Mode

Nodes

Cluster Management

Expanded View

Compact View

Configuration

2. Change operation mode

Operation Mode

Maintenance

Status

Anomaly	Maintenance
Cluster	Maintenance
Device	Maintenance

1.

Because they're still in the process of upgrading, the agents won't be connected. When the upgrade has completed, the agents reconnect to the server and come back online. On the ["blueprint dashboard"](#) on page 400 the **Liveness** anomalies for spine and leaf will also resolve.

### Upgrade On-box Agents (In-Place)

Devices are still connected to the old Apstra server. (see [Devices > Managed Devices](#)). By upgrading the agents, the devices disconnect from the old Apstra server and connect to the new Apstra server.



**CAUTION:** When you initiate agent upgrade you cannot roll back to the previous version. The only way to return to the previous version is to reinstall a new VM with the previous version and restore the database from the backup that you previously made.

1. Log into the Apstra GUI as user **admin**.
2. From the left navigation menu, navigate to **Devices > System Agents > Agents** and select the device(s) to upgrade (up to 100 devices at a time as of Apstra version 4.0.1). You can upgrade multiple on-box agents at the same time, but the order of device upgrade is important.
  - Upgrade agents for superspines first.

- Upgrade agents for spines second.
  - Upgrade agents for leafs third.
3. Click the **Install** button to initiate the install process. The job state changes to **IN PROGRESS**. If agents are using a previous version of the Apstra software, they are automatically upgraded to the new version. Then they connect to the server and push any pending configuration changes to the devices. Telemetry also resumes, and the job states change to **SUCCESS**.
  4. In the **Liveness** section on the ["blueprint dashboard" on page 400](#) confirm that you don't have any device anomalies .

**NOTE:** If you need to roll back to the previous Apstra version after initiating agent upgrade, you must build a new VM with the previous Apstra version and restore the configuration to that VM. For assistance, contact ["Juniper Support" on page 777](#).

### Upgrade Worker Nodes (Apstra Cluster Only)

If you're using an Apstra cluster (for off-box agents and/or IBA probes), you need to upgrade the worker nodes as well as the controller node that you have already upgraded.

1. If you didn't download the Apstra installer package to the worker nodes when you downloaded it to the Apstra server, do that now.
2. From each Apstra worker node, run the `sudo bash aos_<aos_version>.run` command, where `<aos_version>` is the version of the run file. For example, if the version is 4.0.1-1045 the command would be `sudo bash aos_4.0.1-1045.run` (no options). This is the same file you used to upgrade the controller. There are no prompts during the worker node upgrade.

```
admin@aos-server:~$ sudo bash aos_4.0.1-1045.run
Verifying archive integrity... All good.
Uncompressing AOS installer 100%

=====
Backup operation completed successfully.
=====
AOS[2021-10-28_23:15:29]: Removing installed (3.3.0.2-46) AOS package
6babb56be259: Loading layer [=====>] 65.51MB/
65.51MB
4b61bdd19e28: Loading layer [=====>] 5.632kB/
5.632kB
bd9a55afbdce: Loading layer [=====>] 44.03kB/
44.03kB
f495d3ee1163: Loading layer [=====>] 13.31kB/
```

```

13.31kB
0222f30a89f7: Loading layer [=====>] 1.114GB/
1.114GB
15f1b266e91a: Loading layer [=====>] 3.072kB/
3.072kB
3cebea5ed20e: Loading layer [=====>] 5.632kB/
5.632kB
07d63988038c: Loading layer [=====>] 25.6kB/
25.6kB
82bbad94c148: Loading layer [=====>] 88.41MB/
88.41MB
30c5cc7507d8: Loading layer [=====>] 58.8MB/
58.8MB
c3a6272b640d: Loading layer [=====>] 242.4MB/
242.4MB
236ebbddf13a: Loading layer [=====>] 118.3MB/
118.3MB
fcd29376258b: Loading layer [=====>] 25.77MB/
25.77MB
214893e2d628: Loading layer [=====>] 4.608kB/
4.608kB
Loaded image: aos:4.0.1-1045
AOS[2021-10-28_23:16:15]: Installing AOS 4.0.1-1045 package

```

## Reference (Apstra Server)

### IN THIS SECTION

- [Supported Platforms | 71](#)
- [Required Server Resources | 71](#)
- [Required Communication Ports | 72](#)
- [Additional Network Protocols | 73](#)
- [Network Client Services | 73](#)
- [Apstra Server Upgrade Paths | 74](#)
- [Apstra Server Configuration File | 75](#)

## Supported Platforms

Hypervisor	Supported Versions
VMware ESXi	7.0, 6.7, 6.5, 6.0, 5.5
QEMU / KVM for Ubuntu	18.04 LTS
Microsoft Hyper-V	Windows Server 2016 Datacenter Edition
Oracle VirtualBox / VMware Workstation	For lab / evaluation purposes only

## Required Server Resources

Apstra server VM resource requirements are based on the size of the network (blueprint), the scaling of off-box agents and the use of Intent Based Analytics (IBA). If one VM is insufficient for your needs, you can increase resources by ["clustering several VMs" on page 728](#).



**CAUTION:** Although Apstra server VMs might run with fewer resources than recommended below, depending on the size of the network, the CPU and RAM allocations may be insufficient. The system could encounter errors or a critical "segmentation fault" (core dump). If this happens, delete the VM and redeploy it with additional resources.

**Table 2: Recommended VM Resources**

Resource	Recommendation
Memory	64 GB RAM + 300 MB per installed off-box agent*
CPU	8 vCPU
Disk	80 GB
Network	1 network adapter, initially configured with DHCP

\* Off-box agent container memory usage is dependent on the number of IBA collectors enabled. You can add worker nodes to scale off-box agent capacity. We recommend using the ["Apstra Cluster" on page](#)

728 feature in the GUI to monitor usage as shown below.

☆ 🏠 ▶ Platform ▶ Apstra Cluster ▶ Nodes ▶ 10.28.108.4

Nodes Cluster Management

Containers ← Scroll down to the Containers section...

Query: All 1-5 of 5

Page Size: 25

Name	State	Memory Usage, Mb	CPU Usage	Cumulative File Size, Mb
aos-offbox-10_28_108_9-t	launched	175.76	3%	1.80
aos-offbox-10_28_108_12-f	launched	194.71	1%	2.18
aos-offbox-10_28_108_13-f	launched	194.69	1%	2.54
aos_node_keeper_1	launched	344.94	0%	3.89
iba119d95c5	launched	235.77	0%	2.16

... to monitor memory usage

## Required Communication Ports

Open ports and services that run on the Apstra server are listed in the table below. A running iptables instance ensures that network traffic to and from the Apstra server is restricted to the services listed.

Table 3: Apstra Server Network Protocol Requirements

Source	Destination	Protocol	Description
User workstation	Apstra Server	tcp/22 (ssh)	CLI access to Apstra server
User workstation	Apstra Server	tcp/80 (http)	Redirects to tcp/443 (https)
User workstation	Apstra Server	tcp/443 (https)	GUI and REST API
Network Device for device agents	Apstra Server	tcp/80 (http)	Redirects to tcp/443 (https)
Network Device or Off-box Agent	Apstra Server	tcp/443 (https)	Device agent installation and upgrade, Rest API

**Table 3: Apstra Server Network Protocol Requirements (Continued)**

Source	Destination	Protocol	Description
Network Device or Off-box Agent	Apstra Server	tcp/29730-29739	Agent binary protocol (Sysdb)
ZTP Server	Apstra Server	tcp/443 (https)	Rest API for Device System Agent Install
Apstra Server	Network Devices	tcp/22 (ssh)	Device agent installation and upgrade
Off-box Agent	Network Devices	tcp/443 (https) tcp/9443 (nxapi) tcp/830 (for Junos)	Management from Off-box Agent

## Additional Network Protocols

The network protocols in the table below are not required for Apstra server functionality, but they may be required for network device configuration and discovery, and for direct access to devices.

**Table 4: Additional Network Protocols**

Source	Destination	Protocol	Description
Administrator	Network Device	tcp/22 (ssh)	Device management from Administrator
Network Device	DNS Server	udp/53 (dns)	DNS Discovery for Apstra server IP (if applicable)
Network Device	DHCP Server	udp/67-68 (dhcp)	DHCP for automatic management IP (if applicable)

## Network Client Services

Use and configuration of the Apstra server determine the number of network client services that must be enabled.

**Table 5: Apstra server Network Client Services**

Source	Destination	Protocol	Description
Apstra Server	DNS Server	udp/53 (dns)	Server DNS Client

Table 5: Apstra server Network Client Services *(Continued)*

Source	Destination	Protocol	Description
Apstra Server	LDAP Server	tcp/389 (ldap) tcp/636 (ldaps)	Apstra Server LDAP Client (if configured)
Apstra Server	TACACS+ Server	tcp/udp/49 (tacacs)	Apstra Server TACACS+ Client (if configured)
Apstra Server	RADIUS Server	tcp/udp/1812 (radius)	Apstra Server RADIUS Client (if configured)
Apstra Server	Syslog Server	udp/514 (syslog)	Apstra Server Syslog Client (if configured)

## Apstra Server Upgrade Paths

### IN THIS SECTION

- [Supported Upgrade Paths to Version 4.0.2 | 74](#)
- [Supported Upgrade Paths to Version 4.0.1 | 75](#)
- [Known Limitation - Version 4.0.1 | 75](#)

- First major release versions are for new Juniper Apstra installations only. You cannot upgrade to a first major release version (4.0.0 for example).
- You can upgrade to maintenance release versions and later (4.0.2 for example).
- To check your current Apstra version from the GUI navigate to **Platform > About** from the left navigation menu.
- To check your current Apstra version from the CLI run the command `service aos show_version`.

### Supported Upgrade Paths to Version 4.0.2

Table 6: Apstra Server Upgrade Paths to Version 4.0.2

From Version	VM-to-VM (Different VM)	In-Place (Same VM)
4.0.1	Yes	Yes

Table 6: Apstra Server Upgrade Paths to Version 4.0.2 *(Continued)*

From Version	VM-to-VM (Different VM)	In-Place (Same VM)
4.0.0	Yes	Yes

Upgrading from Apstra release versions 3.X are not supported.

Supported Upgrade Paths to Version 4.0.1

Table 7: Apstra Server Upgrade Paths to Version 4.0.1

From Version	VM-to-VM (Different VM)	In-Place (Same VM)
4.0.0	Yes	Yes
3.3.0c	Yes - recommended for production environments	Yes
3.3.0.2	Yes - recommended for production environments	Yes
3.2.2.X	Yes - recommended for production environments	Yes
3.2.X	Yes - recommended for production environments	Yes

Known Limitation - Version 4.0.1

NX-OS MLAG with BGP ECP to shared IP: The automatic creation of the corresponding connectivity template is not supported.

Apstra Server Configuration File

IN THIS SECTION

●

Controller Section | 76

●

Security Section | 76

●

Log Rotate Section | 77

●

Auth Sysdb Log Rotator Section | 77

●

Main Sysdb Log Rotator Section | 78

- Credential Sysdb Log Rotator Section | 79
- Anomaly Sysdb Log Rotator Section | 80
- Device Image Management Section | 81
- Authentication Section | 81
- Device Config Management Section | 82
- Telemetry Init Section | 82
- Telemetry Global Config Section | 83
- Task API Section | 83
- Statistics Section | 83

/etc/aos/aos.conf

### Controller Section

```
admin@aos-server:/etc/aos$ cat aos.conf
[controller]
metadb=eth0

# Role for the controller. Set the option to "slave" in order to setup AOS as a
# slave AOS. The options "metadb" and "node_id" should be also set while
# setting "role" to "slave"
role = controller
# Id of the slave node. Empty in case the server is the controller. The ID is
# generated by the controller.
node_id =
```

### Security Section

```
[security]

# ***EXPERIMENTAL FEATURE*** This feature should not be enabled without Apstra
# engineering assistance. Enable secure connections for AOS system agents.
enable_secure_sysdb_connection = 0
```

## Log Rotate Section

```
[logrotate]

# AOS has builtin log rotate functionality. You can disable it by setting
# <enable_log_rotate> to 0 if you want to use linux logrotate utility to manage
# your log files. AOS agent reopens log file on SIGHUP
enable_log_rotate = 1
# Log file will be rotated when its size exceeds <max_file_size>
max_file_size = 1M
# The most recent <max_kept_backups> rotated log files will be saved. Older
# ones will be removed. Specify 0 to not save rotated log files, i.e. the log
# file will be removed as soon as its size exceeds limit.
max_kept_backups = 5
# Interval, specified as <hh:mm:ss>, at which log files are checked for
# rotation.
check_interval = 1:00:00
```

## Auth Sysdb Log Rotator Section

```
[auth_sysdb_log_rotator]

# AOS has builtin auth sysdb persistence file rotation functionality. Default
# value is 1 which means sysdb retention policy is enabled. You can disable it
# by setting it to 0 and you also can enable it again by setting it to 1. All
# retention policy parameters will be reloaded by restarting AOS service, or
# sending SIGHUP signal to SysdbResourceManager agent via "sudo kill -s 1
# $(pgrep -f SysdbResourceManager)"
enable_auth_sysdb_rotate = 1
# Maximum number of backup copies of valid auth sysdb persistence file groups
# in /var/lib/aos/db. AOS will remove all the older groups. Default value is 5,
# which means AOS will keep the latest 5 groups. Min value is 3. It should be
# specified as a positive number or empty. Leaving it empty means no groups
# number limitation. It will be set to default value if it is configured in
# invalid format. It will be set to minimum value if it is configured to a
# smaller value.
max_kept_backups = 5
# Maximum total size of valid auth sysdb persistence file groups in
# /var/lib/aos/db. Default value is empty, which means no size limitation. It
# should be specified as empty or a positive number ending with k/m/g (case
# insensitive) or no suffix. Otherwise, it will be set to default value. AOS
```

```
# will keep at least 3 valid groups no matter how <max_total_files_size> being
# configured.
max_total_files_size =
# Interval, specified as <hh:mm:ss>, at which auth sysdb persistence files are
# checked for rotation. Default value is 1:00:00. It will be set to default
# value if it is configured in invalid format. Min value is 00:01:00. It will
# be set to min value if it is configured to a smaller value. AOS also update
# all the retention policy parameters per <check_interval> when it is enabled.
check_interval = 1:00:00
```

## Main Sysdb Log Rotator Section

Four parameters for configuring the main graph datastore retention policy.

```
[main_sysdb_log_rotator]

# AOS has builtin main sysdb persistence file rotation functionality. Default
# value is 1 which means sysdb retention policy is enabled. You can disable it
# by setting it to 0 and you also can enable it again by setting it to 1. All
# retention policy parameters will be reloaded by restarting AOS service, or
# sending SIGHUP signal to SysdbResourceManager agent via "sudo kill -s 1
# $(pgrep -f SysdbResourceManager)"
enable_main_sysdb_rotate = 1
# Maximum number of backup copies of valid main sysdb persistence file groups
# in /var/lib/aos/db. AOS will remove all the older groups. Default value is 5,
# which means AOS will keep the latest 5 groups. Min value is 3. It should be
# specified as a positive number or empty. Leaving it empty means no groups
# number limitation. It will be set to default value if it is configured in
# invalid format. It will be set to minimum value if it is configured to a
# smaller value.
max_kept_backups = 5
# Maximum total size of valid main sysdb persistence file groups in
# /var/lib/aos/db. Default value is empty, which means no size limitation. It
# should be specified as empty or a positive number ending with k/m/g (case
# insensitive) or no suffix. Otherwise, it will be set to default value. AOS
# will keep at least 3 valid groups no matter how <max_total_files_size> being
# configured.
max_total_files_size =
# Interval, specified as <hh:mm:ss>, at which main sysdb persistence files are
# checked for rotation. Default value is 1:00:00. It will be set to default
# value if it is configured in invalid format. Min value is 00:01:00. It will
```

```
# be set to min value if it is configured to a smaller value. AOS also update
# all the retention policy parameters per <check_interval> when it is enabled.
check_interval = 1:00:00
```

enable\_main\_sysdb\_rotate = 1 enables and disables the policy.

- Set to **1** to enable the retention policy (default). If you enable the policy after it has been disabled, you must restart the Apstra server for it to be enabled again.
- Set to **0** to disable the retention policy and keep all backups. AOS VM file disk utilization issues may occur. The policy will be disabled during the next retention check (check\_interval). There is no need to restart the Apstra server unless you want to disable the policy immediately.

max\_kept\_backups = 5 maximum number of backups to store in /var/lib/aos/db.

- Leave default of **5** to keep the latest five backups.
- Set to an empty string to keep an unlimited number of backups.
- Setting to an invalid number results in the default value of **5**.
- Setting to a number smaller than **3** (the minimum) results in the minimum value of **3**.

max\_total\_files\_size = maximum file group size to store in /var/lib/aos/db

- Leave default of an empty string for no size limitation.
- Set to a number ending in k, m, or g (case-sensitive) or without a suffix.

The effect of max\_kept\_backups and max\_total\_files\_size is cumulative. For security, Apstra keeps a minimum of three groups of valid Main Graph Datastore persistence files.

check\_interval = 1:00:00 time between retention checks and parameter updates (if file has been updated) (format: <hh:mm:ss>).

- Leave default of **1:00:00** to check every hour.
- Setting to an invalid number results in the default value of **1:00:00**.
- Setting to a number smaller than **00:01:00** (the minimum) results in the minimum value of **1:00:00**.

### Credential Sysdb Log Rotator Section

```
[credential_sysdb_log_rotator]
```

```
# AOS has builtin credential sysdb persistence file rotation functionality.
# Default value is 1 which means sysdb retention policy is enabled. You can
```

```

# disable it by setting it to 0 and you also can enable it again by setting it
# to 1. All retention policy parameters will be reloaded by restarting AOS
# service, or sending SIGHUP signal to SysdbResourceManager agent via "sudo
# kill -s 1 $(pgrep -f SysdbResourceManager)"
enable_credential_sysdb_rotate = 1
# Maximum number of backup copies of valid credential sysdb persistence file
# groups in /var/lib/aos/db. AOS will remove all the older groups. Default
# value is 5, which means AOS will keep the latest 5 groups. Min value is 3. It
# should be specified as a positive number or empty. Leaving it empty means no
# groups number limitation. It will be set to default value if it is configured
# in invalid format. It will be set to minimum value if it is configured to a
# smaller value.
max_kept_backups = 5
# Maximum total size of valid credential sysdb persistence file groups in
# /var/lib/aos/db. Default value is empty, which means no size limitation. It
# should be specified as empty or a positive number ending with k/m/g (case
# insensitive) or no suffix. Otherwise, it will be set to default value. AOS
# will keep at least 3 valid groups no matter how <max_total_files_size> being
# configured.
max_total_files_size =
# Interval, specified as <hh:mm:ss>, at which credential sysdb persistence
# files are checked for rotation. Default value is 1:00:00. It will be set to
# default value if it is configured in invalid format. Min value is 00:01:00.
# It will be set to min value if it is configured to a smaller value. AOS also
# update all the retention policy parameters per <check_interval> when it is
# enabled.
check_interval = 1:00:00

```

## Anomaly Sysdb Log Rotator Section

```

[anomaly_sysdb_log_rotator]

# AOS has builtin anomaly sysdb persistence file rotation functionality.
# Default value is 1 which means sysdb retention policy is enabled. You can
# disable it by setting it to 0 and you also can enable it again by setting it
# to 1. All retention policy parameters will be reloaded by restarting AOS
# service, or sending SIGHUP signal to SysdbResourceManager agent via "sudo
# kill -s 1 $(pgrep -f SysdbResourceManager)"
enable_anomaly_sysdb_rotate = 1
# Maximum number of backup copies of valid anomaly sysdb persistence file
# groups in /var/lib/aos/db. AOS will remove all the older groups. Default

```

```

# value is 5, which means AOS will keep the latest 5 groups. Min value is 3. It
# should be specified as a positive number or empty. Leaving it empty means no
# groups number limitation. It will be set to default value if it is configured
# in invalid format. It will be set to minimum value if it is configured to a
# smaller value.
max_kept_backups = 5
# Maximum total size of valid anomaly sysdb persistence file groups in
# /var/lib/aos/db. Default value is empty, which means no size limitation. It
# should be specified as empty or a positive number ending with k/m/g (case
# insensitive) or no suffix. Otherwise, it will be set to default value. AOS
# will keep at least 3 valid groups no matter how <max_total_files_size> being
# configured.
max_total_files_size =
# Interval, specified as <hh:mm:ss>, at which anomaly sysdb persistence files
# are checked for rotation. Default value is 1:00:00. It will be set to default
# value if it is configured in invalid format. Min value is 00:01:00. It will
# be set to min value if it is configured to a smaller value. AOS also update
# all the retention policy parameters per <check_interval> when it is enabled.
check_interval = 1:00:00

```

## Device Image Management Section

```

[device_image_management]

# Enable version compatibility check. By default version compatibility check is
# enabled. A device will not connect to AOS if its version of AOS device agent
# is not compatible with AOS controller
enable_version_check = 1
# Enable AOS device agent image auto upgrade. By default auto image upgrade is
# disabled. With this option enabled a device can download an image from the
# controller and upgrade itself if needed.
enable_auto_upgrade = 0
# A device will retry in specified timeout (in seconds) if it fails version
# compatibility check or to download/install new image.
retry_timeout = 600

```

## Authentication Section

```

[authentication]

```

```
# Enable authentication/authorization check. By default
# authentication/authorization is enabled. You can disable it by setting enable
# to 0
enable = 1
# Set token expiration time (in seconds). By default token will be expired
# after 24 hours (86400 seconds).
token_expiration = 86400
```

## Device Config Management Section

```
[device_config_management]

# Setting to push quarantine config to unacknowledged devices. By default it is
# disabled as it causes traffic disruptions. Set the value to 1 to enable
# pushing quarantine config, which shuts down all interfaces on the device.
enable_push_quarantine_config = 0
```

## Telemetry Init Section

```
[telemetry_init]

# Number of initial BGP telemetry update rounds before anomaly detection is
# started.
bgp = 4
# Number of initial interface telemetry update rounds before anomaly detection
# is started.
interface = 4
# Number of initial LAG telemetry update rounds before anomaly detection is
# started.
lag = 4
# Number of initial LLDP telemetry update rounds before anomaly detection is
# started.
lldp = 4
# Number of initial route telemetry update rounds before anomaly detection is
# started.
route = 4
# Number of initial MLAG telemetry update rounds before anomaly detection is
# started.
mlog = 4
```

## Telemetry Global Config Section

```
[telemetry_global_config]

# Python multithreading enable/disable knob for telemetry collection
multithreading_config = 1
# Execution timeout for extensible telemetry collectors
command_timeout = 120
```

## Task API Section

```
[task_api]

# Default maximum time in seconds a task can stay in its current state.
default_timeout = 600.0
# Time in seconds a blueprint.create task can stay in its current state.Format:
# "timeout_<task_type>"
timeout_blueprint.create = 360.0
# Time in seconds a blueprint.deploy task can stay in its current state.Format:
# "timeout_<task_type>"
timeout_blueprint.deploy = 300.0
# Time in seconds blueprint.facade.* tasks can stay in their current state.
# Specific facade task overrides prevail over this one.Format:
# "timeout_<task_type>"
timeout_blueprint.facade = 600.0
# Maximum number of tasks, which allowed in the queue. When number of tasks
# becomes higher this value, task rotation will be started.
max_tasks_in_queue = 100
# Maximum number of Bytes in data field which does not require compression. If
# data size is greater than threshold data will be compressed before storing it
# in sysdb.
max_uncompressed_data_size = 1000
```

## Statistics Section

```
[statistics]

# Enable or disable full validation for pod statistics. Disable if Racks and/or
```

```
# Pods tabs load times are excessive  
pod_full_validation = enabled
```

## Design

### IN THIS SECTION

- [Logical Devices \(Design\) | 84](#)
- [Interface Maps \(Design\) | 91](#)
- [Rack Types \(Design\) | 101](#)
- [Templates \(Design\) | 110](#)
- [Configlets \(Design\) | 119](#)
- [Property Sets \(Design\) | 126](#)
- [TCP/UDP Port Aliases \(Design\) | 128](#)
- [Tags \(Design\) | 129](#)

## Logical Devices (Design)

### IN THIS SECTION

- [Logical Device Overview | 84](#)
- [Create Logical Device | 87](#)
- [Edit Logical Device | 90](#)
- [Delete Logical Device | 91](#)

### Logical Device Overview

Logical devices are abstractions of physical devices that specify common device form factors such as number, speed and roles of ports. Vendor-specific information is not included, which lets you plan your

network before selecting vendors and hardware device models. (After selecting hardware devices, logical devices are associated with physical devices with interface maps.) Logical devices are used in rack types and rack-based templates. Some applications of logical devices include:

- Specifying speed and roles for specific ports (For example, the 48th port is always a leaf, or the speed of the 10th port is always 1 Gbps).
- Preparing for port speed transformations (For example, transforming one - 40 GbE port into four - 10 GbE ports)
- Using non-standard port speeds (For example, for a 1 GbE SFP in a 10 GbE port, the underlying hardware is automatically configured correctly.)
- Solving for automatic cable map generation that takes into account failure domains on modular systems (For example, a line card).

Logical devices include the following details:

**Table 8: Logical Device Parameters**

Name	Description
Logical device name	64 characters or fewer
Panel	Port layout based on IP fabric, forwarding engine, line card (slot) or physical layout. A panel contains one or more port groups.
Port Group	A group of ports with the same speed and role(s)
Number of ports	Number of ports in the port group
Speed	Speed of ports in the port group

Table 8: Logical Device Parameters *(Continued)*

Name	Description
Roles	<p>Ports are configured to face the following types of devices:</p> <ul style="list-style-type: none"> <li>• Superspines (5-stage DC fabric only)</li> <li>• Spine</li> <li>• Leaf</li> <li>• Access (limited support) - Port is configured to face an access device. To learn more about this feature and its limitations, contact  Juniper Support &lt;support&gt;  .</li> <li>• Peer (link between two leaf devices)</li> <li>• Unused - not rendered (for example, a dead port)</li> <li>• Generic - Certain roles are not specified in logical devices (for example, a firewall, external router, bare metal server, or load balancer).</li> </ul>

---

From the left navigation menu, navigate to **Design > Logical Devices** to go to logical devices in the global catalog. Click a logical device name to see its details. You can create, clone, edit, and delete logical

devices.

Juniper Apstra™

Design > Logical Devices

Query: All

1-25 of 55

Table View

Page Size: 25

Card View

Capabilities	Panels Count	Ports Count	Ports Summary	Actions
96 × 10 Gbps 8 × 40 Gbps	2	104	<p>96-10-8x40-2</p> <p>96 × 10 Gbps Generic</p> <p>8 × 40 Gbps Superspine • Spine • Generic</p>	<p>Edit Clone Delete</p>
1 × 1 Gbps	1	1	<p>AOS-1x1-1</p> <p>1 × 1 Gbps Leaf • Access</p>	<p>Edit Clone Delete</p>

Click logical device name for details

## Create Logical Device

### IN THIS SECTION

- [Example: Create Logical Device | 88](#)

1. From the left navigation menu, navigate to **Design > Logical Devices** and click **Create Logical Device**.
2. Enter a logical device name.
3. The default panel layout consists of 24 ports (2 rows of 12 ports each). For a different layout, select the number and arrangement of ports to match your requirements by dragging from the bottom-right corner of the layout.
4. Select the ports for the port group by dragging to select contiguous ports, or by clicking individual ports. Clicking a port again deselects it.
5. Select port speed, and applicable role(s) for the selected ports.
6. Click **Create Port Group** (bottom-middle) to create the port group.

7. If unassigned ports remain, repeat the previous two steps until all ports are assigned. For any ports that will not be used, assign them the *Unused* role.
8. To add a panel, click **Add Panel** (bottom-middle) and repeat the steps as for the first panel.
9. Click **Create** (bottom-right) to create the logical device and return to the list view.

#### Example: Create Logical Device

Let's create a logical device with one panel containing one port group with 96 - 10 GbE ports and a second panel containing one port group with 8 - 40 GbE ports.

1. From the left navigation menu, navigate to **Design > Logical Devices** and click **Create Logical Device**.
2. A descriptive name is helpful when referring to the logical device later. For our example we entered **96x10-8x40-2**, which represents the following characteristics:
  - 96x10 - one panel with 96 - 10 GbE ports
  - 8x40 - one panel with 8 - 40 GbE ports

- 2 - number of panels (rack units)

## Create Logical Device ✕

Start creation of a new logical device by filling the form. Alternatively, you can [Import Logical Device](#) from JSON.

Name

96x10-8x40-2 ← Enter name

### PANEL #1

TOTAL

**24 ports**  
0 assigned • 24 available

PORT GROUPS

No port groups created

Connected to ▾

1	3	5	7	9	11	13	15	17	19	21	23
2	4	6	8	10	12	14	16	18	20	22	24

Create port group

Number of ports \*

1 24

Speed \*

Connected To \*  
☐ Superspine  
☐ Spine  
☐ Leaf  
☐ Access  
☐ Peer  
☐ Unused  
☐ Generic

Create Port Group

+

Add Panel

☐ Create Another?

Create

- For the port group in the first panel, drag the bottom-right corner of the port layout to change the default 2x12 configuration to a 3x32 configuration. Leave the number of ports (96) and speed (10

Gbps) as is, and select the **Generic** port role (Connected to).

## Create Logical Device

Start creation of a new logical device by filling the form. Alternatively, you can [Import Logical Device](#) from JSON.

Name  
96x10-8x40-2

**PANEL #1**

TOTAL 96 ports  
0 assigned • 96 available

PORT GROUPS  
No port groups created

Connected to ▾

1	4	7	10	13	16	19	22	25	28	31	34	37	40	43	46	49	52	55	58	61	64	67	70	73	76	79	82	85	88	91	94
2	5	8	11	14	17	20	23	26	29	32	35	38	41	44	47	50	53	56	59	62	65	68	71	74	77	80	83	86	89	92	95
3	6	9	12	15	18	21	24	27	30	33	36	39	42	45	48	51	54	57	60	63	66	69	72	75	78	81	84	87	90	93	96

Create port group

Number of ports \*  
96

Speed \*  
10 Gbps ▾

Connected To \*

- ☐ Superspine
- ☐ Spine
- ☐ Leaf
- ☐ Access
- ☐ Peer
- ☐ Unused
- ☒ Generic

1. Select role

2. Create Port Group

3. Add Panel

☐ Create Another? **Create**

4. Click **Create Port Group** (bottom-middle), then click **Add Panel** (bottom-middle).
5. Drag the bottom-right corner of the port layout to change the configuration to 2x4. Leave the number of ports (8) as is, change the speed to **40 Gbps**, and connect them to **Superspine**, **Spine**, and **Generic**.
6. Click **Create Port Group**, then click **Create** (bottom-right). The new logical device appears in the list view. (In the overview above, it's the first one in the list.)

## Edit Logical Device

If a logical device is linked to an ["interface map" on page 91](#), it cannot be changed. When you change a logical device in the global catalog, rack types and templates that previously embedded that logical device are not affected. This prevents potentially unintended changes to existing rack types and templates. If your intent is for a rack type or template to use a modified logical device, then you must re-import the rack type into the ["template" on page 110](#).

1. Either from the list view (Design > Logical Device) or the details view, click the **Edit** button for the logical device to edit.

2. Make your changes.
  - To change port group details, access the dialog by clicking its description.
  - To add or remove ports from a port group, drag from the bottom-right corner of the port group layout to resize it. If you added ports, enter port speed and role(s).
  - To remove a port group, click the delete button (upper-right).
  - To add a panel, click **Add Panel** and enter relevant port group details.
3. Click **Update** (bottom-right) to update the logical device in the global catalog and return to the list view.

## Delete Logical Device

If a logical device is linked to an ["interface map" on page 91](#), it cannot be deleted.

1. Either from the list view (Design > Logical Devices) or the details view, click the **Delete** button for the logical device to delete.
2. Click **Delete Logical Device** to delete the logical device from the global catalog and return to the list view.

## Interface Maps (Design)

### IN THIS SECTION

- [Interface Map Overview | 91](#)
- [Create Interface Map | 93](#)
- [Example: Create Interface Map with Breakout Ports | 94](#)
- [Example: Inter Port Constraints - Disabled Ports | 97](#)
- [Edit Interface Map | 100](#)
- [Delete Interface Map \(Design\) | 101](#)

## Interface Map Overview

Interface maps consist of interfaces used for achieving the intended network configuration rendering. They map interfaces between logical devices and physical hardware devices (represented with device profiles) while adhering to vendor specifications.

Some characteristics and capabilities of interface maps include:

- Precisely select device ports, transformations and interfaces.
- You are not restricted to selecting interfaces in a contiguous manner.
- Provision QSFP+ breakout ports to transform ports, such as 40GbE ports to 10GbE, 100GbE ports to 25GbE, and so on.
- Port breakouts and available speeds affect possible values of the mapping fields.
- The logical device enables you to plan port and panel mappings accordingly. For example, you can assign a network policy that ensures that spine uplink ports on a leaf switch are always the furthest right ports on a panel.
- If a smaller logical device is mapped to a larger physical device, the unmapped ports in the device profile are marked as **Unused** in the interface map.

From the left navigation menu, navigate to **Design > Interface Maps** to go to interface maps in the global catalog. You can create, clone, edit and delete interface maps.

1. Click on the **Design** menu item in the left navigation bar.

2. Click on the **Interface Maps** sub-menu item.

Click interface map name for details

Interface Map Name	Device Profile	Logical Device	Actions
712-54X_SONIC_BRCM_BUZZNIK_PLUS__AOS-	Accton-AS5712-54X_SONIC_BRCM_BUZZNIK_PLUS	AOS-24x10-2	Edit Clone Delete
712-54X_SONIC_BRCM_BUZZNIK_PLUS__AOS-	Accton-AS5712-54X_SONIC_BRCM_BUZZNIK_PLUS	AOS-48x10+6x40-1	Edit Clone Delete
2-54X-O_Cumulus__AOS-48x10_6x40-1	Accton 5712-54X-O	AOS-48x10+6x40-1	Edit Clone Delete
2-32X-O_Cumulus__AOS-32x40-1	Accton 6712-32X-O	AOS-32x40-1	Edit Clone Delete
Arista DCS-7050QX-32____96-10-8x40-2	Arista DCS-7050QX-32	96-10-8x40-2	Edit Clone Delete



3. Select a logical device from the drop-down list. If you don't see a logical device that fits your requirements, you can ["create" on page 84](#) one.
4. Select a device profile from the drop-down list. If you don't see a device profile that fits your requirements, you can ["create" on page 300](#) one.
5. Map the logical device to the device profile. See example below for details.
6. Click **Create** to create the interface map and return to the list view.

### Example: Create Interface Map with Breakout Ports

To create dense server connectivity, let's create an interface map that breaks out the twenty-four 40 GbE transformable ports of an **Arista DCS-7050QX-32** physical device to ninety-six 10 GbE ports of a **96x10-8x40-2** logical device.

**96x10-8x40-2** is not one of the predefined logical devices that ships with Apstra software, so if you have not created it you will not find it in the drop-down list. If you'd like to follow along with this example, you can create the ["logical device" on page 84](#) before continuing.

1. From the left navigation menu, navigate to **Design > Interface Maps** and click **Create Interface Map**. Leave the name blank. It will populate automatically as you enter more information.
2. From the **Logical Device** drop-down list, select **96x10-8x40-2**. This logical device has 96-10 GbE ports for servers and 8-40 GbE ports for uplinks to spine switches or external routers.
3. From the **Device Profile** drop-down list, select **Arista DCS-7050QX-32**. This device has 24-40 GbE QSFP+ ports that are transformable (4x10 GbE or 1x40 GbE) and 8-40 GbE QSFP+ ports that are not transformable. As soon as both the logical device and device profile are selected, the interface map name is automatically populated.

4. Under **Device profile interfaces** (middle-right), click **Select Interfaces** for the 10 GbE logical ports to go to the port layout.

### Create Interface Map

**Name \***

Arista DCS-7050QX-32\_\_AOS-96x10-8x40-2

**Logical device \***

AOS-96x10-8x40-2

**Device profile \***

Arista DCS-7050QX-32

**Map interfaces**

Logical device port groups		Mapped/required number of interfaces	Device profile interfaces
Speed	Connected To		
10 Gbps	L2 Server • L3 Server	0 / 96	▼ Select interfaces
<div> <div>1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31</div> <div>2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32</div> </div>			
40 Gbps	Superspine • Spine • External Router	0 / 8	► Select interfaces

**Interface map preview** Click on interface to toggle the details

5. Drag to select the first 24 ports. As the ports are selected the white numbers turn gray. When all interfaces are selected the red circle turns green.

- Under **Device profile interfaces** (middle-right), click **Select Interfaces** for the 40 GbE ports to go to the port layout.

## Create Interface Map

[illegible]

7. Drag to select the remaining 8 ports. As the ports are selected the white numbers turn gray. When all interfaces are selected the red circle turns green.

## Create Interface Map

[illegible]

- Click **Create** to create the interface map. You can see the screenshots for the finalized interface map in the overview above.

### Example: Inter Port Constraints - Disabled Ports

## IN THIS SECTION

- Inter Port Constraint Overview | 98
- Disable Unused Ports | 99

## Inter Port Constraint Overview

Inter port constraints for Cumulus devices are handled in both the "device profile" on page 359 and the interface map. For Apstra to generate the correct ports.conf file with these constraints, the unused interfaces must be disabled in the interface map.

For example, if each of the top (odd-numbered) QSFP28 ports in a **Mellanox 2700** device are split into four SFP28 ports, the bottom (even-numbered) QSFP28 ports are blocked. (Source: <https://docs.mellanox.com/display/sn2000pub/Cable+Installation>) The blocked interfaces must be disabled.

### SN2700 and SN2740 Splitting Options

The top QSFP28 ports marked in green are splittable to 4 SFP28 ports, each.

The bottom QSFP28 ports (gray) are blocked when the upper ports are in split mode.

All QSFP28 ports can be split to 2 QSFP28 ports.



Using the predefined interface map **Mellanox\_MSN2700\_Cumulus\_\_AOS-48x10\_8x100-1** as an example, ports 1,3,5,7,9,11,13,15,17,19,21, and 23 were used to generate the 4x10G interfaces, and the



OK and Apstra will automatically set the corresponding ports to disabled.

## Create Interface Map

Name \*

Mellanox MSN2700\_\_AOS-48x10+8x100-1

Logical device \*

AOS-48x10+8x100-1

Device profile \*

Mellanox MSN2700

Map interfaces

Logical device port groups		Mapped/required number of interfaces	Device profile interfaces
Speed	Connected To		
10 Gbps	Access • L2 Server • L3 Server • External Router • Peer	48 / 48	▼ Select interfaces
<div> <div> <div>1</div><div>3</div><div>5</div><div>7</div><div>9</div><div>11</div><div>13</div><div>15</div><div>17</div><div>19</div><div>21</div><div>23</div><div>25</div><div>27</div><div>29</div><div>31</div> </div> <div> <div>2</div><div>4</div><div>6</div><div>8</div><div>10</div><div>12</div><div>14</div><div>16</div><div>18</div><div>20</div><div>22</div><div>24</div><div>26</div><div>28</div><div>30</div><div>32</div> </div> </div> <div> <div> <div>● Transformation #6</div> <div>Interface #1 (12 ports)</div> <div>Interface #2 (12 ports)</div> <div>Interface #3 (12 ports)</div> <div>Interface #4 (12 ports)</div> </div> </div>			
100 Gbps	Spine • External Router	0 / 8	▶ Select interfaces

Do you want to select the disabled interfaces for unused device profile ports?

Select transformed ports

Click OK to disable unused ones

Interface map preview Click on interface to toggle the details



## Edit Interface Map

Changes to interface maps in the global catalog do not affect interface maps that have already been imported into blueprint catalogs, thereby preventing potentially unintended changes to blueprints.



**CAUTION:** Any changes made to predefined interface maps (the ones that ship with Apstra software) are discarded when Apstra is upgraded. To retain a customized interface map through Apstra upgrades, clone the predefined interface map, give it a unique name, and customize it instead of changing the predefined one directly.

- 1. Either from the list view (Design > Interface Maps) or the details view, click the **Edit** button for the interface map to edit.
- 2. Make your changes.
- 3. Click **Update** (bottom-right) to update the interface map and return to the list view.

Delete Interface Map (Design)

- 1. Either from the list view (Design > Interface Maps) or the details view, click the **Delete** button for the interface map to delete.
- 2. Click **Delete Interface Map** to delete it from the global catalog and return to the list view.

Rack Types (Design)

IN THIS SECTION

- Rack Type Overview | 101
- Create Rack Type | 106
- Example: Create Rack Type | 106
- Edit Rack Type in Global Catalog | 109
- Edit Rack Type in Template | 109
- Edit Rack Type in Blueprint | 110
- Delete Rack Type | 110

Rack Type Overview

Rack types define the type and number of leafs, access switches and/or generic systems (as of version 4.0) that are used in rack builds. Since rack types don't define specific vendors or their devices, you can design your network before choosing hardware. If you need to create a "template" on page 110, you'll use rack types to build the structure of your network.

Rack types include the following details:

Summary	Description
Name	17 characters or fewer, and (optional) description

Summary	Description
Fabric connectivity design	<ul style="list-style-type: none"> <li>• L3 Clos - used in 3-stage and 5-stage fabric templates with spines. The spine level connects leafs to each other.</li> <li>• L3 Collapsed - (Junos only) - used in collapsed (spineless) templates (new in version 4.0). Leafs are connected directly to each other via full mesh.</li> </ul>

Leafs	Description
Name	64 characters or fewer
Leaf Logical Device	Used as ToR leaf switch network device(s)
Links per spine, and Link speed (L3 Clos Only)	Number of leaf-spine links and their speed.

*(Continued)*

Leafs	Description
Redundancy Protocol	<p><b>CAUTION:</b> Make sure that the intended platform supports the chosen redundancy protocol. For example, SONiC doesn't support L3 MLAG peers, and ESI is supported on Junos only.</p> <ul style="list-style-type: none"> <li>• <b>None</b> - For single-homed connections</li> <li>• <b>MLAG</b> - For dual-homed connections. Both switches use the same logical device. <ul style="list-style-type: none"> <li>• <b>MLAG Keepalive VLAN ID</b> - If left blank during rack type creation, 2999 is assigned to the peer link during the build phase. If 2999 conflicts with vendors' reserved ranges, enter a different ID. (Ability to specify MLAG VLAN ID from the web interface is new in version 4.0.0.)</li> </ul> </li> </ul> <p><b>NOTE:</b> Network device vendors have varying requirements for "reserved" VLAN ID ranges. For example, Cumulus VLAN-aware Bridge Mode reserves a VLAN ID range, by default, from 3000 to 3999. Cisco NXOS reserves a VLAN ID range from 3968 to 4094. Arista, by default, uses a VLAN ID range from 1006 to 4094 for internal VLANs for routed ports.</p> <ul style="list-style-type: none"> <li>• <b>Peer Links, and Link speed</b> - Number of links between the MLAG devices, and their speed</li> <li>• <b>Peer Link Port Channel ID</b></li> <li>• <b>L3 peer links, and Link speed</b> -Used mainly for BGP peering between border MLAG leafs in non-default routing zones. Mainly used for routed L3 traffic to solve EVPN blackhole issues or if upstream routers go down. L3 peer-links act as backup paths for the north-south traffic. Other than border leaf it can be used on any other ToR leafs as well for avoiding blackholing traffic for a VRF.</li> <li>• <b>L3 Peer Link Port Channel ID</b></li> <li>• <b>ESI (Junos only)</b> - Ethernet Segment ID assigned to the bundled links. Specifying device platforms other than Juniper Junos (such as Cisco, Cumulus, Arista) results in blueprint build errors. See <a href="#">"Juniper EVPN Support" on page 806</a> for information about Juniper ESI support and <a href="#">"ESI MAC MSB settings" on page 603</a> for more information about ESI.</li> </ul>
Tags	<p>(New in version 4.0) User-specified. Select tags from drop-down list generated from global catalog or create tags on-the-fly (which then become part of the global catalog). Tags used in rack types are embedded, so any subsequent changes to tags in the global catalog do not affect the rack type.</p>

**Access Switches - Junos ONLY** - Limited support. See ["Access Layer Switches"](#) on page 805 for more information.

Generic Systems (new in version 4.0)	Description
Name	64 characters or fewer
Generic system count	Number of systems in the set
Port Channel ID Min, and Max	Port channel IDs are used when rendering leaf device port-channel configuration towards generic systems. default: 1-4096. You can customize this field (as of version 3.3.0).
Logical Device	The generic system network device
Tags	(new in version 4.0) User-specified. Select tags from drop-down list generated from global catalog or create tags on-the-fly (which then become part of the global catalog). Useful for specifying generic systems as servers or external routers on nodes and links. Tags used in rack types are embedded, so any subsequent changes to tags in the global catalog do not affect the rack type.

(Continued)

Generic Systems (new in version 4.0)	Description
Logical Link	<ul style="list-style-type: none"> <li>• <b>Name</b> - 64 characters or fewer</li> <li>• <b>Switch</b> - Leaf configured in <b>Leafs</b> section</li> <li>• <b>LAG Mode</b> <ul style="list-style-type: none"> <li>• <b>LACP (Active)</b> - Link Aggregation Control Group (LACP) in active mode - This mode actively advertises LACP BPDU even when the neighbor does not.</li> <li>• <b>LACP (Passive)</b> - Link Aggregation Control Group (LACP) in passive mode - This mode doesn't generate LACP BPDU until it sees one from a neighbor.</li> <li>• <b>Static LAG (no LACP)</b> - Static LAGs don't participate in LACP and will unconditionally operate in forwarding mode.</li> <li>• <b>No LAG</b> - This link is not part of a LAG.</li> </ul> </li> <li>• <b>Physical link count per individual leaf, and Link speed)</b> - Number of links from each generic system to each leaf and their speed. If using dual leaf switches, this number should be half of the total links attached to the generic system.</li> <li>• <b>Tags</b> - (new in version 4.0) User-specified. Select tags from drop-down list generated from global catalog or create tags on-the-fly (which then become part of the global catalog). Useful for specifying generic systems as servers or external routers on nodes and links. Tags used in rack types are embedded, so any subsequent changes to tags in the global catalog do not affect the rack type.</li> </ul>

**NOTE:** You can also add generic systems to blueprints as a Day 2 operation. See ["Add Generic System" on page 448](#).

From the left navigation menu, navigate to **Design > Rack Types** to go to rack types in the design (global) catalog. Click a rack type name to see its details. You can create, clone, edit, and delete rack types.

Juniper Apstra™

Blueprints

Devices

Design

Resources

External Systems

Platform

Favorites

User: admin

Design > Rack Types

Query: All

1-21 of 21

Table view

Card View

Page Size: 25

Leaf Count	Generic System Count	Actions
1 single leaf	2	Edit Clone Delete
1 ESI group	2	Edit Clone Delete
1 single leaf	4	Edit Clone Delete
1 single leaf	40	Edit Clone Delete
1 ESI group	1	Edit Clone Delete
1 single leaf	16	Edit Clone Delete
1 MLAG pair	2	Edit Clone Delete

Click rack type name for details

## Create Rack Type

1. From the left navigation menu, navigate to **Design > Rack Type** and click **Create Rack Type**.
2. Enter a name (17 characters or fewer), (optional) description, then select a fabric connectivity design (L3 Clos, L3 Collapsed).
3. Configure the panel as required for your design.
  - See rack type overview above for parameter details and the example below for a specific use case.
  - To clone or delete a logical link or generic system group within a rack type, click the **Clone** button or **Delete** button (top-right of section).

## Example: Create Rack Type

This example shows how to create a rack type for a dual-connected L2 rack with two AOS-48x10+6x100-1 logical device leaf switches, each with 4-100 GbE spine links and forty-eight dual-connected 10 GbE generic systems.

1. From the left navigation menu, navigate to **Design > Rack Type** and click **Create Rack Type**.
2. Enter a name (**RackType1** in this example), then select **L3 Clos** fabric connectivity design.
3. In the **Leafs** section, enter a name (**MyLeaf1** in this example) and select **AOS-48x10+6x100-1** from the **Leaf Logical Device** drop-down list.

**NOTE:** Instead of scrolling through the list in the **Leaf Logical Device** drop-down list you can start typing in the field to filter the list based on your input.

4. Change the **Links per spine** to **2**. Notice the **Topology** preview on the right side shows the first leaf.

### Create Rack Type

#### Summary

Name \*

RackType1

Description

Fabric connectivity design \*

☒ L3 Clos

Use this option to design rack types used in 3-stage and 5-stage fabric template

☐ L3 Collapsed

Use this option to design rack types used in a collapsed template (spineless)

#### Configuration

Leafs

Access Switches

Generic Systems

Leaf

Name \*

MyLeaf1

Leaf Logical Device \*

AOS-48x10+6x100-1

Links per spine (6 available) \*

2

Link speed \*

100 Gbps

Redundancy Protocol

☒ None

☐ MLAG

☐ ESI

Tags

Select...

+

Add new leaf

#### Preview

Topology

Logical Devices

MyLeaf1\_1

5. Click **Add new leaf** and enter a name for the second leaf (**MyLeaf2** in this example), select **AOS-48x10+6x100-1** from the **Leaf Logical Device** drop-down list, then change the **Links per spine** to **2**. Notice the **Topology** preview on the right side now shows both leaves.
6. Click **Generic Systems**, click **Add new generic system group** and enter a name (**MySystemGroup1** in this example), change the **Generic system count** to **20**, then select **AOS-2x10-1** from the **Logical**

**Device** drop-down list. Notice that the **Topology** preview changes as you configure the rack type.

The screenshot displays the configuration interface for a network setup. On the left, the 'Configuration' tab is active, showing the 'Generic Systems' sub-tab. The 'Generic System Group' form contains the following fields:

- Name:** MySystemGroup1
- Generic system count:** 20
- Port Channel ID Min:** 0
- Port Channel ID Max:** 0
- Logical Device:** AOS-2x10-1
- Tags:** Select...

Below the form, there is a button labeled '+ Add logical link' and another button labeled '+ Add new generic system group'. Red arrows point to the 'Generic Systems' tab, the 'Name' field, the 'Generic system count' field, the 'Logical Device' field, and the '+ Add logical link' button.

On the right, the 'Preview' tab is active, showing a topology diagram. It includes two leaf nodes at the top: 'MyLeaf1\_1' and 'MyLeaf2\_1'. Below them is a grid of 20 system group nodes, labeled 'MySystemGroup1\_1' through 'MySystemGroup1\_20'.

7. Click **Add logical link**, enter a name (**MyLogicalLink1** in this example), select **MyLeaf1** from the **Switch** drop-down list, select **LACP (Active)** for **LAG Mode**, then change **Physical link count per leaf** to **2**.
8. Click **Add new generic system group**, and enter a name (**MySystemGroup2** in this example), change the **Generic system count** to **20**, then from the **Logical Device** drop-down list, select **AOS-2x10-1**.
9. Click **Add logical link**, enter a name (**MyLogicalLink2** in this example), select **MyLeaf2** from the **Switch** drop-down list, select **LACP (Active)** for **LAG Mode** then change **Physical link count per leaf** to **2**.

10. If you'd like to see a preview of the logical devices that you've configured in the rack type, click **Logical Devices** in the **Preview** section.

Create Rack Type ✕

Configuration

Leafs Access Switches **Generic Systems**

**Generic System Group**

Name \*  
MySystemGroup1

Generic system count \*  
20

Port Channel ID Min \* Max \*  
0 0

Logical Device \*  
AOS-2x10-1

Tags  
Select...

**Logical Link**

Name \*  
MyLogicalLink1

Switch \*  
MyLeaf1

LAG Mode  
☒ LACP (Active) ☐ LACP (Passive) ☐ Static LAG (no LACP) ☐ No LAG

Physical link count per leaf (2 available) \*  
2

Link speed \*  
10 Gbps

Done

Preview

Topology **Logical Devices**

**MyLeaf1**

2 x 100 Gbps Links per spine

AOS-48x10+6x100-1  
AOS-48x10+6x100-1

**MyLeaf2**

2 x 100 Gbps Links per spine

AOS-48x10+6x100-1  
AOS-48x10+6x100-1

**MySystemGroup1** 20 generic systems

2 x 10 Gbps MyLogicalLink1 single-homed at MyLeaf1  
LAG Mode: LACP (Active)

AOS-2x10-1  
AOS-2x10-1

**MySystemGroup2** 20 generic systems

2 x 10 Gbps MyLogicalLink2 single-homed at MyLeaf2  
LAG Mode: LACP (Active)

AOS-2x10-1  
AOS-2x10-1

11. Click **Create** to create the rack type in the global catalog and return to the list view.

## Edit Rack Type in Global Catalog

Changes to rack types in the global catalog do not affect rack types that have been embedded into templates (or blueprints that were created from those templates). See the sections below for more information.

1. To edit a rack type in the global catalog, either from the list view (Design > Rack Type) or the details view, click the **Edit** button for the rack type to edit.
2. Make your changes.
3. Click **Update** (bottom-right) to update the rack type in the global catalog and return to the list view.

## Edit Rack Type in Template

If the intent is for a template to use a modified rack type, then after editing the rack type in the global catalog it must be imported into the template. For more information, see [Update Rack Type in Rack Based Template](#) on the ["Templates" on page 110](#) page (Design > Templates > Edit Template).

## Edit Rack Type in Blueprint

You can edit rack types in active running blueprints (as of version 3.2.0). For more information, see ["Edit Rack" on page 480](#) <edit\_rack> (Blueprints > Staged > Physical > Racks).

## Delete Rack Type

Deleting a rack type in the global catalog does not affect templates and blueprints that previously embedded that rack type. For information about deleting racks from blueprints, see ["Delete Rack" on page 480](#) (at Blueprints > Staged > Physical > Racks).

1. To delete a rack type in the global catalog, either from the list view (Design > Rack Type) or the details view, click the **Delete** button for the rack type to delete.
2. Click **Delete** to delete the rack type and return to the list view.

## Templates (Design)

### IN THIS SECTION

- [Template Overview | 110](#)
- [Create Rack Based Template | 116](#)
- [Create Pod Based Template | 117](#)
- [Create Collapsed Template | 118](#)
- [Edit Template | 118](#)
- [Update Rack Type in Rack Based Template | 118](#)
- [Delete Template | 118](#)

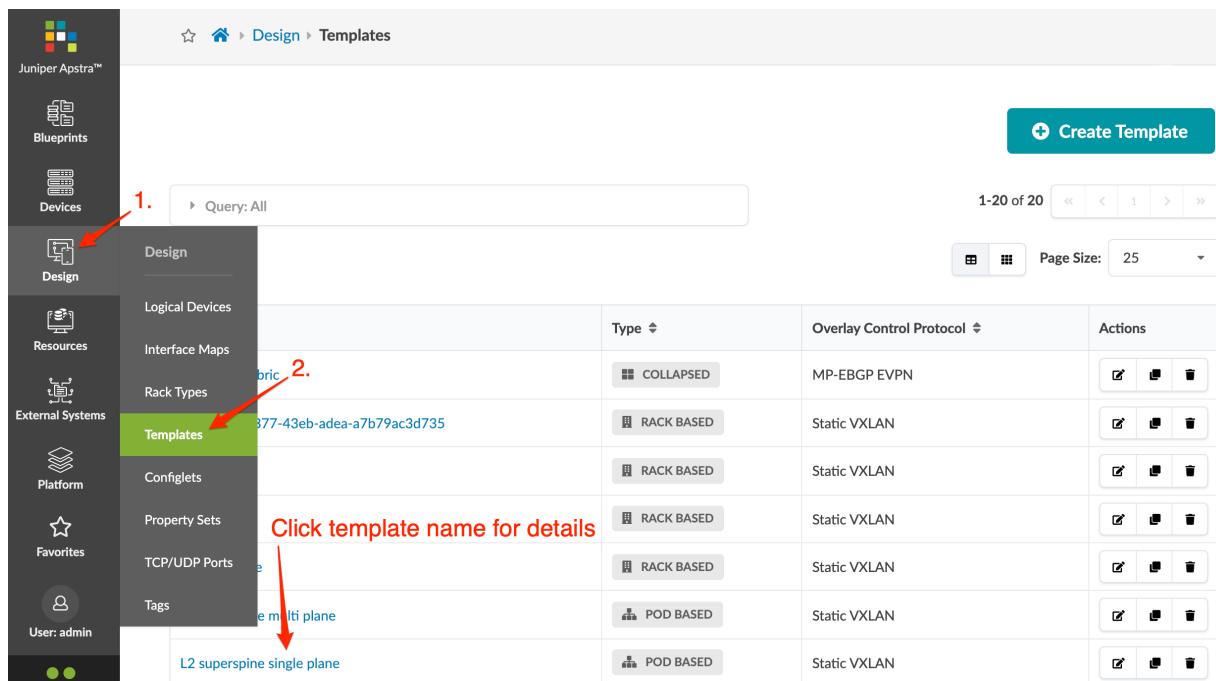
## Template Overview

### IN THIS SECTION

- [Rack-based Template | 111](#)
- [Pod-based Template | 113](#)
- [Collapsed Template | 114](#)

Templates define a network's policy intent and structure. They're used to create blueprints. The global catalog (Design > Templates) includes predefined templates based on common designs.

From the left navigation menu, navigate to **Design > Templates** to go to the templates list view. Many predefined templates are provided for you. Click a template name to see its details. You can create, clone, edit, and delete templates.



See the sections below for details on each type of template.

## Rack-based Template

Rack-based templates define the type and number of racks to connect as top-of-rack (ToR) switches (or pairs of ToR switches). Rack-based templates include the following details:

**Table 9: Rack-based Template Policies**

Policy	Options
ASN Allocation Scheme (spine)	<ul style="list-style-type: none"> <li><b>Unique</b> - applies to 3-stage designs. Each spine is assigned a different ASN.</li> <li><b>Single</b> - applies to 5-stage designs. All spines in each pod are assigned the same ASN, and all superspines are assigned another ASN.</li> </ul>

Table 9: Rack-based Template Policies *(Continued)*

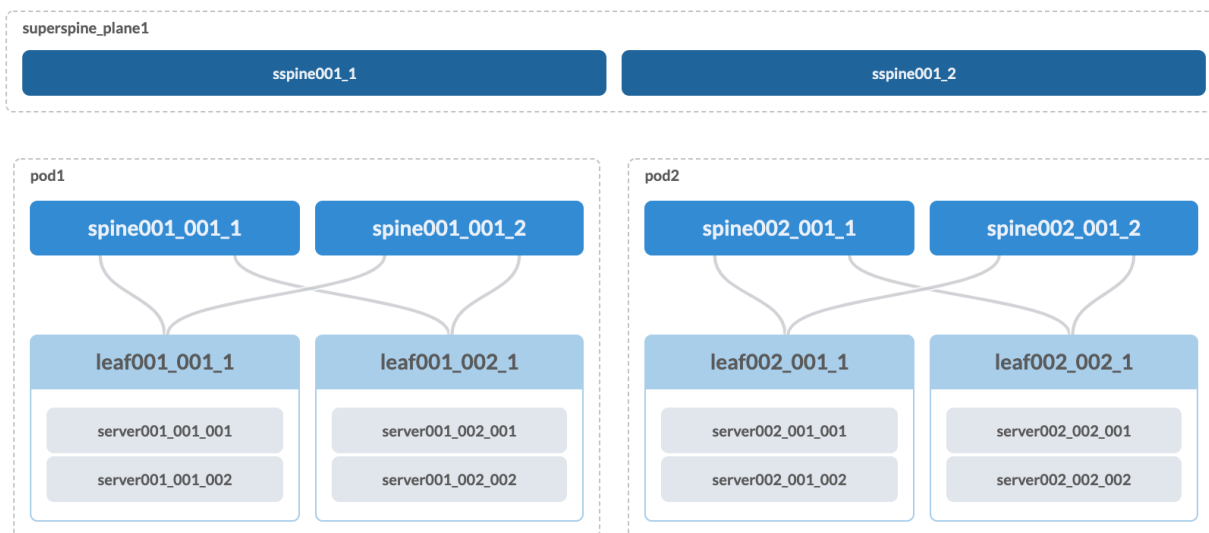
Policy	Options
Routing Policy (import)	<ul style="list-style-type: none"> <li>• <b>Default Only</b> - accepts 0.0.0.0/0 BGP route</li> <li>• <b>ALL</b> - accepts all routes - It sends an internet full table (700k routes), which may cause a crash or other undesirable behavior for fabric network devices that are attached to generic systems. Verify that network devices can accept the appropriate number of routes.</li> </ul>
Overlay Control Protocol	<ul style="list-style-type: none"> <li>• Defines the inter-rack virtual network overlay protocol in the fabric. Overlay control protocol on <i>deployed</i> blueprints can't be changed.</li> <li>• <b>Static VXLAN</b> - uses static VXLAN routing the Head End Replication (HER) flooding to distribute Layer 2 virtual network traffic between racks.</li> <li>• <b>MP-EBGP EVPN</b> - uses EVPN family eBGP sessions between device loopbacks to exchange EVPN routes for hosts (Type 2) and networks (Type 5). Only homogeneous, single-vendor EVPN fabrics are supported. EVPN-VXLAN capabilities for inter-rack virtual networks are dependent on the make and model of network devices used. See "<a href="#">Virtual Networks</a>" on page 488 for more information. External systems must be connected to racks (not spines).</li> </ul>
Spine to Leaf Links Underlay Type	<ul style="list-style-type: none"> <li>• <b>IPv4</b> - uses addresses from "<a href="#">IPv4 resource pools</a>" on page 395.</li> <li>• <b>IPv6 RFC-5549</b> - uses addresses from "<a href="#">IPv6 resource pools</a>" on page 397. Not supported when overlay control protocol is MP-EBGP EVPN.</li> </ul>

Table 10: Rack-based Template Structure

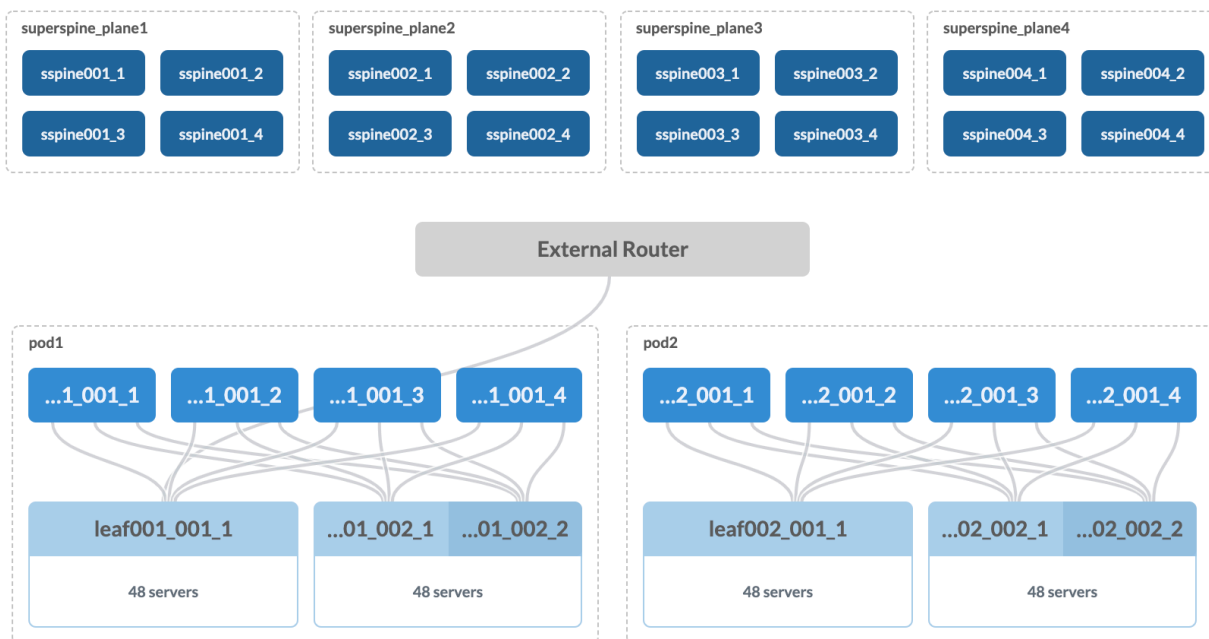
Structure	Options
Rack Types	Type of rack and number of each selected " <a href="#">rack type</a> " on page 101. ESI-based rack types in rack-based templates without EVPN are invalid.
Spines	<ul style="list-style-type: none"> <li>• <b>Spine Logical Device and Count</b> - Type and number of spine "<a href="#">logical devices</a>" on page 84</li> <li>• <b>Links per Superspine Count and Speed</b> - Number and speed of links to any superspines</li> <li>• <b>Tags</b> - (new in version 4.0) User-specified. Select "<a href="#">tags</a>" on page 129 from drop-down list generated from global catalog or create tags on-the-fly (which then become part of the global catalog). Useful for specifying external routers. Tags used in templates are embedded, so any subsequent changes to tags in the global catalog do not affect templates.</li> </ul>

## Pod-based Template

Pod-based templates are used to create large, 5-stage Clos networks, essentially combining multiple rack-based templates using an additional layer of superspines. The following images show examples of 5-stage Clos architectures built using pod-based templates (Superspine links are not shown for readability purposes). See ["5-Stage Clos Architecture" on page 802](#) for more information.



### Single plane, dual superspine



4 x plane, 4 x superspine

Pod-based templates include the following details:

Table 11: Pod-based Template Policies

Policy	Option
Spine to Superspine Links	<ul style="list-style-type: none"> <li>• <b>IPv4</b> - uses addresses from <a href="#">"IPv4 resource pools" on page 395</a>.</li> <li>• <b>IPv6 RFC-5549</b> - uses addresses from <a href="#">"IPv6 resource pools" on page 397</a>. Not supported when overlay control protocol is MP-EBGP EVPN.</li> </ul>
Overlay Control Protocol	<ul style="list-style-type: none"> <li>• Defines inter-rack virtual network overlay protocol used in the fabric. Overlay control protocol on <i>deployed</i> blueprints can't be changed.</li> <li>• <b>Static VXLAN</b> - uses static VXLAN routing the Head End Replication (HER) flooding to distribute Layer 2 virtual network traffic between racks.</li> <li>• <b>MP-EBGP EVPN</b> - uses EVPN family eBGP sessions between device loopbacks to exchange EVPN routes for hosts (Type 2) and networks (Type 5). Only homogeneous, single-vendor EVPN fabrics are supported. EVPN-VXLAN capabilities for inter-rack virtual networks are dependent on the make and model of network devices used. See <a href="#">"Virtual Networks" on page 488</a> for more information. External systems must be connected to racks (not spines).</li> </ul>

Table 12: Pod-based Template Structure

Structure	Options
Pods	Type of rack-based template and number of each selected template
Superspines	<ul style="list-style-type: none"> <li>• <b>Spine Logical Device and Count</b> - Type of spine <a href="#">"logical devices" on page 84</a></li> <li>• <b>Links per Superspine Count and Speed</b> - Number of planes and number of superspines per plane</li> <li>• <b>Tags</b> - (new in version 4.0) User-specified. Select <a href="#">"tags" on page 129</a> from drop-down list generated from global catalog or create tags on-the-fly (which then become part of the global catalog). Useful for specifying external routers. Tags used in templates are embedded, so any subsequent changes to tags in the global catalog do not affect templates.</li> </ul>

## Collapsed Template

Collapsed templates (new in version 4.0) allow you to consolidate leaf, border leaf and spine functions into a single pair of devices. A full mesh topology is created at the leaf level instead of at leaf-spine

connections. This spineless template uses L3 collapsed rack types (new in version 4.0). Collapsed templates have the following limitations:

- No support for upgrading collapsed L3 templates to L3 templates with spines (To achieve the same result you could move devices from the collapsed L3 blueprint to an L3 Clos blueprint.)
- Collapsed L3 templates can't be used as pods in 5-stage templates.
- You can't mix vendors inside redundant leafs - the two leafs must be from the same vendor and model.
- Leaf-to-leaf links can't be added, edited or deleted.
- Inter-leaf connections are limited to full-mesh.
- IPv6 is not supported in version 4.0.

Collapsed templates include the following details:

**Table 13: Collapsed Template Policies**

Policy	Options
Routing Policy (import)	<ul style="list-style-type: none"> <li>• <b>Default Only</b> - accepts 0.0.0.0/0 BGP route</li> <li>• <b>ALL</b> - accepts all routes - It sends an internet full table (700k routes), which may cause a crash or other undesirable behavior for the fabric network devices that are attached to generic systems. Verify that your network devices can accept the appropriate number of routes.</li> </ul>
Overlay Control Protocol	<ul style="list-style-type: none"> <li>• Defines the inter-rack virtual network overlay protocol used in the fabric. Overlay control protocol on deployed blueprints can't be changed.</li> <li>• <b>Static VXLAN</b> - uses static VXLAN routing the Head End Replication (HER) flooding to distribute Layer 2 virtual network traffic between racks.</li> <li>• <b>MP-EBGP EVPN</b> - uses EVPN family eBGP sessions between device loopbacks to exchange EVPN routes for hosts (Type 2) and networks (Type 5). Only homogeneous, single-vendor EVPN fabrics are supported. EVPN-VXLAN capabilities for inter-rack virtual networks are dependent on make and model of network devices used. See <a href="#">"Virtual Networks" on page 488</a> for more information. External systems must be connected to racks (not spines).</li> </ul>

Table 14: Collapse Template Structure

Structure	Options
Rack Types	Type of L3 collapsed rack and number of each selected <a href="#">"rack type" on page 101</a> .
Mesh Links Count and Speed	Defines the link set created between every pair of physical devices, including devices in redundancy groups (MLAG / ESI). These links are always physical L3. No logical links are needed on the mesh level.

## Collapsed Templates in Apstra Versions 4.0.0 and 4.0.1

**NOTE:** This feature has been classified as a Juniper Apstra Technology Preview feature. These features are "as is" and voluntary use. Juniper Support will attempt to resolve any issues that customers experience when using these features and create bug reports on behalf of support cases. However, Juniper may not provide comprehensive support services to Tech Preview features.

For additional information, refer to the Juniper Apstra Technology Previews <tech\_previews> page or contact ["Juniper Support" on page 777](#).

## Create Rack Based Template

You can build a multi-rack environment by selecting multiple rack types, but you can't mix Layer 2 and Layer 3 racks in the same template.

1. If your design requires ["rack types" on page 101](#) and/or ["logical devices" on page 84](#) that are not in the global catalog, create them before proceeding.
2. From the left navigation menu, navigate to **Design > Templates** and click **Create Template**.
3. Enter a name (64 characters or fewer) and select **RACK BASED**.
4. Select applicable policies.
5. Select a rack type from the drop-down list and select the number of that type to include in the template. Notice that as you enter information, the topology preview on the right changes accordingly.
  - To add another rack, click **Add racks**.
6. Select the **Spine Logical Device** from the drop-down list, then select the number of them to include in the template. Make sure to select one that provides a sufficient number of spine ports for your design. For 5-stage designs, make sure to select a logical device that includes the **Superspine** role.
7. For 5-stage designs, enter the number and connection speed of links for **Superspine Connectivity**.

8. Select tags, as applicable (to specify external routers for example), from the drop-down list or create them on-the-fly.
9. Click **Create** to create the template.

Create a ["blueprint" on page 400](#) from the template.

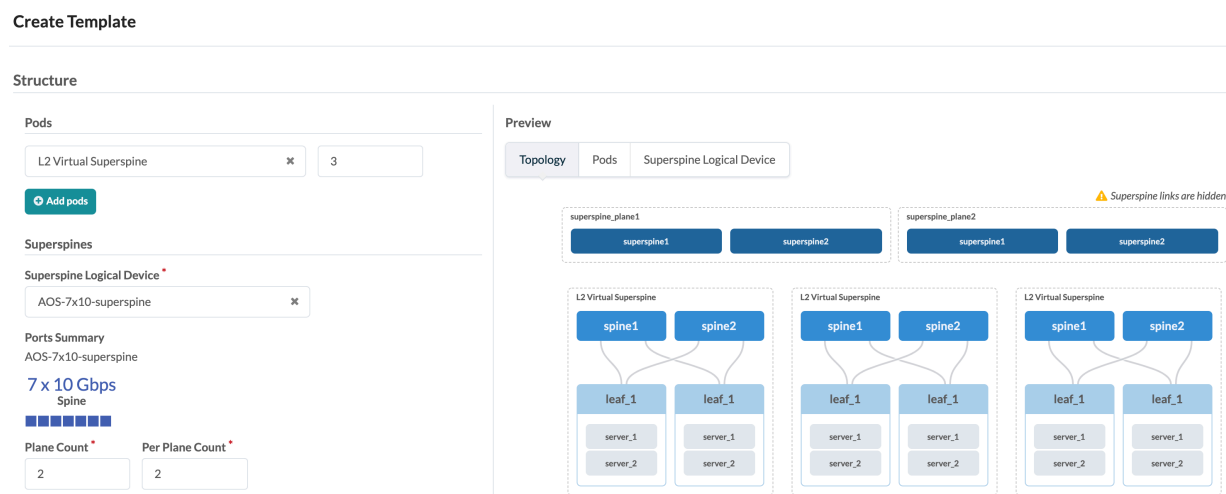
## Create Pod Based Template

A pod-based template consists of multiple rack-based templates; it's essentially a "template of templates" used to build 5-stage Clos networks <5stage>.

1. If your design requires ["templates" on page 110](#), ["rack types" on page 101](#) and/or ["logical devices" on page 84](#) that are not in the global catalog, create them before proceeding.
2. From the left navigation menu, navigate to **Design > Templates** and click **Create Template**.
3. Enter a name (64 characters or fewer) and select **POD BASED**.
4. Select applicable policies.
5. Select a pod from the drop-down list and select the number of that type of pod. Notice that as you enter information, the topology preview on the right changes accordingly.
  - To add another type of pod, click **Add pods** and select another pod from the drop-down list.
6. Select a **Superspines Logical Device** from the drop-down list.
7. Select the number of planes and the number of superspines per plane.
8. Select tags, as applicable (to specify external routers for example), from the drop-down list or create them on-the-fly.
9. Click **Create** to create the template.

Create a ["blueprint" on page 400](#) from the template.

The example below shows a pod-based template with three pods and two planes, each containing two superspines:



## Create Collapsed Template

1. From the left navigation menu, navigate to **Design > Templates** and click **Create Template**.
2. Enter a name (64 characters or fewer) and select **COLLAPSED**.
3. Select applicable policies.
4. Select a ["rack type" on page 101](#) from the drop-down list (only L3 collapsed rack types are available for selecting) and select the number of that type to include in the template. Notice that as you enter information, the topology preview on the right changes accordingly.
5. Click **Create** to create the template.

Create a ["blueprint" on page 400](#) from the template.

## Edit Template

Changes made to a template in the global catalog do not affect blueprints that were previously created with that template, thereby preventing potentially unintended changes to those blueprints.

1. From the left navigation menu, navigate to **Design > Templates** and click the **Edit** button (top-right) for the template to update.
2. Make your changes.
  - To update a rack type in a rack-based template, delete the original rack type from the template (click **X** to the right of the template). Then, *before* clicking **Update**, select the same (modified) rack type from the drop-down list.
3. Click **Update** (bottom-right) to update the template.

## Update Rack Type in Rack Based Template

Changes to a rack type in the global catalog do not affect templates that were previously created with that rack type, thereby preventing potentially unintended changes to those templates. If your intent *is* for the template to use the modified rack type, then you must re-import the rack type into the template.

1. Modify the rack type in the global catalog.
2. From the left navigation menu, navigate to **Design > Templates** and click the **Edit** button (top-right) for the template to update.
3. Click the **X** to the right of the rack type to remove it. Don't click **Update** yet.
4. Select the same rack type from the drop-down list.
5. Click **Update** (bottom-right) to update the template with the modified rack type.

## Delete Template

Do not delete a template if it's referenced by a blueprint.

1. From the left navigation menu, navigate to **Design > Templates** and click the **Delete** button for the template to delete.

2. Click **Delete** to delete the template from the global catalog.

## Configlets (Design)

### IN THIS SECTION

- [Configlet Overview | 119](#)
- [Create Configlet | 124](#)
- [Edit / Delete Configlet \(Design\) | 125](#)

## Configlet Overview

### IN THIS SECTION

- [Configlet Applications | 120](#)
- [When Not to Use Configlets | 120](#)
- [Configlet Parameters | 121](#)
- [Configuration Rendering Order | 123](#)
- [View Configlets \(Design\) | 124](#)

Configlets are configuration templates that augment Apstra's reference design with non-native device configuration. They consist of one or more generators. Each generator specifies a NOS type (config style), when to render the configuration, and CLI commands (and file name as applicable). The section that you select when creating the configlet determines when the configuration is rendered.

When you want to use a configlet, you import it from the global catalog into a blueprint catalog and assign it to one or more roles and/or deployed devices. You can edit the roles and/or devices in a blueprint configlet, but if you want to change the configlet itself, you must export it to the global catalog, modify it, and re-import it into the blueprint.

You can use the same configlets across the entire enterprise, but we recommend creating and applying regionally-specific ["property set" on page 126](#) instead.

**NOTE:** Improperly configured configlets do not raise warnings or restrictions. Testing and validating configlets for correctness is the responsibility of the end user. We recommend that you test configlets on a separate dedicated service to ensure that the configlet performs exactly as intended.

Passwords and other secret keys are not encrypted in configlets.

## Configlet Applications

Some applications for configlets include the following:

- Syslog
- SNMP access policy
- TACACS / RADIUS
- Management ACLs
- Control plane policing
- NTP
- Username / password

## When Not to Use Configlets



**CAUTION:** Using configlets to add non-native configuration is not always appropriate or possible. Configlets are powerful, but if used improperly they pose risks to deployment stability and reference design feature interactions. Testing and validating configlets for correctness is the responsibility of the end user.

Don't use configlets to replace reference design configuration, such as for routing or connectivity. If you change interface configuration, the Apstra-intended interface configuration could be overwritten. For example, if a configlet creates a network span port, you must apply the configlet to an **Unused** port, or it might inadvertently overwrite one that is already in use.

On Cisco NX-OS and Arista EOS devices, do not use configlets to configure multi-line banners (such as banner motd) because of a problematic extra non-ASCII character that cannot be entered. Instead, configure multi-line banners with Cisco POAP (Power-on Auto Provisioning) or ZTP (Arista Zero Touch Provisioning) before installing the device agent. The banner configuration becomes part of the device's pristine configuration and persists throughout the Apstra configuration. Another option is to manually configure multi-line banners on the device. This method causes a *configuration deviation* anomaly that

you can clear by accepting the new configuration as the Golden Configuration<golden\_config>. For more information, see ["Configuration Deviation" on page 671](#).

### Configlet Parameters

Configlets include the following details. The selected config style (NOS type) and section determine whether template text, negation template text and filename are required:

**Table 15: Configlet Parameters**

Name	Description
Configlet Name	64 characters or fewer
Config Style (NOS Type)	Junos, NX-OS, EOS, SONiC, Cumulus
<ul style="list-style-type: none"> <li>Section: System (Junos [on Apstra version 4.0 and 4.0.1], NX-OS, EOS, SONiC, Cumulus)</li> <li>Section: Top-Level: Hierarchical (previously called System) (Junos on Apstra version 4.0.2)</li> </ul>	<ul style="list-style-type: none"> <li>Runs commands as root user. Improper changes could break the functionality of the reference design and <b>take down a network</b>.</li> <li>When a device is unassigned from a node, the negation template text removes configuration. For example, if the template text is <code>username example privilege 15 secret 0 MyPassword</code>, the negation template text might be <code>no username example</code>.</li> <li>Can be used in conjunction with File configlets to restart processes or perform administrative tasks after File configlets render.</li> <li>System configlets can nest other configuration.</li> <li>For NX-OS and EOS, the appropriate configure terminal context is applied. It doesn't need to be part of the configlet.</li> </ul>
Section: Top-Level: Set / Delete (Junos on Apstra version 4.0.2)	Author configlets using Juniper "Set" style rather than structured JSON

Table 15: Configlet Parameters (*Continued*)

Name	Description
<ul style="list-style-type: none"> <li>• Section: Interface (NX-OS, EOS, Cumulus)</li> <li>• Section: Interface-Level: Hierarchical (Junos on Apstra version 4.0.2)</li> </ul>	<ul style="list-style-type: none"> <li>• For physical devices only.</li> <li>• You specify the interface when you <a href="#">"import" on page 615</a> the configlet into a blueprint (scope).</li> <li>• For Cumulus: net clu (NCLU) syntax is no longer used. The config line must match /etc/network/interfaces syntax exactly.</li> </ul>
Section: Interface-Level: Set (Junos on Apstra version 4.0.2)	Author configlets using Juniper "Set" command rather than structured JSON. Text is validated to begin with 'set'.
Section: Interface-Level: Delete (Junos on Apstra version 4.0.2)	Author configlets using Juniper "Delete" command rather than structured JSON. Text is validated to begin with 'delete'.
Section: File (SONiC, Cumulus)	<ul style="list-style-type: none"> <li>• The entire contents of the file must be present within the configlet because the entire file is overwritten; there is no versioning or storing of the original file contents, so it can't be restored to its original contents. <b>Improper use can take down a network.</b> Do not use on config files of critical processes (such as /etc/frr/frr.conf or /etc/network/interfaces/).</li> <li>• Contents are written, as root user, to the /etc directory file (because of Apstra's Docker container host mount). To write to a file outside of /etc (/usr for example) build the File configlet, then use a System configlet to move the file afterwards.</li> </ul>
Section: System Top (NX-OS, EOS)	Ensures that a setting can be overwritten to implement programmed intent. When the reference design is applied, any needed features that were "turned off" in this configlet are reenabled.
Section: FRR (SONiC, Cumulus)	<ul style="list-style-type: none"> <li>• Configlet configuration is appended to the end of the Apstra-generated /etc/frr/frr.conf file and becomes part of FRR intent. Configuration is incrementally included in frr-reload.</li> <li>• <b>Template text is not validated.</b> Errors are likely to cause deployment errors, unintended configuration and device impact.</li> </ul>
Template Text	CLI commands to add configuration to devices. Issued directly to devices without validation.

**Table 15: Configlet Parameters** *(Continued)*

Name	Description
Negation Template Text	CLI commands to disable configlet functionality (when a device is unassigned). Issued directly to devices without validation.
Filename	For File configlets

### Configuration Rendering Order

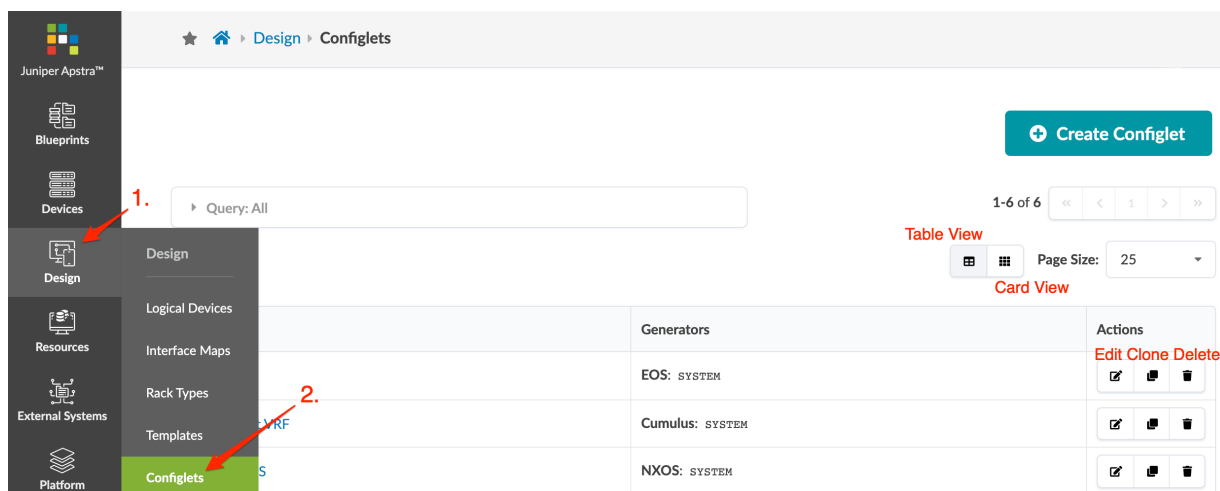
Configuration rendering order is as follows:

1. System Top: negation template text (NX-OS, EOS)
2. System Top: template text (NX-OS, EOS)
3. Apstra reference design
4. Interface: negation template text (NX-OS, EOS, Cumulus)
5. System: negation template text (Junos, NX-OS, EOS, SONiC, Cumulus)
6. File (SONiC, Cumulus)
7. System: template text (Junos, NX-OS, EOS, SONiC, Cumulus)
8. Interface: template text (NX-OS, EOS, Cumulus)

To control the order of operations within a section, create configlets with numeric names. For example, 01\_syslog renders before 02\_ntp. Configlets are then ordered based on the condition of the configlet (for example the spine or leaf role), and then by the Node ID of the configlet.

## View Configlets (Design)

From the left navigation menu, navigate to **Design > Configlets** to go to configlets in the design (global) catalog. You can create, clone, edit and delete configlets.



## Create Configlet

1. From the left navigation menu, navigate to **Design > Configlets** and click **Create Configlet**.
2. Enter a configlet name, and select a NOS type (config style).
3. Select the section where you want the configlet to be rendered. Available choices depend on the selected config style. (As of Apstra release 4.0.0, OSPF for external routers is no longer supported. While OSPF configlets still appear in the Apstra GUI, they should not be used.)

For Cumulus, depending on the section selected, the following applies:

- Cumulus System: **Template Text** and **Negation Template Text** require the `net commit` command.
  - Cumulus Interface: `net clu` (NCLU) syntax is no longer used. Config line must match `/etc/network/interfaces` syntax exactly.
  - Cumulus File: All files are overwritten. Removing the configlet doesn't restore original content.
  - Cumulus FRR: all configlet content is appended to the `/etc/frr/frr.conf` file.
4. In the **Template Text** and **Negation Template Text** fields (as applicable), enter CLI commands. See Configlet examples in the Reference section. Avoid using shortened versions of commands. The exact commands may be validated, but after changes are pushed, they may be returned as-is in the rendered configuration.



**CAUTION:** Using a raw text editor (OSX TextEdit, Windows Notepad++) is critical. Hidden characters can cause unforeseen issues when the configlet is deployed.

**NOTE:** Instead of hard-coding data into a configlet, you can refer to a ["property set" on page 126](#) (key-value pairs). For an example, see the ["Arista NTP example" on page 1057](#) in the References section.

5. If **Negation Template Text** is required, enter the CLI commands to remove the configuration.
6. For File configlets, enter the filename in the **Filename** field.
7. To add another generator, click **Add a style** and enter details. (Tip: Configlets can contain syntax for multiple vendors. Create one single-purpose configlet with a generator for each vendor NOS type to include its own syntax.)
8. Click **Create** to add the configlet to the global catalog.

When you're ready to use the configlet in a blueprint, ["import" on page 615](#) it into the blueprint's catalog.

## Edit / Delete Configlet (Design)

### IN THIS SECTION

- [Edit Configlet | 125](#)
- [Delete Configlet | 125](#)

### Edit Configlet

Changing configlets in the design (global) catalog doesn't affect configlets in blueprint catalogs. If your intent is for a blueprint to use a modified configlet, see ["Edit \(Blueprint\) Configlet" on page 617](#) for the workflow.

1. From the list view (Design > Configlets) or the details view, click the name of the configlet to edit.
2. Make your changes (name, config style, section, template text, negation template text, filename, as applicable).
3. Click **Update** (bottom-right) to update the configlet in the global catalog and return to the list view.

### Delete Configlet

Deleting configlets in the design (global) catalog doesn't affect configlets in blueprint catalogs.

1. Either from the list view (Design > Configlets) or the details view, click the **Delete** button for the configlet to delete.

2. Click **Delete** to delete the configlet from the global catalog and return to the list view.

## Property Sets (Design)

### IN THIS SECTION

- [Property Set Overview | 126](#)
- [Create Property Set | 127](#)
- [Edit Property Set | 127](#)
- [Delete Property Set \(Design\) | 127](#)

### Property Set Overview

Property sets are collections of key-value pairs that you import into blueprint catalogs to use in configlets and IBA probes.

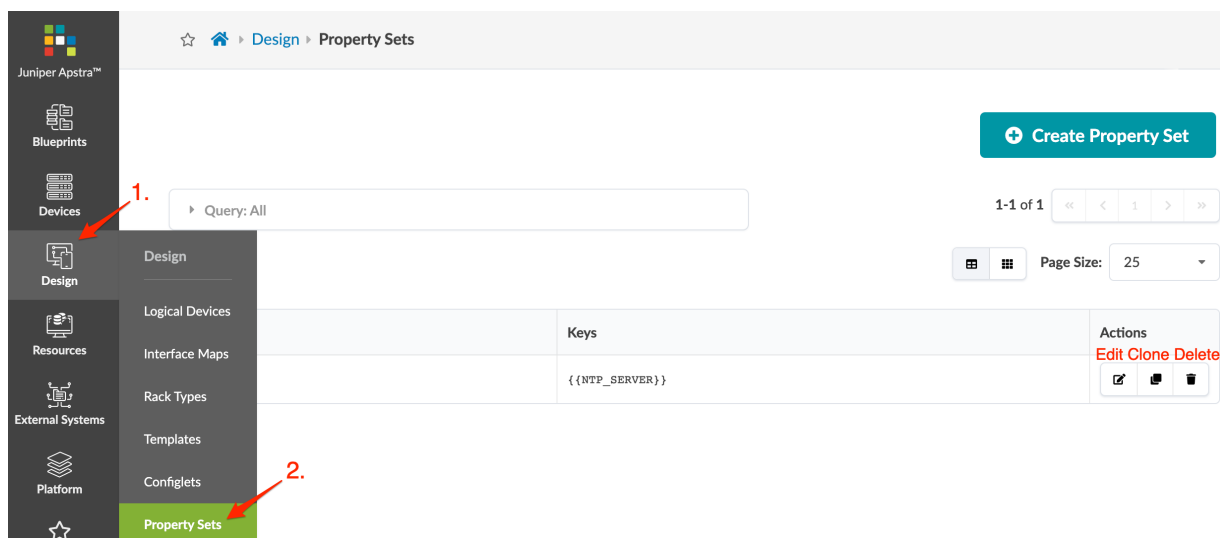
For example, instead of hard-coding values for NTP servers, SMTP servers, and syslog servers, you can define them in a property set, associate the property set with a configlet, then import both of them into a blueprint catalog.

For "[Intent-Based Analytics](#)" on [page 403](#), you can parameterize macro level SLAs or individual business units by applying property sets to IBA probe definitions.

Property sets include the following details:

- **Name** - Identifies the property set
- **Properties** - key-value pair(s)

From the left navigation menu, navigate to **Design > Property Sets** to go to property sets. You can create, clone, edit and delete property sets.



## Create Property Set

1. From the left navigation menu, navigate to **Design > Property Sets** and click **Create Property Set**.
2. Enter a name, then a key in the left property field and a value in the right property field. Do not add curly braces `{{ }}` to the key; they are added for you. To add another property, click **Add a Property**.
3. Click **Create** to create the property set and return to the list view.

## Edit Property Set

To prevent potentially unintended changes to existing blueprints, changes to property sets in the global catalog do not affect property sets in the blueprint catalog. If your intent is for a blueprint to use a modified property set, then you must re-import the revised property set into the blueprint.

1. From the left navigation menu, navigate to **Design > Property Sets** and click the name of the property set to edit.
2. Click the **Edit** button (top-right) and make your changes.
3. Click **Update** (bottom-right) to update the Property Set.

## Delete Property Set (Design)

If a property set is assigned to a "configlet" on page 119, it cannot be deleted.

1. Either from the list view (Design > Property Sets) or the details view, click the **Delete** button for the property set to delete.
2. Click **Delete** to delete the property set from the global catalog and return to the list view.

## TCP/UDP Port Aliases (Design)

### IN THIS SECTION

- [TCP/UDP Port Alias Overview | 128](#)
- [Create TCP/UDP Port Alias | 128](#)
- [Edit TCP/UDP Port Alias | 129](#)
- [Delete TCP/UDP Port Alias | 129](#)

### TCP/UDP Port Alias Overview

When you create a security policy and add rules for TCP or UDP protocols, a source port and destination port are specified. You can enter port numbers or you can create aliases ahead of time that can be entered instead of the port numbers. For example, you could create an alias with name *SSH* and a value of *22*.

From the left navigation menu, navigate to **Design > TCP/UDP Ports** to go to TCP/UDP ports. You can create, clone, edit and delete port aliases.

The screenshot shows the Juniper Apstra web interface. The left navigation menu is open, showing the 'Design' menu item highlighted with a red arrow labeled '1.'. The 'Design' menu is expanded, showing sub-items: Logical Devices, Interface Maps, Rack Types, Templates, Configlets, Property Sets, and TCP/UDP Ports. The 'TCP/UDP Ports' sub-item is highlighted with a red arrow labeled '2.'. The main content area shows the 'TCP/UDP Ports' page with a 'Create Port Alias' button, a search bar, and a table of port aliases. The table has columns for 'Value' and 'Actions'. The 'Value' column contains the number '1025'. The 'Actions' column contains links for 'Edit', 'Clone', and 'Delete'.

### Create TCP/UDP Port Alias

1. From the left navigation menu, navigate to **Design > TCP/UDP Ports** and click **Create Port Alias**.
2. Enter a name and one or more values.

3. Click **Create**. When you add a rule for TCP or UDP protocols to a security policy, the TCP/UDP port alias appears in the drop-down list.

## Edit TCP/UDP Port Alias

1. From the left navigation menu, navigate to **Design > TCP/UDP Ports** and click the **Edit** button for the port alias to edit.
2. Make your changes.
3. Click **Update** to update the TCP/UDP port alias and return to the list view.

## Delete TCP/UDP Port Alias

1. From the left navigation menu, navigate to **Design > TCP/UDP Ports** and click the **Delete** button for the port alias to delete.
2. Click **Delete** to delete the TCP/UDP port alias from the system.

## Tags (Design)

### IN THIS SECTION

- [Tags Overview | 129](#)
- [Create Tag \(Design\) | 131](#)
- [Edit Tag \(Design\) | 131](#)
- [Delete Tag \(Design\) | 131](#)

## Tags Overview

Tags add user-defined information to nodes and links. You can add tags to the following elements:

- Rack types (Design)
- Templates (Design)
- Connectivity Templates (Blueprints)
- Intent-Based Analytics (IBA) Probes (Blueprints)
  - ECMP Imbalance (External Interfaces) probe

- Total East/West Traffic probe
- Critical Services: Utilization, Trending, Alerting probe (new in 4.0.1)
- Leafs Hosting Critical Services: Utilization, Trending, Alerting probe (new in 4.0.1)

For example, you assign servers and external routers the **generic** port role in logical devices (new in version 4.0), and then tag them with their specific roles when you design rack types and templates. When you create a blueprint, tags from the relevant design elements are embedded into the tag section of the blueprint catalog.

Changes you may subsequently make to tags in the design elements do not affect the blueprint that had previously used those tags. If you want a blueprint to use revised tags from a design element, you can ["import " on page 622](#) them.

You can ["export" on page 622](#) tags that you created in a blueprint to the global catalog (as long as they have a unique name) where they can be used in subsequent design elements.

Tags include the following details:

- **Name** - Case-insensitive. They must be unique across all tags defined in the design.
- **Description** - Optional field to add any details (for example, server roles, external router roles or customer name).

From the left navigation menu, navigate to **Design > Tags** to go to tags in the global catalog. Four tags (Bare Metal, Firewall, Hypervisor, Router) are predefined for you. You can create, clone, edit and delete tags in the global catalog.

The screenshot shows the Juniper Apstra interface. The left navigation menu is open, and the 'Design' menu item is selected. The 'Tags' sub-item is highlighted in green. A red arrow labeled '1.' points to the 'Design' menu item, and another red arrow labeled '2.' points to the 'Tags' sub-item. The main content area shows the 'Design > Tags' page. It includes a 'Create Tag' button, a search bar with 'Query: All', and a table of predefined tags.

Description	Actions
Bare Metal Servers.	<a href="#">Edit</a> <a href="#">Clone</a> <a href="#">Delete</a>
L2 and L3 Firewalls.	<a href="#">Edit</a> <a href="#">Clone</a> <a href="#">Delete</a>
Hypervisor/Compute Nodes.	<a href="#">Edit</a> <a href="#">Clone</a> <a href="#">Delete</a>
External Routers, Virtual Routers, etc.	<a href="#">Edit</a> <a href="#">Clone</a> <a href="#">Delete</a>

## Create Tag (Design)

1. From the left navigation menu, navigate to **Design > Tags** and click **Create Tag**.
2. Enter a name and (optional) description.
3. Click **Create** to create the tag and return to the list view.

## Edit Tag (Design)

You cannot change tag names directly; You can only change tag descriptions.

**NOTE:** You can change a tag name indirectly by creating a tag with the preferred name, applying the tag to the rack type or template, then deleting the tag with the original name from the rack type or template (then deleting the original tag).

To change a tag name indirectly:

1. Create a tag with the preferred name.
  2. Apply the tag to the rack type or template.
  3. Delete the tag with the original name from the rack type or template.
  4. Delete the original tag.
1. Either from the list view (Design > Tags) or the details view, click the **Edit** button for the tag to change.
  2. Change the description.
  3. Click **Update** to update the tag description and return to the list view.

## Delete Tag (Design)

Deleting a tag from the design (global) catalog does not affect rack types and templates that have previously been assigned the tag.

1. Either from the list view (Design > Tags) or the details view, click the **Delete** button for the tag to delete.
2. Click **Delete** to delete the tag and return to the list view.

# Devices

## IN THIS SECTION

- [Managed Devices \(Devices\) | 132](#)
- [Device Configuration Lifecycle | 140](#)
- [Add Device | 150](#)
- [Deploy Device | 150](#)
- [Drain Device Traffic | 152](#)
- [Upgrade Device NOS | 158](#)
- [Remove Device | 166](#)
- [Reuse Device | 168](#)
- [Device AAA Support | 169](#)
- [Telemetry \(Devices\) | 171](#)
- [Agents \(Devices\) | 184](#)
- [Agent Profiles \(Devices\) | 260](#)
- [Packages \(Devices\) | 262](#)
- [OS Images \(Devices\) | 264](#)
- [Apstra ZTP \(Devices\) | 267](#)
- [Device Profiles \(Devices\) | 300](#)

## Managed Devices (Devices)

### IN THIS SECTION

- [Managed Devices Overview | 133](#)
- [Acknowledge Device\(s\) | 134](#)
- [Set Admin State on Devices | 134](#)
- [Update User Config | 135](#)

- Edit System | 135
- Modular Devices and Device Profiles | 137
- Edit Pristine Config | 138
- Update Pristine Config from Device | 139
- Delete System(s) | 139
- Managed Device Telemetry | 140

## Managed Devices Overview

Devices with installed "device agents" on page 184 appear in the managed devices list. From the left navigation menu, navigate to **Devices > Managed Devices** to go to the managed devices list view.

Juniper Apstra™

☆ 🏠 > Devices > Managed Devices

Query: All

1-5 of 5

Columns (11/12) Page Size: 25

Managed Devices

all selected only unselected only

Key	Device Profile	Operation Mode	Management IP	Apstra Version	Hostname	OS	Acknowledged?	State	Blueprint	Comms
2	Arista DCS-7280SR-48C6	FULL CONTROL	10.85.68.149	AOS_4.0.1_OB.1044	jtac-rack1-001-leaf1	EOS 4.24.5M	✓	IS-ACTIVE	JTAC-Arista-Blueprint	👤
3	Arista DCS-7280SR-48C6	FULL CONTROL	10.85.68.150	AOS_4.0.1_OB.1044	jtac-rack1-001-leaf2	EOS 4.24.5M	✓	IS-ACTIVE	JTAC-Arista-Blueprint	👤
4	Arista DCS-7050CK3-32S	FULL CONTROL	10.85.68.151	AOS_4.0.1_OB.1044	jtac-rack2-001-leaf1	EOS 4.24.5M	✓	IS-ACTIVE	JTAC-Arista-Blueprint	👤
5	Arista DCS-7050CK3-32S	FULL CONTROL	10.85.68.152	AOS_4.0.1_OB.1044	spine1	EOS 4.24.5M	✓	IS-ACTIVE	JTAC-Arista-Blueprint	👤
6	Arista DCS-7050CK3-32S	FULL CONTROL	10.85.68.153	AOS_4.0.1_OB.1044	spine2	EOS 4.24.5M	✓	IS-ACTIVE	JTAC-Arista-Blueprint	👤

From **Managed Devices**, you can perform tasks, such as acknowledging devices (to bring them under Apstra management), setting admin states, updating user configuration, changing associated device

profiles or admin states, and deleting devices.

1-5 of 5

Columns (11/12) Page Size: 25

Filter selected by ☒ all ☐ selected only ☐ unselected only

<input checked="" type="checkbox"/>	Device Key	Device Profile	Operation Mode	Management IP	Apstra Version	Hostname	OS	Acknowledged?	State	Blueprint	Comms
<input checked="" type="checkbox"/>	SS-...	Arista DCS-7280SR-48C6	FULL CONTROL	10.85.68.149	AOS_4.0.1_OB.1044	jtac-rack1-001-leaf1	EOS 4.24.5M	<input checked="" type="checkbox"/>	IS-ACTIVE	JTAC-Arista-Blueprint	
<input checked="" type="checkbox"/>	SS-...	Arista DCS-7280SR-48C6	FULL CONTROL	10.85.68.150	AOS_4.0.1_OB.1044	jtac-rack1-001-leaf2	EOS 4.24.5M	<input checked="" type="checkbox"/>	IS-ACTIVE	JTAC-Arista-Blueprint	
<input checked="" type="checkbox"/>	JPI-...	Arista DCS-7050CK3-32S	FULL CONTROL	10.85.68.151	AOS_4.0.1_OB.1044	jtac-rack2-001-leaf1	EOS 4.24.5M	<input checked="" type="checkbox"/>	IS-ACTIVE	JTAC-Arista-Blueprint	
<input checked="" type="checkbox"/>	JPI-...	Arista DCS-7050CK3-32S	FULL CONTROL	10.85.68.152	AOS_4.0.1_OB.1044	spine1	EOS 4.24.5M	<input checked="" type="checkbox"/>	IS-ACTIVE	JTAC-Arista-Blueprint	
<input checked="" type="checkbox"/>	JPI-...	Arista DCS-7050CK3-32S	FULL CONTROL	10.85.68.153	AOS_4.0.1_OB.1044	spine2	EOS 4.24.5M	<input checked="" type="checkbox"/>	IS-ACTIVE	JTAC-Arista-Blueprint	

**NOTE:** For more information about managing devices in the Apstra environment see the device guides in the Guides section.

## Acknowledge Device(s)

Acknowledging devices puts them in the **Ready** state and signals the intent to have Apstra manage them.

1. From the left navigation menu, navigate to **Devices > Managed Devices** and select the device(s) to be managed by Apstra.
2. Above where you just clicked, click the checkmark to **Acknowledge** the selected device(s).
3. Click **Confirm** to acknowledge the device(s) and return to the list view. The **Acknowledged?** field for the device changes to a green checkmark and the device state changes to **OOS-READY**.

## Set Admin State on Devices

1. From the left navigation menu, navigate to **Devices > Managed Devices** and select the device(s) to update.
2. Click the button for the state to change the selection(s) to (MAINT, NORMAL, DECOMM).



### CAUTION:

- If you are decommissioning a device, after setting the admin state to **DECOMM**, you must uninstall the device agent. See the device guide for ["removing a device" on page 166](#). If you would subsequently like Apstra to manage the device you can ["add the device" on page 150](#) back.

- If you're ["upgrading a device network operating system" on page 158](#) the admin state must be set to **NORMAL**.

3. Click **Confirm** to set the admin state and return to the list view.

### Update User Config

1. From the left navigation menu, navigate to **Devices > Managed Devices** and select the device(s) to update.
2. Click the **Update user config** button, then to override device profile, admin state, and/or location, check the appropriate boxes.
3. Click **Confirm** to update the user config and return to the list view.

### Edit System

There may be cases where you need to change managed device details. For example, if you want to set up a hyperloop interface to ["enable VXLAN Routing on a Cumulus device equipped with Tomahawk" on page 845](#) so it supports Routing In and Out of Tunnels (RIOT) you must change the device profile used in the managed device. For another example, see the Modular Devices and Device Profiles section below.

1. To edit a managed device, from the left navigation menu, navigate to **Devices > Managed Devices** and select the device key for the device to update.

☆
🏠
›
Devices
›
Managed Devices

›
Query: All

<input type="checkbox"/>	Device Key ⇅	Device Profile ⇅	Operation Mode ⇅
0 selected			
<input type="checkbox"/>	HS	Arista DCS-7504N 4x7500R-36CQ	<b>FULL CONTROL</b>

2. Click the **Edit** button (top-right) and select a different device profile, admin state, and/or location, as applicable.

## Edit System ✕

Device Profile <sup>\*</sup>

Arista DCS-7504N 4x7500R-36CQ ✕

Admin State <sup>\*</sup>

☐ DECOMM ☐ MAINT ☐ NORMAL

Location

Update

3. Click **Update** to update the device and return to the list view.

# Modular Devices and Device Profiles

Multiple "device profiles" on page 300 may exist for one modular device model to represent different line card configurations.

☆
🏠
»
Devices
»
Device Profiles

+
Create Device Profile

Query: Name == "7504"

1-2 of 2

Page Size: 25

Name	Manufacturer	Hardware Model	Modular?	OS Family	OS Version	Actions
Arista DCS-7504N 4x7500R-36CQ	Arista	DCS-7504N	yes	EOS	4.(18 20 21 22 23 24)..*	🔍 📄 🗑️
Arista DCS-7504N 4x7500R-36Q	Arista	DCS-7504N	yes	EOS	4.(18 20 21 22 23 24)..*	🔍 📄 🗑️

Apstra selects the first device profile, based on the selector model field, that it encounters that matches the model of the device chassis (DCS-7504N for example).

☆
🏠
»
Devices
»
Managed Devices

+
Stock Device

Query: All

1-1 of 1

Page Size: 25

	Device Key	Device Profile	Operation Mode	Management IP	AOS Version	Hostname	Location	OS	Acknowledged?	State	Blueprint	Comms
0 selected	HSH	Arista DCS-7504N 4x7500R-36CQ	FULL CONTROL	172.20.140.6	AOS_3.30.1_0B.88	localhost		EOS 4.22.3M	🚫	OOS-QUARANTINED	Not assigned	🟢

Apstra does not match device profiles based on line card configuration.

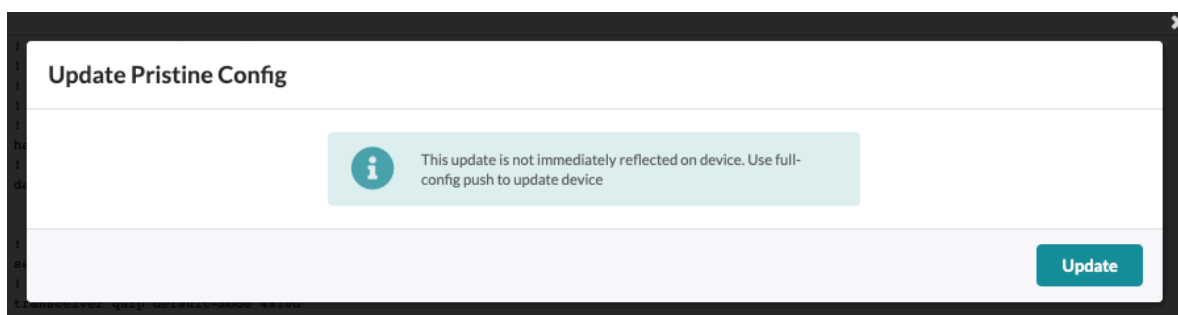
If you're using a modular device in your network, check that the correct device profile is associated with it. If it's not, edit the system to update the device profile (see Edit System section above for details). You



## Update Pristine Config from Device

It is also possible to update (customize) the pristine config by copying the running device configuration. To use this function, the device must be removed from a blueprint and out of service (OOS-READY or OOS-MAINT). Once you have your device in OOS state, make sure that all the necessary changes are done manually via CLI on the device before updating pristine config from the device.

1. Unassign the device from the blueprint.
2. Make any necessary changes to the running device configuration via CLI.
3. From the left navigation menu in the Apstra GUI, navigate to **Devices > Managed Devices** and select the **Device Key** of the device.
4. Click the **Pristine Config** tab (top-left), then click the **Update From Device** button (top-right).
5. Click **Update** to update Pristine Config from the device.



After the "Update From Device" operation, verify the Pristine Config again. As you have copied the running config of the device in OOS state which should be the device "Discovery-1" config, it may include additional configuration such as interface "speed" commands. You can edit Pristine Config again and delete those additional configuration manually. Contact ["Juniper Suuport" on page 777](#) for assistance as needed.

**NOTE:** Beginning in Apstra 4.0.1, if a device is not assigned to a blueprint, you can update the device pristine configuration on a device by clicking the **Collect Pristine Config** button for the device system agent in **Devices > System Agents > Agents**.

## Delete System(s)



**CAUTION:** Devices that have been acknowledged cannot simply be deleted - as there is still an active agent on the device communicating with the Apstra server, the devices would re-appear within seconds. See the device guides for the complete workflow for ["removing a device" on page 166](#) from Apstra management.

Decommission devices and uninstall their agents before deleting devices from the managed devices list.

1. From the left navigation menu, navigate to **Devices > Managed Devices** and select the device(s) to remove.
2. Click the **Delete system(s)** button, then click **Confirm** to remove the device(s) and return to the list view.

## Managed Device Telemetry

1. From the left navigation menu, navigate to **Devices > Managed Devices** and select the **Device Key** of the device.
2. Click the **Telemetry** tab (top-left), then click a tab to see the relevant telemetry.

From the **Config** tab, you can **Apply Full Config** or **Accept Changes**, as applicable. You can also ["apply full config" on page 665](#) from the blueprint that it's used in. For more information about when to apply a full config, see Configuration Deviation in ["Device Configuration Lifecycle" on page 140](#).

## Device Configuration Lifecycle

### IN THIS SECTION

- [Terminology | 140](#)
- [Configuration Stages: Overview | 141](#)
- [Configuration Stages: Detail | 143](#)
- [Configuration Deviations | 147](#)
- [Device Offline \(Unavailable\) | 148](#)
- [Manually Apply Full Config | 148](#)
- [Deploy Modes | 148](#)



**CAUTION:** A good understanding of the Apstra device configuration lifecycle (from the moment it is on-boarded to the moment the device is decommissioned) is essential. We strongly recommend that you fully understand the following content before working with devices in the Apstra environment.

## Terminology

The following terminology is used to identify configuration stages:

Config	Description
Pristine Config	Consists of pre-existing config plus config added during agent installation. Normally, the pristine config does not change throughout the device's lifecycle.
Discovery 1 Config	Initial basic configuration is added to the device when it is <i>acknowledged</i> . This includes enabling LLDP on all interfaces.
Discovery 2 Config	Additional basic configuration is added to the device when it is assigned to a blueprint and deploy mode is <i>ready</i> . This includes device hostnames, interface descriptions and port speed / breakout config.
Service Config	When the deploy mode is set to <i>deploy</i> , configuration that's required for the Apstra environment is added. <i>Service Config</i> is Discovery 1 config, Discovery 2 config and this additional config combined.
Rendered Config	Complete Apstra-rendered configuration for the device, per the Apstra Reference Design.
Incremental Config	Staged changes. The configuration that will be applied when the staged changes are committed.
Golden Config	A successful commit is followed with a new collection of the running config. This is called the Golden Config, and serves as Intent: Running configuration is continuously matched against this Golden config. When a deployment fails, Golden Config is unset.

## Configuration Stages: Overview

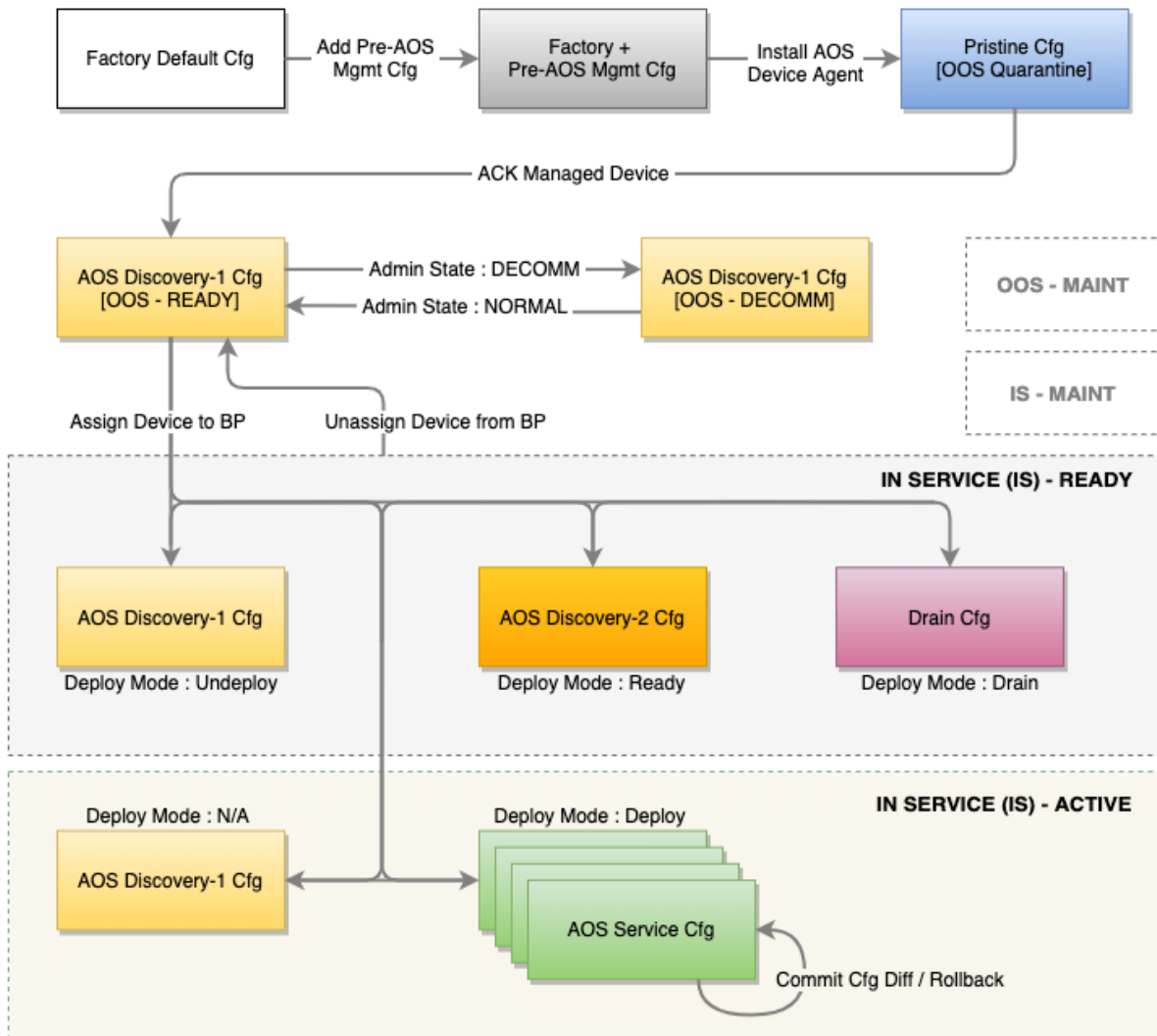
The following table describes the various config events and resulting config and device states and states of a device as it exists within a blueprint:

Event	Resulting Device Configuration	Resulting Apstra Managed Device State	Apstra Blueprint Deployment Mode
New Device	Factory Default Configuration	N/A	Not Assigned
Add Pre-Apstra [mgmt] Configuration to device	Factory + Pre-Apstra	N/A	Not Assigned
Install Apstra Device System Agent	<b>Pristine Config:</b> Factory + Pre-Apstra + Agent Install config	OOS-QUARANTINED	Not Assigned

*(Continued)*

Event	Resulting Device Configuration	Resulting Apstra Managed Device State	Apstra Blueprint Deployment Mode
Acknowledge Device	<b>Discovery 1:</b> Pristine, plus Interfaces Enabled	OOS-READY	Not Assigned
Assign Device to blueprint (no deploy)	<b>Discovery 2:</b> Discovery 1, plus various basic config	IS-READY	Ready
Deploy Device	<b>Service Config:</b> Discovery 2, plus full Apstra-Rendered config	IS-ACTIVE	Deploy
Add/Commit Incremental Configuration	Delta of resulting config changes from blueprint modifications	IS-ACTIVE	Deploy
Drain Device	"Drain" Configuration is added	IS-READY	Drain
Undeploy Device	Apstra-rendered config is removed	IS-READY	Undeploy
Unassign Device	Discovery 1 config is re-applied	OOS-READY	Not Assigned

## AOS Device Lifecycle



Note: This diagram does not include the flows for 'Admin State : MAINT'. When device admin state is set to MAINT, device state will be either 'IN SERVICE (IS) - MAINT' or 'OUT OF SERVICE (OOS) - MAINT' but the device config will not be changed.



**CAUTION:** If there's any configuration on the device before you install an agent it becomes part of the Pristine Config and therefore becomes part of the devices' entire configuration lifecycle. Any corrections that you need to make will be service impacting.

## Configuration Stages: Detail

### IN THIS SECTION

● [New Device \(Factory Default\) | 144](#)

- Add Pre-Apstra Config (User-required) | 144
- Install Agent (Pristine) | 144
- Acknowledge Device (Discovery 1 / Ready) | 145
- Assign Device (Discovery 2 / Ready) | 145
- Deploy Device (Rendered / Active) | 146
- Stage Device Update (Incremental / Active) | 147
- Commit Device Again (Rendered-Updated / Active) | 147

## New Device (Factory Default)

The lifecycle of a device begins with the **factory default** configuration stage.

### Add Pre-Apstra Config (User-required)

Certain minimum base configuration must be included in the entire config lifecycle, such as that required for agent installation or device connectivity. You must configure management IP connectivity between devices and the Apstra server out-of-band (OOB). Configuring it in-band is not supported and could cause connectivity issues when changes are made to the blueprint.

This **User-required** config can be bootstrapped with "[Apstra ZTP](#)" on page 267, or added with scripts (or other methods).



**CAUTION:** Adding configuration to the device at the pre-Apstra stage should be limited to changes required for connectivity or Apstra Agent installation, or any config that is known to be required **throughout the device's lifecycle**, for example Banners or NTP / SNMP / syslog server IP addresses. You can add required configuration that is not rendered by Apstra with "[configlets](#)" on page 119.

### Install Agent (Pristine)

When an Apstra device agent is installed on a device (or in the case of an offbox agent, installed server-side) the device connects and registers to Apstra in the **Quarantined** state. A partial config is applied and any "Pre-Apstra Config" that has already been added becomes part of the **Pristine configuration**. This pristine configuration is the basis for all subsequent device configuration.



**CAUTION:** For Cumulus Linux, enabling configuration services replaces all config in `/etc/network/interfaces`.

### Acknowledge Device (Discovery 1 / Ready)

Acknowledging a device puts it in the **Ready** state and signals the intent to have Apstra manage the device. In this **Discovery 1** stage minimal base configuration essential to Apstra agent operation is added to the pristine config. Discovery 1 applies a *complete* configuration (a.k.a. "Full config push"), overwriting all existing configuration to ensure config integrity.

- All interfaces are rendered with interface speeds for the assigned Device Profile.
- All interfaces are *no shutdown* to allow you to view LLDP neighbor information.
- All interfaces are moved to L3 mode (default) to prevent the device from participating in the fabric.
- Cumulus: DHCP relay configuration is removed and wiped.
- Cumulus: Quagga configuration is removed and wiped.
- Cumulus: Hostname is learned when agent first starts up, or re-used if agent ran previously. Hostnames can also be learned via DHCP before the device is acknowledged.

**NOTE:** Devices that have been acknowledged cannot simply be deleted from the Apstra environment. As there is still an active agent on the device communicating with the Apstra server, devices would re-appear within seconds. For details on how to remove a device from Apstra, see the device guide for ["removing a device" on page 166](#).

### Assign Device (Discovery 2 / Ready)

Assigning a device to a blueprint and setting its Deploy Mode to **Ready** puts it in the **Discovery 2** configuration stage. The device has been staged, but not yet committed (deployed) to the active blueprint. Discovery 2 applies a *complete* configuration (aka. "Full config push") to ensure config integrity. The discovery2 configuration brings up network interfaces and configures interface descriptions and validates telemetry, such as LLDP, to ensure it is properly wired and configured. This configuration is non-disruptive to other services in the fabric. Links are up, but they are configured in L3-mode to prevent STP/L2 operations.

- Hostname is configured per blueprint intent.
- All interface descriptions are changed per blueprint intent.

- Interfaces are rendered with blueprint interface speeds.
- No routing or BGP is configured.
- No L3 information is configured on interfaces.
- Fabric MTU is modified for spines to 9050 bytes.
- Cumulus: Quagga configuration is still empty (`etc/quagga/Quagga.conf`), but the file is created if it does not exist
- Cumulus: DHCP configuration is not modified

### Deploy Device (Rendered / Active)



**CAUTION:** The first time a device is assigned, the Deploy Mode is set to "Deployed" and the blueprint is committed (**full config push** is triggered for the device) effectively overwriting the complete running config with the Pristine Configuration then adding the full rendered Apstra Configuration. Any config that is not part of the Apstra-rendered config is discarded.

When a device is committed, it becomes **Active**, and Apstra deploys the service configuration, moving the device into the **Rendered** configuration stage. Rendered config contents are derived from the pristine config, selected reference design/topology, NOS, and device model. The first rendered config applies a *complete* configuration (removing all existing configuration from the Apstra server per Jinja) to ensure configuration integrity. This is the full end-state of Apstra. A full configuration has been pushed, all interfaces are running, and routing within IP fabric is configured. Full configuration rendering, intent-based telemetry, and standard service operations occur here.

- Hostname is configured per blueprint intent.
- All interface descriptions are changed per blueprint intent.
- Interfaces are rendered with blueprint interface speeds.
- Interface VLANs, LAGS, MLAG, VXLAN, etc are managed.
- All L3 information is rendered.
- BGP configuration is fully rendered for all BGP peering information.
- DHCP configuration is configured for any required DHCP relay agents.
- Cumulus: FRR configuration is fully rendered (`/etc/frr/frr.conf`), including all BGP peering information.
- The device is added to the graph database.

After the full configuration is successfully deployed to the Device Apstra takes a snapshot of the Device Configuration (e.g. show running-config) and store it as the **Golden Configuration**.



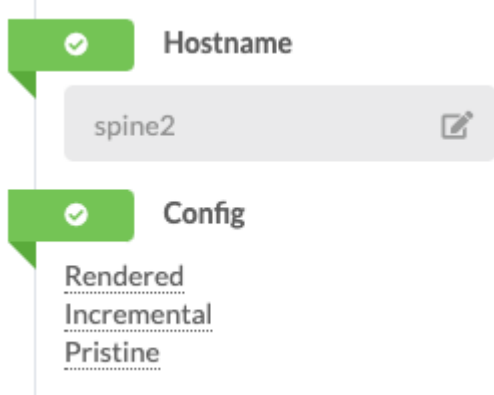
**CAUTION:** Adding extra configuration to Apstra at this time will result in a configuration deviation anomaly, a difference between the current Device Configuration and the stored **Golden Configuration**. Apstra will fail subsequent deployment tasks until the deviation is resolved. Correct the anomaly to proceed.

To see the rendered config file after committing the blueprint, select the device in the **Active** blueprint and click **Config** (right-side).

A running configuration can be modified in multiple ways. To modify a config that is not part of the reference design, use ["configlets" on page 119](#).

### Stage Device Update (Incremental / Active)

Staging changes to a running blueprint creates an **Incremental** configuration. You can preview the incremental config before committing the changes to the network. From the staged blueprint, select the device, and click **Incremental** in the **Config** section (lower-right). (Config previews for **Rendered** and **Pristine** are also accessible from here). When the changes have been committed, the Incremental Config is empty.



### Commit Device Again (Rendered-Updated / Active)

When a change is committed to a blueprint that affects the device's configuration, a partial config updates the rendered config.

## Configuration Deviations

After each **successful** config deploy Apstra collects the running config and stores it internally as the **Golden** configuration. Intent is the cornerstone of Apstra. As such, any difference between the actual

running config and this Golden config results in a Config Deviation Anomaly on the blueprint's Dashboard. The Golden config is updated every time config is successfully applied to a device.

Some important points to know:

- Golden Config is updated upon each *successful* configuration deployment
- When config deployment fails for some reason, Golden config is not set. This means both a config deviation and deployment failure anomaly is raised.
- Running configuration telemetry is continuously collected and matched against the Golden config. Any difference results in a Deviation anomaly.
- Configuration Anomalies can be 'suppressed' using the "Accept Changes feature". This does **NOT** mean the change is added to Golden config or Intent.

See ["Anomalies \(Service\)" on page 671](#) for details.

### Device Offline (Unavailable)

A managed device (one that has been acknowledged) that is not connected to the Apstra server is in the **unavailable** state. A device could be offline if the device agent interface is offline, if the service is not running, or if a network connectivity error occurs.

### Manually Apply Full Config

The **Discovery 1** and **Deploy Device** configuration stages initiate full config pushes. In rare cases, you may need to manually apply a full config push. For example, if the required config is not in place for a blueprint with NX-OS devices that require TCAM carving, the device config will fail. The TCAM config error must be corrected, followed by manually pushing a full config.

**NOTE:** Perform a full configuration push with the utmost caution, as it is very likely to impact all services running on the box. Exact impact depends on changes being pushed. Also note **all** Out of Band changes are overwritten upon a full push.

### Deploy Modes

Managed devices in blueprints can be in one of several modes. ["Devices \(Staged\)" on page 423](#) for steps for changing deploy modes.

Not Set	The initial state of a device. The device is not active in the fabric.
Deploy	A deployed device is an active device in the fabric.

Ready	When a device is assigned to a blueprint it is in ready mode; discovery 2 configuration is added (hostnames, interface descriptions, port speed / breakout configuration). It is not yet active in the fabric. Changing from deploy to ready removes Apstra-rendered configuration.
Drain	<p>Draining a device for physical maintenance enables it to be taken out of service gracefully without impacting existing TCP flows. Depending on the device being drained, Apstra uses one of two methods:</p> <p>For L2 Servers</p> <ul style="list-style-type: none"> <li>• MLAG peer-links port channels and bond interfaces on any NOS are not changed.</li> <li>• For Arista EOS, Cisco NX-OS, all interfaces towards L2 servers in the blueprint are shutdown.</li> <li>• For Cumulus, all bond interfaces towards L2 servers in the blueprint are deleted and member ports are removed from /etc/network/interfaces. As a result LAG/MLAG anomalies are generated.</li> </ul> <p>For Network L3 Switches</p> <p>Use Inbound/Outbound route-maps 'deny' statements to block any advertisements to 0.0.0.0/0 le 32.</p> <p>This allows existing L3 TCP flows to continue without interruption. After a second or two, the TCP sessions should be re-established by the src/dst devices, or they should negotiate a new TCP port. The new TCP port forces the devices to be hashed onto a new ECMP path from the list of available links. Since no ECMP routes to the destination are available in the presence of a route map, the traffic does not flow through the device that is in maintenance mode. The device is effectively <b>drained</b> of traffic and can be removed from the fabric (by changing Deploy mode to Undeploy).</p> <p>While TCP sessions drain (which could take some time, especially for EVPN blueprints) BGP anomalies are expected. When configuration deployment is complete, the temporary anomalies are resolved. See the device draining guide &lt;device_drain&gt; for more information.</p>
Undeploy	Undeploying a device removes the complete service configuration. If a device is carrying traffic it is best to put it in drain mode first (and commit the change) before undeploying the device.

## Add Device

Before working with devices, have a good understanding of the ["Device configuration lifecycle" on page 140](#).

If you're adding a device to replace a decommissioned faulty one (for RMA) and you plan to use the same management IP address, make sure that you ["decommission" on page 166](#) the original device before you assign the new device. Apstra software expects each device to have a unique management IP address.

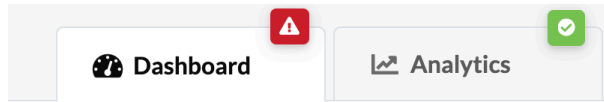
1. ["Install" on page 184](#) device agent(s). You can navigate to **Devices > Managed Devices** to see that they are in the out-of-service quarantined state.
2. ["Acknowledge" on page 132](#) device(s). The state changes to out-of-service ready, and they are now managed by the Apstra software.
3. After creating the ["blueprint" on page 400](#), ["assign interface map\(s\)" on page 422](#) (device profiles) to leafs and spines.
4. ["Assign device system ID\(s\)" on page 423](#) to the devices and confirm that deploy mode(s) are set to **Deploy**. This puts them in the in-service ready state.
5. ["Commit" on page 648](#) the device assignment(s) to the blueprint to put them in the active state.

## Deploy Device

### IN THIS SECTION

- [Juniper Junos | 151](#)

When you set the deploy mode of a device to **Deploy** and commit that change in the blueprint, the **full** Apstra-rendered configuration is applied to the device making it an active device in the fabric. Immediately after deploying devices, while intent is being verified, several anomalies may be raised, which appear on the dashboard. One could say Apstra 'assumes' an anomaly until it is verified to be OK. At this point, telemetry data is being checked against Intent, and if they match, the anomalies are cleared. This can take a fair amount of time in some cases, BGP sessions that come up and routes that are being advertised, for example.



## Deployment Status

### Service Config

✓ 4 SUCCEEDED

⌚ 0 PENDING

! 0 FAILED

The running configuration is continuously collected and compared with the golden configuration. Protocol related anomalies like BGP or LLDP are only raised if devices at both ends are deployed.

It is common to have a committed blueprint without any deployed devices. You can deploy devices as and when required, be it in batches, one by one, or all in one go.

**NOTE:** For information about other deploy modes, see ["Device Configuration Lifecycle" on page 140](#).

Implications of deploying a device can vary across different vendors. Below are known NOS characteristics that result in differences in the way anomalies are raised in the Apstra environment.

### Juniper Junos

Depending on the device vendor, deploying devices can have different implications. Juniper Junos devices have the following characteristics with regards to raising anomalies:

- `show interface` commands don't list interfaces on ports that do not have a transceiver plugged in. This means *Interface Down* anomalies cannot be raised for these interfaces. Such interfaces can be recognized using the `show virtual-chassis vc-port`, and have a status of 'Absent'.
- If a virtual network endpoint is configured on a leaf interface, Apstra expects an EVPN type 3 route for that interface. If this interface is down, Junos does not advertise the RT-3, resulting in a "Missing Route" anomaly in Apstra. If this anomaly is undesirable we recommend that you remove the interface from the VN until the interface is up.

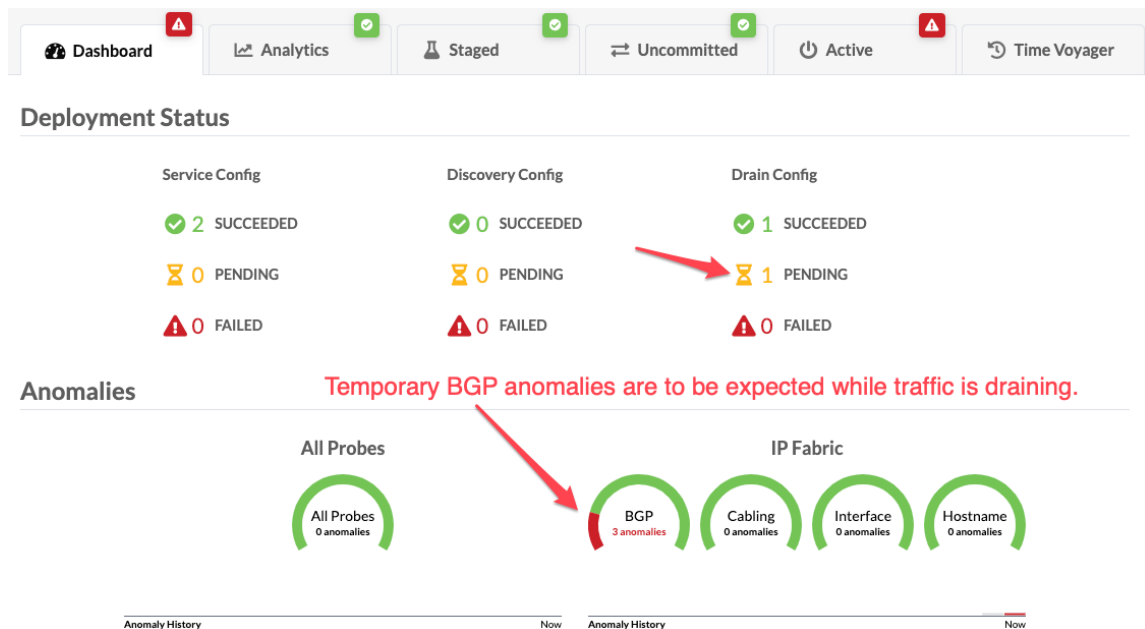
## Drain Device Traffic

### IN THIS SECTION

- Upgrade MLAG Pair | 154

You can gracefully take devices out-of-service for maintenance (or decommissioning) by draining them of traffic.

1. From the blueprint, navigate to **Staged > Physical > Build > Devices** and change the "deploy mode" on [page 423](#) on the device to **Drain**.
2. Click **Uncommitted** and commit the staged change to activate it.
3. While TCP sessions drain (which could take some time, especially for EVPN blueprints) BGP anomalies are expected. You can monitor draining progress from various locations in the Apstra GUI. When drain configuration is complete, the temporary anomalies are resolved.
  - You can monitor drain status from the **Deployment Status** section of the blueprint dashboard (Drain Config).



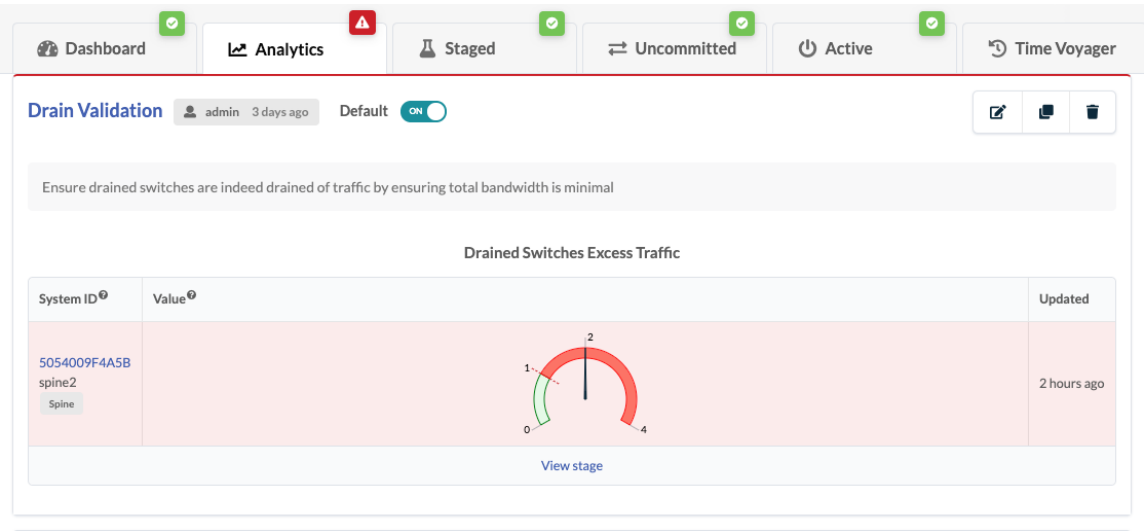
- You can monitor drain status from **Active > Physical** in the **Status** panel (Deployment Status: Drain).

The screenshot shows the 'Active' status panel with the 'Physical' tab selected. The 'Status' panel on the right displays a list of anomalies and deployment status. A red arrow points to the 'Deployment Status: Drain' entry, which shows a status of '1/0/0'.

Category	Status
Anomalies: All Services	0
Anomalies: BGP	0
Anomalies: Cabling	0
Anomalies: Config	0
Anomalies: Hostname	0
Anomalies: Interface	0
Anomalies: LAG	0
Anomalies: Liveness	0
Anomalies: MLAG	0
Anomalies: Probes	0
Anomalies: Route	0
Deploy Mode	3/0/1/2
Deployment Status: Discovery	0/0/0
Deployment Status: Drain	1/0/0
Deployment Status: Service	3/0/0
Traffic Heat	0

- If you "[instantiate](#)" on [page 403](#) the predefined **Drain Validation** dashboard, you can monitor drain status from **Analytics > Dashboards**. (If you set the dashboard as default, you can see it on the blueprint dashboard as well as on the analytics dashboard). In the image below, traffic is in the

process of draining.



After performing device maintenance, change the deploy mode back to **Deploy** and "[commit](#)" on [page 648](#) the change to bring the device back into active service.

Upgrade MLAG Pair

If MLAG is supported across two Network Operating Systems (NOS) for a vendor, you can minimize traffic downtime by draining the device when you upgrade the MLAG pair. This example is for the Cumulus NOS upgrade workflow. You'll drain and upgrade one device at a time as part of an MLAG pair.

It is to exhibit similar steps in consultation with other vendors can be performed to utilise Apstra device **Draining** functionality to upgrade MLAG pair. It isn't free from any traffic downtime but helps liaise minimum traffic disruption possible.

- 1. From the blueprint, navigate to **Staged > Physical > Node > (Node\_name) > Node > Device > Deploy Mode**, set deploy mode to **Drain** and commit the blueprint. The first upgrade is done on a switch with

a **secondary clag** role. To confirm this, log into the switch and confirm that switch role is secondary.

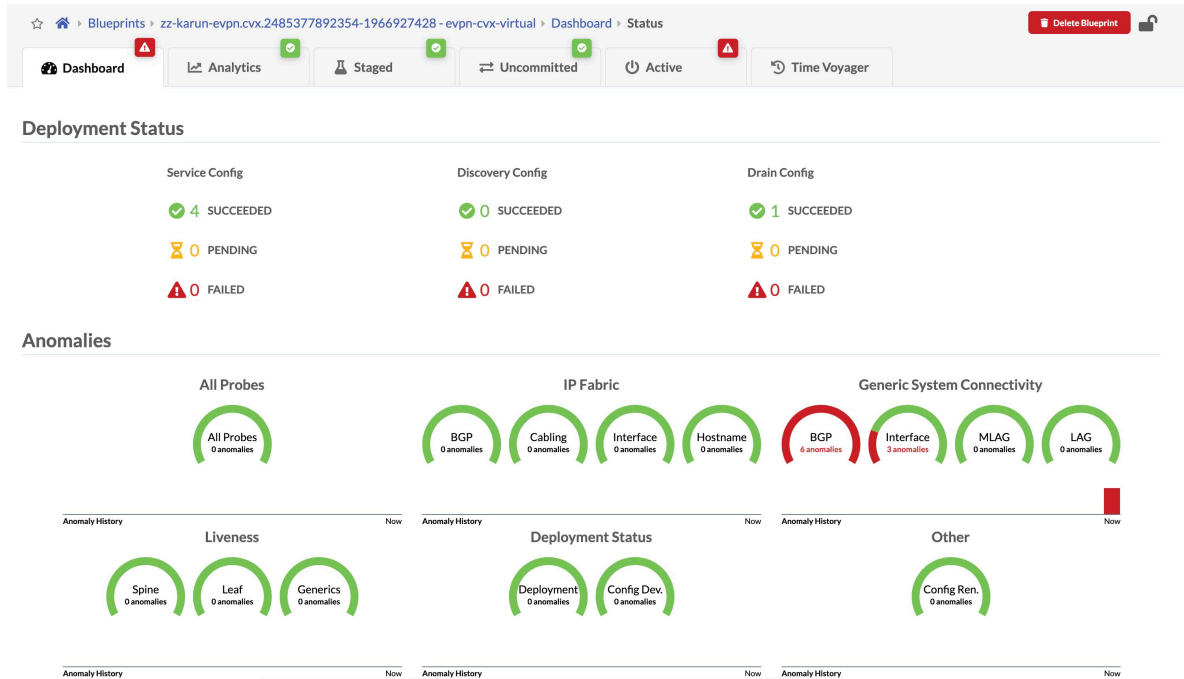
The screenshot shows the Cumulus CloudForge interface. The top navigation bar includes tabs for Dashboard, Analytics, Staged, Uncommitted, Active, and Time Voyager. The main area displays a topology diagram with nodes like leaf1, rack1-server1, leaf2, rtr\_leaf1\_leaf2, spine2, spine1, and switch1-server1. A sidebar on the right shows the configuration for leaf1, including its role (Leaf), group label (evpn-mlag), and various settings like Deploy Mode (Drain), S/N (525400ECE1C5), Device Info (Management IP: 172.20.35.13, OS: Cumulus 4.2.1), and Hostname (leaf1).



**CAUTION:** Shutting down **Generic System** links towards generic systems before draining BGP sessions can result in minimal traffic disruption until it is switched over to the peer leaf.

- When the device is set to **Drain** mode, any **Interface** and **BGP** anomalies appear in the dashboard under **Generic System Connectivity**. Also, all L2 ports towards generic systems are shutdown and all bond interfaces are set to **NotConfigured**. Also, to drain ingress and egress traffic, route-map is rendered on device denying traffic from any prefix in the underlay and Overlay BGP sessions within

Fabric and towards generic systems. Traffic continues to flow via the primary switch.



- For extra caution and to avoid rare case scenarios you can bring down the MLAG peer-link using Cumulus System based configlet as an example. However, this step varies from vendor to vendor and should be performed as per the vendor recommendation.

### Create Configlet

The screenshot shows the 'Create Configlet' form in the Cumulus dashboard. The form is titled 'Name' and contains the following fields and options:

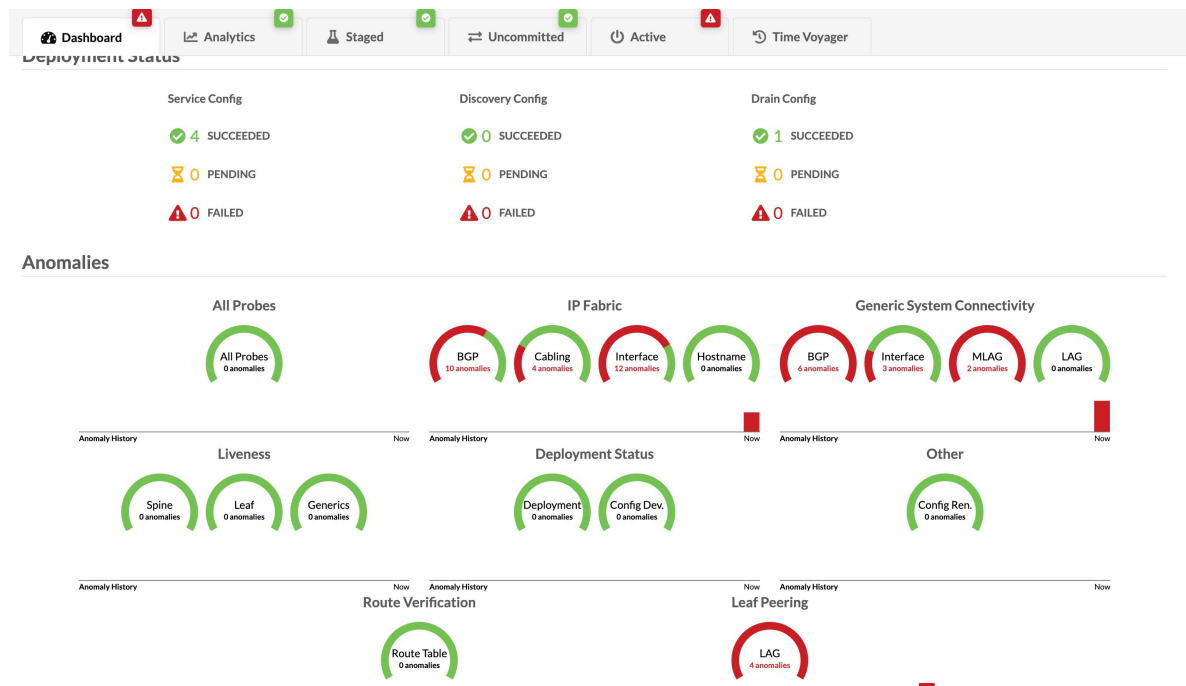
- Name:** Peer-Link down
- Generators:**
  - Config Style:** Cumulus (selected), NXOS, EOS, Junos, SONIC.
  - Section:** SYSTEM (selected), INTERFACE, FILE, OSPF, FRR.
  - Template Text:**

```
ifdown SWP3 --admin-state
ifdown SWP4 --admin-state
```
  - Negation Template Text:**

```
ifup SWP3 --admin-state
ifup SWP4 --admin-state
```
- Buttons:** 'Add a style' (blue), 'Create Another?' (checkbox), and 'Create' (blue).

Once committed, anomalies appear on the dashboard related to MLAG and interface, However, there

is no impact to traffic.



1. Upgrade the drained node with the ["Device Network Operating System Upgrade"](#) on page 158 procedure.
2. Bring back up the peer-link so that you can change the clag priority. Do this by removing the System based configlet applied before. All the Interface, MLAG and BGP anomalies related to peer-link should disappear in the web interface.
3. Put the **secondary** device back into the **Deploy Mode** and make sure peer is alive and clag bond interfaces are up. BGP and Interface anomalies for generic systems should clear now in the GUI. The secondary node has successfully upgraded. For upgrading primary node, first change the clag priority so that secondary node becomes primary and follow the same procedure on the new secondary switch.

The example below is for a system based configlet that changes priority on Cumulus secondary node:

### Create Configlet

Name \*

Clag\_Priority\_Change

Generators \*

Config Style \*

☒ Cumulus
 ☐ NXOS
 ☐ EOS
 ☐ Junos
 ☐ SONIC

Section \*

☒ SYSTEM
 ☐ INTERFACE
 ☐ FILE
 ☐ OSPF
 ☐ FRR

Template Text \*

```
sudo ctagctl priority 2048
```

Negation Template Text

```
sudo ctagctl priority 32768
```

+ Add a style

☐ Create Another?
 

Create

## Upgrade Device NOS

### IN THIS SECTION

- NOS Upgrade Overview | 158
- Update User-defined Device Profiles | 160
- Upgrade OS | 162

We highly recommend that you become familiar with this procedure before performing a device NOS upgrade.

### NOS Upgrade Overview

To upgrade a device NOS directly in the Apstra environment, you register the new OS image that you obtained from the vendor, then click a button to start the upgrade.

**NOTE:** We highly recommend that you upgrade directly from the Apstra environment to ensure that requirements are automatically taken care of for you. If you manually upgrade the NOS outside of the Apstra environment (which is not recommended), you must manually update pristine configuration afterward. Unassign the system ID and set the deploy mode to **Undeploy**. From the **Actions** panel at Devices > Managed Devices, click **Collect Pristine Config**. Then re-assign the system ID and set the deploy mode to **Deploy**.

For information about supported upgrade paths, see ["NOS Upgrade Paths" on page 906](#) in the Reference section.

Apstra software ships with built-in device profiles that support specific OS versions. When you upgrade the Apstra server, you're also updating these device profiles with the OS versions that are supported in the new Apstra version. You can then upgrade the NOS to one of the newly supported versions.

For example, Apstra version 4.0.0 supports Arista EOS versions as shown in the OS version selector (4.(18|20|21|22|23|24)) in the device profile. That is, it supports versions 4.18, 4.20, 4.21, 4.22, 4.23, and 4.24. Whereas, Apstra version 4.02 supports EOS versions 4.18, 4.20, 4.21, 4.22, 4.23, 4.24, and 4.25 (4.(18|20|21|22|23|24|25)). 4.25 is a newly supported version. If you upgrade the Apstra server to version 4.0.2, you can upgrade Arista devices to EOS version 4.25.

However, device profiles that you've created (cloned) yourself, are not managed in the Apstra environment, so they aren't automatically updated with newly supported versions when you upgrade the Apstra version. You'll need to follow a few extra steps as described in the next section.

Before beginning the process, make sure of the following:

- Make sure that you understand the ["device configuration lifecycle" on page 140](#) and that you're comfortable with managing deploy modes.
- Make sure that the device you're upgrading is being managed by the Apstra software. Navigate to **Devices > Managed Devices** and confirm that your device is on the list and that it is acknowledged (with a green check mark).
- Before upgrading NOS, delete any device AAA/TACACS+ configlets from the blueprint. After the upgrade is complete you can reapply them.
- Make sure that the Admin state of the device is set to **NORMAL**. Navigate to **Devices > Managed Devices**, click on the **device key** of the device to confirm the admin state. (Do NOT set the Admin state to MAINT/DECOMM or the device could enter an unrecoverable state.)
- Make sure that the Apstra version specified is the same on both the Apstra server and the device. If they are different, you cannot upgrade the device. If you attempt to upgrade with different versions, you will not receive a warning; the task status remains in the "in progress" state indefinitely.

- If you're upgrading a Cumulus Linux device, the DHCP server must be on the device management network that is configured with a static DHCP assignment for the Cumulus Linux device. The Cumulus Linux device **must** get the same IP from the DHCP server as before the upgrade. Astra software uses the Open Network Install Environment (ONIE) process to add a completely new Cumulus Linux to the device and return it to a factory default state. The eth0 management interface comes up configured for DHCP.

## Update User-defined Device Profiles

Make sure that your devices are in the appropriate states for upgrading as described in the overview above.

If you've created (cloned) your own device profiles, you'll need to manually specify OS versions in the device profile and the blueprint that uses that device profile. (If your devices use built-in device profiles, then proceed to the next section to upgrade the NOS.)

1. From the left navigation menu in the Apstra GUI, navigate to **Devices > Device Profiles**, select your device and update the OS version in the **Selector** section.

### Edit Device Profile

Device profiles need to accurately model various characteristics of a switch model. Make sure you update the profile to match the new switch model(s) you intend to use this profile for.

Updating the device profile ports may not be allowed because it is referenced by [Arista\\_DCS-7160TC\\_EOS\\_AOS-48x10\\_6x100-1](#) interface map.

Summary

Selector<sup>?</sup>

Capabilities

Ports

**Manufacturer \***

Arista

**Model \***

DCS-7160-48TC6

**OS family \***

EOS

**Version \***

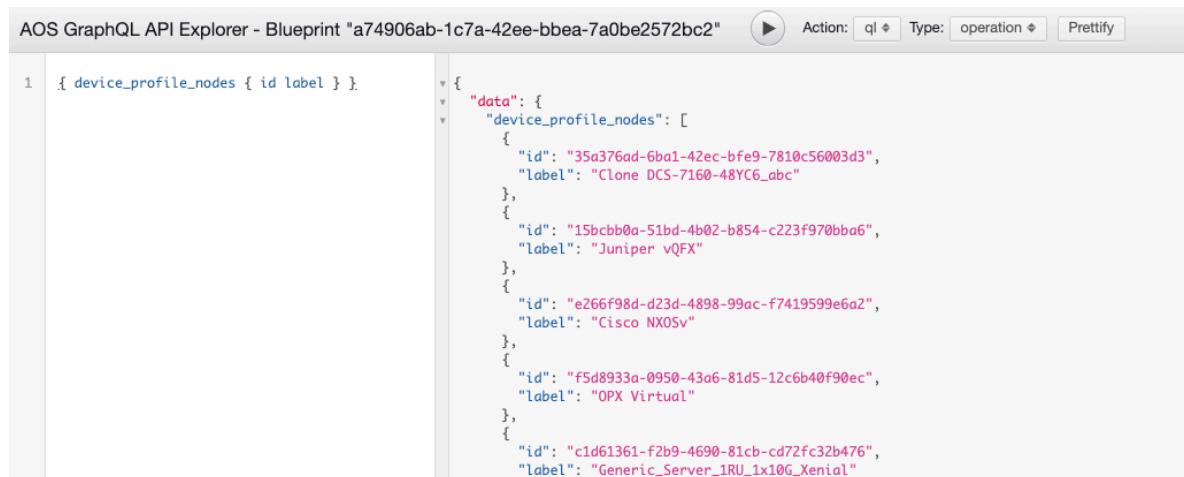
4\.(18|20|21|22|23)\.\*

4\.(18|20|21|22|23)\.\*

Update

2. From the left navigation menu, navigate to **Platform > Developers > Graph Explorer** and find the ID for the device profile. You can find it with the query variables `{ device_profile_nodes { id label } }`

In this example, the "id" for the label "Clone DCS-7160-48YC6\_abc" is "35a376ad-6ba1-42ec-bfe9-7810c56003d3".



The screenshot shows the AOS GraphQL API Explorer interface. The title bar indicates the blueprint ID is "a74906ab-1c7a-42ee-bbea-7a0be2572bc2". The interface has a query editor on the left and a response viewer on the right. The query is: `{ device_profile_nodes { id label } }`. The response is a JSON object with a "data" field containing an array of device profile nodes. The first node in the array has the ID "35a376ad-6ba1-42ec-bfe9-7810c56003d3" and the label "Clone DCS-7160-48YC6\_abc". Other nodes include "Juniper vQFX", "Cisco NXOSv", "OPX Virtual", and "Generic\_Server\_1RU\_1x10G\_Xenial".

```

1 { device_profile_nodes { id label } }

{
  "data": {
    "device_profile_nodes": [
      {
        "id": "35a376ad-6ba1-42ec-bfe9-7810c56003d3",
        "label": "Clone DCS-7160-48YC6_abc"
      },
      {
        "id": "15bcb0a-51bd-4b02-b854-c223f970bba6",
        "label": "Juniper vQFX"
      },
      {
        "id": "e266f98d-d23d-4898-99ac-f7419599e6a2",
        "label": "Cisco NXOSv"
      },
      {
        "id": "f5d8933a-0950-43a6-81d5-12c6b40f90ec",
        "label": "OPX Virtual"
      },
      {
        "id": "c1d61361-f2b9-4690-81cb-cd72fc32b476",
        "label": "Generic_Server_1RU_1x10G_Xenial"
      }
    ]
  }
}

```

3. Use the `aos-cli` utility to update the device profile. For more information about using the `aos-cli`, see [Juniper Support Knowledge Base article KB36747](#)

You can use your blueprint ID and the node ID from the previous step, then set the proper model ID ("DCS-7160-48YC6" for example), and execute.

`aos-cli` command format:

```

blueprint set-node-property --blueprint <your blueprint ID> --node_type
device_profile --node <node ID from Step2> --property selector
--value-fn '{"os_version":"4\.(18|20|21|22|23)\..*","model":<your model>'"
,"os": "EOS","manufacturer": "Arista"}'

```

Example:

```

aos> blueprint set-node-property --blueprint
a74906ab-1c7a-42ee-bbea-7a0be2572bc2 --node_type device_profile
--node 35a376ad-6ba1-42ec-bfe9-7810c56003d3 --property selector
--value-fn '{"os_version":"4\.(18|20|21|22|23)\..*","model":"DCS-7160-48YC6",
"os": "EOS","manufacturer": "Arista"}'

```

- From the Apstra GUI, navigate to your blueprint, click **Uncommitted** and commit the changes.

Query: All 1-2 of 2 Page Size: 25

Node ID	Node Type	Action	Staged Value	Active Value
35a376ad-6ba1-42ec-bfe9-7810c56003d3	device_profile	CHANGED	<pre>{   "selector": {     "os_version": "4\\.([0 20 21 22 23])\\.\\..*"   } }</pre>	<pre>{   "selector": {     "os_version": "4\\.([0 20 21 22 23])\\.\\..*"   } }</pre>
ba92af7a-a257-4b3f-932d-acc40a8a7e11	metadata	CHANGED	see value	see value

- Proceed to the next section to upgrade the OS in the same manner as for devices using predefined device profiles.

## Upgrade OS

Make sure that your devices are in the appropriate states for upgrading as described in the overview above.

This procedure is for devices that use built-in device profiles, as well as devices that use user-defined device profiles that have completed the previous section.

- Upload or register the necessary device ["OS images"](#) on page 264.



**CAUTION:** Make sure to select a compatible device operating system image for the device that you're upgrading. If you use an incompatible image and the upgrade fails, the deployment lock will not be released automatically, even if you recover the device. To release the deployment lock and activate the device again, remove the device assignment from the blueprint, decommission and normalize the device (from Devices > Managed Devices), then reassign the device to the blueprint. For assistance, contact ["Juniper Support"](#) on page 777.

- From the left navigation menu, navigate to **Devices > Agents** and select the device(s) to upgrade. You can search for specific devices (such as for all EOS devices) by entering a query.

**NOTE:** All selected devices must be of the same type, and they must be upgraded to the same image and version.

- Click the **OS Upgrade** button (in the right **Actions** panel).

The screenshot shows the AIOps Agents management interface. At the top, there are tabs for 'ONBOX' (5) and 'OFFBOX' (0). A 'Create Agent(s)' button is in the top right. Below the tabs is a search bar with 'Query: All' and pagination controls showing '1-5 of 5' and 'Page Size: 25'. The main table lists 5 devices with columns for Device Address, Operation Mode, Platform, Platform Version, AOS Version, Job State, System ID, Hostname, and Device State. The 'Job State' column shows 'SUCCESS' for all devices. The 'Actions' column contains icons for various operations, with the 'OS Upgrade' icon (a person with a plus sign) highlighted by a green box.

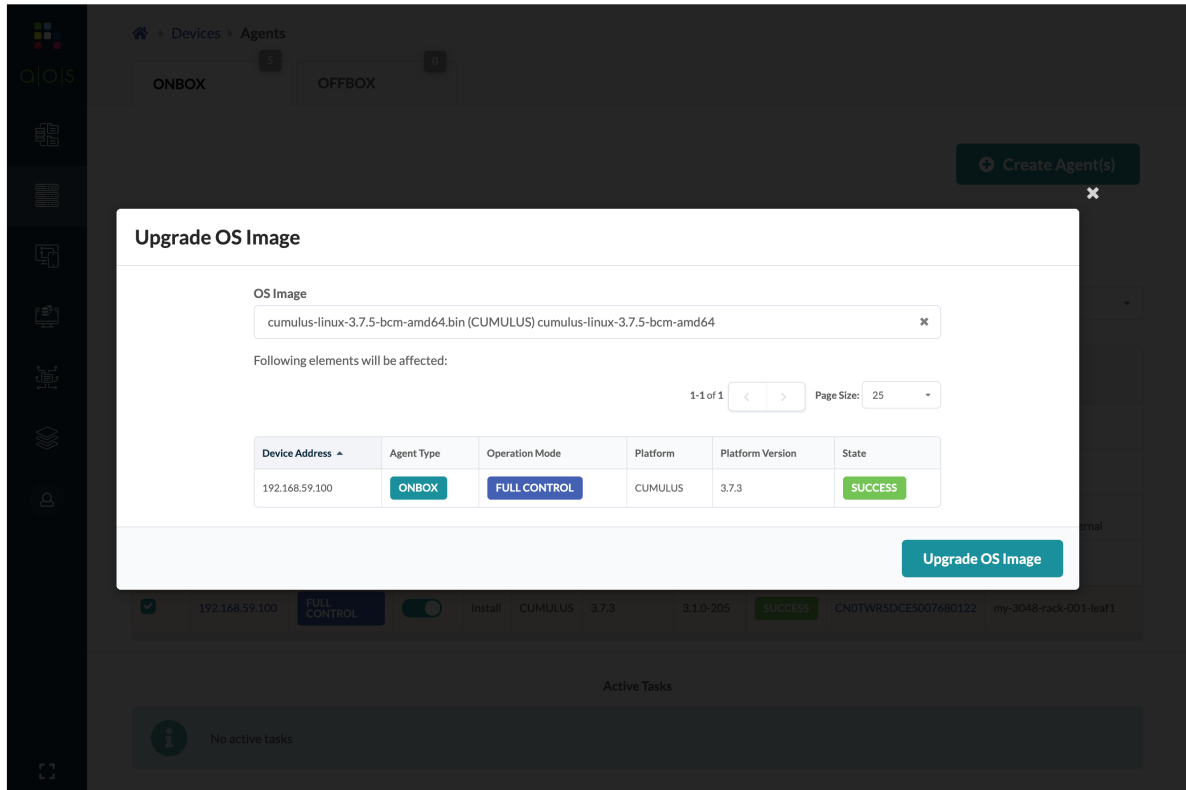
Device Address	Operation Mode	Platform	Platform Version	AOS Version	Job State	System ID	Hostname	Device State	Actions
172.20.191.16	FULL CONTROL	EOS	4.22.3M	3.2.0-127	SUCCESS	5054000BF029	I2-virtual-003-leaf1	IS-ACTIVE	[Icons: Check, Download, Upload, <b>OS Upgrade</b> , Eye, Refresh, Delete]
172.20.191.15	FULL CONTROL	CUMULUS	3.7.9	3.2.0-127	SUCCESS	525400651bb3	I2-virtual-001-leaf1	IS-ACTIVE	[Icons: Check, Download, Upload, OS Upgrade, Eye, Refresh, Delete]
172.20.191.17	FULL CONTROL	NXOS	7.0(3)I7(7)	3.2.0-127	SUCCESS	525400D591C4	switch	OOS-READY	[Icons: Check, Download, Upload, OS Upgrade, Eye, Refresh, Delete]
172.20.191.13	FULL CONTROL	CUMULUS	3.7.9	3.2.0-127	SUCCESS	525400460c1e	spine1	IS-ACTIVE	[Icons: Check, Download, Upload, OS Upgrade, Eye, Refresh, Delete]
172.20.191.14	FULL CONTROL	EOS	4.22.2F	3.2.0-127	SUCCESS	5054009CDB33	spine2	IS-ACTIVE	[Icons: Check, Download, Upload, OS Upgrade, Eye, Refresh, Delete]

Active Jobs: 0

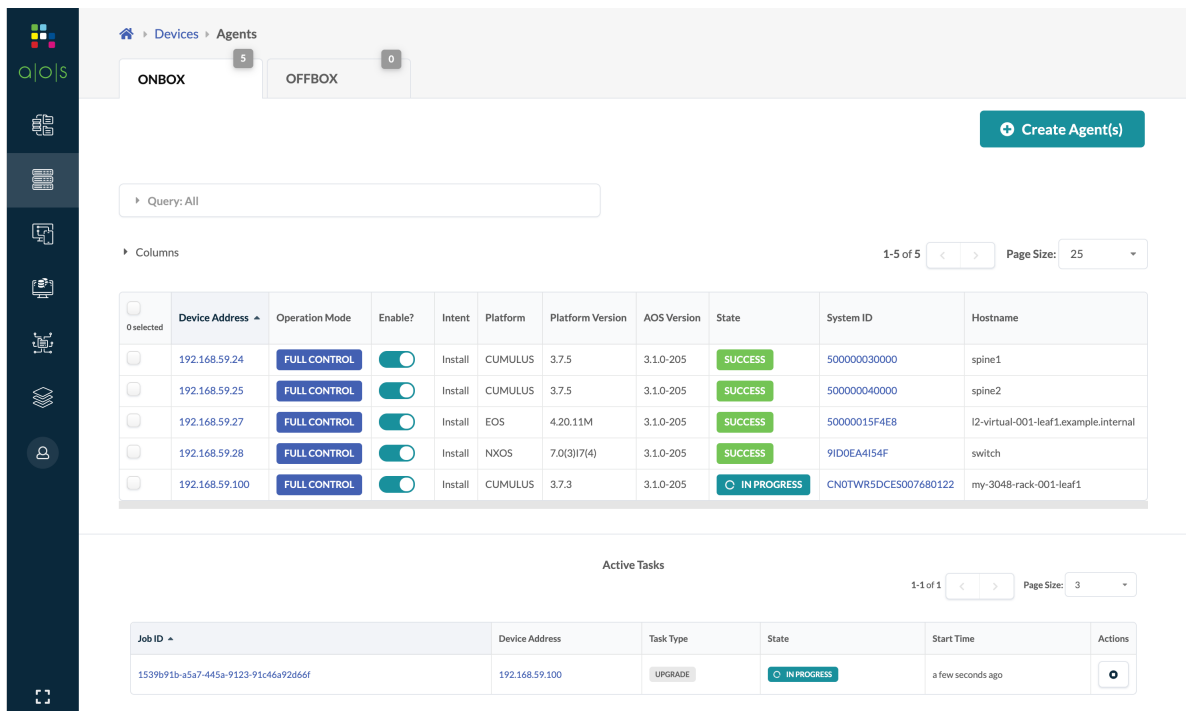
- The **Upgrade OS Image** dialog lists the available OS images that match the selected devices. Select the appropriate OS image and click **Upgrade OS Image**. The image uploads to the device.

For Cumulus Linux:

- The local device username and password that was used to install the agent is recreated.
- The same Cumulus Linux license is reinstalled and Apstra configuration and deployment state is restored.



5. You can monitor the upgrade status from the **Active Jobs** section at the bottom of the **Agents** page .



6. After the image is uploaded, if a checksum is provided with the OS image, the image checksum is verified. If the MD5/SHA512 checksum is incorrect, or if any other failures occur (such as for

insufficient disk space or incorrect remote URL), the job state changes to **FAIL** and the device does not reboot.

**NOTE:** If an issue arises with the OS image (such as interrupted download or invalid URL) during a NOS upgrade, you are informed before any device configuration is changed (new in Apstra version 4.0.1). You can then resolve the issue and restart the upgrade process.

7. If the job fails, click the agent to view errors. You can also click the **Show Log** button to view the detailed Ansible job. If an upgrade fails, you must manually resolve the issue causing the failure. For example, with a checksum error, you must either correct the invalid checksum or register a new OS image with a correct checksum, then repeat the upgrade process.
8. If the checksum is correct and no other failures occur, the job state changes to **SUCCESS** and the device reboots.
9. When the device has rebooted with the new image and has reestablished its agent connection with the controller, the upgrade is complete. The **Agents** page displays the new OS version.

The screenshot shows the Apstra web interface's 'Agents' page. The breadcrumb navigation is 'Devices > Agents'. There are two tabs: 'ONBOX' (5 agents) and 'OFFBOX' (0 agents). A 'Create Agent(s)' button is in the top right. Below the tabs is a search bar with 'Query: All' and a 'Columns' dropdown. A table lists 5 agents, all with a 'SUCCESS' state. Below the table is an 'Active Tasks' section showing 'No active tasks'.

	Device Address	Operation Mode	Enable?	Intent	Platform	Platform Version	AOS Version	State	System ID	Hostname
	192.168.59.24	FULL CONTROL	<input checked="" type="checkbox"/>	Install	CUMULUS	3.7.5	3.1.0-205	SUCCESS	500000030000	spine1
	192.168.59.25	FULL CONTROL	<input checked="" type="checkbox"/>	Install	CUMULUS	3.7.5	3.1.0-205	SUCCESS	500000040000	spine2
	192.168.59.27	FULL CONTROL	<input checked="" type="checkbox"/>	Install	EOS	4.20.11M	3.1.0-205	SUCCESS	50000015F4E8	I2-virtual-001-leaf1.example.internal
	192.168.59.28	FULL CONTROL	<input checked="" type="checkbox"/>	Install	NXOS	7.0(3i)7(4)	3.1.0-205	SUCCESS	9ID0EA4I54F	switch
	192.168.59.100	FULL CONTROL	<input checked="" type="checkbox"/>	Install	CUMULUS	3.7.5	3.1.0-205	SUCCESS	CN0TWR5DCE5007680122	cumulus

## Remove Device

### IN THIS SECTION

- [Remove Device from Blueprint | 166](#)
- [Remove Device from Apstra Management | 168](#)
- [Replace Device | 168](#)

After installing agents on devices and acknowledging them, they are managed by Apstra software and can be assigned to blueprints. You can unassign devices from blueprints and remove them from Apstra management, as needed. See below for details.

### Remove Device from Blueprint

1. Remove the device assignment from the blueprint.



## [l2\\_virtual\\_ext\\_002\\_leaf1](#)

Role: Leaf

Group label: leaf

**Device** Properties

✓ **Deploy Mode**

deploy

✓ **S/N**

505400457C3D (172.20.225.11)

✓ **Hostname**

l2-virtual-ext-002-leaf1

✓ **Config**

Rendered

Incremental

Pristine

2. ["Commit" on page 648](#) the change to remove the assignment. The device has been disassociated from the blueprint. It is still managed by Apstra and it is ready and available to be assigned to the same blueprint or another one.

## Remove Device from Apstra Management

Devices that are no longer managed by Apstra are disconnected from the Apstra server and removed from the Apstra database. For successful device removal, follow the steps in the order listed

1. Remove the device assignment from the blueprint (if it had been previously assigned).
2. ["Commit" on page 648](#) the change to remove the device assignment from the blueprint, as applicable.
3. Set the admin state of the device to **DECOMM**.
4. ["Uninstall" on page 193](#) the device agent.
5. ["Delete" on page 194](#) the device agent.
6. ["Delete" on page 132](#) the device system from the managed devices list.

**NOTE:** If you are decommissioning a faulty device and replacing it with another one for RMA using the same management IP address, make sure that the original device has been decommissioned before the new device is assigned, as each device is expected to have a unique management IP address.

## Replace Device

If you are removing a device that will be replaced, follow the steps above for removing a device from Apstra management, then follow the steps for ["adding a device" on page 150](#).

## Reuse Device

There may be times when you want to remove a device from one blueprint and reuse it in a different one. In this case, you'll need to take the device configuration to a pre-Apstra state and then completely re-onboard the device. Refer to the two device guides below for details:

1. ["Remove Device" on page 166](#).
2. ["Add Device" on page 150](#).

# Device AAA Support

IN THIS SECTION

- [Device AAA Overview | 169](#)
- [Supported Platforms | 170](#)

## Device AAA Overview

Device AAA (authentication, authorization and accounting) frameworks are supported, specifically RADIUS and TACACS+, on certain platforms. Device AAA is optional and correct implementation is the responsibility of the end user. Minimum requirements for correct Apstra AAA implementations are described below.



**CAUTION:** We recommend adding a local Apstra user to devices when using AAA framework. If AAA authentication or authorization fails when Apstra performs a full configuration push, manual recovery (config push) is required.

You can apply AAA configuration in one of two ways:

Table 16: Device AAA Configuration Methods

Device AAA Method	Description
Configlets (Recommended)	<p>Configuration is added to configlets which are then imported into blueprints. Local credentials must be available from the Apstra environment so the device can be added and the configlet can be applied. See <a href="#">"Configlets" on page 119</a> for details.</p> <p><b>CAUTION:</b> Before upgrading the Apstra server, device agents, or NOSes, you <b>must</b> delete device AAA/TACACS configlets from blueprints. After the upgrade is complete, you can re-apply them.</p>
User-required	<p>To ensure pre-existing configuration is retained when a device is brought under Apstra management, make sure the config is in place before unacknowledging the device. See <a href="#">"Device Configuration Lifecycle " on page 140</a>.</p>

## Supported Platforms

### IN THIS SECTION

- [Juniper Junos | 170](#)
- [Cisco NX-OS | 170](#)
- [Arista EOS | 171](#)

### Juniper Junos



**CAUTION:** Credentials for the Junos off-box system agent user must always be valid and available. When using the AAA framework we recommend that you add a local user to devices and use it for Apstra off-box system agents. Always have “password” be first in Junos config for authentication-order as follows:

```
authentication-order [ password radius ]
```

### Cisco NX-OS



**CAUTION:** A remote user could erratically be removed from NX-OS devices, causing authentication and authorization failures. The user (role 'network-admin') must exist on the device in order to manage the device. If not, Apstra functions such as agent installation, telemetry collection and device configuration may fail. The only known workaround is to use local authentication.

The example NX-OS configuration below has been tested to work correctly with Apstra software. This uses both authentication and authorization:

```
tacacs-server key 7 "<key>"
tacacs-server timeout <timeout>
tacacs-server host <host>
aaa group server tacacs+ <group>
  server <host>
  use-vrf management
  source-interface mgmt0
```

```

aaa authentication login default group <group>
aaa accounting default group <group> local
aaa authentication login error-enable
aaa authentication login ascii-authentication

```

## Arista EOS



**CAUTION:** When TACACS+ AAA is configured on EOS devices, device agent upgrades could fail while files are copied from the Apstra server to the device. This commonly happens if TACACS+ uses a custom password prompt. To prevent this type of failure, temporarily disable all TACACS+ AAA where device authentication uses an admin-level username and password for any device agent operations, including upgrades.

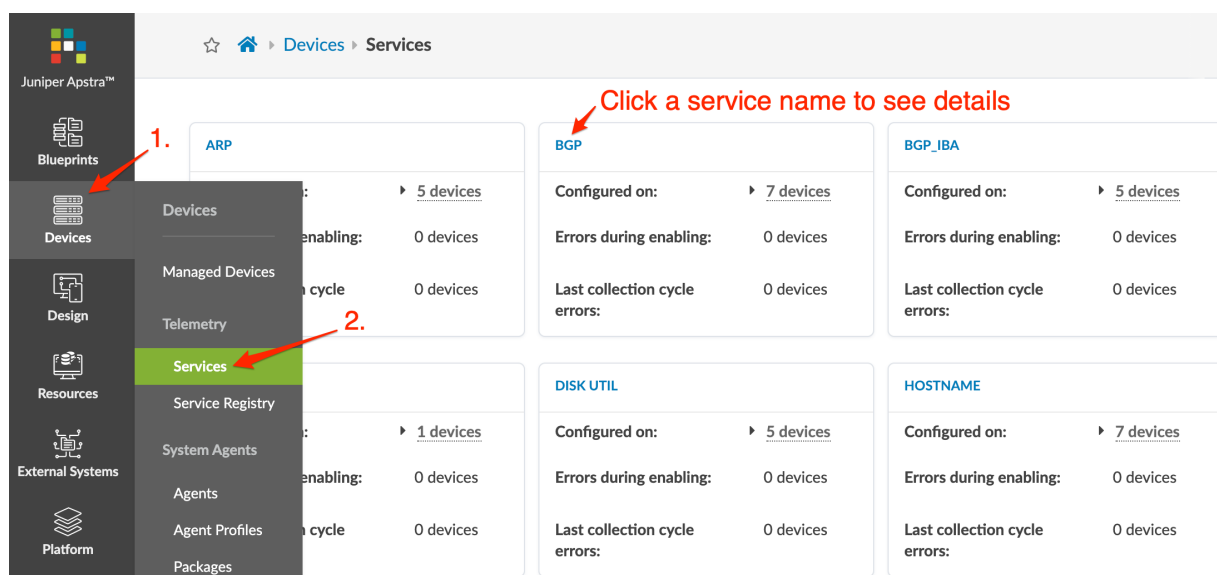
## Telemetry (Devices)

### IN THIS SECTION

- [Services | 172](#)
- [Service Registry | 174](#)
- [Telemetry Collection Statistics | 176](#)
- [Telemetry Streaming | 177](#)
- [Route Anomalies for a Host - Example | 178](#)
- [Telemetry Command Reference | 181](#)
- [Cisco Telemetry | 181](#)
- [Arista Telemetry Commands | 182](#)
- [Cumulus | 182](#)
- [Linux Servers | 183](#)
- [Debugging Telemetry | 183](#)

## Services

From the left navigation menu, navigate to **Devices > Telemetry > Services** to go to a summary of telemetry services.



Telemetry services include the following:

Service	Description
ARP	ARP telemetry shows an ARP table. This information can be queried via API. No anomalies are generated.
BGP	BGP telemetry shows role(s), VRF name, address family, source and destination information, expected and actual states, intent status, last fetched/modified, and BGP peer state.
Config	<p>Devices with deviations between the rendered discovered/service config and the actual config are flagged with a config deviation error. When configuration changes are made outside of Apstra management, alarms are generated immediately. The risk with a configuration deviation is that it is possible for Apstra to overwrite the deviated configuration with a configuration re-write.</p> <p>The correct way to deal with a config deviation alarm is to understand the configuration change being made, and consider setting it up as a <a href="#">"configlet" on page 119</a> instead.</p>
Counters	Counter telemetry provides information about interface in/out packets, interface errors, statistics, and so on. This feature is consumed by other advanced downstream features like telemetry streaming. No anomalies are generated.
Hostname	When you assign a device with deploy mode <b>Ready</b> to a blueprint, the device enters the <b>Discovery 2 Config</b> stage. Hostname telemetry is collected that validates the device hostname against intent. Mismatches result in anomalies.
Interface	When you assign a device with deploy mode <b>Ready</b> to a blueprint, the device enters the <b>Discovery 2 Config</b> stage. Interface telemetry is collected that compares intent with the

Service	Description
	up/down state of physical interfaces. It does not include LLDP, LAG or any other attachment information.
LAG	LAG telemetry shows the health of all the LACP bonds facing servers and between MLAG switches.
LLDP (Cabling)	When you assign a device with deploy mode <b>Ready</b> to a blueprint, the device enters the <b>Discovery 2 Config</b> stage. Every node is part of intent. On each link, there are expected neighbor hostnames, interfaces and connections. Physical cabling and links must match the specified intent. Any deviations result in anomalies that must be corrected by either recabling to match the blueprint or by modifying the blueprint to match cabling already in place.
MAC	MAC Address-table telemetry shows which MAC addresses appear on which interfaces, and which VLANs.
MLAG	<p>MLAG telemetry tracks the health status of the MLAG domain itself - the control-protocols required between two leaf switches communicating with each other properly for the MLAG domain state. Implementation detail differences exist between multiple vendors, but the intent is the same -the switches should be healthy among each other. MLAG telemetry is only available for L2 blueprints that have at least one virtual network assigned in an MLAG pair.</p> <p>If an MLAG-attached server is not fully connected, the state changes from 'active_full' to 'active_partial'.</p> <div> <p><b>NOTE:</b> Cisco MLAG (VPC) commands cannot derive the status of the LAG on the VPC peer switch. Accordingly, the state <i>dual-active</i> cannot actually gather the command. This is a limitation from Cisco.</p> </div>
Route	Routing telemetry analyzes the routing table on every managed spine and leaf. Since the entire IP fabric is managed, you can derive and predict full IP table information from the network topology. Deviations in the network routing telemetry (for example, a missing next-hop IP address for a default route) cause an alarm.
Transceivers	Transceiver telemetry gives the network operator statistics on optical interfaces, showing DOM statistics, light levels, lossy interfaces, and other optical statistics. No anomalies are generated.
Utilization (On-box agents only)	<p>Utilization telemetry allows the network operator to view some vital statistics on the device - CPU and Memory utilization. No anomalies are generated.</p> <p>Utilization telemetry is not available on devices using off-box agents (Junos for example). Therefore, the utilization tab contains the error <b>Network Device not found</b>.</p>

# Service Registry

IN THIS SECTION

- [Service Registry Overview | 174](#)
- [Import Service Schemas | 175](#)
- [Delete Service Registry | 176](#)

## Service Registry Overview

From the left navigation menu, navigate to **Devices > Service Registry** to go to the service registry. You can view, import and delete telemetry service schemas via the GUI (as of Apstra version 4.0.1). For information about developing extensible telemetry, see the ["Extensible Telemetry Guide" on page 787](#).

Juniper Apstra™

Blueprints

Devices

Design

Resources

External Systems

Platform

Favorites

User: admin

Devices

Managed Devices

Telemetry

Services

Service Registry

System Agents

Agents

Agent Profiles

Packages

OS Images

ZTP Status

Devices

Services

Device Profiles

Devices > Service Registry

1-17 of 17

Page Size: 25

	Storage Schema Path	Description	Builtin?	Actions
	aos.sdk.telemetry.schemas.generic		no	
	aos.sdk.telemetry.schemas.iba_integer_data		no	
	aos.sdk.telemetry.schemas.iba_integer_data		no	
	aos.sdk.telemetry.schemas.iba_string_data		no	
	aos.sdk.telemetry.schemas.iba_integer_data		no	
	aos.sdk.telemetry.schemas.iba_string_data		no	
	aos.sdk.telemetry.schemas.iba_integer_data		no	
	aos.sdk.telemetry.schemas.iba_string_data		no	
	aos.sdk.telemetry.schemas.iba_integer_data		no	
	aos.sdk.telemetry.schemas.iba_string_data		no	
	aos.sdk.telemetry.schemas.iba_integer_data		no	

1.

2.

Click a service name to see details

To see service registry details, click a service name.

The screenshot shows the Juniper Apstra web interface. On the left is a dark navigation sidebar with icons for Blueprints, Devices, Design, Resources, External Systems, Platform, and Favorites. The top of the main area has a breadcrumb trail: [Devices](#) > [Service Registry](#) > [evpn\\_vxlan\\_type5](#). The main content area displays details for the service 'evpn\_vxlan\_type5' in a table-like format.

Service Name	evpn_vxlan_type5
Storage Schema Path	aos.sdk.telemetry.schemas.iba_integer_data
Description	
Builtin?	no
Version	version_0
Application Schema	<pre>{   "required": [     {       "key",       "value"     }   ],   "type": "object",   "properties": {     "key": {       "required": [         "l3_vni",         "subnet",         "ip_family",         "nexthop",         "rd",         "route_target"       ],       "type": "object",       "properties": {         "subnet": {           "type": "string"         },         "ip_family": {           "type": "string"         },         "route_target": {           "type": "string"         },         "nexthop": {           "type": "string"         },         "rd": {           "type": "string"         },         "l3_vni": {           "type": "string"         }       }     },     "value": {       "type": "integer"     }   } }</pre>

At the bottom of the sidebar, it shows 'User: admin'.

## Import Service Schemas

1. From the left navigation menu, navigate to **Devices > Service Registry** and click **Import Service Schemas**.
2. Either click **Choose File** and navigate to the file on your computer, or drag and drop the file from your computer into the dialog window and click **Import**.

### Delete Service Registry

1. Either from the list view (Devices > Service Registry) or the details view, click the **Delete** button for the service to delete.
2. Click **Delete Service Schema** to remove the schema from the system and return to the service registry screen.

### Telemetry Collection Statistics

To go to collection statistics for devices using a specific service, click a service name. Telemetry collection statistics include the following details:

Collection Statistics	Description
Device	The device key
Service Started?	Has the service started?
Interval	How frequently the service is configured to run on the device (in seconds)
Input	The input that is provided to the service for its processing
Run Count	The number of times the collector is scheduled to run
Success Count	The number of times the collector successfully executed
Failure Count	The number of times the collector failed execution
Max Run Count	User-specified maximum number of times for the collector to run
Execution Time	The time it took for collection during the last iteration (in milliseconds)
Waiting Time	A device runs multiple collectors. If some collectors monopolize CPU, other collector executions are deferred. Waiting time is the amount of time that the collector was deferred (in milliseconds).
Last Run Timestamp	Timestamp at which the collector was scheduled to run
Last Error Timestamp	Timestamp at which the collector last reported an error
Error message	Error message from the last collector iteration.

From the collection statistics screen, you can see if there are any service errors that were generated during the telemetry collection process (in the **Error message** column). Click the **Show error** link to see its details.

From this screen you can also go to all telemetry services for a specific device by clicking the device name.

Click a device name to go to all telemetry services for that device

Click to see details about any errors

Device	Service Started?	Interval, s	Input	Run Count	Success Count	Failure Count	Max Run Count	Execution Time, ms	Waiting Time, ms	Last Run Timestamp	Last Error Timestamp	Error message
5254007AF13C (localhost, 10.28.81.7)	yes	5		5427	0	5427		10.48	0.83	2021-10-18, 10:52:46	2021-10-18, 10:52:46	Show error
505400CFEACB (leaf-1-1, 10.28.81.9)	yes	5		63826	63826	0		34.43	0.67	2021-10-18, 10:53:36		N/A

To go to collection statistics for all services on a specific device, click **Collection Statistics**.

Collection Statistics

Service Name	Service Started?	Interval (s)	Input	Run Count	Success Count	Failure Count	Max Run Count	Execution Time, ms	Waiting Time, ms	Last Run Timestamp	Last Error Timestamp	Error message
ARP	yes	120		2719	2719	0		21.37	0.78	2021-10-18, 11:02:22		N/A
MLAG	yes	10		32610	32610	0		24.24	157.62	2021-10-18, 11:03:51		N/A

## Telemetry Streaming

The Apstra server transmits the following content to user-defined end-hosts for further processing of the data and for use within your own internal systems:

Data Type	Description
Counter Data	Performance Monitoring (PM) data is time-series numerical values such as interface counters, CPU memory utilization, and CPU usage. This information is typically stored and graphed for visual analysis. Typical tools used for this purpose include Graphite and Cacti.

Data Type	Description
Event Data	Event data is a collection of status information that you may need to refer back to in order to troubleshoot your network. The best reference for example event data is syslog. You need a general amount of event history so that you can perform troubleshooting activities over a period of time. While this is an undefined amount of time, you generally want as much time as possible, because you don't get to troubleshoot a problem the instant that it occurs.
Alert Data	Alert data is a collection of information that requires your attention to resolve an issue. In the best cases, alerts tell you what is wrong relative to the network service, and provide the necessary data to allow you to identify root-cause and resolve the issue as fast as possible.

The data streams themselves are implemented with Google Protocol Buffers (GPB). The format of the data streams is defined and also implemented using GPBs. GPBs allow software developers to use a language-agnostic definition of events and data types.

GPB offers support for C++, Python, Go, and possibly more languages in the future. Apstra has example Python code named "[AOSOM Streaming](#)" on page 828 that is available for Google Protocol Buffers. The AOSOM Streaming demo software is open source and can be downloaded from github: <https://github.com/Apstra/aosom-streaming>.

Developers have various language options : C++, Python, Go. This means it integrates nicely with our C++ infrastructure. And then Infrastructure Engineers can use Python or Go for the client.

## Route Anomalies for a Host - Example

HTTP GET [https://aos-server/api/blueprints/{blueprint\\_id}/anomalies](https://aos-server/api/blueprints/{blueprint_id}/anomalies) (output has been truncated to only show example of one missing route. Actual GET response will return entire routing table)

```
{
  "items": [
    {
      "actual": {
        "value": "missing"
      },
      "anomaly_type": "route",
      "expected": {
        "value": "up"
      },
      "id": "547bcbc9-963f-4477-904b-712482aa6428",
      "identity": {
        "anomaly_type": "route",
        "destination_ip": "0.0.0.0/0",
        "system_id": "000C29202526"
      }
    }
  ]
}
```

```

    },
    "last_modified_at": "2017-06-09T17:28:13.773324Z",
    "role": "unknown",
    "severity": "critical"
  },
  {
    "actual": {
      "value": "partial"
    },
    "anomaly_type": "route",
    "expected": {
      "value": "up"
    },
    "id": "92a6804a-42ff-4cbd-a52b-5c6acadc1d23",
    "identity": {
      "anomaly_type": "route",
      "destination_ip": "0.0.0.0/0",
      "system_id": "000C29EA59A7"
    },
    "last_modified_at": "2017-06-09T17:28:44.787604Z",
    "role": "unknown",
    "severity": "critical"
  },
  {
    "actual": {
      "value": "partial"
    },
    "anomaly_type": "route",
    "expected": {
      "value": "up"
    },
    "id": "25886eb7-e629-4f56-9479-686fe1e53c64",
    "identity": {
      "anomaly_type": "route",
      "destination_ip": "0.0.0.0/0",
      "system_id": "000C29E808A1"
    },
    "last_modified_at": "2017-06-09T17:28:13.773423Z",
    "role": "unknown",
    "severity": "critical"
  },
  {
    "actual": {

```

```

      "value": "partial"
    },
    "anomaly_type": "route",
    "expected": {
      "value": "up"
    },
    "id": "2b7a77ac-fd12-41fe-acfc-a53678b177ed",
    "identity": {
      "anomaly_type": "route",
      "destination_ip": "0.0.0.0/0",
      "system_id": "000C2982786A"
    },
    "last_modified_at": "2017-06-09T17:28:13.773389Z",
    "role": "unknown",
    "severity": "critical"
  },
  {
    "actual": {
      "value": "partial"
    },
    "anomaly_type": "route",
    "expected": {
      "value": "up"
    },
    "id": "50a1e0d6-e483-4bc4-bed8-cbc5666569f8",
    "identity": {
      "anomaly_type": "route",
      "destination_ip": "0.0.0.0/0",
      "system_id": "000C2998C7E7"
    },
    "last_modified_at": "2017-06-09T17:28:13.773453Z",
    "role": "unknown",
    "severity": "critical"
  },
  {
    "actual": {
      "value": "down"
    },
    "anomaly_type": "bgp",
    "expected": {
      "value": "up"
    },
    "id": "ab9f4273-e86f-456c-8cc7-7115f3aafa45",

```

```

    "identity": {
      "anomaly_type": "bgp",
      "destination_asn": "1",
      "destination_ip": "10.1.1.1",
      "source_asn": "65417",
      "source_ip": "10.0.0.5",
      "system_id": "000C29202526"
    },
    "last_modified_at": "2017-06-09T17:28:13.727949Z",
    "role": "to_external_router",
    "severity": "critical"
  }
],
"count": 6
}

```

## Telemetry Command Reference

This section assists network administrators in understanding why telemetry alarms exist, and how they are generated. This is not an exhaustive list of interface commands.

### Cisco Telemetry

Cisco telemetry is derived from the NX-API with 'show' commands and embedded event manager applets that provide context data to the device agent while it is running. Most commands are run as their CLI version wrapped into JSON output.

Service	Command
Interface counters	show interface counters   json
Interface error counters	show interface counters errors   json
Interface status	show interface status   json
LLDP neighbors	show lldp neighbors detail   json
BGP Sessions	show bgp session   json
Hostname	show hostname   json and show hosts   json
ARP	show ip arp vrf default   json
MAC Table	show mac address-table   json
Routing table	show ip route   json
Port-channel	show port-channel summary   json
MLAG	show vpc   json

## Arista Telemetry Commands

Arista EOS uses a few techniques from the EOS SDK API to directly subscribe to event notifications from the switch, for example 'interface down' or 'new route' notifications. When using an event-based notification, you do not have to continually render 'show' commands every few seconds. The EOS SDK gives you the information immediately as soon as the switch has the status.



**CAUTION:** Event-based subscription requires the EOSProxySDK agent. For details, see ["Arista Device Agents" on page 219](#).

When the Arista API does not provide information (LLDP statistics), Apstra runs CLI commands at a regular interval to derive telemetry expectations.

Interface counters	show interface counters
Interface error counters	show interfaces counters errors
Interface status	show interfaces status
LLDP neighbors	show lldp neighbors detail
BGP Sessions	show ip bgp summary
Hostname	show hostname
ARP	ARP collection is done using an event-monitor for performance. show event-monitor arp and show ip arp
MAC Table	MAC address collection is done using an event-monitor for performance. show event-monitor mac and show mac address-table
Routing table	show ip route
Port-channel	show port-channel summary
MLAG	show mlag and show mlag interfaces

## Cumulus

Cumulus switches use a combination of the Cumulus netshow command and standard Linux sockets.

Service	Command
Interface counters	ethtool -m
Interface error counters	ethtool -m
Interface status	Interface status is collected using the netlink api (AF_INET)
LLDP neighbors	lldpctl -f json

Service	Command
BGP Sessions	<code>vysh -c 'show ip bgp summary json'</code>
Hostname	<code>hostname</code>
ARP	<code>ARP</code>
MAC Table	MAC address collection is done using an event-monitor for performance. <code>show event-monitor mac</code> and <code>show mac address-table</code>
Routing table	<code>show ip route</code> and the <code>AF_INET</code> linux socket
Port-channel	<code>netshow bondmems --json</code>
MLAG	<code>clagctl -j</code>

## Linux Servers

Linux Servers use simple CLI commands and standard Linux sockets for most of the telemetry collection.

Interface counters	<code>ethtool -m</code>
Interface error counters	<code>ethtool -m</code>
Interface status	Interface status is collected using the netlink api ( <code>AF_INET</code> )
LLDP neighbors	<code>lldpctl -f xml</code>
BGP Sessions	<code>vysh -c 'show ip bgp summary json'</code>
Hostname	<code>hostname</code>
ARP	<code>ip -4 neigh</code>
MAC Table	<code>brctl showmacs</code>
Routing table	<code>show ip route</code> and the <code>AF_INET</code> linux socket
Port-channel	<code>netshow bondmems --json</code>
MLAG	<code>clagctl -j</code>

## Debugging Telemetry

Enable trace options to debug telemetry output. On the Device Agent, in `/etc/aos.conf` (usually), set these options and restart the agent.

```
[DeviceTelemetryAgent]
log_config = aos.infra.core.entity_util:DEBUG,aos.device.DeviceTelemetryAgent:DEBUG
trace_config = MountFacility/0-8,DHT,AgentHeartbeat,TelemetryProxy
```

Log files containing trace information for telemetry agents will then be viewable in `/var/log/aos/DeviceTelemetryAgent.<pid>.<timestamp>.log`. These log files are verbose, but they may point to various rendering and parsing issues in the environment. When you finish troubleshooting, be sure to disable logging.

## Agents (Devices)

### IN THIS SECTION

- [Create On-box Agent | 189](#)
- [Create Off-box Agent | 190](#)
- [Uninstall Agent | 193](#)
- [Edit / Delete Agent | 194](#)
- [Juniper Device Agent | 195](#)
- [SONiC Device Agent | 199](#)
- [Cisco Device Agent | 207](#)
- [Arista Device Agent | 219](#)
- [Cumulus Device Agent | 239](#)

Apstra device system agents handle configuration management, device-to-server communication, and telemetry collection. If you're not using ["Apstra ZTP" on page 267](#) to bootstrap your devices (or if you have a one-off installation) you can use this device installer to automatically install and verify devices. You can install device agents in the following ways:

- on-box - agent is installed on the device
- off-box - agent is installed on the Apstra server and communicates with devices via API

The types of agents supported on devices are as follows:

**Table 17: On-box and Off-box Agent Support**

Device	On-box	Off-box
Juniper Junos OS	No	Yes

Table 17: On-box and Off-box Agent Support *(Continued)*

Device	On-box	Off-box
Juniper Junos OS Evolved	No	No
Cisco NX-OS	Yes	Yes
Arista EOS	Yes	Yes
Cumulus Linux	Yes	No
Enterprise SONiC	Yes	No

The following configuration must not be on the device when you install a device agent:

- VLANs other than VLAN 1
- VRFs other than "management"
- Interface IP addresses other than "management"
- Loopback interfaces
- VLAN interfaces
- VXLAN interfaces
- AS-Path access-lists
- IP prefix-lists
- Route maps or policies
- BGP configuration

During the agent install process, device configuration is validated, and if the device contains configuration that could prevent the deployment of service configuration, the agent install process raises an error (as of Apstra 4.0.1).


Status

System ID	JPI
Operation Mode	NOT INSTALLED
Apstra Version	absent
State	FAILED
Has Credentials?	yes
Platform	eos
Platform Version	4.24.5M
Error	Conflicting pre-AOS configuration found in device. See device logs for more details.
Current Task	Collect pristine

In this case, manually remove conflicting configuration and start the agent installation process again.

If you must complete the agent installation with configuration validation errors, you can disable pristine configuration validation. To do this, from **Devices > System Agents > Agents** select **Advanced Settings** and select **Skip Pristine Configuration Validation**.

### Advanced Settings


☒ **Skip Pristine Configuration Validation**  
Pristine Configuration Validation is done as part of Apstra agent installation to check if the initial device configuration contains any configuration that may conflict with future Apstra managed configuration and abort the agent installation to avoid future problems. Checking this option is not recommended but can be done to force install Apstra agents even if possible conflicting configuration is found

☐ **Skip Revert to Pristine on Uninstall**  
When uninstalling Apstra agents, the device configuration is automatically reverted back to its Pristine Configuration. Checking this option will keep the device configuration untouched when Apstra agent is uninstalled

Update

For information about retaining pre-existing configuration when bringing devices under Apstra management, see ["Device Configuration Lifecycle" on page 140](#). For more information about managing devices in the Apstra environment see the guides.

**NOTE:** On some platforms (Junos for example) you can configure rate-limiting for management traffic (SSH for example). When the Apstra server interacts directly with devices it can be more bursty than when it interacts with a user. Rate-limiting configurations that are used for hardening security can impact device management, and lead to deployment failures and other agent-related issues.

Agents include the following parameters:

Table 18: Agent Creation Parameters

Parameter	Description
Device addresses	Management IP(s) of the device(s)
Operation Mode	<ul style="list-style-type: none"> <li>• Full Control - deploys configuration and collects telemetry</li> <li>• Telemetry Only - configuration is not deployed</li> </ul>
Platform (off-box only)	For off-box agents only: drop-down list includes supported platforms.
Username / Password	If you're not using an agent profile with credentials, check these boxes and add credentials.
Agent Profile	If you don't want to manually enter credentials and packages, use agent profiles that you previously defined.
Job to run after creation	<ul style="list-style-type: none"> <li>• Install (default) - installs the agent on the device</li> <li>• Check - creates the agent, but does not install it. It appears in the list view where you can install it later.</li> </ul>
Install Requirements (servers only)	For servers only: If servers don't have Internet connectivity, uncheck the box.
Packages	Before creating the agent, install required packages so they are available. Packages associated with selected agent profiles are listed here as well.
Open Options (off-box only)	<p>Passes configured parameters to off-box agents. For example, to use HTTPS as the API connection from off-box agents to devices, use the key-value pair: proto-https - port-443. The following default values can be overridden with open options:</p> <ul style="list-style-type: none"> <li>• commit_timeout - 60 (integer: seconds)</li> <li>• telemetry_timeout - 100 (integer: seconds)</li> <li>• probe_timeout: 5 (integer: seconds)</li> <li>• log_config_diff - True (boolean)</li> </ul>

From the left navigation menu, navigate to **Devices > System Agents > Agents** to go to the agent list view.

Juniper Apstra™

Devices > Agents

ONBOX 4 OFFBOX 0

Advanced Settings

Create Onbox Agent(s)

1-4 of 4

Columns (10/10) Page Size: 25

Address	Operation Mode	Platform	Platform Version	Apstra Version	Job State	System ID	Hostname	Device State	Actions
10.85.68.152	FULL CONTROL	EOS	4.24.5M	4.0.1-1044	SUCCESS	JPC	spine1	IS-ACTIVE	[Check] [Uninstall] [Revert] [Refresh] [Filter] [Reset] [Delete]
10.85.68.151	FULL CONTROL	EOS	4.24.5M	4.0.1-1044	SUCCESS	JPC	jtac-rack2-001-leaf1	IS-ACTIVE	[Check] [Uninstall] [Revert] [Refresh] [Filter] [Reset] [Delete]
10.85.68.150	FULL CONTROL	EOS	4.24.5M	4.0.1-1044	SUCCESS	SSJ	jtac-rack1-001-leaf2	IS-ACTIVE	[Check] [Uninstall] [Revert] [Refresh] [Filter] [Reset] [Delete]
10.85.68.149	FULL CONTROL	EOS	4.24.5M	4.0.1-1044	SUCCESS	SSJ	jtac-rack1-001-leaf1	IS-ACTIVE	[Check] [Uninstall] [Revert] [Refresh] [Filter] [Reset] [Delete]

You can select one or more agents or select actions for individual agents.

Juniper Apstra™

Devices > Agents

ONBOX 5 OFFBOX 0

Advanced Settings

Create Onbox Agent(s)

1-5 of 5

Columns (10/10) Page Size: 25

Check Uninstall Install OS Upgrade Delete

Filter selected by all selected only unselected only

Device Address	Operation Mode	Platform	Platform Version	Apstra Version	Job State	System ID	Hostname	Device State	Actions
10.85.68.153	FULL CONTROL	EOS	4.24.5M	4.0.1-1044	SUCCESS	JPC	localhost	OOS-QUARANTINED	[Check] [Uninstall] [Revert] [Refresh] [Filter] [Reset] [Delete]
10.85.68.149	FULL CONTROL	EOS	4.24.5M	4.0.1-1044	SUCCESS	SSJ	jtac-rack1-001-leaf1	IS-ACTIVE	[Check] [Uninstall] [Revert] [Refresh] [Filter] [Reset] [Delete]
10.85.68.150	FULL CONTROL	EOS	4.24.5M	4.0.1-1044	SUCCESS	SSJ	jtac-rack1-001-leaf2	IS-ACTIVE	[Check] [Uninstall] [Revert] [Refresh] [Filter] [Reset] [Delete]
10.85.68.151	FULL CONTROL	EOS	4.24.5M	4.0.1-1044	SUCCESS	JPC	jtac-rack2-001-leaf1	IS-ACTIVE	[Check] [Uninstall] [Revert] [Refresh] [Filter] [Reset] [Delete]
10.85.68.152	FULL CONTROL	EOS	4.24.5M	4.0.1-1044	SUCCESS	JPC	spine1	IS-ACTIVE	[Check] [Uninstall] [Revert] [Refresh] [Filter] [Reset] [Delete]

You can delete an agent only if that agent has been uninstalled and is no longer running on a device.

Additional actions are available on the line with the agent. For example, if a device is not assigned to a blueprint, you can restore the device's pristine configuration by clicking the **Revert to Pristine Config** button (as of Apstra version 4.0.1).

Juniper Apstra™

Devices > Agents

ONBOX 5 OFFBOX 0

Advanced Settings

Create Onbox Agent(s)

Query: All

1-5 of 5

Columns (10/10) Page Size: 25

Filter selected by ☒ all ☐ selected only ☐ unselected only

Device Address	Operation Mode	Platform	Platform Version	Apstra Version	Job State	System ID	Hostname	Device State	Actions
10.85.68.153	FULL CONTROL	EOS	4.24.5M	4.0.1-1044	SUCCESS	JP1	localhost	OOS-QUARANTINED	✓, ⚙, 🔄, 📄, 🗑, 🔄, 📄, 🗑
10.85.68.149	FULL CONTROL	EOS	4.24.5M	4.0.1-1044	SUCCESS	SS	j1ac-rack1-001-leaf1	IS-ACTIVE	✓, ⚙, 🔄, 📄, 🗑, 🔄, 📄, 🗑
10.85.68.150	FULL CONTROL	EOS	4.24.5M	4.0.1-1044	SUCCESS	SS	j1ac-rack1-001-leaf2	IS-ACTIVE	✓, ⚙, 🔄, 📄, 🗑, 🔄, 📄, 🗑
10.85.68.151	FULL CONTROL	EOS	4.24.5M	4.0.1-1044	SUCCESS	JP1	j1ac-rack2-001-leaf1	IS-ACTIVE	✓, ⚙, 🔄, 📄, 🗑, 🔄, 📄, 🗑
10.85.68.152	FULL CONTROL	EOS	4.24.5M	4.0.1-1044	SUCCESS	JP1	spine1	IS-ACTIVE	✓, ⚙, 🔄, 📄, 🗑, 🔄, 📄, 🗑

Revert to Pristine  
Collect Pristine  
Show Log  
Cancel Job

## Create On-box Agent

To create on-box agents, you must have **full admin / root** privileges. We recommend creating a dedicated user on the device using ["Apstra ZTP" on page 267](#) or other means. Make sure that you've:

- Added login credentials for the devices.
- Configured management IP connectivity between devices and the Apstra server. You must do this before installing agents so it's out-of-band (OOB). Configuring management connectivity in-band (through the fabric) is not supported and could cause connectivity issues when changes are made to the blueprint.
- Uploaded required packages.

Before creating/installing on-box device agents on Cisco NX-OS and Arista EOS, configure the following minimum configuration on them as shown below. (Cumulus Linux and SONiC Enterprise have no specific configuration requirements other than Management Network and privileged user access.)

### Cisco NX-OS On-box Agent Minimum Configuration

```
!
copp profile strict
!
username admin password <admin-password> role network-admin
!
vrf context management
  ip route 0.0.0.0/0 <management-default-gateway>
!
interface mgmt0
```

```
ip address <address>/<cidr>
!
```

### Arista EOS On-box Agent Minimum Configuration

```
!
service routing protocols model multi-agent
!
aaa authorization exec default local
!
username admin privilege 15 role network-admin secret <admin-password>
!
interface Management1
    ip address <address>/<cidr>
!
ip route vrf management 0.0.0.0/0 <management-default-gateway>
!
```

1. Confirm that you've installed the minimum configuration as described above, and that the device doesn't contain configuration that would raise validation errors. (For details, see Agent overview.)
2. From the left navigation menu, navigate to **Devices > System Agents > Agents** and click **Create Onbox Agent(s)**.
3. Specify agent details as described in the Device Agent Overview section above.
4. Click **Create**. While the task is active you can view its progress at the bottom of the screen in the **Active Jobs** section. The job status changes from **Initialized** to **In Progress** to **Succeeded**.

### Create Off-box Agent

Make sure that you've:

- Added login credentials for the devices.
- Configured management IP connectivity between devices and the Apstra server. You must do this before installing agents so it's out-of-band (OOB). Configuring management connectivity in-band (through the fabric) is not supported and could cause connectivity issues when changes are made to the blueprint.
- Uploaded required packages.
- On Juniper devices, add Junos license configuration. (This is *not* the preferred method for adding license configuration. For more information, see ["Juniper Device Agent" on page 195.](#))

Before creating/installing off-box device agents on Juniper Junos, Cisco NX-OS and Arista EOS, configure the following minimum configuration on them as shown below.

## Juniper Junos Off-box Agent Minimum Configuration

```

system {
  login {
    user aosadmin {
      uid 2000;
      class super-user;
      authentication {
        encrypted-password "xxxxx";
      }
    }
  }
  services {
    ssh;
    netconf {
      ssh;
    }
  }
  management-instance;
}
interfaces {
  em0 {
    unit 0 {
      family inet {
        address <address>/<cidr>;
      }
    }
  }
}
routing-instances {
  mgmt_junos {
    routing-options {
      static {
        route 0.0.0.0/0 next-hop <management-default-gateway>;
      }
    }
  }
}
}

```

For more information, see ["Juniper Device Agent" on page 195](#).

## Cisco NX-OS Off-box Agent Minimum Configuration

```

!
feature nxapi
feature bash-shell
feature scp-server
feature evmed
copp profile strict
nxapi http port 80
!
username admin password <admin-password> role network-admin
!
vrf context management
    ip route 0.0.0.0/0 <management-default-gateway>
!
nxapi http port 80
!
interface mgmt0
    ip address <address>/<cidr>
!

```

## Arista EOS Off-box Agent Minimum Configuration

```

!
service routing protocols model multi-agent
!
aaa authorization exec default local
!
username admin privilege 15 role network-admin secret <admin-password>
!
vrf definition management
    rd 100:100
!
interface Management1
    vrf forwarding management
    ip address <address>/<cidr>
!
ip route vrf management 0.0.0.0/0 <management-default-gateway>
!
management api http-commands
    protocol http

```

```

no shutdown
!
vrf management
    no shutdown
!

```

1. Confirm that you've installed the minimum configuration as described above, and that the device doesn't contain configuration that would raise validation errors. (For details, see Agent overview.).
2. From the left navigation menu, navigate to **Devices > System Agents > Agents**, click the **OFFBOX** tab, and click **Create Offbox Agent(s)**.
3. Specify agent details as described in the Device Agent Overview section.
4. Click **Create**. While the task is active you can view its progress at the bottom of the screen in the **Active Jobs** section. The job status changes from **Initialized** to **In Progress** to **Succeeded**.

## Uninstall Agent

If you are uninstalling an agent because you want to remove a device from Apstra management, see the device guide for ["removing a device" on page 166](#) for the complete workflow.

1. From the left navigation menu, navigate to **Devices > System Agents > Agents** and click the **Uninstall** button for the agent to uninstall.
2. Click **Confirm** to start the uninstall process and return to the list view.


Beginning in Apstra 4.0.1, by default, before you uninstall an agent from a device you must remove the device from the blueprint, restoring the pristine configuration to the device. If you wish to remove the device from the blueprint and remove the agent from the device, you must follow the following workflow.

1. From the left navigation menu, navigate to **Devices > Managed Devices** and set the device admin state to **MAINT**. See ["Managed Devices" on page 132](#) for information.
2. Unassign the device from the blueprint and commit changes.

- From the left navigation menu, navigate to **Devices > System Agents > Agents**, select **Advanced Settings** and select **Skip Revert to Pristine on Uninstall**.

### Advanced Settings

☐ **Skip Pristine Configuration Validation**  
Pristine Configuration Validation is done as part of Apstra agent installation to check if the initial device configuration contains any configuration that may conflict with future Apstra managed configuration and abort the agent installation to avoid future problems. Checking this option is not recommended but can be done to force install Apstra agents even if possible conflicting configuration is found


☒ **Skip Revert to Pristine on Uninstall**  
When uninstalling Apstra agents, the device configuration is automatically reverted back to its Pristine Configuration. Checking this option will keep the device configuration untouched when Apstra agent is uninstalled

Update

- Uninstall device system agent.

## Edit / Delete Agent

### IN THIS SECTION

- Edit Agent | [194](#)
- Delete Agent | [194](#)

### Edit Agent



**CAUTION:** Changing a user requires completely re-onboarding the device. Changing the password involves several steps that are not straightforward (changing the password on the device, device agents, and pristine config). If you need to change a password, we recommend contacting ["Juniper Support" on page 777](#).

- From the left navigation menu, navigate to **Devices > System Agents > Agents** and click the **Edit** button for the agent to edit.
- Make your changes (including package upgrades that you previously uploaded).
- Click **Update** to update the agent and return to the list view.

### Delete Agent

If you are deleting an agent because you want to remove a device from Apstra management, see the device guides for the complete workflow for ["removing a device" on page 166](#). There are additional steps before and after deleting the agent.

As of Apstra version 4.0.1, you must remove devices from a blueprint and uninstall agents before you can delete a device system agent.

1. From the left navigation menu, navigate to **Devices > System Agents > Agents** and click the **Delete** button for the agent to delete.
2. Click **Delete** to delete the agent and return to the list view.

## Juniper Device Agent

### IN THIS SECTION

- [Juniper ZTP | 195](#)
- [Disable ZTP | 195](#)
- [Apply Initial Juniper Junos Configuration | 196](#)
- [Configure super-user User | 197](#)
- [Configure IP address and Management VRF | 198](#)
- [Configure SSH and NETCONF | 199](#)
- [Add Junos License Configuration | 199](#)

This document describes how to manually install Juniper device agents.

### Juniper ZTP

For an option that's simpler and easier to support at scale, see ["Apstra ZTP" on page 267](#), which shows you how to automatically boot and install Apstra device agents and prerequisite switch configuration.

### Disable ZTP

If you want to install agents manually because a previous attempt to install them with Apstra ZTP failed, you must first delete the ZTP mode (since it remains active) with the command `delete chassis auto-image-upgrade`.

If you're going to provision the Juniper switch without ZTP (ZTP Disabled), make sure that the ZTP process is disabled before proceeding. After logging into the switch for the first time and setting system root-authentication, configure `delete chassis auto-image-upgrade`.

```
{master:0}
root> edit
```

```

Entering configuration mode

{master:0}[edit]
root# delete chassis auto-image-upgrade

{master:0}[edit]
root# commit and-quit
configuration check succeeds
commit complete
Exiting configuration mode

{master:0}
root>

```

### Apply Initial Juniper Junos Configuration

Before installing Apstra device system agents on Juniper Junos devices, apply the minimum configuration below to the devices.

```

system {
  login {
    user aosadmin {
      uid 2000;
      class super-user;
      authentication {
        encrypted-password "xxxxx";
      }
    }
  }
  services {
    ssh;
    netconf {
      ssh;
    }
  }
  management-instance;
}
interfaces {
  em0 {
    unit 0 {
      family inet {

```

```

        address <address>/<cidr>;
    }
}
}
}
routing-instances {
    mgmt_junos {
        routing-options {
            static {
                route 0.0.0.0/0 next-hop <management-default-gateway>;
            }
        }
    }
}
}

```

### Configure super-user User

For the device system agent to connect to the Juniper Junos device, you must configure a local device user with class super-user.

```

{master:0}
root> edit
Entering configuration mode

{master:0}[edit]
root# set system login user aosadmin class super-user

{master:0}[edit]
root# set system login user aosadmin authentication plain-text-password
New password:
Retype new password:

{master:0}[edit]
root# commit and-quit
configuration check succeeds
commit complete
Exiting configuration mode

{master:0}
root>

```

**NOTE:** If you intend to use a different authentication method for device access (such as RADIUS), you must use local password authentication first.

```
system authentication-order [ password radius ]
```

## Configure IP address and Management VRF

Device system agents use the Junos `mgmt_junos` management-instance VRF and the management interface (such as `em0`).

```
{master:0}
root> edit
Entering configuration mode

{master:0}[edit]
root# set system management-instance

{master:0}[edit]
root# set interfaces em0.0 family inet address 192.168.59.11/24

{master:0}[edit]
root# set routing-instances mgmt_junos routing-options static route 0.0.0.0/0 next-hop
192.168.59.1

{master:0}[edit]
root# commit and-quit
configuration check succeeds
commit complete
Exiting configuration mode

{master:0}
root>
```

If the Juniper device uses a different management interface (such as `vme.0`), configure the management IP address on it instead.

## Configure SSH and NETCONF

Device system agents require Junos SSH and NETCONF access to be configured under system services.

```
{master:0}
root> edit
Entering configuration mode

{master:0}[edit]
root# set system services ssh

{master:0}[edit]
root# set system services netconf ssh

{master:0}[edit]
root# commit and-quit
configuration check succeeds
commit complete
Exiting configuration mode

{master:0}
root>
```

## Add Junos License Configuration

You can add license configuration before installing the system agent (to make it part of the pristine configuration), but the preferred method is to add license configuration with ["configlets" on page 119](#).

## SONiC Device Agent

### IN THIS SECTION

- [SONiC Device Agent Overview | 200](#)
- [Configure Management IP Manually \(SONiC\) | 200](#)
- [Install Agent Manually \(SONiC\) | 202](#)
- [Uninstall Agent Manually \(SONiC\) | 206](#)

## SONiC Device Agent Overview

Although the preferred method of installing ["device system agents" on page 184](#) is by creating agents in the Apstra GUI, you *can* manually install Apstra agents from the CLI. Only in rare exceptions is it needed to manually install agents, which is error-prone and requires more effort. An in-depth understanding of the various device states, configuration stages, and agent operations is required before manually installing agents. For assistance, contact ["Juniper Support" on page 777](#).

The only Software for Open Networking in the Cloud (SONiC) distribution that is supported is SONiC Enterprise (starting with Apstra version 3.3.0a).

The SONiC device agent manages the following files in the filesystem:

- `/etc/sonic/config_db.json` - The main configuration file for SONiC, specifying interfaces, IP addresses, port breakouts etc.
- `/etc/sonic/frr/frr.conf` - `frr.conf` contains all of the routing application configuration for BGP on the device.



**CAUTION:** Do not edit the `config_db.json` or `frr.conf` files manually at any time, before or after device system agent installation. The agent overwrites any existing configuration in these files.

## Configure Management IP Manually (SONiC)

SONiC automatically creates a management VRF for the "eth0" management interface. By default, "eth0" gets a DHCP address from the management network. In most cases, no management configuration should be needed.

However, if you need to manually configure a SONiC device management IP address, you **must** configure it using the `sonic-cli` interface.

```
admin@sonic:~$ sonic-cli
sonic# show interface Management 0
eth0 is up, line protocol is up
Hardware is MGMT
Description: Management0
Mode of IPV4 address assignment: not-set
Mode of IPV6 address assignment: not-set
IP MTU 1500 bytes
LineSpeed 1GB, Auto-negotiation True
Input statistics:
    11 packets, 1412 octets
```

```

    0 Multicasts, 0 error, 4 discarded
Output statistics:
    31 packets, 5290 octets
    0 error, 0 discarded
sonic# configure terminal
sonic(config)# interface Management 0
sonic(conf-if-eth0)# ip address 192.168.59.7/24 gwaddr 192.168.59.1
sonic(conf-if-eth0)# exit
sonic(config)# exit
sonic# write memory
sonic# show interface Management 0
eth0 is up, line protocol is up
Hardware is MGMT
Description: Management0
IPV4 address is 192.168.59.7/24
Mode of IPV4 address assignment: MANUAL
Mode of IPV6 address assignment: not-set
IP MTU 1500 bytes
LineSpeed 1GB, Auto-negotiation True
Input statistics:
    18 packets, 2494 octets
    0 Multicasts, 0 error, 6 discarded
Output statistics:
    38 packets, 6455 octets
    0 error, 0 discarded
sonic#

```

You can check the Managment VRF from the SONiC Linux command line.

```

admin@leaf1:~$ show mgmt-vrf

ManagementVRF : Enabled

Management VRF interfaces in Linux:
48: mgmt: <NOARP,MASTER,UP,LOWER_UP> mtu 65536 qdisc noqueue state UP mode DEFAULT group default
qlen 1000
    link/ether 8e:32:49:6c:ec:71 brd ff:ff:ff:ff:ff:ff promiscuity 0
    vrf table 5000 addrngenmode eui64 numtxqueues 1 numrxqueues 1 gso_max_size 65536 gso_max_segs
65535
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast master mgmt state UP mode
DEFAULT group default qlen 1000
    link/ether 52:54:00:c1:ac:1b brd ff:ff:ff:ff:ff:ff

```

```
49: lo-m: <BROADCAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc noqueue master mgmt state UNKNOWN mode
DEFAULT group default qlen 1000
    link/ether c2:39:a7:6c:4b:be brd ff:ff:ff:ff:ff:ff
admin@leaf1:~$ show mgmt-vrf routes
```

```
Routes in Management VRF Routing Table:
default via 172.20.9.1 dev eth0 metric 201
broadcast 127.0.0.0 dev lo-m proto kernel scope link src 127.0.0.1
127.0.0.0/8 dev lo-m proto kernel scope link src 127.0.0.1
local 127.0.0.1 dev lo-m proto kernel scope host src 127.0.0.1
broadcast 127.255.255.255 dev lo-m proto kernel scope link src 127.0.0.1
broadcast 172.20.9.0 dev eth0 proto kernel scope link src 172.20.9.7
172.20.9.0/24 dev eth0 proto kernel scope link src 172.20.9.7
local 172.20.9.7 dev eth0 proto kernel scope host src 172.20.9.7
broadcast 172.20.9.255 dev eth0 proto kernel scope link src 172.20.9.7
admin@leaf1:~$
```

## Install Agent Manually (SONiC)

To manually install SONiC device agents you'll download, install and configure the agent software, then *acknowledge* it to bring it under Apstra management.

1. Download the Apstra agent with the `sudo cgexec -g l3mdev:mgmt curl -o /tmp/aos.run -k -O https://{aos-ip-address}/device_agent_images/aos_device_agent{{aos-version}}-{{aos-build}}.run` command.

```
admin@sonic:~$ sudo cgexec -g l3mdev:mgmt curl -o /tmp/aos.run -k -O
https://172.20.74.3/device_agent_images/aos_device_agent_3.3.0a-93.run
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload  Total  Spent    Left  Speed
100 111M 100 111M    0     0 328M      0 --:--:-- --:--:-- --:--:-- 328M
admin@sonic:~$
```

2. Install the Apstra agent with the `sudo /bin/bash /tmp/aos.run -- --no-start` command.

```
admin@sonic:~$ sudo /bin/bash /tmp/aos.run -- --no-start
Verifying archive integrity... All good.
Uncompressing AOS Device Agent installer 100%
+ set -o pipefail
+++ dirname ./agent_installer.sh
++ cd .
++ pwd
+ script_dir=/tmp/selfgz334323135
```

```

+ systemd_available=false
++ date
+ echo 'Device Agent Installation : Mon' Oct 19 19:02:01 UTC 2020
Device Agent Installation : Mon Oct 19 19:02:01 UTC 2020
+ echo

+ UNKNOWN_PLATFORM=1
+ WRONG_PLATFORM=1
+ CANNOT_EXECUTE=126
+ '[' 0 -ne 0 ']'
+ arg_parse --no-start
+ start_aos=True
+ [[ 1 > 0 ]]
+ key=--no-start
+ case $key in
+ start_aos=False
+ shift
+ [[ 0 > 0 ]]
+ supported_platforms=(["centos"]="install_centos" ["eos"]="install_on_arista"
["nxos"]="install_on_nxos" ["cumulus"]="install_sysvinit_deb" ["opx"]="install_systemd_deb
opx" ["trusty"]="install_sysvinit_deb" ["xenial"]="install_sysvinit_deb"
["icos"]="install_sysvinit_rpm" ["snaproute"]="install_sysvinit_deb"
["simulation"]="install_sysvinit_deb" ["sonic"]="install_systemd_deb sonic"
["bionic"]="install_sysvinit_deb")
+ declare -A supported_platforms
++ /tmp/selfgz334323135/aos_get_platform
+ current_platform=sonic
+ installer='install_systemd_deb sonic'
+ [[ -z install_systemd_deb sonic ]]
+++ readlink /sbin/init
++ basename /lib/systemd/systemd
+ [[ systemd == systemd ]]
+ systemd_available=true
+ [[ -x /etc/init.d/aos ]]
+ echo 'Stopping AOS'
Stopping AOS
+ true
+ systemctl stop aos
+ install_systemd_deb sonic
++ pwd
+ local pkg_dir=/tmp/selfgz334323135/sonic
+ install_deb /tmp/selfgz334323135/sonic
+ local pkg_dir=/tmp/selfgz334323135/sonic

```

```

+ dpkg -s aos-device-agent
+ dpkg --purge aos-device-agent
(Reading database ... 34189 files and directories currently installed.)
Removing aos-device-agent (3.3.0a-93) ...
Purging configuration files for aos-device-agent (3.3.0a-93) ...
Processing triggers for systemd (232-25+deb9u12) ...
+ dpkg -i /tmp/selfgz334323135/sonic/aos-device-agent-3.3.0a-93.amd64.deb
Selecting previously unselected package aos-device-agent.
(Reading database ... 34180 files and directories currently installed.)
Preparing to unpack .../aos-device-agent-3.3.0a-93.amd64.deb ...
Unpacking aos-device-agent (3.3.0a-93) ...
Setting up aos-device-agent (3.3.0a-93) ...
Synchronizing state of aos.service with SysV service script with /lib/systemd/systemd-sysv-
install.
Executing: /lib/systemd/systemd-sysv-install enable aos
/var/lib/dpkg/info/aos-device-agent.postinst: line 7: /usr/sbin/aosconfig: No such file or
directory
Processing triggers for systemd (232-25+deb9u12) ...
+ mkdir -p /opt/aos
+ cp aos_device_agent.img /opt/aos
+ post_install_common
+ /etc/init.d/aos config_gen
+ [[ False == \T\r\u\e ]]
+ true
+ systemctl enable aos
Synchronizing state of aos.service with SysV service script with /lib/systemd/systemd-sysv-
install.
Executing: /lib/systemd/systemd-sysv-install enable aos
admin@sonic:~$

```

3. Set the IP of the Apstra server and enable configuration service by editing `/etc/aos/aos.conf` with the `sudo vi /etc/aos/aos.conf` command.

- For the following, replace "aos-server" with the IP address or valid FQDN of your Apstra server.

```

[controller]
# <metadb> provides directory service for AOS. It must be configured properly
# for a device to connect to AOS controller.
metadb = tbt://aos-server:29731

```

- For example

```
[controller]
# <metadb> provides directory service for AOS. It must be configured properly
# for a device to connect to AOS controller.
metadb = tbt://172.20.74.3:29731
```

- For the following, add the management interface (usually eth0).

```
# <interface> is used to specify the management interface. This is currently
# being used only on server devices and the AOS agent on the server device will
# not come up unless this is specified.
interface = eth0
```

- For the following, set "enable\_configuration\_service" to **1** to enable "full control" mode from Apstra.

```
[service]
# AOS device agent by default starts in "telemetry-only" mode. Set following
# variable to 1 if you want AOS agent to manage the configuration of your
# device.
enable_configuration_service = 1
```

- Add the following, "credential" configuration with "username = " and the local Linux user to be used for the agent (usually "admin").

```
[credential]
username = admin
```

4. Start the agent with the `sudo service aos start` command and check its status with the `sudo service aos status` command.

```
admin@sonic:~$ sudo service aos start
admin@sonic:~$ sudo service aos status
• aos.service - AOS Device Agent
  Loaded: loaded (/etc/systemd/system/aos.service; enabled; vendor preset: enabled)
  Active: active (running) since Mon 2020-10-19 19:22:50 UTC; 19s ago
  Process: 23375 ExecStart=/etc/init.d/aos start (code=exited, status=0/SUCCESS)
```

```

Main PID: 23521 (tacspawner)
  Tasks: 22 (limit: 4915)
  Memory: 367.1M
  CPU: 15.278s
  CGroup: /system.slice/aos.service
          └─23521 tacspawner --daemonize=/var/log/aos/aos.log --pidfile=/host_var_run/
aos.pid --name=5254001B4A4D --hostname=5254001B4A4D --domainSocket=aos_spawner_sock --hostS
          └─23528 tacsysdb --sysdbType=leaf --agentName=5254001B4A4D-
LocalTasks-5254001B4A4D-0 --partition= --storage-mode=persistent --eventLogDir=. --
eventLogSev=
          └─23541 /usr/bin/python /usr/bin/aos_agent --
class=aos.device.common.ProxyCountersAgent.ProxyCountersAgent --name=CounterProxyAgent
device_type=Sonic serial_number=@(S
          └─23544 /usr/bin/python /usr/bin/aos_agent --
class=aos.device.sonic.SonicTelemetryAgent.SonicTelemetryAgent --name=DeviceTelemetryAgent
serial_number=@(SYSTEM_UNIQUE_I
          └─23551 /usr/bin/python /usr/bin/aos_agent --
class=aos.device.common.DeviceKeeperAgent.DeviceKeeperAgent --name=DeviceKeeperAgent
serial_number=@(SYSTEM_UNIQUE_ID)
          └─23617 /usr/bin/python /usr/bin/aos_agent --
class=aos.device.common.ProxyDeploymentAgent.ProxyDeploymentAgent --name=DeploymentProxyAgent
device_type=Sonic serial_num
          └─25007 sh -c aos_host_exec show interface transceiver eeprom Ethernet12 2>&1
          └─25010 /usr/bin/python /usr/bin/show interface transceiver eeprom Ethernet12
admin@sonic:~$

```

5. From the left navigation menu in the Apstra GUI, navigate to **Devices > Managed Devices** to acknowledge the device, then you can assign it to a blueprint.

### Uninstall Agent Manually (SONiC)

To manually uninstall SONiC Apstra device agents you'll stop Apstra server, uninstall the agent, and remove any remaining Apstra files.

1. Stop the Apstra agent with the `sudo service aos stop` command.

```

admin@sonic:~$ sudo service aos stop
admin@sonic:~$

```

2. Uninstall the Apstra agent with the `sudo dpkg --purge --force-all aos-device-agent` command.

```

admin@sonic:~$ sudo dpkg --purge --force-all aos-device-agent
(Reading database ... 34189 files and directories currently installed.)

```

```
Removing aos-device-agent (3.3.0a-93) ...
Purging configuration files for aos-device-agent (3.3.0a-93) ...
Processing triggers for systemd (232-25+deb9u12) ...
admin@sonic:~$
```

3. Remove remaining Apstra files with the `sudo rm -fr /etc/aos /var/log/aos /mnt/persist/.aos /opt/aos /run/aos /run/lock/aos /tmp/aos_show_tech /usr/sbin/aos*` command.

```
admin@sonic:~$ sudo rm -fr /etc/aos /var/log/aos /mnt/persist/.aos /opt/aos /run/aos /run/
lock/aos /tmp/aos_show_tech /usr/sbin/aos*
admin@sonic:~$
```

## Cisco Device Agent

### IN THIS SECTION

- [Cisco NX-OS Device Agent Overview | 207](#)
- [Device Configuration Requirements | 208](#)
- [Resize and Enable Guestshell | 208](#)
- [Download Agent Installer | 209](#)
- [Install Cisco Device Agent | 210](#)
- [Device Agent Configuration File | 210](#)
- [Activate Apstra Devices on the Apstra Server | 211](#)
- [Deploy Device | 211](#)
- [Reset Apstra Device Agent | 211](#)
- [Uninstall Apstra Device Agent | 211](#)
- [Remove Apstra EEM Scripts | 212](#)
- [Cisco Agent Troubleshooting | 212](#)

## Cisco NX-OS Device Agent Overview

Although the preferred method of installing ["device system agents" on page 184](#) is by creating agents in the Apstra GUI, you *can* manually install Apstra agents from the CLI. Only in rare exceptions is it needed to manually install agents, which requires more effort and is error-prone. An in-depth understanding of the various device states, configuration stages, and agent operations is required before manually installing agents. For assistance, contact ["Juniper Support" on page 777](#).

Manually installing an agent for Cisco devices involves the following steps:

- Modify the guestshell disk size, memory and cpu, and restart the guestshell in order to take effect.
- Copy the device agent from the Apstra Server and installing it.
- Modify the aos\_config file.



**CAUTION:** The Cisco GuestShell is not partitioned to be unique with Apstra. If there are other applications hosting on the guestshell, any changes in the guestshell could impact them.



**CAUTION:** Commands in the "Bootstrap" or "Pristine" configuration may interfere with Apstra configuration added during fabric deployment.  
Adding NX-OS configuration "system jumbomtu" with a value lower than MTUs used by Apstra causes Apstra MTU commands to fail.

### Device Configuration Requirements

Configuration steps must happen in order on NX-OS - VRF, NXAPI, GuestShell, Create Management VRF. Apstra's device agent requires the use of VRF of the name `management` to allow for agent-server communication. Ensure these lines appear in the running configuration.

```
!
no password strength-check
username admin password admin-password role network-admin
copp profile strict
!
vrf context management
  ip route 0.0.0.0/0 <Management Default Gateway>
!
interface mgmt0
  vrf member management
  ip address <Management CIDR Address>
!
```

### Resize and Enable Guestshell

Either the guestshell is running or not restarting or enabling the service is required after the the following step.

Resize the guestshell disk space, memory and cpu by executing the next commands:

```
guestshell resize rootfs 1024
guestshell resize memory 2048
guestshell resize cpu 6
```

If the guestshell is not enable, proceed to activate it by executing "guestshell enable", otherwise, if it was already running please run "guestshell reboot" command in order to restart the shell.

Verify that the guestshell is activated again:

```
switch# show guestshell detail
```

## Download Agent Installer

We can easily copy the installation agents over HTTPS from the Apstra server. After downloading, confirm the MD5sum of your downloaded copy matches what Apstra stores.

**NOTE:** The Cisco device needs to connect to the Apstra server using HTTPS in order to retrieve the agent file, please make sure that this connectivity is OK before proceeding.

Apstra ships the agent from the Apstra Server. We can copy it to the /volatile, or volatile: filesystem location. Apstra also ships with an md5sum file in the /home/admin folder on the Apstra Server.

Replace the aos\_server\_ip variable and aos\_version from the run file below, you can find this exact version from the Apstra Server, Platform --> About (i.e '3.2.2-12')

```
switch# guestshell run sudo chvrf management wget --no-check-certificate -o /volatile/
aos_download.log
-O /volatile/aos.run https://<aos_server_ip>/device_agent_images/
aos_device_agent_<aos_version>.run

guestshell run sudo chvrf management wget --no-check-certificate -o /volatile/aos_download.log
-O /volatile/aos.run.md5 https://<aos_server_ip>/device_agent_images/
aos_device_agent_<aos_version>.run.md5
```

Validate that the file was downloaded correctly.

```
switch# show file volatile:aos.run md5
a28780880a8d674f6eb6a397509db101

switch# show file volatile:aos.run.md5
a28780880a8d674f6eb6a397509db101  aos_device_agent_<aos_version>.run
```

## Install Cisco Device Agent

The Apstra agent on Cisco is installed by running it as a shell script directly as root on the Cisco NXOS switch. This command must be done within the guest shell. After installing the agent and before starting the service, aos.conf file needs to be modified to connect to the server.

**NOTE:** We recommend that you save your current running-config to the startup-config 'copy running-config startup-config' to save your latest changes in case of any issue.

```
switch# guestshell run sudo chmod +x /volatile/aos.run
switch# guestshell run sudo /volatile/aos.run -- --no-start
<omitted output>
created 7855 files
created 1386 directories
created 602 symlinks
created 0 devices
created 0 fifos
+ [[ True == \T\r\u\e ]]
+ true
+ systemctl enable aos
```

Change the required parameters in the Apstra configuration file before enabling the Apstra service (see next steps).

## Device Agent Configuration File

You can configure the Cisco NX-OS device agent configuration file directly at `/etc/aos/aos.conf`. See ["Apstra device agent configuration file" on page 901](#) for parameters. After updating the file, start the Apstra device agent. with the command `service aos start`.

## Activate Apstra Devices on the Apstra Server

When the Apstra device agent communicates with Apstra, it uses a 'device key' to identify itself. For Cisco NXOS switches, the device key is the MAC address of the management interface 'eth0'.

```
root@Cisco:/etc/aos# ip link show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT qlen
1000
link/ether 08:00:27:8a:39:05 brd ff:ff:ff:ff:ff:ff
```

## Deploy Device

From the left navigation menu of the Apstra GUI, navigate to **Devices > Managed Devices**. When the agent is up and running it appears in this list, and can be acknowledged and assigned to a Blueprint using the GUI per standard procedure.

## Reset Apstra Device Agent

If you need to reset the Apstra agent for some reason (changing blueprints, redeploying, restoring device from backup, etc.) it's best to clear the Apstra agent metadata, re-register the device, and redeploy to the blueprint.

```
C9K-172-20-65-5# guestshell
[guestshell@guestshell ~]$ sudo su -
[root@guestshell ~]# systemctl stop aos
[root@guestshell ~]# rm -rf /var/log/aos/*
[root@guestshell ~]# systemctl start aos

Starting AOS Agents...root@guestshell ~]#
```

## Uninstall Apstra Device Agent

To uninstall the agent, first undeploy and unassign it from the blueprint per standard procedures using the GUI. You can also delete it entirely from the Managed Devices page.

To remove the Apstra package from NX-OS, destroy the guestshell. Do this only if no other applications are using the guestshell:

```
C9K-172-20-65-5# guestshell destroy
```

Remove remaining AOS data from system

Removing the guest-shell deletes most of the data left by AOS. Some files are still on the bootflash:/.aos folder.

```
C9K-172-20-65-5# delete bootflash:./aos no-prompt
```

## Remove Apstra EEM Scripts

The Apstra device agent installs some event manager applets to assist with telemetry. These can be safely removed

```
C9K-172-20-65-5(config)# no event manager applet AOS_PROTO_VSH_LAUNCH
```

```
C9K-172-20-65-5(config)# no event manager applet AOS_STATS_VSH_LAUNCH
```

```
C9K-172-20-65-5(config)# no event manager applet aos_bgp_applet C9K-172-20-65-5(config)# no
event manager applet aos_ifdown_applet C9K-172-20-65-5(config)# no event manager applet
aos_ifup_applet
```

## Cisco Agent Troubleshooting

### IN THIS SECTION

- [Confirm Network Reachability to Apstra | 214](#)
- [Confirm Agent Installation | 214](#)
- [Check that Apstra Agent is Running | 215](#)
- [Check for Presence of Files in /etc/aos | 216](#)
- [Check for Apstra Data in /var/log/aos | 216](#)
- [Determine Apstra Agent Version | 217](#)
- [DNS Resolution Failure | 217](#)
- [Apstra Service Takes Long Time to Start on Cisco NX-OS | 218](#)
- [Apstra Stops and ails Without Errors \(MGMT VRF\) | 218](#)
- [Verify MGMT VRF in NX-OS Guest Shell | 219](#)

The Apstra agent runs under the NXOS guestshell to interact with the underlying bash and Linux environments. This is an internal Linux Container (LXC) in which Apstra operates. Under LXC, Apstra makes use of the NXAPI and other methods to directly communicate with NXOS. For security reasons, Cisco partitions much of the LXC interface away from the rest of the NXOS device, so we must drop to the guest shell bash prompt to perform more troubleshooting commands.

Confirm the Guest Shell is running on NX-OS The Apstra agent runs under the NXOS Guest Shell to interact with the underlying bash and linux environments. This is an internal Linux Container (LXC) in which Apstra operates. We are checking to make sure the guest shell is activated and running.

```
C9K-172-20-65-5# show guestshell detail
Virtual service guestshell+ detail
  State           : Activated
  Package information
Name              : guestshell.ova
Path              : /isanboot/bin/guestshell.ova
Application
  Name            : GuestShell
  Installed version : 2.1(0.0)
  Description     : Cisco Systems Guest Shell
Signing
  Key type        : Cisco release key
  Method          : SHA-1
Licensing
  Name            : None
  Version         : None
Resource reservation
Disk              : 1024 MB
Memory           : 3072 MB
CPU              : 6% system CPU

Attached devices
Type      Name      Alias
-----
Disk      _rootfs
Disk      /cisco/core
Serial/shell
Serial/aux
Serial/Syslog      serial2
Serial/Trace       serial3
```

Showing registered services

```
C9K-172-20-65-5# show virtual-service list
```

```
Virtual Service List:
```

Name	Status	Package Name
-----		
guestshell+	Activated	guestshell.ova

### ***Confirm Network Reachability to Apstra***

Check ICMP Ping to the Apstra Server by pinging within the guest shell. On NXOS, we have to use the 'chvrf <vrf>' command to run commands within the context of a VRF. In this case, 'management' VRF.

```
[guestshell@guestshell ~]$ chvrf management ping 172.20.65.3
PING 172.20.65.3 (172.20.65.3) 56(84) bytes of data.
64 bytes from 172.20.65.3: icmp_seq=1 ttl=64 time=0.239 ms
64 bytes from 172.20.65.3: icmp_seq=2 ttl=64 time=0.215 ms
```

### ***Confirm Agent Installation***

Check if the Apstra device agent package is installed. In NXOS, the Apstra agent installs to /etc/rc.d/init.d/aos to start when the guestshell instance starts.

```
[guestshell@guestshell ~]$ systemctl status aos
aos.service - LSB: Start AOS device agents
   Loaded: loaded (/etc/rc.d/init.d/aos)
   Active: active (running) since Tue 2016-11-15 00:10:49 UTC; 3h 54min ago
   Process: 30 ExecStart=/etc/rc.d/init.d/aos start (code=exited, status=0/SUCCESS)
   CGroup: /system.slice/aos.service
           └─113 tacspawner --daemonize=/var/log/aos/aos.log --pidfile=/var/run/aos.pid --
name=SAL2028T5NE --hostname=localhost --domainSocket=aos_spawner_sock --hostSysdbAddress=tb...
           └─115 tacleafsysdb --agentName=SAL2028T5NE-LocalTasks-SAL2028T5NE-0 --partition= --
storage-mode=persistent --eventLogDir=. --eventLogSev=TaccSpawner/error,Mounter/error,M...
           └─116 /usr/bin/python /bin/aos_agent --
class=aos.device.common.ProxyDeploymentAgent.ProxyDeploymentAgent --name=DeploymentProxyAgent
device_type=Cisco serial_number=@(SWI...
           └─117 /usr/bin/python /bin/aos_agent --
class=aos.device.common.ProxyCountersAgent.ProxyCountersAgent --name=CounterProxyAgent
device_type=Cisco serial_number=@(SWITCH_UNI...
           └─118 /usr/bin/python /bin/aos_agent --
class=aos.device.cisco.CiscoTelemetryAgent.CiscoTelemetryAgent --name=DeviceTelemetryAgent
serial_number=@(SWITCH_UNIQUE_ID)
```

### Check that Apstra Agent is Running

Check the running system state with the 'service' command, and check running processes with the 'ps' command. We are looking to confirm aos\_agent is running properly.

```
[root@guestshell ~]# service aos status
aos is running

[root@guestshell ~]# ps wax
  PID TTY          STAT       TIME COMMAND
  1 ?        Ss   0:00   /sbin/init
  9 ?        Ss   0:00   /usr/lib/systemd/systemd-journald
  19 ?       Ss   0:00   /bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-
activation
  22 ?       Ss   0:00   /usr/lib/systemd/systemd-logind
  29 ?       Ss   0:00   /usr/sbin/sshd -D -f /etc/ssh/sshd_config-cisco -p 17682 -o
ListenAddress=localhost
  38 ?       Ss   0:00   /usr/sbin/crond -n
  55 pts/1Ss+0:00 /sbin/agetty --noclear ttyS1
  56 pts/0Ss+0:00 /sbin/agetty --noclear ttyS0
 113 ?      Sl   0:01   tacspawner --daemonize=/var/log/aos/aos.log --pidfile=/var/run/aos.pid --
name=C9K --hostname=localhost --domainSocket=aos_spawner_sock --hostSysdbAdd
 115 ?      S    0:03   tacleafsysdb --agentName=C9K-LocalTasks-C9K-0 --partition= --storage-
mode=persistent --eventLogDir=. --eventLogSev=TaccSpawner/error,Mounter/
 116 ?      Sl   0:01   /usr/bin/python /bin/aos_agent --
class=aos.device.common.ProxyDeploymentAgent.ProxyDeploymentAgent --name=DeploymentProxyAgent
device_type=Cisco serial_numbe
 117 ?      Sl   0:19   /usr/bin/python /bin/aos_agent --
class=aos.device.common.ProxyCountersAgent.ProxyCountersAgent --name=CounterProxyAgent
device_type=Cisco serial_number=@(SWI
 118 ?      Sl   0:02   /usr/bin/python /bin/aos_agent --
class=aos.device.cisco.CiscoTelemetryAgent.CiscoTelemetryAgent --name=DeviceTelemetryAgent
serial_number=@(SWITCH_UNIQUE_ID)
 700 ?     Ss   0:00   sshd: guestshell [priv]
 702 ?     S    0:00   sshd: guestshell@pts/4
 703 pts/4Ss   0:00   bash -li
 732 pts/4S   0:00   sudo su -
 733 pts/4S   0:00   su -
 734 pts/4S   0:00   -bash
 823 pts/4R+  0:00   ps wax
```

### *Check for Presence of Files in /etc/aos*

Under the guest shell, Apstra stores a number of configuration files under /etc/aos.

```
[root@guestshell aos]# ls -lah /etc/aos
total 44K
drwxr-xr-x  2 root root 4.0K Nov 15 00:05 .
drwxr-xr-x 63 root root 4.0K Nov 15 00:09 ..
-rwxr-xr-x  1 root root 1.1K Nov 14 22:26 agent.json
-rw-r--r--  1 root root 1.1K Nov 15 00:05 aos.conf
-rwxr-xr-x  1 root root  992 Nov 14 22:26 common_functions
-rwxr-xr-x  1 root root 1.4K Nov 14 22:26 health_check_functions
-rwxr-xr-x  1 root root  450 Nov 14 22:26 iproute2_functions
-rwxr-xr-x  1 root root  916 Nov 14 22:26 lsb_functions
-rwxr-xr-x  1 root root 4.5K Nov 14 22:26 platform_functions
-rwxr-xr-x  1 root root  156 Nov 14 22:26 version
```

### *Check for Apstra Data in /var/log/aos*

Apstra writes the internal database to /var/log/aos

```
[root@guestshell aos]# ls -lah /var/log/aos
total 500K
drwxr-xr-x  2 root root  480 Nov 15 00:10 .
drwxr-xr-x  3 root root  120 Nov 15 00:10 ..
-rw-r--r--  1 root root 3.2K Nov 15 00:11 CounterProxyAgent.117.1479168658.log
-rw-r--r--  1 root root 289K Nov 15 02:27 CounterProxyAgent.err
-rw-r--r--  1 root root0 Nov 15 00:10 CounterProxyAgent.out
-rw-----  1 root root  31K Nov 15 00:11
CounterProxyAgentC9K_2016-11-15--00-10-59_117-2016-11-15--00-10-59.tel
-rw-r--r--  1 root root  104 Nov 15 00:45 DeploymentProxyAgent.116.1479168650.log
-rw-r--r--  1 root root  12K Nov 15 00:45 DeploymentProxyAgent.err
-rw-r--r--  1 root root0 Nov 15 00:10 DeploymentProxyAgent.out
-rw-----  1 root root  31K Nov 15 00:10
DeploymentProxyAgentC9K_2016-11-15--00-10-51_116-2016-11-15--00-10-51.tel
-rw-r--r--  1 root root 4.1K Nov 15 00:11 DeviceTelemetryAgent.118.1479168657.log
-rw-r--r--  1 root root 1.4K Nov 15 00:11 DeviceTelemetryAgent.err
-rw-r--r--  1 root root0 Nov 15 00:10 DeviceTelemetryAgent.out
-rw-----  1 root root  31K Nov 15 00:11
DeviceTelemetryAgentC9K_2016-11-15--00-10-58_118-2016-11-15--00-10-58.tel
-rw-r--r--  1 root root0 Nov 15 00:10 C9K-0.115.1479168649.log
```

```
-rw-r--r-- 1 root root0 Nov 15 00:10 C9K-0.err
-rw-r--r-- 1 root root0 Nov 15 00:10 C9K-0.out
-rw----- 1 root root 39K Nov 15 00:10 C9K-LocalTasks-
C9K-0_2016-11-15--00-10-50_115-2016-11-15--00-10-50.tel
-rw----- 1 root root 36K Nov 15 00:10 Spawner-
C9K_2016-11-15--00-10-49_111-2016-11-15--00-10-49.tel
-rw----- 1 root root 634 Nov 15 00:10 _C9K-00000000582a528a-0001744b-checkpoint
-rw-r--r-- 1 root root0 Nov 15 00:10 _C9K-00000000582a528a-0001744b-checkpoint-valid
-rw----- 1 root root0 Nov 15 00:10 _C9K-00000000582a528a-0001744b-log
-rw-r--r-- 1 root root0 Nov 15 00:10 _C9K-00000000582a528a-0001744b-log-valid
-rw-r--r-- 1 root root0 Nov 15 00:10 aos.log
[root@guestshell aos]#
```

### ***Determine Apstra Agent Version***

The Apstra agent version is available in /etc/aos/version. Before executing this command we need to attach to aos service.

```
[root@guestshell admin]# service aos attach
aos@guestshell:/# cat /etc/aos/version
VERSION=99.0.0-3874
BUILD_ID=AOS_latest_0B.3874
BRANCH_NAME=master
COMMIT_ID=d3eb2585608f0509a11b95fb9d07aed6e26d6c32
BUILD_DATETIME=2018-05-20_10:22:32_PDT
AOS_DI_RELEASE=2.2.0-169
aos@guestshell:/#
```

### ***DNS Resolution Failure***

Apstra agent is sensitive to the DNS resolution of the metadb connection. Ensure that the IP and/or DNS from /etc/aos/aos.conf is reachable from the device eth0 management port.

```
[root@guestshell ~]# aos_show_tech | grep -i dns
[2016/10/20 23:04:20.534538UTC@event-'warning']:(textMsg=Failing outgoing mount to <'tbt://aos-
server:29731/Data/ReplicaStatus?flags=i','/Metadb/ReplicaStatus'>' due to code 'resynchronizing'
and reason 'Dns lookup issue "Temporary failure in name resolution" Unknown error
18446744073709551613)
[2016/10/20 23:04:21.540444UTC@OutgoingMountConnectionError-'warning']:(connectionName=--
NONE--,localPath=/Metadb/ReplicaStatus,remotePath=tbt://aos-server:29731/Data/ReplicaStatus?
```

```
flags=i,msg=Tac::ErrnoException: Dns lookup issue "Temporary failure in name resolution" Unknown
error 18446744073709551613)
[2016/10/20 23:04:21.541174UTC@event-'warning']:(textMsg=Failing outgoing mount to <'tbt://aos-
server:29731/Data/ReplicaStatus?flags=i','/Metadb/ReplicaStatus'>' due to code 'resynchronizing'
and reason 'Dns lookup issue "Temporary failure in name resolution" Unknown error
18446744073709551613)
```

Insufficient Guestshell filesystem size

An error message 'AOS Agent needs XXMB on the / filesystem' will occur if the rootfs partition is not at least 1GB large. Please make sure to resize the guestshell filesystem to 2gb ram, 1gb disk, and 6% CPU.

```
<snip>
+ popd
/tmp/selfgz18527139
+ rpm -Uvh --nodeps --force /tmp/selfgz18527139/aos-device-agent-1.1.0-0.1.1108.x86_64.rpm
Preparing... ##### [100%]
installing package aos-device-agent-1.1.0-0.1.1108.x86_64 needs 55MB on the / filesystem
```

### ***Apstra Service Takes Long Time to Start on Cisco NX-OS***

The GuestShell feature on Cisco NXOS takes a few minutes to initialize the NXAPI within the LXC container. Apstra does not have control over this to make it any faster. Apstra Engineering has added a wait-delay to the initialization of the Apstra scripts to account for this delay. This wait is normal.

### ***Apstra Stops and ails Without Errors (MGMT VRF)***

Ensure that the guestshell is properly behind management VRF.

We should not be able to ping the Apstra server when running 'ping' command by default:

Below - we expect a ping from global default routing table to Apstra server at 172.20.156.3 to fail, but succeed under the guest shell.

```
SAL2028T5PP-172-20-156-5# ping 172.20.156.3
PING 172.20.156.3 (172.20.156.3): 56 data bytes
ping: sendto 172.20.156.3 64 chars, No route to host
^C
--- 172.20.156.3 ping statistics ---
1 packets transmitted, 0 packets received, 100.00% packet loss
SAL2028T5PP-172-20-156-5# ping 172.20.156.3 vrf management
PING 172.20.156.3 (172.20.156.3): 56 data bytes
```

```
64 bytes from 172.20.156.3: icmp_seq=0 ttl=63 time=0.649 ms
64 bytes from 172.20.156.3: icmp_seq=1 ttl=63 time=0.449 ms
64 bytes from 172.20.156.3: icmp_seq=2 ttl=63 time=0.428 ms
64 bytes from 172.20.156.3: icmp_seq=3 ttl=63 time=0.423 ms
64 bytes from 172.20.156.3: icmp_seq=4 ttl=63 time=0.404 ms
^C
```

### *Verify MGMT VRF in NX-OS Guest Shell*

```
[root@guestshell ~]# ping 172.20.157.3
connect: Network is unreachable

[root@guestshell ~]# sudo ip netns exec management ping 172.20.156.3
PING 172.20.156.3 (172.20.156.3) 56(84) bytes of data.
64 bytes from 172.20.156.3: icmp_seq=1 ttl=64 time=0.226 ms
64 bytes from 172.20.156.3: icmp_seq=2 ttl=64 time=0.232 ms
^C
```

## Arista Device Agent

### IN THIS SECTION

- [Initial Arista EOS Configuration | 220](#)
- [Decommission Device | 222](#)
- [Remove Apstra Package from Device | 223](#)
- [Restart System | 224](#)
- [Manually Install Arista Device Agent | 225](#)
- [Device Agent Configuration File | 227](#)
- [Arista Agent Troubleshooting | 227](#)

The section below describes how to manually install Apstra device agents on Arista devices. You have the option of automatically booting and installing agents and prerequisite configuration on switches with Apstra ZTP. Using Apstra ZTP is simpler and easier to support at scale than manually installing agents. For more information, see "[Apstra ZTP](#)" on page 267.

## Initial Arista EOS Configuration

### IN THIS SECTION

- [Disable ZTP | 220](#)
- [Configure AAA and network-admin User | 220](#)
- [Configure IP Address and Management VRF | 220](#)
- [Configure DNS for EOS | 221](#)
- [Configure HTTP API for EOS | 221](#)
- [Configure multi-agent for EVPN | 222](#)

### *Disable ZTP*

If you are provisioning the switch without ZTP (ZTP Disabled), ensure that the ZTP process is disabled before proceeding. After logging into the switch for the first time, run the command `zerotouch disable`. This requires a device reload.

```
localhost login: admin
localhost> zerotouch disable
```

### *Configure AAA and network-admin User*

To install or manage the agent, a network-admin user must be configured on the device with a known password.

```
aaa authorization exec default local
username admin privilege 15 role network-admin secret <admin-password>
```

### *Configure IP Address and Management VRF*

**NOTE:** If you are installing an on-box agent, you don't need to configure the management VRF. If it's needed, the agent installer automatically configures the management VRF.

The agent uses the management VRF. Move any management interfaces from the default (none) VRF into the management VRF.

The agent uses the Management1 interface by default. On modular chassis such as the Arista 7504 or 7508, the management interface is Management0 - check your platform to see if management interfaces appear as Management1 or Management1/1, Management1/2, and Management0. Management0 is a shared management interface between both supervisors.



**CAUTION:** If you are logging into this switch remotely, make sure you have an out-of-band connection prior to issuing the `vrf forwarding management` command under an interface. This immediately removes the IP address from the NIC and potentially locks you out of your system.

```
vrf definition management
  rd 100:100
interface management1
  vrf forwarding management
  ip address <address>/<cidr>
  ip route vrf management 0.0.0.0/0 <management-default-gateway>
```

### *Configure DNS for EOS*

Apstra server discovery supports DNS-based discovery if you are manually configuring the agent. By default, the `aos-config` file looks for `tbt://aos-server:29731` - accordingly, you can use a DNS nameserver to resolve `aos-server`.

```
ip name-server vrf management <dns-server-ip>
ip name-server vrf management <dns-server-ip>
```

### *Configure HTTP API for EOS*

**NOTE:** If you are installing an on-box agent, you don't need to configure HTTP API. If it's needed, the agent installer automatically configures the HTTP API.

HTTP API and Unix sockets are used to connect to the EOS API for configuration rendering and telemetry commands. The API must be made available for both the default route and the management VRF. The agent connects using the unix-socket locally on the filesystem.

```
management api http-commands
  protocol unix-socket
  no shutdown
  vrf management
    no shutdown
```

### *Configure multi-agent for EVPN*

To run EVPN with Arista devices running EOS 4.22, you must run the service routing protocols model multi-agent. You must also reboot the device to apply the configuration.

```
localhost(config)#service routing protocols model multi-agent
! Change will take effect only after switch reboot
localhost(config)#
```

To ensure that it is added to the pristine configuration of the device, we recommend that you add multi-agent configuration to the device before installing the agent. After adding the configuration, save the device configuration and reload the device.

```
localhost(config)#wr mem
Copy completed successfully.
localhost(config)#reload now

Broadcast message from root@localhost (Mon Sep 21 20:25:03 2020):

The system is going down for reboot NOW!
```

### **Decommission Device**

1. From the left navigation menu of the Apstra GUI, navigate to **Devices > Managed Devices** and select the check box for the device to decommission.
2. Click the **DECOMM** button (above the table), then click **Confirm** to change the admin state and return to the list view.
3. With the device still selected, click the **Delete system(s)** button, then click **Confirm** to remove the device and return to the list view.

## Remove Apstra Package from Device

### IN THIS SECTION

- [Uninstall Agent using EOS CLI | 223](#)
- [Uninstall Agent using Bash | 223](#)
- [Remove Remaining Apstra Data from System | 224](#)
- [Save Config File | 224](#)

### *Uninstall Agent using EOS CLI*

Erasing the startup-configuration does not delete the installed EOS extension files. You must explicitly remove the agent. Follow these steps in order.

```
localhost#no extension aos-device-agent-2.0.0-0.1.210.i386.rpm
localhost#delete extensions:no extension aos-device-agent-2.0.0-0.1.210.i386.rpm
localhost#copy boot-extensions installed-extensions
```

### *Uninstall Agent using Bash*

To use the Bash CLI you, must edit `/mnt/flash/boot-extensions` to remove the reference to the extension and delete the extension from `/mnt/flash/.extensions/aos-device-agent.i386.rpm` - This filename is unique depending on the installed Apstra version.

```
localhost#dir /all flash:.extensions/
Directory of flash:/.extensions

-rwx      1798948      May 31 02:11  EosSdk-1.8.1-4.16.6M.i686.rpm
-rwx         36199      May 31 02:25  aos-device-agent-1.2.0-0.1.137.i386.rpm
localhost#more flash:boot-extensions
EosSdk-1.8.1-4.16.6M.i686.rpm
aos-device-agent-1.2.0-0.1.137.i386.rpm
```

```
[admin@localhost ~]$ vi /mnt/flash/boot-extensions
```

### ***Remove Remaining Apstra Data from System***

Apstra-related data is retained on the filesystem in a few locations. Manually remove these data as shown below:



**CAUTION:** If you don't remove Apstra files (especially `/mnt/flash/.aos/` which includes checkpoint files) the next time you install Apstra software, configuration is replaced by the configuration that was rendered last (including any quarantine configuration) which could shut down all interfaces.

When you're removing Apstra data be sure to remove `/mnt/flash/.aos/`.

```
root@Arista:~# rm -rf /mnt/flash/aos*
root@Arista:~# rm -rf /mnt/flash/.aos*
root@Arista:~# rm -rf /var/log/aos
root@Arista:~# rm -rf /.aos
```

### ***Save Config File***

For the extension to be removed from bootup, run the command `wr mem` to ensure the extension no longer appears in boot-extensions. If the RPM is still installed in available extensions, the agent may start up again .

### **Restart System**

After uninstalling the Apstra software, reboot the system. To ensure the extension is removed from the boot extension, select 'yes' to save configuration.

```
localhost#reload
System configuration has been modified. Save? [yes/no/cancel/diff]:yes
Proceed with reload? [confirm]

Broadcast message from root@localhost (Thu Oct 19 02:03:28 2020):

The system is going down for reboot NOW!
```

When you remove the agent, configuration that is running on the switch is not modified or changed in any way; the network is not disrupted.

## Manually Install Arista Device Agent

### IN THIS SECTION

- [Download Agent Installer | 225](#)
- [Install Arista Device Agent | 225](#)



**CAUTION:** Manually installing agents requires an in-depth understanding of various device states, configuration stages and agent operation. Since it requires more effort and is error-prone we recommend manual installation in rare cases only. We, instead, recommend using the Apstra GUI to automatically install agents. For details, see ["Agents" on page 184](#) documentation. To proceed with manually installation see sections below. For assistance, contact ["Juniper Support" on page 777](#).

### *Download Agent Installer*

The agent is available over HTTPs from the Apstra server from the base URL [https://aos-server/device\\_agent\\_images/aos\\_device\\_agent.run](https://aos-server/device_agent_images/aos_device_agent.run)

```
spine1#routing-context vrf management
spine1(vrf:management)#copy https://192.168.25.250/device_agent_images/aos_device_agent.run
flash:
Copy completed successfully.
```

### *Install Arista Device Agent*

Run the command `aos_device_agent.run` to install the agent.

```
localhost#bash sudo /mnt/flash/aos_device_agent.run
Verifying archive integrity... All good.
Uncompressing AOS Device Agent installer 100%
+ set -o pipefail
+++ dirname ./agent_installer.sh
++ cd .
++ pwd
+ script_dir=/tmp/selfgz726322812
```

```

++ date
+ echo 'Device Agent Installation : Wed' Oct 18 20:34:11 UTC 2017
Device Agent Installation : Wed Oct 18 20:34:11 UTC 2017
+ echo

+ UNKNOWN_PLATFORM=1
+ WRONG_PLATFORM=1
+ CANNOT_EXECUTE=126
+ '[' 0 -ne 0 ']'
+ arg_parse
+ start_aos=True
+ [[ 0 > 0 ]]
+ supported_platforms=(["centos"]="install_sysvinit_rpm" ["eos"]="install_on_arista"
["nxos"]="install_on_nxos" ["cumulus"]="install_sysvinit_deb" ["trusty"]="install_sysvinit_deb"
["icos"]="install_sysvinit_rpm" ["snaproute"]="install_sysvinit_deb"
["simulation"]="install_sysvinit_deb")
+ declare -A supported_platforms
++ /tmp/selfgz726322812/aos_get_platform
+ current_platform=eos
+ installer=install_on_arista
+ [[ -z install_on_arista ]]
+ [[ -x /etc/init.d/aos ]]
+ echo 'Stopping AOS'
Stopping AOS
+++ readlink /sbin/init
++ basename upstart
+ [[ systemd == upstart ]]
+ /etc/init.d/aos stop
+ install_on_arista
++ pwd
+ local pkg_dir=/tmp/selfgz726322812/arista
+ local to_be_installed=
+ local flash_dir_from_bash=/mnt/flash/aos-installer
+ local flash_dir_from_cli=flash:/aos-installer
+ cp aos_device_agent.img /mnt/flash/
+ mkdir -p /mnt/flash/aos-installer
++ ls /mnt/flash/.extensions/aos-device-agent-2.0.0-0.1.138.i386.rpm
+ existing_aos=/mnt/flash/.extensions/aos-device-agent-2.0.0-0.1.138.i386.rpm
+ for aos_rpm in '${existing_aos}'
++ basename /mnt/flash/.extensions/aos-device-agent-2.0.0-0.1.138.i386.rpm
+ ip netns exec default FastCli -p15 -c 'no extension aos-device-agent-2.0.0-0.1.138.i386.rpm'
++ basename /mnt/flash/.extensions/aos-device-agent-2.0.0-0.1.138.i386.rpm
+ ip netns exec default FastCli -p15 -c 'delete extension:aos-device-

```

```

agent-2.0.0-0.1.138.i386.rpm'
+ pushd /tmp/selfgz726322812/arista
/tmp/selfgz726322812/arista /tmp/selfgz726322812
++ ls aos-device-agent-2.0.0-0.1.138.i386.rpm
+ aos_rpm=aos-device-agent-2.0.0-0.1.138.i386.rpm
+ cp aos-device-agent-2.0.0-0.1.138.i386.rpm /mnt/flash/aos-installer
+ ip netns exec default FastCli -p15 -c 'copy flash:/aos-installer/aos-device-
agent-2.0.0-0.1.138.i386.rpm extension:'
Copy completed successfully.
+ ip netns exec default FastCli -p15 -c 'extension aos-device-agent-2.0.0-0.1.138.i386.rpm force'
+ popd
/tmp/selfgz726322812
+ ip netns exec default FastCli -p15 -c 'copy installed-extensions boot-extensions'
Copy completed successfully.
+ rm -rf /mnt/flash/aos-installer
+ /etc/init.d/aos config_gen
+ [[ True == \T\r\u\e ]]
+ aos_starter -f

```

## Device Agent Configuration File

The Arista device agent manages the running-configuration file. No other configuration files are modified throughout the agent lifecycle. You can manage configuration by editing the configuration file directly. The Arista EOS device agent config file is located at `/mnt/flash/aos-config`. See ["Agent Configuration file" on page 901](#) for parameters. After updating the file, restart the agent.

```

localhost# bash sudo systemctl stop aos
localhost# bash sudo systemctl start aos

```

## Arista Agent Troubleshooting

### IN THIS SECTION

- [Apstra Log Files | 228](#)
- [Verify Agent is Running | 229](#)
- [DNS Resolution Failure | 231](#)
- [List Running Processes | 231](#)

- ['Unable to Connect' error during Installation | 239](#)

### *Apstra Log Files*

Apstra logs to a number of files in the `/var/log/aos` directory.

Confirm that the agent package is installed.

```
-bash-4.1# rpm -q --info aos-device-agent
Name       : aos-device-agent      Relocations: /
Version    : 1.0.1                Vendor: (none)
Release    : 0.1.15              Build Date: Thu Oct  6 21:21:08 2016
Install Date: Fri Oct 21 04:14:07 2016  Build Host: 6539ff88c5b0
Group      : Unspecified          Source RPM: aos-device-agent-1.0.1-0.1.15.src.rpm
Size       : 87227369             License: Copyright 2014-present, Apstra, Inc. All
rights reserved.
Signature  : (none)
Summary    : AOS device agent package for Arista switches
Description:
AOS device agent for Arista switches

localhost#show extension detail
      Name: EosSdk-1.8.1-4.16.6M.i686.rpm
      Version: 1.8.1
      Release: 3206305.idboiseeosdk
      Presence: available
      Status: installed
      Vendor:
      Summary: EOS Software Development Kit
      RPMS: EosSdk-1.8.1-4.16.6M.i686.rpm 1.8.1/3206305.idboiseeosdk
      Total size: 8073886 bytes
      Description:
The EOS Software Development Kit provides a set of stable C++ interfaces for
high-performance access to EOS primitives, for onbox programming beyond what
can be done with Python.

      Name: aos-device-agent-1.2.0-0.1.137.i386.rpm
      Version: 1.2.0
      Release: 0.1.137
```

```

Presence: available
Status: installed
Vendor:
Summary: AOS device agent package for Arista switches
RPMs: aos-device-agent-1.2.0-0.1.137.i386.rpm 1.2.0/0.1.137
Total size: 88651 bytes
Description:
AOS device agent for Arista switches

```

### ***Verify Agent is Running***

```

localhost#bash sudo service aos status
AOS is running

```

```

localhost#dir flash:aos*
Directory of flash:/aos*

-rwx      2228      May 31 02:26  aos-config
-rwx    55668736    May 31 02:25  aos_device_agent.img
-rwx    54889549    May 31 02:10  aos_device_agent_1.2.0-137_eos.run

Directory of flash:/aos

drwx      4096      May 31 02:25  plugins

4025892864 bytes total (3392516096 bytes free)

```

```

localhost#dir file:/var/log/aos
Directory of file:/var/log/aos

-rw-        0      May 31 02:37  000C29E808A1-0.4602.1496198223.log
-rw-        0      May 31 02:37  000C29E808A1-0.err
-rw-        0      May 31 02:37  000C29E808A1-0.out
-rw-    63643      May 31 02:40  000C29E808A1-
LocalTasks-000C29E808A1-0_2017-05-31--02-37-03_4602-2017-05-31--02-37-03.tel
-rw-        0      May 31 02:37  CounterProxyAgent.4604.1496198231.log
-rw-        0      May 31 02:37  CounterProxyAgent.4684.1496198239.log
-rw-    1490      May 31 02:37  CounterProxyAgent.err
-rw-        0      May 31 02:37  CounterProxyAgent.out

```

```

-rw-      33589      May 31 02:37
CounterProxyAgent000C29E808A1_2017-05-31--02-37-12_4604-2017-05-31--02-37-12.tel
-rw-      42562      May 31 02:37
CounterProxyAgent000C29E808A1_2017-05-31--02-37-20_4684-2017-05-31--02-37-20.tel
-rw-         0      May 31 02:37 DeploymentProxyAgent.4603.1496198226.log
-rw-         0      May 31 02:37 DeploymentProxyAgent.4629.1496198235.log
-rw-      1569      May 31 02:37 DeploymentProxyAgent.err
-rw-         0      May 31 02:37 DeploymentProxyAgent.out
-rw-      33618      May 31 02:37
DeploymentProxyAgent000C29E808A1_2017-05-31--02-37-07_4603-2017-05-31--02-37-07.tel
-rw-      39585      May 31 02:37
DeploymentProxyAgent000C29E808A1_2017-05-31--02-37-16_4629-2017-05-31--02-37-16.tel
-rw-         0      May 31 02:37 DeviceKeeperAgent.4606.1496198231.log
-rw-      510      May 31 02:37 DeviceKeeperAgent.err
-rw-         0      May 31 02:37 DeviceKeeperAgent.out
-rw-      38221      May 31 02:37
DeviceKeeperAgent000C29E808A1_2017-05-31--02-37-12_4606-2017-05-31--02-37-12.tel
-rw-         0      May 31 02:37 DeviceTelemetryAgent.4605.1496198230.log
-rw-      158      May 31 02:37 DeviceTelemetryAgent.4670.1496198242.log
-rw-      2580      May 31 02:37 DeviceTelemetryAgent.err
-rw-         0      May 31 02:37 DeviceTelemetryAgent.out
-rw-      33597      May 31 02:37
DeviceTelemetryAgent000C29E808A1_2017-05-31--02-37-12_4605-2017-05-31--02-37-12.tel
-rw-      56620      May 31 02:37
DeviceTelemetryAgent000C29E808A1_2017-05-31--02-37-23_4670-2017-05-31--02-37-23.tel
-rw-      50737      May 31 02:37
Spawner-000C29E808A1_2017-05-31--02-37-02_4597-2017-05-31--02-37-02.tel
-rw-      640      May 31 02:37 _000C29E808A1-00000000592e2c4f-00054c50-
checkpoint
-rw-         0      May 31 02:37 _000C29E808A1-00000000592e2c4f-00054c50-
checkpoint-valid
-rw-         0      May 31 02:37 _000C29E808A1-00000000592e2c4f-00054c50-log
-rw-         0      May 31 02:37 _000C29E808A1-00000000592e2c4f-00054c50-log-valid
-rw-         0      May 31 02:37 aos.log

```

291463168 bytes total (260136960 bytes free)

### ***DNS Resolution Failure***

The agent is sensitive to the DNS resolution of the metadb connection. Ensure that the IP and/or DNS from the config file is reachable from the device management port.

```
localhost# bash sudo service aos show_tech | grep -i dns
[2016/10/20 23:04:20.534538UTC@event-'warning']:(textMsg=Failing outgoing mount to <'tbt://aos-server:29731/Data/ReplicaStatus?flags=i','/Metadb/ReplicaStatus'>' due to code 'resynchronizing' and reason 'Dns lookup issue "Temporary failure in name resolution" Unknown error 18446744073709551613')
[2016/10/20 23:04:21.540444UTC@OutgoingMountConnectionError-'warning']:(connectionName=--NONE--,localPath=/Metadb/ReplicaStatus,remotePath=tbt://aos-server:29731/Data/ReplicaStatus?flags=i,msg=Tac::ErrnoException: Dns lookup issue "Temporary failure in name resolution" Unknown error 18446744073709551613)
[2016/10/20 23:04:21.541174UTC@event-'warning']:(textMsg=Failing outgoing mount to <'tbt://aos-server:29731/Data/ReplicaStatus?flags=i','/Metadb/ReplicaStatus'>' due to code 'resynchronizing' and reason 'Dns lookup issue "Temporary failure in name resolution" Unknown error 18446744073709551613')
```

### ***List Running Processes***

List the Apstra agent processes that run alongside other management components on the switch with the `ps wax` command.

```
localhost#bash sudo service aos attach
aos@localhost:/# ps wax
```

PID	TTY	STAT	TIME	COMMAND
1	?	Ss	0:03	/sbin/init
2	?	S	0:00	[kthreadd]
3	?	S	0:00	[ksoftirqd/0]
4	?	S	0:00	[kworker/0:0]
6	?	S	0:00	[migration/0]
8	?	S<	0:00	[khelper]
9	?	S<	0:00	[netns]
10	?	S	0:00	[kworker/u:1]
168	?	S	0:00	[sync_supers]
170	?	S	0:00	[bdi-default]
172	?	S<	0:00	[kblockd]
179	?	S<	0:00	[ata_sff]
189	?	S	0:00	[khubd]
290	?	S	0:00	[dst_gc_task]

```

375 ?      S      0:00 [arp_cache-prd]
376 ?      S      0:00 [icmp_unreachabl]
377 ?      S<     0:00 [rpciod]
380 ?      S<     0:00 [ecc_log_wq]
388 ?      S      0:00 [khungtaskd]
389 ?      S      0:00 [khungtaskd2]
394 ?      S      0:00 [kswapd0]
395 ?      S      0:00 [fsnotify_mark]
396 ?      S<     0:00 [nfsiod]
397 ?      S<     0:00 [crypto]
467 ?      S<     0:00 [pcielwd]
506 ?      S      0:00 [scsi_ah_0]
509 ?      S      0:00 [scsi_ah_1]
512 ?      S      0:00 [kworker/u:2]
599 ?      S<     0:00 [edac-poller]
631 ?      S      0:00 [ndisc_cache-prd]
635 ?      S<     0:00 [deferwq]
951 ?      S<     0:00 [loop0]
1244 ?     S<S    0:00 /sbin/udev -d
1374 ?     S      0:01 [kworker/0:2]
1471 ?     S<     0:00 /sbin/udev -d
1730 ?     S      0:00 python /usr/bin/immortalize --daemonize --log=/var/log/agents/ConnMgr
--logpidsuffix --maxcredits=5 --cos
1732 ?     S      0:00 /usr/bin/ConnMgr -p /var/run/ConnMgr.pid
1750 ?     S      0:00 python /usr/bin/immortalize --daemonize --log=/var/log/agents/
TimeAgent --logpidsuffix --maxcredits=5 --c
1751 ?     S<     0:00 /usr/bin/TimeAgent -c /etc/TimeAgent.conf -p /var/run/TimeAgent.pid
1762 ?     S      0:00 watchdog
1763 ?     S<     0:00 wdog-cl
1786 ?     S      0:00 python /usr/bin/inotifyrun -c pax -x sv4cpio -O -w -f /mnt/flash/
persist/local.new . && mv /mnt/flash/per
1788 ?     Ss+    0:00 inotifywait -m -r -e modify -e create -e delete -e attrib -e move .
1798 ?     S      0:00 python /usr/bin/inotifyrun -c pax -x sv4cpio -O -w -f /mnt/flash/
persist/sys.new . && mv /mnt/flash/persi
1799 ?     Ss+    0:00 inotifywait -m -r -e modify -e create -e delete -e attrib -e move .
1811 ?     S      0:00 python /usr/bin/inotifyrun -c shred --exact --iterations=1 /mnt/flash/
persist/secure; pax -x sv4cpio -O -
1813 ?     Ss+    0:00 inotifywait -m -r -e modify -e create -e delete -e attrib -e move .
1820 ?     S      0:00 [watchdog/0]
1964 ?     S      0:00 /usr/bin/EosOomAdjust
1968 ?     Ss      0:00 /usr/sbin/mcelog --daemon --no-syslog --logfile /var/log/mcelog
1979 ?     S      0:00 [kbf_v4v6_rx]
1980 ?     S      0:00 [kbf_v4v6_echo]

```

```

1981 ?      S<      0:00 [kbfd_tx]
1982 ?      S<      0:00 [kbfd_rx_expire]
1983 ?      S<      0:00 [kbfd_tx_reset]
1984 ?      S<      0:00 [kbfd_echo_tx]
1985 ?      S<      0:00 [kbfd_echo_rx_ex]
1986 ?      S<      0:00 [kbfd_echo_tx_re]
1987 ?      S<      0:00 [kbfd_echo_exp_r]
2030 ?      Ss      0:00 crond
2079 ?      S       0:00 netnsd-watcher -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2081 ?      S       0:00 netnsd-server -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2091 ?      S       0:00 ProcMgr-mast -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2092 ?      S       0:02 ProcMgr-work -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2093 ?      S       0:14 Sysdb -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2094 ?      S       0:02 /usr/bin/SlabMonitor
2095 ?      S       0:03 FastClid-ser -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2131 ?      S       0:01 Fru -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2136 ?      S       0:02 Launcher -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2222 ?      S       0:01 /usr/bin/EosProxySdkAgent --agenttitle=EosSdk-EosProxySdkAgent --
demuxerOpts=172749640510,172743984283,tb
2244 ?      S       0:00 netns --agenttitle=LacpTxAgent --
demuxerOpts=176938128982,176937081924,tbl://sysdb/+n,Sysdb (pid:2093) --
2249 ?      Ss      0:00 netnsd-session -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2250 ?      S       0:00 LacpTxAgent -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2264 ?      S       0:00 netns --agenttitle=Ipv6RouterAdvt --
demuxerOpts=177054066724,176993113047,tbl://sysdb/+n,Sysdb (pid:2093)
2266 ?      Ss      0:00 netnsd-session -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2267 ?      S       0:00 Ipv6RouterAd --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2286 ?      S       0:00 netns --agenttitle=AgentMonitor --
demuxerOpts=180713744050,180503816091,tbl://sysdb/+n,Sysdb (pid:2093) -
2289 ?      Ss      0:00 netnsd-session -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize

```

```

2290 ?      S      0:02 AgentMonitor -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2294 ?      S      0:00 netns --agenttitle=Mirroring --
demuxerOpts=181173742385,181026608825,tbl://sysdb/+n,Sysdb (pid:2093) --sy
2295 ?      Ss     0:00 netnsd-session -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2296 ?      S      0:00 Mirroring -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2315 ?      S      0:00 netns --agenttitle=Acl --demuxerOpts=184720501541,181293026506,tbl://
sysdb/+n,Sysdb (pid:2093) --sysdbfd=
2316 ?      Ss     0:00 netnsd-session -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2317 ?      S      0:00 Acl -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2328 ?      S      0:00 IgmpSnooping -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2359 ?      S      0:01 SuperServer -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2446 ?      S      0:00 netns --agenttitle=Dot1x --
demuxerOpts=193890685273,189430843618,tbl://sysdb/+n,Sysdb (pid:2093) --sysdbf
2447 ?      Ss     0:00 netnsd-session -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2448 ?      S      0:00 Dot1x -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2467 ?      S      0:00 FastClidCapi -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2503 ?      S      0:00 FastClid-ses -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2504 ?      Ssl    0:13 FastCapi -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2540 ?      S      0:00 netns --agenttitle=EventMgr --
demuxerOpts=198435198068,198381904787,tbl://sysdb/+n,Sysdb (pid:2093) --sys
2541 ?      Ss     0:00 netnsd-session -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2542 ?      S      0:00 EventMgr -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2544 ?      S      0:00 netns --agenttitle=TopoAgent --
demuxerOpts=207004990826,206854969014,tbl://sysdb/+n,Sysdb (pid:2093) --sy
2546 ?      Ss     0:00 netnsd-session -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2547 ?      S      0:00 TopoAgent -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2568 ?      S      0:00 netns --agenttitle=PortSec --

```

```

demuxerOpts=211114755521,211113859019,tbl://sysdb/+n,Sysdb (pid:2093) --sysd
2570 ?      Ss      0:00 netnsd-session -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2571 ?      S      0:00 PortSec      -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2573 ?      S      0:00 netns --agenttitle=Bfd --demuxerOpts=211236786399,211177838833,tbl://
sysdb/+n,Sysdb (pid:2093) --sysdbfd=
2576 ?      Ss      0:00 netnsd-session -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2580 ?      S      0:00 Bfd          -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2595 ?      S      0:00 netns --agenttitle=Ira --demuxerOpts=214768824794,211370899495,tbl://
sysdb/+n,Sysdb (pid:2093) --sysdbfd=
2596 ?      Ss      0:00 netnsd-session -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2598 ?      S      0:00 Ira          -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2618 ?      S      0:00 netns --agenttitle=LedPolicy --
demuxerOpts=215245146330,215100253912,tbl://sysdb/+n,Sysdb (pid:2093) --sy
2619 ?      Ss      0:00 netnsd-session -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2621 ?      S      0:00 LedPolicy    -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2628 ?      Sl      0:00 Aaa          -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2648 ?      S      0:00 netns --agenttitle=CapiApp-CapiApp --
demuxerOpts=219306529482,219133267319,tbl://sysdb/+n,Sysdb (pid:2093)
2651 ?      Ss      0:00 netnsd-session -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2657 ?      Sl      0:01 uwsgi        -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2661 ?      S      0:00 netns --agenttitle=StpTxRx --
demuxerOpts=219560663096,219463089954,tbl://sysdb/+n,Sysdb (pid:2093) --sysd
2668 ?      Ss      0:00 netnsd-session -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2669 ?      S      0:00 StpTxRx      -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2681 ?      S      0:00 netns --agenttitle=Macsec --
demuxerOpts=219852379174,219704155526,tbl://sysdb/+n,Sysdb (pid:2093) --sysdb
2682 ?      Ss      0:00 netnsd-session -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2683 ?      S      0:00 Macsec       -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize

```

```

2718 ?      S      0:00 MplsUtilLsp    -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2744 ?      Ss     0:00 nginx: master process /usr/sbin/nginx -c /etc/nginx/nginx.conf -g
pid /var/run/nginx.pid;
2748 ?      S      0:00 nginx: worker process
2910 ?      S      0:00 netns --agenttitle=MaintenanceMode --
demuxerOpts=236329384403,223871866307,tbl://sysdb/+n,Sysdb (pid:2093)
2916 ?      Ss     0:00 netnsd-session -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2920 ?      S      0:00 MaintenanceM          -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2963 ?      S      0:00 netns --agenttitle=Arp --demuxerOpts=236663705062,236485011967,tbl://
sysdb/+n,Sysdb (pid:2093) --sysdbfd=
2971 ?      Ss     0:00 netnsd-session -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2974 ?      Sl     0:00 Arp                    -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
2980 ?      Ss     0:00 /usr/sbin/sshd
2997 ?      S      0:00 netns --agenttitle=PowerManager --
demuxerOpts=240546963425,236860990252,tbl://sysdb/+n,Sysdb (pid:2093) -
3002 ?      Ss     0:00 netnsd-session -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
3004 ?      S      0:00 PowerManager          -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
3007 ?      S      0:00 netns --agenttitle=Mpls --demuxerOpts=241249655231,241228647018,tbl://
sysdb/+n,Sysdb (pid:2093) --sysdbfd
3014 ?      Ss     0:00 netnsd-session -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
3015 ?      S      0:00 Mpls                  -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
3031 ?      S      0:01 CliSessionMg         -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
3040 ?      S<     0:00 /sbin/udev -d
3070 ?      S      0:00 netns --agenttitle=Fhrp --demuxerOpts=245198240050,244921462712,tbl://
sysdb/+n,Sysdb (pid:2093) --sysdbfd
3075 ?      Ss     0:00 netnsd-session -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
3077 ?      S      0:00 Fhrp                  -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
3118 ?      Sl     0:00 /sbin/rsyslogd -i /var/run/syslogd.pid -c 5
3122 ?      S      0:00 netns --agenttitle=Qos --demuxerOpts=249452799773,245803103371,tbl://
sysdb/+n,Sysdb (pid:2093) --sysdbfd=
3131 ?      Ss     0:00 netnsd-session -d -i --dlopen -p -f -l libLoadDynamicLibs.so

```

```

procmgr libProcMgrSetup.so --daemonize
3136 ?      S      0:00 Qos          -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
3184 ?      S      0:00 netns --agenttitle=Thermostat --
demuxerOpts=253407320281,249878057576,tbl://sysdb/+n,Sysdb (pid:2093) --s
3185 ?      Ss     0:00 netnsd-session -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
3187 ?      S      0:00 Thermostat -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
3189 ?      S      0:00 netns --agenttitle=Lldp --demuxerOpts=254384000160,254383598162,tbl://
sysdb/+n,Sysdb (pid:2093) --sysdbfd
3190 ?      Ss     0:00 netnsd-session -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
3192 ?      S      0:00 Lldp          -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
3198 ?      S      0:00 Lag           -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
3217 ?      S      0:00 EventMon      -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
3220 ?      S      0:00 /usr/bin/conlogd
3222 ?      S      0:00 sh -c /usr/bin/tail -n 0 --retry --follow=name --pid=3220 /var/log/
eos-console | sed 's/\(.*\)/\1\r/'
3223 ?      S      0:00 /usr/bin/tail -n 0 --retry --follow=name --pid=3220 /var/log/eos-
console
3224 ?      S      0:00 sed s/\(.*\)/\1\r/
3233 ?      S      0:01 PhyEthtool    -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
3264 ?      S      0:00 netns --agenttitle=StpTopology --
demuxerOpts=262614958826,262505739622,tbl://sysdb/+n,Sysdb (pid:2093) --
3269 ?      Ss     0:00 netnsd-session -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
3277 ?      S      0:00 StpTopology    -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
3278 ?      S      0:00 netns --agenttitle=Stp --demuxerOpts=262947885263,262802812166,tbl://
sysdb/+n,Sysdb (pid:2093) --sysdbfd=
3279 ?      Ss     0:00 netnsd-session -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
3280 ?      S      0:00 Stp            -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
3281 ?      S      0:07 Etba          -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
3289 ?      S      0:00 netns --agenttitle=Ebra --demuxerOpts=267068997224,266942848299,tbl://
sysdb/+n,Sysdb (pid:2093) --sysdbfd

```

```

3290 ?      Ss      0:00 netnsd-session -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
3291 ?      S      0:00 Ebra -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
3295 ?      S      0:00 netns --agenttitle=KernelFib --
demuxerOpts=270859722189,270754589714,tbl://sysdb/+n,Sysdb (pid:2093) --sy
3296 ?      Ss      0:00 netnsd-session -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
3297 ?      S      0:00 KernelFib -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
3298 ?      S      0:02 /usr/sbin/ribd -N
3496 ?      Ss      0:00 /usr/sbin/sshd-management -f /etc/ssh/sshd_config-management
3554 ttyS0   Ss+     0:00 /sbin/mingetty --noclear /dev/ttyS0
3564 tty1    Ss+     0:00 /sbin/mingetty /dev/tty1
3566 tty2    Ss+     0:00 /sbin/mingetty /dev/tty2
3569 tty3    Ss+     0:00 /sbin/mingetty /dev/tty3
3571 tty4    Ss+     0:00 /sbin/mingetty /dev/tty4
3573 tty5    Ss+     0:00 /sbin/mingetty /dev/tty5
3575 tty6    Ss+     0:00 /sbin/mingetty /dev/tty6
3618 ?      S      0:02 /usr/sbin/ribd -N -z client name management ns-name ns-management
vrfname management servername vre_serve
4566 ?      S<      0:00 [loop1]
4600 ?      Sl      0:00 tacspawner --daemonize=/var/log/aos/aos.log --pidfile=/var/run/
aos.pid --name=000C29E808A1 --hostname=000
4602 ?      S      0:00 tacleafsysdb --agentName=000C29E808A1-LocalTasks-000C29E808A1-0 --
partition= --storage-mode=persistent --
4606 ?      Sl      0:00 /usr/bin/python /usr/bin/aos_agent --
class=aos.device.common.DeviceKeeperAgent.DeviceKeeperAgent --name=D
4629 ?      Sl      0:00 /usr/bin/python /usr/bin/aos_agent --
class=aos.device.common.ProxyDeploymentAgent.ProxyDeploymentAgent --
4670 ?      Sl      0:00 /usr/bin/python /usr/bin/aos_agent --
class=aos.device.arista.AristaTelemetryAgent.AristaTelemetryAgent --
4684 ?      Sl      0:00 /usr/bin/python /usr/bin/aos_agent --
class=aos.device.common.ProxyCountersAgent.ProxyCountersAgent --name
5366 ?      S      0:00 FastClidHelp -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
5371 ?      S      0:00 FastClid-ses -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
5372 ?      Ssl     0:00 Cli [interac -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
5483 ?      S      0:00 FastClidHelp -d -i --dlopen -p -f -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
5488 ?      S      0:00 FastClid-ses -d -i --dlopen -p -f -l libLoadDynamicLibs.so

```

```

procmgr libProcMgrSetup.so --daemonize
5489 ?      Ssl    0:00 Cli [interac  -d -i --dlopen -p -f  -l libLoadDynamicLibs.so
procmgr libProcMgrSetup.so --daemonize
5506 ?      Ss     0:00 sshd-management: admin [priv]
5531 ?      S      0:00 sshd-management: admin@pts/3
5534 ?      Ssl+   0:00 FastCli
5579 ?      S      0:00 sudo service aos attach
5581 ?      S      0:00 /bin/sh /sbin/service aos attach
5589 ?      S      0:00 /bin/bash /etc/init.d/aos attach
5616 ?      S      0:00 /bin/bash
5622 ?      R+     0:00 ps wax

```

### *'Unable to Connect' error during Installation*

When you install an Arista EOS device agent, you might receive an Unable to connect: Connection refused error.

```

Unable to connect: Connection refused
+ status=
+ [[ '' =~ .*Status: installed.* ]]
+ return 1
+ cp aos-device-agent-1.2.1-0.1.72.i386.rpm /mnt/flash/aos-installer
+ FastCli -p15 -c 'copy flash:/aos-installer/aos-device-agent-1.2.1-0.1.72.i386.rpm extension:'
Unable to connect: Connection refused
'sudo /mnt/flash/aos_device_agent_1.2.1-72_eos.run' returned error code:255

```

This error could be caused from:

- the SDK not running
- the unix-socket not listening
- attempting to run the device installer in the management VRF.

To resolve this error, switch the routing-contexts to default.

## Cumulus Device Agent

### IN THIS SECTION

 [Quick start | 240](#)

- Cumulus Initial Configuration | 242
- Download Agent Installer | 245
- Install Cumulus Device Agent | 246
- Device Agent Configuration File | 247
- Device Agent Management | 248
- Deploy Device | 249
- Uninstall Apstra Device Agent | 249
- Cumulus Agent Troubleshooting | 251

Although the preferred method of installing "[device system agents](#)" on [page 184](#) is by creating agents in the Apstra GUI, you *can* manually install Apstra agents from the CLI. Only in rare exceptions is it needed to manually install agents, which requires more effort and is error-prone. An in-depth understanding of the various device states, configuration stages, and agent operations is required before manually installing agents. For assistance, contact "[Juniper Support](#)" on [page 777](#).

## Quick start

This section is a quick-start steps for installing the Cumulus device agent. The remaining sections describe the steps in detail.

### 1. Configure management IP and VRF

```
admin@cumulus:mgmt-vrf:~$ vi /etc/network/interfaces
```

```
auto mgmt
iface mgmt
    address 127.0.0.1/8
    vrf-table auto
    post-up sudo service aos start

# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

auto eth0
```

```

iface eth0 inet dhcp
    vrf mgmt

```

```

admin@cumulus:mgmt-vrf:~$ ifreload -a

```

## 2. Activate cumulus license.

```

admin@cumulus:mgmt-vrf:~$ cl-license -i user@domain|cumulus-license-key
admin@cumulus:mgmt-vrf:~$ service switchd restart

```

## 3. Download Apstra agent.

```

admin@cumulus:mgmt-vrf:~$ sudo vrf task exec mgmt wget -nv \
    --no-check-certificate \
    https://aos-server/device_agent_images/aos_device_agent.run

```

## 4. Set IP of Apstra server and enable configuration service.

```

admin@cumulus:mgmt-vrf:~$ sudo vi /etc/aos/aos.conf

```

```

[controller]
metadb = tbt://aos-server:29731

[service]
enable_configuration_service = 1

```

## 5. Restart Apstra agent

```

admin@cumulus:mgmt-vrf:~$ sudo service aos stop
admin@cumulus:mgmt-vrf:~$ sudo service aos start

```

## 6. Approve device in Apstra.

## 7. Assign device to blueprint.

## Cumulus Initial Configuration

### IN THIS SECTION

- [CumulusVX | 242](#)
- [Management Interface | 242](#)
- [Install Cumulus License | 244](#)

Prior to being used with the Apstra software, Cumulus device agents require certain configuration. We've added a local username **admin** with password **admin** as part of our provisioning process. By default, the Cumulus credentials are username **cumulus** and password **CumulusLinux!** - Apstra does not depend on the username to be changed.

### *CumulusVX*

If you're deploying CumulusVX, (Virtual appliance), ensure the virtual switch is provided at least 1 vCPU and 2GB RAM. Cumulus VX 3.1.1 OVA template provides only for 512MB RAM. This will need to be increased.

### *Management Interface*



**CAUTION:** Cumulus device agents require a management VRF to be set up before running the Apstra device installer. The 'bash' shell must also be running under the context of the management VRF prior to installation to ensure the Apstra agent can communicate to the Apstra server.

By default, Cumulus management VRF is not activated.

```
root@cumulus:~# cat /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*.intf

# The loopback network interface
auto lo
iface lo inet loopback
```

```
# The primary network interface
auto eth0
iface eth0 inet dhcp
```

Add a new auto mgmt interface as a VRF and activate the eth0 interface for vrf mgmt.

```
root@cumulus:~# vi /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*.intf

# The loopback network interface
auto lo
iface lo inet loopback

auto mgmt
iface mgmt
    address 127.0.0.1/8
    vrf-table auto
    post-up sudo service aos start

auto eth0
iface eth0 inet dhcp
    vrf mgmt
```

After the VRF is activated, reload the network file with `ifreload -a`, and log back into the switch afterwards. This ensures the Bash prompt is under the management VRF routing context. Pay particular note to the new bash prompt, `admin@cumulus:mgmt-vrf:~$` which indicates bash is running under the mgmt-vrf context.

```
root@cumulus:/etc/network# ifreload -a
<reconnect with SSH>
Welcome to Cumulus VX (TM)

Cumulus VX (TM) is a community supported virtual appliance designed for
experiencing, testing and prototyping Cumulus Networks' latest technology.
For any questions or technical support, visit our community site at:
http://community.cumulusnetworks.com
```

The registered trademark Linux (R) is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a world-wide basis.

Last login: Fri May 26 12:41:13 2017 from 192.168.25.1

admin@cumulus:mgmt-vrf:~\$

### *Install Cumulus License*

To run properly on most platforms, a license must be installed for the switch. If a license is not installed, deployment failures may occur because the command `ifreload -a` will fail because the `switchd` service isn't running.

You can pre-provision a Cumulus license for the Apstra ZTP server. You can install it manually.

If a license isn't installed, and this is not a cumulus VX platform, the command `cl-license` will fail.

```
admin@cumulus:mgmt-vrf:~$ sudo service switchd start
Job for switchd.service failed. See 'systemctl status switchd.service' and 'journalctl -xn' for details.

admin@cumulus:mgmt-vrf:~$ cl-license
1508122225.507863 2017-10-16 02:50:25 license.c:318 CRIT No license file.
No license installed!

admin@cumulus:mgmt-vrf:~$ sudo cat /var/log/syslog
2017-10-16T02:53:04.661850+00:00 cumulus systemd[1]: Failed to start Cumulus Linux Switch Daemon.
2017-10-16T02:53:04.662375+00:00 cumulus systemd[1]: Dependency failed for Cumulus Linux Port Watch Event Daemon.
2017-10-16T02:53:04.663217+00:00 cumulus systemd[1]: Dependency failed for Cumulus Linux acltool.
2017-10-16T02:53:04.664004+00:00 cumulus systemd[1]: Unit switchd.service entered failed state.
2017-10-16T02:53:04.687029+00:00 cumulus Failure: Not running cl-support for portwd.service, failure #2 status: Result=success ExecMainCode=2 ExecMainStatus=15
2017-10-16T02:53:04.730186+00:00 cumulus Failure: Not running cl-support for switchd.service, failure #2 status: Result=exit-code ExecMainCode=1 ExecMainStatus=2
```

Add the license key.

```
admin@cumulus:mgmt-vrf:~$ sudo cl-license -i
Paste license text here, then hit ctrl-d
user@company.com|example_license_text/here
```

```
License file installed.
Service 'switchd' is not running.
Run this command:
sudo systemctl restart switchd
```

```
Or reboot to enable functionality.
License file installed.
```

After the license is installed, restart the switchd service.

```
admin@cumulus:mgmt-vrf:~$ sudo service switchd stop
admin@cumulus:mgmt-vrf:~$ sudo service switchd start
```

## Download Agent Installer

Apstra device agent installation files for Cumulus are available from the Apstra server, served from the URL [https://aos-server/device\\_agent\\_images/aos\\_device\\_agent.run](https://aos-server/device_agent_images/aos_device_agent.run)

For validating the downloaded file, a .md5 file is also available. Copy the .run file to the Cumulus switch, with the command `vrf task exec mgmt wget -nv --no-check-certificate https://aos-server/device_agent_images/aos_device_agent.run`

**NOTE:** This command assumes bash is running under the 'mgmt' vrf context. If this is not the case, omit the section `vrf task exec mgmt` and download the file normally.

```
root@cumulus:mgmt-vrf:~# vrf task exec mgmt wget -nv --no-check-certificate https://aos-server/
device_agent_images/aos_device_agent.run
WARNING: The certificate of 'aos-server' is not trusted.
WARNING: The certificate of 'aos-server' hasn't got a known issuer.
The certificate's owner does not match hostname 'aos-server'
2017-10-15 23:32:55 URL:https://aos-server/device_agent_images/aos_device_agent.run
[57245356/57245356] -> "aos_device_agent.run" [1]
```

```
root@cumulus:mgmt-vrf:~#vrf task exec mgmt wget -nv --no-check-certificate https://aos-server/
device_agent_images/aos_device_agent.run.md5
WARNING: The certificate of 'aos-server' is not trusted.
WARNING: The certificate of 'aos-server' hasn't got a known issuer.
The certificate's owner does not match hostname 'aos-server'
```

```
2017-10-15 23:34:50 URL:https://aos-server/device_agent_images/aos_device_agent.run.md5 [65/65] -
> "aos_device_agent.run.md5" [1]
```

```
root@cumulus:mgmt-vrf:~# root@cumulus:mgmt-vrf:~# cat aos_device_agent.run.md5
70d58a0aaa5ed4519b87ced74003476a aos_device_agent_2.0.0-210.run
root@cumulus:mgmt-vrf:~# root@cumulus:mgmt-vrf:~# md5sum aos_device_agent.run
70d58a0aaa5ed4519b87ced74003476a aos_device_agent.run
```

## Install Cumulus Device Agent

To install the Apstra device agent on cumulus, run the '.run' file available from the Apstra Server. Once the file is downloaded, run it as a shell command. Make sure that the Apstra device agent is installed while bash is under the mgmt-vrf routing-context.

Run `sudo sh aos_device_agent.run`

```
admin@cumulus:mgmt-vrf:~$ sudo sh aos_device_agent.run
Verifying archive integrity... All good.
Uncompressing AOS Device Agent installer 100%
+ set -o pipefail
+++ dirname ./agent_installer.sh
++ cd .
++ pwd
+ script_dir=/tmp/selfgz8796
++ date
+ echo 'Device Agent Installation : Mon' Oct 16 00:19:57 UTC 2017
Device Agent Installation : Mon Oct 16 00:19:57 UTC 2017
+ echo

+ UNKNOWN_PLATFORM=1
+ WRONG_PLATFORM=1
+ CANNOT_EXECUTE=126
+ '[' 0 -ne 0 ']'
+ arg_parse
+ start_aos=True
+ [[ 0 > 0 ]]
+ supported_platforms=(["centos"]="install_sysvinit_rpm" ["eos"]="install_on_arista"
["nxos"]="install_on_nxos" ["cumulus"]="install_sysvinit_deb" ["trusty"]="install_sysvinit_deb"
["icos"]="install_sysvinit_rpm" ["snaprout""]="install_sysvinit_deb"
["simulation"]="install_sysvinit_deb")
+ declare -A supported_platforms
++ /tmp/selfgz8796/aos_get_platform
```

```

+ current_platform=cumulus
+ installer=install_sysvinit_deb
+ [[ -z install_sysvinit_deb ]]
+ [[ -x /etc/init.d/aos ]]
+ install_sysvinit_deb
++ pwd
+ local pkg_dir=/tmp/selfgz8796/sysvinit_deb
+ dpkg -s aos-device-agent
+ dpkg --purge aos-device-agent
(Reading database ... 25364 files and directories currently installed.)
Removing aos-device-agent (2.0.0-210) ...
Purging configuration files for aos-device-agent (2.0.0-210) ...
+ dpkg -i /tmp/selfgz8796/sysvinit_deb/aos-device-agent-2.0.0-210.amd64.deb
Selecting previously unselected package aos-device-agent.
(Reading database ... 25364 files and directories currently installed.)
Preparing to unpack ../aos-device-agent-2.0.0-210.amd64.deb ...
Unpacking aos-device-agent (2.0.0-210) ...
Setting up aos-device-agent (2.0.0-210) ...
Processing triggers for systemd (215-17+deb8u4) ...
+ mkdir -p /opt/aos
+ cp aos_device_agent.img /opt/aos
+ post_install_common
+ /etc/init.d/aos config_gen
grep: /etc/aos/aos.conf: No such file or directory
+ [[ True == \T\r\u\e ]]
+++ readlink /sbin/init
++ basename /lib/systemd/systemd
+ [[ systemd == systemd ]]
+ systemctl start aos

```

If an `aos.conf` file doesn't exist at first agent startup, the Apstra software creates one.

### Device Agent Configuration File

You manage device agent configuration by editing the device agent configuration file directly. The Cumulus device agent config file is located at `/etc/aos/aos.conf`. See ["Apstra device agent configuration file" on page 901](#) for parameters. After updating the file, restart the Apstra device agent.

```

service aos stop
service aos start

```

## Device Agent Management

### IN THIS SECTION

- [Bootstrap Configuration | 248](#)
- [Cumulus Device Configuration Management | 248](#)

### *Bootstrap Configuration*

The concept of bootstrap configuration relates to the `/etc/network/interfaces` configuration section as applicable to the management IP address. Bootstrap configuration is prepended to configuration jobs that Apstra pushes. This helps ensure that Apstra does not overwrite any network configuration that may prevent the agent from reaching the controller.

The bootstrap configuration is typically pushed by configuration by the user, or automated with software such as Apstra's Aeon ZTP server, or the Apstra device installer project.

Bootstrap configuration contains the minimum necessary network settings for Apstra to connect to the Apstra controller.

```
auto mgmt
iface mgmt
address 127.0.0.1/8
vrf-table auto
post-up sudo service aos start

# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

auto eth0
iface eth0 inet dhcp
    vrf mgmt
```

### *Cumulus Device Configuration Management*

The Cumulus device agent manages the following files on the filesystem:

- `/etc/cumulus/ports.conf` - specifies how port breakouts are consumed on the Cumulus platform

- **/etc/frr/frr.conf** - contains all routing information for BGP on the device
- **/etc/network/interfaces** - handles all Layer 2 and Layer 3 configuration on the device, including CLAG, VLANs, VXLAN, IP Routing
- **/etc/hostname** - the file that Apstra manages the device hostname through
- **/etc/default/isc-dhcp-relay** - specifies interfaces that participate in DHCP Relay

## Deploy Device

Once the Agent is up and running it appears under Managed Devices, and can be Acknowledged and assigned to a Blueprint using the GUI per standard procedure.

## Uninstall Apstra Device Agent

### IN THIS SECTION

- [Stop Apstra Service | 249](#)
- [Remove Device | 250](#)
- [Remove Apstra Package from Cumulus | 250](#)

To remove the Apstra device agent you'll stop the agent, remove it from Apstra, then uninstall the agent.

You can uninstall the Apstra device Agent on Cumulus with the steps below. Since the Apstra agent itself is mostly a compressed (squashfs) image, it can be uninstalled in a few steps.

When uninstalling the Apstra configuration applied to the various files (frr.conf, /etc/network/interfaces, etc) is removed.

### *Stop Apstra Service*

To prevent the agent from immediately re-registering to the Apstra server, stop Apstra service on the agent before uninstalling the agent.

```
admin@cumulus:mgmt-vrf:~$ sudo service aos stop
admin@cumulus:mgmt-vrf:~$ sudo service aos status
aos.service - LSB: Start AOS device agents
Loaded: loaded (/etc/init.d/aos)
Active: inactive (dead) since Fri 2017-05-26 17:27:37 UTC; 5s ago
```

```
Process: 32086 ExecStop=/etc/init.d/aos stop (code=exited, status=0/SUCCESS)
Process: 31268 ExecStart=/etc/init.d/aos start (code=exited, status=0/SUCCESS)
```

### ***Remove Device***

Using the Apstra GUI, undeploy and unassign the device from the blueprint per standard procedures. You can also delete it entirely from the Managed Devices page.



**CAUTION:** If you uninstall the agent before removing it from the Apstra GUI, existing configuration can no longer be erased.

### ***Remove Apstra Package from Cumulus***

Remove the Apstra package with the following command:

```
sudo dpkg -r aos-device-agent
```

```
admin@cumulus:mgmt-vrf:~$ sudo dpkg -r aos-device-agent
(Reading database ... 25366 files and directories currently installed.)
Removing aos-device-agent (2.0.0-210) ...
Processing triggers for systemd (215-17+deb8u4) ...
```

Clean up any other lingering files left on the filesystem.

```
sudo rm -rf /opt/aos/
sudo rm -rf /etc/aos
sudo rm -rf /var/log/aos
sudo rm -rf /run/aos/*
sudo rm -rf /mnt/persist/.aos
sudo rm -rf /run/aos
sudo rm -rf /run/lock/aos
sudo rm -rf /tmp/aos_show_tech
sudo rm -rf /usr/sbin/aos*
```

Check if any other Apstra files remain on the filesystem.

```
admin@cumulus:mgmt-vrf:~$ sudo find / -iname '*aos*'
find: File system loop detected; `/.snapshots/1/snapshot' is part of the same file system loop
as `/'.
```

```
/home/admin/aos_device_agent_1.2.0-137_cumulus.run
/var/lib/dpkg/info/aos-device-agent.postrm
/var/lib/dpkg/info/aos-device-agent.list
```

Optionally, remove the management VRF from /etc/network/interfaces.

## Cumulus Agent Troubleshooting

### IN THIS SECTION

- [Check Apstra Status | 251](#)
- [List Running Processes | 252](#)
- [Display and Read Apstra Log Files | 257](#)
- [Configuration not Pushed | 259](#)
- [Can't Ping Apstra or Use Other Network Tools | 259](#)
- [Avg State is CRITICAL | 260](#)

### *Check Apstra Status*

Running `service aos status` provides output of Apstra service status.

```
root@cumulus:mgmt-vrf:~# service aos status
aos.service - LSB: Start AOS device agents
  Loaded: loaded (/etc/init.d/aos)
  Active: active (running) since Fri 2017-05-26 17:42:39 UTC; 1min 59s ago
  Process: 32086 ExecStop=/etc/init.d/aos stop (code=exited, status=0/SUCCESS)
  Process: 32381 ExecStart=/etc/init.d/aos start (code=exited, status=0/SUCCESS)
  CGroup: /system.slice/aos.service
          └─32468 tacspawner --daemonize=/var/log/aos/aos.log --pidfile=/var/run/aos.pid --
name=000C29CBF3A8 --hostname=000C29CBF3A8 --domainSocket=aos_spawner_sock --
hostSysdbAddress=tbl://aos_localtasks_sock --jsonConfig=/etc/aos/cumulus/agent.json --
eventLogSev=TaccSpawner/error,Mounter/error,Mountee/error,Nb...
```

```

└─32470 tacleafsystdb --agentName=000C29CBF3A8-LocalTasks-000C29CBF3A8-0 --partition=
--storage-mode=persistent --eventLogDir=. --eventLogSev=TaccSpawner/error,Mounter/error,Mountee/
error,NboAttrLog/error
└─32471 /usr/bin/python /usr/bin/aos_agent --
class=aos.device.common.ProxyDeploymentAgent.ProxyDeploymentAgent --name=DeploymentProxyAgent
device_type=Cumulus serial_number=@(SYSTEM_UNIQUE_ID)
└─32474 /usr/bin/python /usr/bin/aos_agent --
class=aos.device.common.DeviceKeeperAgent.DeviceKeeperAgent --name=DeviceKeeperAgent
serial_number=@(SYSTEM_UNIQUE_ID)
└─32484 /usr/bin/python /usr/bin/aos_agent --
class=aos.device.common.ProxyCountersAgent.ProxyCountersAgent --name=CounterProxyAgent
device_type=Cumulus serial_number=@(SYSTEM_UNIQUE_ID)
└─32511 /usr/bin/python /usr/bin/aos_agent --
class=aos.device.cumulus.CumulusTelemetryAgent.CumulusTelemetryAgent --name=DeviceTelemetryAgent
serial_number=@(SYSTEM_UNIQUE_ID)
└─32748 sh -c curl -k -f -sS -H "Content-Type: application/json" -X POST -d
'{"version":"1.2.0-137","serial_number":"000C29CBF3A8","platform":"cumulus"}' https://aos-
server/api/versions/device 2>&1
└─32749 curl -k -f -sS -H Content-Type: application/json -X POST -d
{"version":"1.2.0-137","serial_number":"000C29CBF3A8","platform":"cumulus"} https://aos-
server/api/versions/device

```

### List Running Processes

You can attach to the Apstra container with `service aos attach` command, then run normal Linux commands within the container. Any changes within this container are lost/destroyed after the container restarts again, it's a read-only instance.

```

root@cumulus:mgmt-vrf:/var/log/aos# service aos attach
aos@mclag-compute-1-leaf1:/# ps wax
  PID TTY          STAT       TIME COMMAND
    1 ?           Ss        0:21 /sbin/init
    2 ?           S          0:00 [kthreadd]
    3 ?           S          0:04 [ksoftirqd/0]
    5 ?          S<         0:00 [kworker/0:0H]
    7 ?           S          1:28 [rcu_sched]
    8 ?           S          0:00 [rcu_bh]
    9 ?           S          0:00 [migration/0]
   10 ?          S          0:00 [watchdog/0]
   11 ?           S          0:00 [watchdog/1]
   12 ?           S          0:00 [migration/1]

```

```

13 ?      S      0:04 [ksoftirqd/1]
15 ?      S<     0:00 [kworker/1:0H]
16 ?      S      0:00 [watchdog/2]
17 ?      S      0:00 [migration/2]
18 ?      S      0:08 [ksoftirqd/2]
20 ?      S<     0:00 [kworker/2:0H]
21 ?      S      0:00 [watchdog/3]
22 ?      S      0:00 [migration/3]
23 ?      S      0:07 [ksoftirqd/3]
25 ?      S<     0:00 [kworker/3:0H]
26 ?      S<     0:00 [khelper]
27 ?      S      0:00 [kdevtmpfs]
28 ?      S<     0:00 [netns]
29 ?      S<     0:00 [perf]
30 ?      S      0:00 [khungtaskd]
31 ?      S<     0:00 [writeback]
33 ?      SN     0:00 [ksmd]
34 ?      SN     0:00 [khugepaged]
35 ?      S<     0:00 [crypto]
36 ?      S<     0:00 [kintegrityd]
37 ?      S<     0:00 [bioset]
38 ?      S<     0:00 [kblockd]
39 ?      S<     0:00 [ata_sff]
40 ?      S<     0:00 [edac-poller]
42 ?      S<     0:00 [rpciod]
43 ?      S      0:00 [kswapd0]
44 ?      S      0:00 [fsnotify_mark]
45 ?      S<     0:00 [nfsiod]
54 ?      S<     0:00 [kthrotld]
57 ?      S      0:00 [scsi_eh_0]
58 ?      S<     0:00 [scsi_tmf_0]
59 ?      S      0:00 [scsi_eh_1]
60 ?      S<     0:00 [scsi_tmf_1]
61 ?      S      0:00 [scsi_eh_2]
62 ?      S<     0:00 [scsi_tmf_2]
63 ?      S      0:00 [scsi_eh_3]
64 ?      S<     0:00 [scsi_tmf_3]
69 ?      S      0:00 [scsi_eh_4]
70 ?      S<     0:00 [scsi_tmf_4]
71 ?      S      0:00 [scsi_eh_5]
72 ?      S<     0:00 [scsi_tmf_5]
76 ?      S<     0:00 [ipv6_addrconf]
77 ?      S<     0:00 [deferwq]

```

```

128 ?      S<      0:00 [bioset]
133 ?      S        0:00 [scsi_eh_6]
134 ?      S<      0:00 [scsi_tmf_6]
135 ?      S        0:03 [usb-storage]
136 ?      S        0:00 [scsi_eh_7]
137 ?      S<      0:00 [scsi_tmf_7]
138 ?      S        0:01 [usb-storage]
146 ?      S<      0:00 [kworker/0:1H]
147 ?      S<      0:00 [kworker/1:1H]
148 ?      S<      0:00 [kworker/3:1H]
149 ?      S<      0:00 [kworker/2:1H]
166 ?      S<      0:00 [btrfs-worker]
168 ?      S<      0:00 [btrfs-worker-hi]
169 ?      S<      0:00 [btrfs-delalloc]
170 ?      S<      0:00 [btrfs-flush_del]
171 ?      S<      0:00 [btrfs-cache]
172 ?      S<      0:00 [btrfs-submit]
173 ?      S<      0:00 [btrfs-fixup]
174 ?      S<      0:00 [btrfs-endio]
175 ?      S<      0:00 [btrfs-endio-met]
176 ?      S<      0:00 [btrfs-endio-met]
177 ?      S<      0:00 [btrfs-endio-rai]
178 ?      S<      0:00 [btrfs-endio-rep]
179 ?      S<      0:00 [btrfs-rmw]
180 ?      S<      0:00 [btrfs-endio-wri]
181 ?      S<      0:00 [btrfs-freespace]
182 ?      S<      0:00 [btrfs-delayed-m]
183 ?      S<      0:00 [btrfs-readahead]
184 ?      S<      0:00 [btrfs-qgroup-re]
185 ?      S<      0:00 [btrfs-extent-re]
186 ?      S        0:00 [btrfs-cleaner]
187 ?      S        0:06 [btrfs-transacti]
262 ?      S        0:00 [kauditd]
266 ?      Ss       0:05 /lib/systemd/systemd-journald
269 ?      Ss       0:00 /lib/systemd/systemd-udev
337 ?      S<      0:00 [kvm-irqfd-clean]
542 ?      S<s1     0:00 /sbin/auditd -n
694 ?      SNs      0:00 /usr/sbin/cron -f -L 38
697 ?      Ss       0:00 /lib/systemd/systemd-logind
702 ?      Ss       0:02 /usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile
--systemd-activation
708 ?      Ss       0:00 /usr/sbin/mcelog --daemon
710 ?      Ss       0:04 /usr/sbin/irqbalance --pid=/var/run/irqbalance.pid

```

```

715 ?      Ss      0:00 /usr/sbin/acpid
719 ?      Ssl     0:01 /usr/sbin/rsyslogd -n
733 ?      Ss      0:00 /usr/sbin/wd_keepalive
740 ?      Ss      9:17 /usr/bin/python /usr/sbin/smond
741 ?      Ss      0:24 /usr/bin/python /usr/sbin/ledmgrp
743 ?      S       0:00 /usr/sbin/dnsmasq -x /var/run/dnsmasq/dnsmasq.pid -u dnsmasq -7 /etc/
dnsmasq.d,.dpkg-dist,.dpkg-old,.dpkg
744 ?      Ss      0:43 /usr/bin/python /usr/sbin/pwmd
745 ?      S<s     0:05 /sbin/mstpd -d -v2
1053 ?     Ss      0:00 /usr/sbin/uuid --socket-activation
1196 ?     Ssl     1:34 /usr/bin/python /usr/bin/arp_refresh
1200 ?     Ss      0:00 /usr/bin/python /usr/lib/python2.7/dist-packages/clcmd_server.py
> /dev/null 2>&1
1240 ?     Ss      0:02 /usr/sbin/ntpd -n -u ntp:ntp -g
1242 ?     Ss      0:00 /usr/sbin/ptmd -l INFO
1255 ?     Ss      0:00 /usr/sbin/sshd -D
1978 ?     Ss      0:00 /sbin/dhclient -pf /run/dhclient.eth0.pid -lf /var/lib/dhcp/
dhclient.eth0.leases eth0
2310 ?     S       0:00 [kworker/2:2]
4633 ?     Ss      0:00 sshd: admin [priv]
4661 ?     S       0:00 sshd: admin@pts/2
4662 ?     Ss      0:00 -bash
4699 ?     S       0:00 sudo su -
4704 ?     S       0:00 su -
4726 ?     S       0:00 -su
5853 ?     S       0:00 [kworker/3:2]
8515 ?     S       0:00 [kworker/u8:1]
9264 ?     S<      0:00 [kworker/u9:6]
11195 ?    S       0:00 [kworker/2:1]
13477 ?    S<      0:00 [kworker/u9:0]
13480 ?    S<      0:00 [kworker/u9:1]
13481 ?    S<      0:00 [kworker/u9:2]
13482 ?    S<      0:00 [kworker/u9:3]
13810 ?    S       0:00 /bin/bash /etc/init.d/aos attach
13824 ?    S       0:00 /lib/systemd/systemd-udevd
13832 ?    S       0:00 /bin/bash
13835 ?    R+      0:00 ps wax
14541 ?    S       0:00 sudo su -
14547 ?    S       0:00 su -
14575 ?    S       0:00 -su
14645 ?    S       0:00 [kworker/u8:2]
14675 ?    S+      0:00 tail -f DeploymentProxyAgent.err
15057 ?    S<      0:00 [kloopd0]

```

```

15097 ?      S<      0:00 [kworker/u9:7]
15099 ?      Sl      0:00 tacspawner --daemonize=/var/log/aos/aos.log --pidfile=/var/run/
aos.pid --name=571254X1448071 --hostname=5
15103 ?      S        0:00 tacleafsysdb --agentName=571254X1448071-LocalTasks-571254X1448071-0 --
partition= --storage-mode=persisten
15107 ?      Sl      0:02 /usr/bin/python /usr/bin/aos_agent --
class=aos.device.common.DeviceKeeperAgent.DeviceKeeperAgent --name=D
15117 ?      Sl      0:03 /usr/bin/python /usr/bin/aos_agent --
class=aos.device.common.ProxyCountersAgent.ProxyCountersAgent --name
15138 ?      Sl      0:06 /usr/bin/python /usr/bin/aos_agent --
class=aos.device.cumulus.CumulusTelemetryAgent.CumulusTelemetryAgent
15139 ?      Sl      0:02 /usr/bin/python /usr/bin/aos_agent --
class=aos.device.common.ProxyDeploymentAgent.ProxyDeploymentAgent --
15373 ttyS1   Ss      0:00 /bin/login --
15403 ttyS1   S        0:00 -bash
15440 ttyS1   S        0:00 sudo su
15445 ttyS1   S        0:00 su
15467 ttyS1   S+      0:00 bash
15756 ?      Ss      0:00 dhclient -lf /var/lib/dhcp/dhclient.eth0.leases eth0
16287 ?      S<sl    5:55 /usr/sbin/switchd
16583 ?      Ds      0:29 /usr/bin/python /usr/sbin/portwd
16586 ?      S        0:00 [kworker/1:0]
16763 ?      S<s     0:00 /usr/lib/quagga/zebra -s 90000000 --daemon -A 127.0.0.1
16770 ?      S<s     0:00 /usr/lib/quagga/bgpd --daemon -A 127.0.0.1
16777 ?      S<s     0:00 /usr/lib/quagga/watchquagga -adz -r /usr/sbin/
servicebBquaggabBrestartbB%s -s /usr/sbin/servicebBquaggabB
16787 ?      Ss      0:00 sshd: admin [priv]
16815 ?      S        0:00 sshd: admin@pts/1
16816 ?      Ss      0:00 -bash
16853 ?      S        0:00 sudo su -
16858 ?      S        0:00 su -
16880 ?      S        0:00 -su
17070 ?      S+      0:00 tail -f DeploymentProxyAgent.err
18757 ?      Ss      0:00 lldpd: monitor .
18760 ?      S        0:01 lldpd: connected to dutmgmtsw2-eth0.dc1.apstra.com
21990 ?      S        0:00 [kworker/2:0]
22942 ?      S        0:00 [kworker/3:0]
24846 ?      S        0:00 [kworker/0:0]
27138 ?      S        0:00 [kworker/3:1]
27389 ?      S<      0:00 [bond41]
27584 ?      Ssl     0:06 /usr/bin/python /usr/sbin/clagd --daemon 10.0.0.63 bond41.2999
44:38:39:ff:00:01 --priority 32768 --backu
27665 ?      S        0:00 /sbin/bridge monitor fdb

```

```

27959 ?      S      0:00 [kworker/1:1]
28342 ?      S      0:00 [kworker/u8:0]
31882 ?      Ss     0:00 sshd: admin [priv]
31912 ?      S      0:00 sshd: admin@pts/0
31913 ?      Ss     0:00 -bash
32063 ?      S      0:00 [kworker/0:2]

```

### *Display and Read Apstra Log Files*

Apstra log files are stored in the `/var/log/aos` folder. There is typically one log file for each Apstra agent that runs. The Apstra device agent spawns off a series of other device agents for various purposes - telemetry, configuration rendering, counters, and agent health. You can read the plain-text `.err` files with any text editor.

```

root@cumulus:mgmt-vrf:/var/log/aos# ls -lah /var/log/aos
total 876K
drwxr-xr-x 1 root root 4.9K Oct 16 01:42 .
drwxr-xr-x 1 root root 360 Oct 16 01:44 ..
-rw----- 1 root root 644 Oct 16 01:42 _571254X1448071-0000000059e40e8c-000ea27f-checkpoint
-rw-r--r-- 1 root root 0 Oct 16 01:42 _571254X1448071-0000000059e40e8c-000ea27f-checkpoint-
valid
-rw----- 1 root root 0 Oct 16 01:42 _571254X1448071-0000000059e40e8c-000ea27f-log
-rw-r--r-- 1 root root 0 Oct 16 01:42 _571254X1448071-0000000059e40e8c-000ea27f-log-valid
-rw-r--r-- 1 root root 0 Oct 16 01:40 571254X1448071-0.14640.1508118008.log
-rw-r--r-- 1 root root 0 Oct 16 01:42 571254X1448071-0.15103.1508118156.log
-rw-r--r-- 1 root root 0 Oct 16 01:40 571254X1448071-0.err
-rw-r--r-- 1 root root 0 Oct 16 01:40 571254X1448071-0.out
-rw----- 1 root root 61K Oct 16 01:40 571254X1448071-
LocalTasks-571254X1448071-0_2017-10-16--01-40-08_14640-2017-10-16--01-40-08.tel
-rw----- 1 root root 105K Oct 16 02:16 571254X1448071-
LocalTasks-571254X1448071-0_2017-10-16--01-42-36_15103-2017-10-16--01-42-36.tel
-rw-r--r-- 1 root root 0 Oct 16 01:42 aos.log
-rw-r--r-- 1 root root 0 Oct 16 01:40 CounterProxyAgent.14642.1508118012.log
-rw-r--r-- 1 root root 0 Oct 16 01:40 CounterProxyAgent.14665.1508118023.log
-rw-r--r-- 1 root root 0 Oct 16 01:42 CounterProxyAgent.15105.1508118159.log
-rw-r--r-- 1 root root 0 Oct 16 01:42 CounterProxyAgent.15117.1508118171.log
-rw----- 1 root root 33K Oct 16 01:40
CounterProxyAgent571254X1448071_2017-10-16--01-40-14_14642-2017-10-16--01-40-14.tel
-rw----- 1 root root 42K Oct 16 01:40
CounterProxyAgent571254X1448071_2017-10-16--01-40-24_14665-2017-10-16--01-40-24.tel
-rw----- 1 root root 33K Oct 16 01:42

```

```

CounterProxyAgent571254X1448071_2017-10-16--01-42-41_15105-2017-10-16--01-42-41.tel
-rw----- 1 root root 42K Oct 16 01:42
CounterProxyAgent571254X1448071_2017-10-16--01-42-52_15117-2017-10-16--01-42-52.tel
-rw-r--r-- 1 root root 3.0K Oct 16 01:42 CounterProxyAgent.err
-rw-r--r-- 1 root root 0 Oct 16 01:40 CounterProxyAgent.out
-rw-r--r-- 1 root root 0 Oct 16 01:40 DeploymentProxyAgent.14641.1508118018.log
-rw-r--r-- 1 root root 0 Oct 16 01:42 DeploymentProxyAgent.15104.1508118166.log
-rw-r--r-- 1 root root 0 Oct 16 01:42 DeploymentProxyAgent.15139.1508118176.log
-rw----- 1 root root 39K Oct 16 01:40
DeploymentProxyAgent571254X1448071_2017-10-16--01-40-19_14641-2017-10-16--01-40-19.tel
-rw----- 1 root root 33K Oct 16 01:42
DeploymentProxyAgent571254X1448071_2017-10-16--01-42-47_15104-2017-10-16--01-42-47.tel
-rw----- 1 root root 39K Oct 16 01:43
DeploymentProxyAgent571254X1448071_2017-10-16--01-42-58_15139-2017-10-16--01-42-58.tel
-rw-r--r-- 1 root root 31K Oct 16 02:03 DeploymentProxyAgent.err
-rw-r--r-- 1 root root 0 Oct 16 01:40 DeploymentProxyAgent.out
-rw-r--r-- 1 root root 0 Oct 16 01:40 DeviceKeeperAgent.14644.1508118014.log
-rw-r--r-- 1 root root 0 Oct 16 01:42 DeviceKeeperAgent.15107.1508118166.log
-rw----- 1 root root 38K Oct 16 01:40
DeviceKeeperAgent571254X1448071_2017-10-16--01-40-15_14644-2017-10-16--01-40-15.tel
-rw----- 1 root root 38K Oct 16 01:43
DeviceKeeperAgent571254X1448071_2017-10-16--01-42-48_15107-2017-10-16--01-42-48.tel
-rw-r--r-- 1 root root 1.3K Oct 16 01:42 DeviceKeeperAgent.err
-rw-r--r-- 1 root root 0 Oct 16 01:40 DeviceKeeperAgent.out
-rw-r--r-- 1 root root 0 Oct 16 01:40 DeviceTelemetryAgent.14643.1508118012.log
-rw-r--r-- 1 root root 0 Oct 16 01:40 DeviceTelemetryAgent.14674.1508118021.log
-rw-r--r-- 1 root root 0 Oct 16 01:42 DeviceTelemetryAgent.15106.1508118165.log
-rw-r--r-- 1 root root 0 Oct 16 01:42 DeviceTelemetryAgent.15138.1508118177.log
-rw----- 1 root root 33K Oct 16 01:40
DeviceTelemetryAgent571254X1448071_2017-10-16--01-40-15_14643-2017-10-16--01-40-15.tel
-rw----- 1 root root 56K Oct 16 01:40
DeviceTelemetryAgent571254X1448071_2017-10-16--01-40-23_14674-2017-10-16--01-40-23.tel
-rw----- 1 root root 33K Oct 16 01:42
DeviceTelemetryAgent571254X1448071_2017-10-16--01-42-47_15106-2017-10-16--01-42-47.tel
-rw----- 1 root root 56K Oct 16 01:43
DeviceTelemetryAgent571254X1448071_2017-10-16--01-42-59_15138-2017-10-16--01-42-59.tel
-rw-r--r-- 1 root root 8.8K Oct 16 01:49 DeviceTelemetryAgent.err
-rw-r--r-- 1 root root 0 Oct 16 01:40 DeviceTelemetryAgent.out
-rw----- 1 root root 50K Oct 16 01:40
Spawner-571254X1448071_2017-10-16--01-40-08_14635-2017-10-16--01-40-08.tel
-rw----- 1 root root 51K Oct 16 01:43
Spawner-571254X1448071_2017-10-16--01-42-36_15096-2017-10-16--01-42-36.tel

```

- **DeviceTelemetryAgent.err** - contains all diagnostic output as it relates to device telemetry.
- **DeviceKeeperAgent.err** - tracks the health of the Apstra agent itself and how it mounts Apstra Graph Datastore to the Apstra Server
- **DeploymentProxyAgent.err** - This file logs all configuration options managed by Apstra, and describes whether the configuration job is a complete apply or partial apply. All configuration changes by Apstra are seen here.
- **CounterProxyAgent.err** - captures any custom counter collectors deployed on the switch
- **aos.log** - Unused
- **XXXXXXXXXXXXXXXX-0.err** - Unused

The .tel files are tacc event logs output, used for internal support at Apstra.

### *Configuration not Pushed*

If configuration did not get pushed, the device agent could be in 'telemetry-only' mode. Check/etc/aos/aos.conf for enable\_configuration\_service = 0.

You may also observe these log lines in /var/log/aos/DeviceProxyAgent.err, stating Configuration service disabled. Not setting mount timer.

Correcting this shows handle device deployment config in the log file.

```
2017-10-16 01:42:58,571 15139:INFO:aos.device.common.ProxyDeploymentSm:Device init sanity check
completed at 1508118178.571466
2017-10-16 01:42:58,571 15139:INFO:aos.device.common.ProxyDeploymentSm:Device rebooted: False
2017-10-16 01:42:58,572 15139:INFO:aos.device.common.ProxyDeploymentSm:Device undergo ztp: True
2017-10-16 01:42:58,630 15139:INFO:aos.device.common.ProxyDeploymentSm:handle device deployment
config: 571254X1448071 ddc: 1 dds: 1
2017-10-16 01:42:58,630 15139:INFO:aos.device.common.ProxyDeploymentSm:deploy ddc: 1 dds: 1
2017-10-16 01:42:58,635 15139:INFO:aos.device.common.ProxyDeploymentSm:Config up-to-date after
restart, not re-applying
```

### *Can't Ping Apstra or Use Other Network Tools*

If you ping the Apstra Server on the CLI while the device is configured in a VRF, you may get confusing error messages. Make sure to use ping -I mgmt on an ICMP echo to source it from the management VRF properly.

For other commands, use `vrf task exec mgmt`.

```
admin@cumulus:mgmt-vrf:~$ ping aos-server
sudo: unable to resolve host mclag-compute-1-leaf1
connect: Network is unreachable

admin@cumulus:mgmt-vrf:~$ ping -I mgmt aos-server
sudo: unable to resolve host mclag-compute-1-leaf1
ping: Warning: source address might be selected on device other than mgmt.
PING aos-server (172.20.85.3) from 172.20.85.6 mgmt: 56(84) bytes of data.
64 bytes from aos-server (172.20.85.3): icmp_seq=1 ttl=64 time=0.245 ms
```

### *Avg State is CRITICAL*

When the Cumulus VM runs out of RAM, we get kernel panics and the system continues to restart by itself. The cumulus default OVA VM ships with 512MB by default.

The root cause here is there was not enough RAM to run the VM.

```
Broadcast message from root@cumulus (somewhere) (Mon Jan 15 20:44:51 2018):

Avg state is CRITICAL. Last 10 values: [100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0, 100.0]. System will shutdown in 26 secs
```

## Agent Profiles (Devices)

### IN THIS SECTION

- [Create Agent Profile | 261](#)
- [Edit / Delete Agent Profile | 262](#)

Agent profiles enable the logical link between device credentials, a key-value store to be used in the device configuration, and a selection of user uploaded packages. With agent profiles, you can configure parameters for a certain class of devices that exist in the network and edit their device agent settings as a group. Agent profiles include the following details:

Table 19: Agent Profile Parameters

Name	Description
Name	To identify the device agent profile
Platform	OS family (EOS, Junos, NX-OS)
Username / Password	Admin/root username and password on the device
Open Options (off-box only)	<p>Passes configured parameters to off-box agents. For example, to use HTTPS as the API connection from off-box agents to devices, use the key-value pair: proto-https - port-443. The following default values can be overridden with open options:</p> <ul style="list-style-type: none"> <li>• commit_timeout - 60 (integer: seconds)</li> <li>• telemetry_timeout - 100 (integer: seconds)</li> <li>• probe_timeout: 5 (integer: seconds)</li> <li>• log_config_diff - True (boolean)</li> </ul>
Packages	Admin-provided software packages stored on the Apstra server that can be applied to each device agent that is created using the profile.

From the left navigation menu, navigate to **Devices > System Agents > Agent Profiles** to go to the agent profile list view. You can create, clone, edit, and delete agent profiles.

The screenshot shows the Juniper Apstra web interface. On the left, a navigation menu is open, showing the path: **Devices** (indicated by a red arrow and '1.') > **System Agents** > **Agent Profiles** (indicated by a red arrow and '2.'). The main content area displays the 'Agent Profiles' list view. At the top right, there is a '+ Create Agent Profile' button (indicated by a red arrow). Below the button, there is a search bar and a table of agent profiles. The table has columns: Platform, Has Username?, Has Password?, Packages Count, Open Options Count, and Actions. The first row shows a Junos profile with 'yes' for both username and password, 0 packages, and 0 open options. The Actions column contains icons for edit, clone, and delete (indicated by red arrows).

## Create Agent Profile

Before creating an agent profile, upload any "packages" on page 262 that are to be included in the agent profile.

1. From the left navigation menu, navigate to **Devices > System Agents > Agent Profiles** and click **Create Agent Profile**.
2. Select the platform from the drop-down list (optional).
3. Set a username and password (optional).
4. Add open options (optional).
5. Select package(s) (optional).
6. Click **Create** to create the agent profile and return to the list view.

## Edit / Delete Agent Profile

### IN THIS SECTION

- [Edit Agent Profile | 262](#)
- [Delete Agent Profile | 262](#)

### Edit Agent Profile

1. Either from the list view (Devices > System Agents > Agent Profiles) or the details view, click the **Edit** button for the profile to edit.
2. Make your changes.
3. Click **Update** to update the profile and return to the list view.

### Delete Agent Profile

1. Either from the list view (Devices > System Agents > Agent Profiles) or the details view, click the **Delete** button for the profile to delete.
2. Click **Delete** to delete the profile and return to the list view.

## Packages (Devices)

### IN THIS SECTION

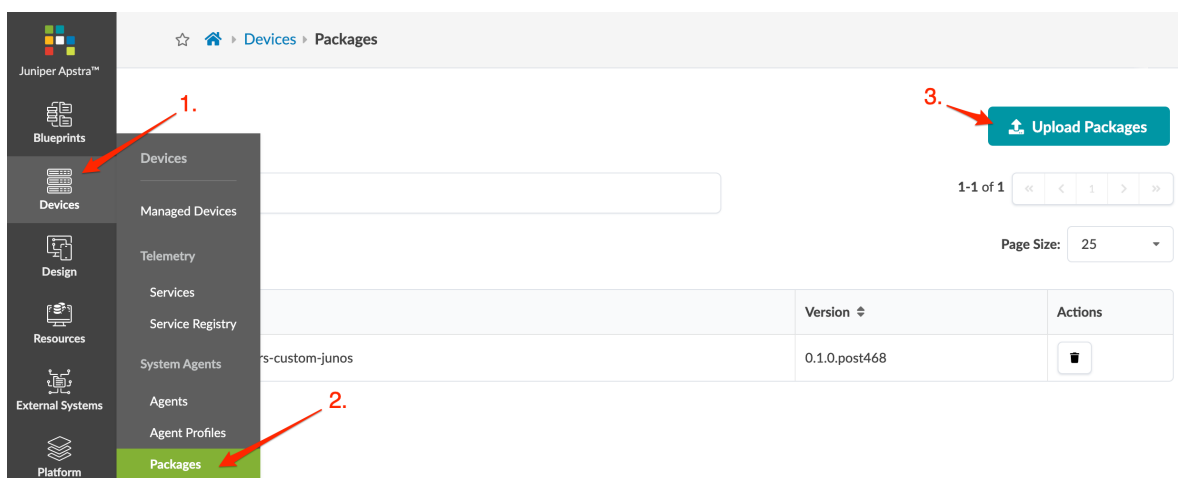
- [Packages Overview | 263](#)

## Packages Overview

You can extend Apstra capabilities by adding support for network operating systems (NOS), new telemetry collectors, third party software, and more. You upload packages (sometimes referred to as plugins) to the Apstra server, then include them in ["device agents" on page 184](#) and ["agent profiles" on page 260](#). Valid package types include .egg, .whl (Python wheel package) and .gz. One package can include one or more collectors for one or more OS platforms.

## Upload Packages

1. Download the required package(s) from the [Juniper Support Portal](#).
2. From the left navigation menu, navigate to **Devices > System Agents > Packages** and click **Upload Packages**.



3. For each package to upload, either click **Choose File** and navigate to the downloaded file, or drag and drop the file into the dialog window.
4. Click **Upload**, then close the dialog to return to the list view.

## OS Images (Devices)

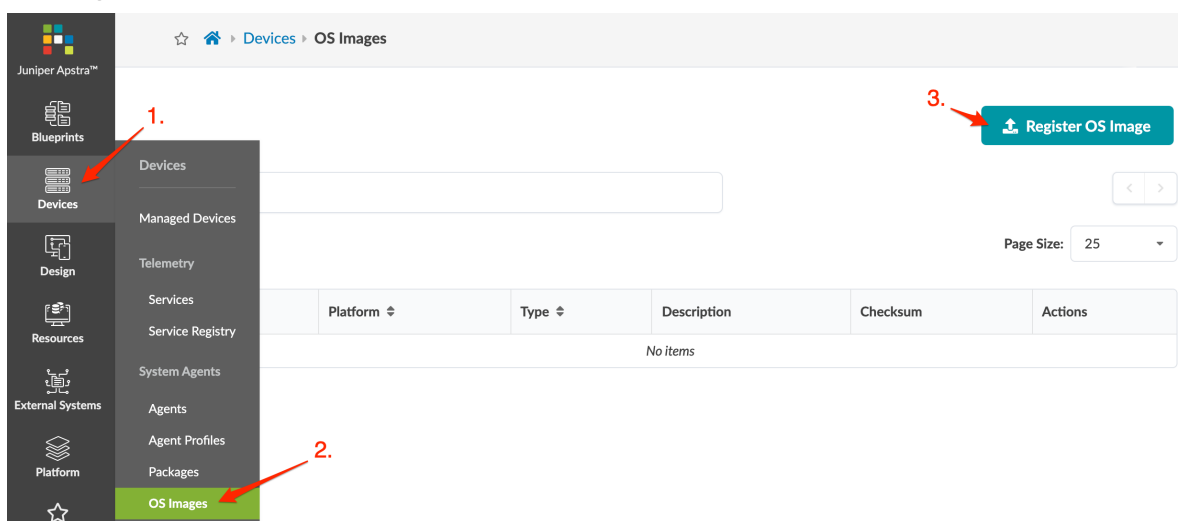
### IN THIS SECTION

- [Register OS Image | 264](#)
- [Method One: Upload Image | 265](#)
- [Method Two: Provide Image URL | 265](#)
- [Add Checksum \(Optional\) | 266](#)
- [Edit OS Image | 267](#)
- [Delete OS Image | 267](#)

In preparation for "NOS upgrade" on page 158, you must register the new OS image.

### Register OS Image

1. Obtain the OS image from the device vendor.
2. From the left navigation menu, navigate to **Devices > System Agents > OS Images** and click **Register OS Image** (top-right).



3. Select the platform from the drop-down list (EOS, CUMULUS, NXOS, SONIC, JUNOS) and enter a description.
4. Proceed by either uploading the image directly to the Apstra server or by providing a URL download link pointing to an image file on an accessible HTTP server (described in sections below).

## Method One: Upload Image

1. Select **Upload Image**, then either click **Choose File** and navigate to the image on your computer, or drag and drop the image from your computer into the dialog window and click **Open**.

### Register Device OS Image

Platform \*

NXOS

Description \*

EOS-4.22.5M

☒ Upload Image ☐ Provide Image URL

Image \*

Drag and drop file here or choose file by clicking the button.

 Choose File

Checksum

dbfd28d3597777a6ee5946b52277205fc714e11ab992574b7ef1156ffcd6e379979979f8c009f665fc212

SHA512 checksum (128 characters)

2. Add a checksum (optional) (described in section below).
3. Click **Upload** to upload and register the image with the Apstra software.
4. If the (optional) checksum can't be verified, the upgrade stops before rebooting the device.

## Method Two: Provide Image URL

If another HTTP server is accessible to the devices being upgraded via their network management port, you can register the OS Image instead of uploading it. Only HTTP URLs are supported. HTTPS, FTP, SFTP, SCP and others are not.

### 1. Select **Provide Image URL**.

#### Register Device OS Image

---

**Platform \***

**Description \***

☐ Upload Image
 ☒ Provide Image URL

**Image URL \***

**Checksum**

SHA512 checksum (128 characters)

2. Enter the URL that's pointing to the image on the other server.
3. Add a checksum (optional) (described in section below).
4. Click **Register** to register the image with the Apstra software.
5. If the (optional) checksum can't be verified, the upgrade stops before rebooting the device.

### Add Checksum (Optional)

The type of checksum is determined by the platform:

- Juniper Junos - MD5 (32 characters)
- Enterprise SONiC - MD5 (32 characters)
- Cisco NX-OS - SHA512 (128 characters)
- Arista EOS - SHA512 (128 characters)
- Cumulus Linux - MD5 (32 characters)

If the device vendor provides a checksum file, we recommend that you download it and copy it to the Checksum field. If a checksum file is not available, you can generate a checksum with the Linux **md5sum** or **shasum** commands, as applicable, or with equivalent programs.

```
$ shasum -a 512 EOS-4.20.11M.swi
dbfd28d3597777a6ee5946b52277205fc714e11ab992574b7ef1156ffcd6e379979979f8c009f665fc21212e4d38d1794
a412d79bab149f859aa72be417c0975 EOS-4.20.11M.swi
$
```

```
$ md5sum cumulus-linux-3.7.5-bcm-amd64.bin
MD5 (cumulus-linux-3.7.5-bcm-amd64.bin) = 8ffe069651cf4ffe8cfbe1162491761a
$
```

## Edit OS Image

To prevent failure, make sure to disable AAA authentication before proceeding with new image upgrades.

1. From the left navigation menu, navigate to **Devices > System Agents > OS Images** and click the **Edit** button for the OS Image to edit.
2. Make your changes.
3. Click **Update** to update the OS image.

## Delete OS Image

1. From the left navigation menu, navigate to **Devices > System Agents > OS Images** and click the **Delete** button for the OS Image to delete.
2. Click **Delete** to delete the OS image.

## Apstra ZTP (Devices)

### IN THIS SECTION

- [Apstra ZTP Overview | 268](#)
- [Download and Deploy Apstra ZTP VM | 273](#)
- [Configure Static Management IP Address \(Apstra ZTP\) | 274](#)

- [Configure ZTP User | 275](#)
- [Configure DHCP Server | 276](#)
- [Configure Controller IP Address for ZTP | 279](#)
- [Edit Apstra ZTP Configuration File | 279](#)
- [Apstra ZTP - Juniper Junos | 286](#)
- [Apstra ZTP - SONiC | 289](#)
- [Apstra ZTP - Cisco | 291](#)
- [Apstra ZTP - Arista | 294](#)
- [Apstra ZTP - Cumulus | 297](#)

## Apstra ZTP Overview

**NOTE:** This document applies to Apstra ZTP 4.0 versions. Use the Apstra ZTP version corresponding to the Juniper Apstra version you are using. Apstra versions earlier than 4.0 use Apstra ZTP versions 1.0.0 or 2.0.0. For more information, see the Juniper Apstra 3.3.0 User Guide.

Apstra ZTP is a Zero-Touch-Provisioning server for data center infrastructure systems. (Apstra ZTP replaces the community-supported Aeon-ZTPS software that was previously used for ZTP implementation in the Apstra environment.) Apstra ZTP enables you to bootstrap Apstra data center devices without considering the differences in underlying NOS mechanisms. ZTP, from an Apstra perspective, is a process that takes a device from initial boot to a point where it is managed by Apstra via device agents.

Depending on how ZTP is configured, the process may include (but not always) the following capabilities:

- A DHCP service
- Setting the device admin/root password
- Creating a device user for device system agent
- Upgrading / downgrading NOS
- Installing license (Cumulus only)
- On-box or Off-box Device System Agent installation

See also vendor-specific information:

- ["Juniper Junos" on page 286](#)
- ["Enterprise SONiC" on page 289](#)
- ["Cisco NX-OS" on page 291](#)
- ["Arista EOS" on page 294](#)
- ["Cumulus Linux" on page 297](#)

**NOTE:** To prevent being locked out of a device when there is a problem during the ZTP process, ZTP uses default, hard-coded credentials. These credentials are:

- root / admin
- aosadmin / aosadmin

You can use an Apstra-provided VM image (.ova, .qcow2.gz, .vhdx.gz) or build your own ZTP server and use the Apstra-provided device provisioning scripts as part of the existing ZTP/DHCP process to automatically install agents on devices as part of the boot process. The Apstra ZTP reference implementation consists of the following three phases:

#### 1. Generic DHCP Phase

- The device requests an IP address via DHCP.
- The device receives the assigned IP address and a pointer to a script to execute (or an OS image to install if using the Apstra-provided VM image).

#### 2. Initialization Phase

- The device downloads the ZTP script using TFTP.
- The device executes the downloaded script to prepare it to be managed. This includes verifying that the device is running a supported OS.

#### 3. I Agent Installation Phase

- The ZTP script makes an API call to install a device system agent on the device.

### Apstra ZTP 4.0 VM Server Resource Requirements

Apstra ZTP 4.0 runs as an Ubuntu 18.04 LTS server running a DHCP, HTTP, and TFTP server and includes Apstra provided ZTP scripts that must be customized for the your environment. The table below shows the minimum server specifications for a production environment:

Resource	Setting
Guest OS Type	Ubuntu 18.04 LTS 64-bit
Memory	2 GB
CPU	1 vCPU
Disk Storage	64 GB
Network	At least 1 network adapter. Configured for DHCP initially

### Apstra ZTP 4.0 Network requirements

Source	Destination	Ports	Role
Device agents	DHCP Server (renewals) & Broadcast (requests)	udp/67 -> udp/68	DHCP Client
Device agents	Apstra ZTP	any -> tcp/80	Bootstrap and API scripts
Arista and Cisco Device agents	Apstra ZTP	any -> udp/69	TFTP for POAP and ZTP
Apstra ZTP	Controller	any -> tcp/443	Device System Agent Installer API

In addition to the ZTP-specific network requirements, the Apstra ZTP server and device agents require connectivity to the controller. Refer to ["Required Communication Ports" on page 72](#) for more information.

Beginning with Apstra ZTP 4.0.0 and Apstra 4.0.0, you can monitor device ZTP status from the Apstra GUI.

From the left navigation menu, navigate to **Devices > ZTP Status > Devices**.

Juniper Apstra™

☆ 🏠 > Devices > ZTP Status > Devices

Query: All

1-6 of 6

Page Size: 25

System ID	ZTP Status	ZTP Latest Event	Last Updated	Actions
FLM...	Completed	Device Ready	2021-10-26, 14:05:37	👁️ 🗑️
FLN...	Completed	Device Ready	2021-10-26, 14:05:58	👁️ 🗑️
FLM...	Completed	Device Ready	2021-10-26, 13:58:06	👁️ 🗑️
FLM...	Completed	Device Ready	2021-10-26, 14:03:03	👁️ 🗑️
FD...	Completed	Device Ready	2021-10-26, 14:32:07	👁️ 🗑️
	Unknown		2021-10-28, 15:49:46	👁️ 🗑️

Each device interacting with DHCP and ZTP is listed along with its System ID (serial number) if known, ZTP Status, ZTP Latest Event and when the device status was last updated.

To see the full DHCP and ZTP log for the device, click the "Show Log" icon.

Log Preview

```

2021-05-11 22:33:06 DHCP OFFER on 172.20.22.7 to 50:c7:09:f5:87:b8 via eth0
2021-05-11 22:33:31 ZTP Start JNP48Y8C-CHAS 172.20.22.7
2021-05-11 22:33:31 Ztp conf: {'sonic-versions': [], 'sonic-image': 'aos_sonic_image', 'license': 'cumulus_license', 'aos-password': '**masked**'}
2021-05-11 22:33:31 Connected (version 2.0, client OpenSSH_7.5)
2021-05-11 22:33:31 Authentication (password) successful!
2021-05-11 22:33:32 Connected (version 2.0, client OpenSSH_7.5)
2021-05-11 22:33:32 Authentication (password) successful!
2021-05-11 22:33:32 Ensuring AOS supported OS..
2021-05-11 22:33:32 Connected (version 2.0, client OpenSSH_7.5)
2021-05-11 22:33:33 Authentication (password) successful!
2021-05-11 22:33:34 Junos version: 20.2R2-S3.5
2021-05-11 22:33:34 Configuring root user..
2021-05-11 22:33:34 Connected (version 2.0, client OpenSSH_7.5)
2021-05-11 22:33:35 Authentication (password) successful!
2021-05-11 22:33:36 Configuring device user..
2021-05-11 22:33:36 Connected (version 2.0, client OpenSSH_7.5)
2021-05-11 22:33:36 Authentication (password) successful!
2021-05-11 22:33:38 Custom config..
2021-05-11 22:33:38 Connected (version 2.0, client OpenSSH_7.5)
2021-05-11 22:33:38 Authentication (password) successful!
2021-05-11 22:33:38 Connected (version 2.0, client OpenSSH_7.5)
2021-05-11 22:33:39 Authentication (password) successful!
2021-05-11 22:33:39 Running custom configuration script
2021-05-11 22:33:39 Connected (version 2.0, client OpenSSH_7.5)
2021-05-11 22:33:39 Authentication (password) successful!

```

Download Log File

Any device that interacts with DHCP or ZTP is listed. If you don't need the logs for a device anymore, click the "Delete" icon.

Log files for all processes can be found in the /containers\_data/logs directory

```

root@apstra-ztp:/containers_data/logs# ls -l
total 7132
-rw-r--r-- 1 root root 6351759 Oct 28 17:47 debug.log
drwxr-xr-x 2 root root 4096 Oct 27 19:20 devices
-rw----- 1 root root 0 Oct 23 20:02 dhcpd.leases
-rw-r--r-- 1 root root 926980 Oct 28 17:39 info.log
-rw----- 1 root root 58 Oct 23 20:02 README
-rw----- 1 root root 469 Oct 27 02:13 rsyslog.log
root@apstra-ztp:/containers_data/logs# tail info.log
2020-10-28 17:16:38,786 root.status INFO Incoming: dhcpd dhcpd[18]: DHCPACK on
192.168.59.9 to 04:f8:f8:6b:36:91 via eth0
2020-10-28 17:18:04,299 root.status INFO Incoming: dhcpd dhcpd[18]: DHCPREQUEST for
192.168.59.9 from 04:f8:f8:6b:36:91 via eth0
2020-10-28 17:18:04,300 root.status INFO Incoming: dhcpd dhcpd[18]: DHCPACK on
192.168.59.9 to 04:f8:f8:6b:36:91 via eth0
2020-10-28 17:19:29,250 root.status INFO Incoming: dhcpd : -- MARK --
2020-10-28 17:19:29,442 root.status ERROR Failed to update status of all
containers: /api/ztp/service 404 b'{"errors":"Resource not found"}'
2020-10-28 17:33:29,353 root.status INFO Incoming: tftp : -- MARK --
2020-10-28 17:33:29,538 root.status ERROR Failed to update status of all
containers: /api/ztp/service 404 b'{"errors":"Resource not found"}'
2020-10-28 17:33:34,768 root.status INFO Incoming: status : -- MARK --
2020-10-28 17:39:29,349 root.status INFO Incoming: dhcpd : -- MARK --
2020-10-28 17:39:29,539 root.status ERROR Failed to update status of all
containers: /api/ztp/service 404 b'{"errors":"Resource not found"}'
root@apstra-ztp:/containers_data/logs#

```

Beginning with Apstra ZTP 4.0.0 and Apstra 4.0.0, you can monitor the ZTP services on the Apstra ZTP Server from the Apstra GUI.

From the left navigation menu, navigate to **Devices > ZTP Status > Services**.

The screenshot shows the Juniper Apstra web interface. The left navigation menu is open, highlighting 'Services' under 'ZTP Status'. The main content area displays a table of services with columns for IP Address, Service Status, and Last Updated. The table shows five services, all with a status of 'Up'.

Managed Devices	IP Address	Service Status	Last Updated
Telemetry	192.168.31.20	Up	2021-10-28, 15:45:29
Services	192.168.31.21	Up	2021-10-28, 15:45:29
Service Registry	192.168.31.17	Up	2021-10-28, 15:45:29
System Agents		Up	2021-10-28, 15:45:29
Agents	192.168.31.18	Up	2021-10-28, 15:45:29

Each service name includes its Docker IP Address, Service Status and when the service status was last updated.

## Download and Deploy Apstra ZTP VM

Apstra ZTP software is delivered on a standalone Apstra ZTP VM

1. As a registered support user, download the appropriate Apstra VM image from [Juniper Support Downloads](#).

VMware OVA image	apstra-ztp-4.0.*-<build-version>.ova (example: apstra-ztp-4.0.1-11.ova)
Microsoft Hyper-V	apstra-ztp-4.0.*-<build-version>.vhdx.gz (example: apstra-ztp-4.0.1-11.vhdx.gz)
Linux KVM QCOW2 image	apstra-ztp-4.0.*-<build-version>.qcow2.gz (example: apstra-ztp-4.0.1-11.qcow2.gz)

2. Validate the downloaded file against the SHA512/MD5 checksums provided.
3. Deploy the VM with the appropriate resources.
4. TFTP, NGINX (HTTP), DHCPd, Status. and MySQL Docker containers are enabled and run by default.

admin@apstra-ztp:~\$ docker ps			
CONTAINER ID	IMAGE	COMMAND	CREATED

```

STATUS
PORTS
NAMES
b0f398b45755      apstra/tftp      "sh /init.sh"      2 weeks ago      Up 28
minutes          0.0.0.0:69->69/
udp                                                      tftp
d93eed5d8dbe      apstra/nginx     "sh /init.sh"      2 weeks ago      Up 28
minutes          0.0.0.0:80->80/tcp, 0.0.0.0:443->443/tcp, 0.0.0.0:8080->8080/tcp, 0.0.0.0:31415-
>31415/tcp  nginx
8ed71af7cb2b      apstra/status    "sh /init.sh"      2 weeks ago      Up 28
minutes          8080/
tcp                                                      status
e7cf8ecb187f      apstra_ztp_dhcpd "sh /init.sh"      2 weeks ago      Up 28
minutes
                dhcpd
87ae2a77a1f9      mysql:8          "docker-entrypoint.s..." 2 weeks ago      Up 28
minutes          3306/tcp, 33060/
tcp                                                      db
admin@apstra-ztp:~$

```

5. If you do not want to use the Apstra ZTP DHCP Server, stop and disable the dhcpd container.

```

admin@apstra-ztp:~$ docker stop dhcpd
dhcpd
admin@apstra-ztp:~$ docker update --restart=no dhcpd
dhcpd
admin@apstra-ztp:~$

```

## Configure Static Management IP Address (Apstra ZTP)

By default, the Apstra ZTP Server attempts to assign an IP address for its eth0 interface via DHCP. If you're using the Apstra ZTP Server as a DHCP server, you must set a static management IP address.

1. SSH into the Apstra server as user **admin**. (ssh admin@<apstra-server-ip> where <apstra-server-ip> is the IP address of the Astra server.)
2. Edit the /etc/netplan/01-netcfg.yaml file to configure the static management IP address. See example below. (For more information about using netplan, see <https://netplan.io/examples>)

```

admin@apstra-ztp:~$ sudo vi /etc/netplan/01-netcfg.yaml
[sudo] password for admin:

```

```
# This file describes the network interfaces available on your system
# For more information, see netplan(5).
network:
  version: 2
  renderer: networkd
  ethernets:
    eth0:
      dhcp4: no
      addresses: [192.168.59.4/24]
      gateway4: 192.168.59.1
      nameservers:
        search: [example.com, example.net]
        addresses: [69.16.169.11, 69.16.170.11]
```

3. Apply the change with one of the following methods:

- Reboot the Apstra server with the command `sudo reboot`.
- Run the command `sudo netplan apply`.

## Configure ZTP User

You can use any configured Apstra GUI user that has API write access (such as admin), but we recommend that you create a designated user (for example "ztp") that is assigned the predefined role **device\_ztp**. The `device_ztp` role allows users with that role to make API calls to the controller to request

device system agent installation. For more information, see ["User / Role Management"](#) on page 699.

The screenshot shows the 'Create User' modal in the Apstra web interface. The modal has a dark header with the title 'Create User'. The main content area contains several input fields: 'Username' (with a red asterisk), 'First Name', 'Last Name', 'Email', 'Password' (with a red asterisk), and 'Repeat Password' (with a red asterisk). The 'Username' field contains the text 'ztp'. Below the password fields, there are two sections of radio buttons. The first section, 'Global Roles', has four options: 'administrator', 'device\_ztp' (which is selected with a blue checkmark), 'user', and 'viewer'. The second section, 'Per-Blueprint Roles', is currently empty. At the bottom right of the modal, there is a 'Create' button in a teal color and a 'Create Another?' checkbox.

## Configure DHCP Server

Apstra software comes with an ISC DHCP server for the device management network. If you use a different DHCP server, it's your responsibility to configure the same options as described in this guide for the Apstra-supplied DHCP server. For example, if you're using Juniper devices, you must ensure the server contains the following, so devices download the Apstra ZTP `junos_apstra_ztp_bootstrap.sh` file.

```
option space JUNIPER;
option JUNIPER.config-file-name code 1 = text;
option JUNIPER-encapsulation code 43 = encapsulate JUNIPER;
class "juniper" {
    match if (substring(option vendor-class-identifier, 0, 7) = "Juniper");
    option JUNIPER.config-file-name "junos_apstra_ztp_bootstrap.sh";
}
```

DHCP configuration files are on the Apstra ZTP VM in the `/containers_data/dhcp` directory.

```
admin@apstra-ztp:~$ sudo ls -l /containers_data/dhcp
total 16
-rw----- 1 root root 2533 Oct 21 00:35 dhcpd.conf
```

```
-rw----- 1 root root 146 Oct 21 00:35 Dockerfile
-rw----- 1 root root 932 Oct 21 00:35 init.sh
-rw----- 1 root root 1896 Oct 21 00:35 rsyslog.conf
admin@apstra-ztp:~$
```

**NOTE:** All configuration files are owned by root. You must use sudo to run commands as root using the sudo command or after becoming root with the sudo -s command.

1. Edit the dhcpd.conf file with vi or nano text editor.

```
admin@apstra-ztp:~$ sudo nano /containers_data/dhcp/dhcpd.conf
```

2. Add a "group" corresponding to the management network:

```
group {
    option tftp-server-name "192.168.59.4";
    subnet 192.168.59.0 netmask 255.255.255.0 {
        range 192.168.59.21 192.168.59.99;
        option routers 192.168.59.1;
    }
    host my-switch {
        hardware ethernet 34:17:eb:1e:41:80;
        fixed-address 192.168.59.100;
    }
}
```

tftp-server-name	IP address of ZTP server (not a URL)
subnet	IP management network and netmask
range	Range of dynamic DHCP IP addresses. Ensure the full range is available and no statically configured IP addresses from that range are used.
option routers	Default gateway router for management network
host	Static DHCP IP address

hardware ethernet of the management interface used for DHCP negotiations

fixed-address for device with hardware ethernet MAC. Use the Switch MAC address

3. The following DHCP parameters are optional:

```
ddns-update-style none;
option domain-search "example.internal";
option domain-name "example.internal";
option domain-name-servers 8.8.8.8, 8.8.4.4;
```

4. If you're using ZTP with Cumulus Linux, you must edit the following:

```
class "cumulus" {
    match if (substring(option host-name, 0, 7) = "cumulus");
    option cumulus-provision-url "tftp://192.168.59.4/ztp.py";
}
```

- cumulus-provision-url: TFTP URL with IP address of ZTP server

5. If you're using ZTP with SONiC, you must edit the following:

```
class "sonic" {
    match if (substring(option host-name, 0, 5) = "sonic");
    option sonic-provision-url "tftp://192.168.59.4/ztp.py";
}
```

sonic-provision-url: TFTP URL with IP address of ZTP server

6. After modifying any DHCP configuration, restart the Apstra ZTP DHCP process with the `sudo docker restart dhcpd` command.

```
admin@apstra-ztp:~$ docker restart dhcpd
dhcpd
admin@apstra-ztp:~$
```

## Configure Controller IP Address for ZTP

The controller IP and the Apstra ZTP username must be configured in the `/containers_data/status/app/aos.conf` file on the Apstra ZTP 4.0 Server.

```
admin@apstra-ztp:~$ sudo nano /containers_data/status/app/aos.conf
admin@apstra-ztp:~$ sudo nano /containers_data/status/app/aos.conf
```

```
{
  "ip": "192.168.59.3",
  "user": "ztp",
  "password": "ztp-user-password"
}
```

ip	IP Address of the controller
user	Username of the ZTP or admin user
password	User's password

## Edit Apstra ZTP Configuration File

Apstra ZTP VM includes a TFTP and nginx HTTP server. These servers do not require configuration. Both servers serve files out of the `/containers_data/tftp` directory.

```
admin@apstra-ztp:~$ sudo ls -l /containers_data/tftp/
total 232
-rwxr-xr-x 1 root root 2448 Apr 24 00:47 config_verifier.py
-rwxr-xr-x 1 root root 393 Apr 24 00:47 container_init.sh
-rwxr-xr-x 1 root root 170 Apr 24 00:47 cumulus_custom.sh
-rwxr-xr-x 1 root root 55 Apr 24 00:47 cumulus_license_file
-rwxr-xr-x 1 root root 192 Apr 24 00:47 Dockerfile
-rwxr-xr-x 1 root root 107 Apr 24 00:47 eos_custom.sh
-rwxr-xr-x 1 root root 5393 Apr 24 00:47 junos_apstra_ztp_bootstrap.sh
-rwxr-xr-x 1 root root 1799 Apr 24 00:47 junos_custom.sh
-rwxr-xr-x 1 root root 86 Apr 24 00:47 nxos_custom.sh
-rwxr-xr-x 1 root root 205 Apr 24 00:47 poap-md5sum
-rwxr-xr-x 1 root root 1843 Apr 24 00:47 rsyslog.conf
-rwxr-xr-x 1 root root 170 Apr 24 00:47 sonic_custom.sh
```

```
-rwxr-xr-x 1 root root 1910 Apr 24 00:47 ztp.json
-rwxr-xr-x 1 root root 86599 Apr 24 00:48 ztp.py
-rw----- 1 root root 86556 Apr 24 00:48 ztp.py.md5
admin@apstra-ztp:~$
```

The ztp.json file contains all configuration for the Apstra ZTP script ztp.py.

1. Edit the ztp.json file with vi or nano text editor.

```
admin@apstra-ztp:~$ sudo nano /containers_data/tftp/ztp.json
```

2. The ztp.json file is organized by the following:

**defaults** - Values are used for all devices unless more specific keys are defined.

```
"defaults": {
  "device-root-password": "root-password-123",
  "device-user": "admin",
  "device-user-password": "admin-password-123",
  "system-agent-params": {
    "agent_type": "onbox",
    "install_requirements": false
  }
}
```

**platform** - Values are used for all devices for a network platform ("cumulus", "nxos", "eos", "junos", "sonic") unless more specific keys are defined.

```
"cumulus": {
  "cumulus-versions": ["3.7.13"],
  "cumulus-image": "http://192.168.59.4/cumulus-linux-3.7.13-bcm-amd64.bin",
  "license": "cumulus_license_file",
  "custom-config": "cumulus_custom.sh",
}
```

**model** - Values are used for all devices for a specific device model (for example "N9K-C93180YC-FX").

```
"N9K-C93180YC-FXC3396": {
  "custom-config": "93180_cumulus_custom.sh",
}
```

**serial number** - Values are used for a device matching a specific device serial number ("525400B3C311" for example).

```
"525400B3C311": {
  "cumulus-versions": [ "3.7.13" ],
  "cumulus-image": "http://192.168.59.4/cumulus-linux-3.7.13-bcm-amd64.bin"
}
```

More specific data takes precedence over other data. For example, data for a specific serial number takes precedence over any other data, then model, then platform, then finally default data.

### 3. The ztp.json file uses the following keys:

**junos-versions** - Valid versions for Juniper Junos devices. If a device is not running a version in this list, ZTP upgrades the device with the junos-image image.

```
"junos-versions": [ "20.2R2-S3.5" ]
```

**junos-image** - Filename of the Juniper Junos TGZ image to be loaded if the running version does not match a version in the junos-versions list.

- By default, the image name is loaded from the ZTP server via TFTP from the ZTP server's / container\_data/tftp/ directory. For example: "junos-image": "jinstall-host-qfx-5-20.2R2-S3.5-signed.tgz"
- To use any HTTP server for image transfer, enter a valid HTTP URL with IP address. For example:  
"junos-image": "http://192.168.59.4/jinstall-host-qfx-5-20.2R2-S3.5-signed.tgz"

This example uses HTTP from the controller to transfer the Juniper Junos image.

**sonic-versions** - Valid versions for SONiC devices. If a device is not running a version in this list, ZTP upgrades the device with the sonic-image image.

```
"sonic-versions": [ "SONiC-OS-3.1.0a-Enterprise_Base" ]
```

sonic-image - Filename of the SONiC ONIE BIN image to be loaded if the running version does not match a version in the sonic-versions list.

- By default, the image name is loaded from the ZTP server via TFTP from the ZTP server's / container\_data/tftp/ directory. For example: "sonic-image": "sonic-3.1.0a-bcm.bin"
- To use any HTTP server for image transfer, enter a valid HTTP URL with IP address. For example: "sonic-image": "http://192.168.59.3/sonic-3.1.0a-bcm.bin"

This example uses HTTP from the controller to transfer the SONiC image.

nxos-versions - Valid versions for NX-OS devices. If a device is not running a version in this list, ZTP upgrades the device with the nxos-image image.

"nxos-versions": [ "9.2(2)", "9.3(6)" ]

nxos-image - Filename of the NX-OS image to be loaded if the running version does not match a version in the nxos-versions list.

- By default, the image name is loaded from the ZTP server via TFTP from the ZTP server's / container\_data/tftp/ directory. For example: "nxos-image": "nxos.9.3.6.bin"
- To use any HTTP server for image transfer, enter a valid HTTP URL with IP address. For example: "nxos-image": "http://192.168.59.4/nxos.9.3.6.bin"

This example uses HTTP from the ZTP server to transfer the Cisco NX-OS image.

eos-versions - Valid versions for Arista EOS devices. If a device is not running a version in this list, ZTP upgrades the device with the eos-image image.

"eos-versions": [ "4.22.3M", "4.24.5M" ]

`eos-image` - Filename of the Arista EOS SWI image to be loaded if the running version does not match a version in the `eos-versions` list.

- By default, the image name is loaded from the ZTP server via TFTP from the ZTP server's / `container_data/tftp/` directory. For example: `"eos-image": "EOS-4.24.5M.swi"`
- To use any HTTP server for image transfer, enter a valid HTTP URL with IP address. For example: `"eos-image": "http://192.168.59.3/dos_images/EOS-4.24.5M.swi"`

This example uses HTTP from the controller to transfer the Arista EOS image.

`cumulus-versions` - Valid versions for Cumulus Linux devices. If a device is not running a version in this list, ZTP upgrades the device with the `cumulus-image` image.

`"cumulus-versions": [ "3.7.12", "3.7.13" ]`

`cumulus-image` - Filename of the Cumulus Linux ONIE BIN image to be loaded if the running version does not match a version in the `cumulus-versions` list.

- By default, the image name is loaded from the ZTP server via TFTP from the ZTP server's / `container_data/tftp/` directory. For example: `"cumulus-image": "cumulus-linux-3.7.13-bcm-amd64.bin"`
- To use any HTTP server for image transfer, enter a valid HTTP URL with IP address. For example: `"cumulus-image": "http://192.168.59.4/cumulus-linux-3.7.13-bcm-amd64.bin"`

This example uses HTTP from the ZTP server to transfer the Cumulus Linux image.

`device-root-password` - The ZTP process sets the device root password to this value. For Arista EOS and Cisco NX-OS devices, the `device-root-password` is used to set the password for the system admin password.

`"device-root-password": "root-admin-password"`

`device-user` / `device-user-password` - Username and password that is used for the device system agent. Also, if necessary, the ZTP process creates a user on the device with this username and password.

`"device-user": "aosadmin",`  
`"device-user-password": "aosadmin-password"`

license - Filename of the Cumulus Linux license file in the TFTP directory or a URL pointing to the file on a HTTP server.	"license": "cumulus_license_file"
custom-config - The filename of the custom configuration shell script in the TFTP directory or a URL pointing to the file on a HTTP server. This shell script runs during ZTP allowing you to add custom configuration to the device. See Platform Specific Information section below for more information.	"custom-config": "cumulus_custom.sh"
system-agent-params	Information that is used to create new users and device system agents on devices, as described below.
agent_type - Agent type, onbox or offbox	"agent_type": "onbox"
install_requirements - Always set to false. Not currently needed for any supported Network Operating System.	"install_requirements": false

job\_on\_create - Set to install to install the agent on the device.

"job\_on\_create": "install"

#### Junos Example

```
{
  "junos": {
    "junos-versions": ["21.2R1-S2.2"],
    "junos-image": "http://10.85.24.52/
juniper/21.2R1-S2.2/jinstall-host-qfx-5e-
x86-64-21.2R1-S2.2-secure-signed.tgz",
    "device-root-password": "root123",
    "device-user": "admin",
    "device-user-password": "admin",
    "system-agent-params": {
      "platform": "junos",
      "agent_type": "offbox",
      "job_on_create": "install"
    }
  },
  "QFX10002-36Q": {
    "junos-versions": ["21.2R1-S2.2"],
    "junos-image": "http://10.85.24.52/
juniper/21.2R1-S2.2/jinstall-host-qfx-10-f-
x86-64-21.2R1-S2.2-secure-signed.tgz"
  },
  "JNP10002-60C [QFX10002-60C]": {
    "junos-versions": ["21.2R1-S1.3"],
    "junos-image": "http://10.85.24.52/
juniper/21.2R1-S1.3/junos-vmhost-install-qfx-
x86-64-21.2R1-S1.3.tgz"
  }
}
```

platform - (Required for off-box agents only) Set to the device platform ("eos", "nxos", "junos"). Must be in all lowercase.

"platform": "junos"

open\_options - (off-box agents only) Set to enable HTTPS between off-box agent to device API interface. If open\_options is not defined, the connection defaults to HTTP.

```
"open_options": {
  "proto": "https",
  "port": "443"
}
```

```

packages - Set to configure which additional SDK or
extended telemetry packages to upload to the
system agent.
"packages": [
  "aos-deployment-helper-nxos",
  "aosstdcollectors-builtin-nxos",
  "aosstdcollectors-custom-nxos"
]

```

For REST API documentation for all available system-agent-params options in /api/system-agents, refer to Swagger.

## Apstra ZTP - Juniper Junos

### IN THIS SECTION

- [Juniper Junos and ZTP Disk Space | 286](#)
- [Example: Juniper Junos ztp.json | 287](#)
- [Juniper Junos Bootstrap File | 287](#)
- [Juniper Junos Custom Config File | 287](#)
- [Restart Juniper Junos ZTP | 288](#)
- [Troubleshoot Juniper Junos ZTP | 289](#)

EX switches require Junos OS version 21.2 or higher. The Python module that's required for ztp is missing on EX switches using Junos OS versions below 21.2

### Juniper Junos and ZTP Disk Space

Apstra ZTP manages the bootstrap and lifecycle of Juniper Junos devices. It uses a custom script to create offbox agents, create local users and set other system configuration. As part of the ZTP process a new OS image is copied to the switch. Before installing Apstra ZTP ensure that the switch has sufficient disk space for the OS image.

```

root@leaf001-001-2> show system storage
Filesystem      Size  Used  Avail  Capacity  Mounted on
/dev/gpt/junos  6.0G  1.0G   4.5G    18%    /.mount
<...>

```

**Example: Juniper Junos ztp.json**

Juniper Junos Offbox Agent / Apstra ZTP 4.0

```
{
  "junos": {
    "junos-versions": [ "20.2R2-S3.5" ],
    "junos-image": "http://192.168.59.4/jinstall-host-qfx-5-20.2R2-S3.5-signed.tgz",
    "device-root-password": "root-password",
    "device-user": "admin",
    "device-user-password": "admin-password",
    "custom-config": "junos_custom.sh",
    "system-agent-params": {
      "platform": "junos",
      "agent_type": "offbox",
      "job_on_create": "install"
    }
  }
}
```

**Juniper Junos Bootstrap File**

Apstra ZTP uses a Python script to provision the device during ZTP. To allow the Python script (ztp.py) to run on the Junos device, additional configuration is required. Use the `junos_apstra_ztp_bootstrap.sh` script to bootstrap Apstra ZTP on Junos. It downloads and runs the ZTP script.

**Juniper Junos Custom Config File**

When configuring `custom-config` for Juniper Junos devices, refer to the example `junos_custom.sh`, a bash file executed during the ZTP process. It can set system configuration (such as Syslog, NTP, SNMP authentication) prior to device system agent installation.

```
#!/bin/sh

SOURCE_IP=$(cli -c "show conf interfaces em0.0" | grep address | sed 's/.*address \([0-9.]*\).*\n/\1/')

# Syslog
SYSLOG_SERVER="192.168.59.4"
SYSLOG_PORT="514"

# NTP
```

```

NTP_SERVER="192.168.59.4"
# SNMP
SNMP_NAME="SAMPLE"
SNMP_SERVER="192.168.59.3"

# Syslog
cli -c "configure; \
set system syslog host $SYSLOG_SERVER any notice ; \
set system syslog host $SYSLOG_SERVER authorization any ; \
set system syslog host $SYSLOG_SERVER port $SYSLOG_PORT ; \
set system syslog host $SYSLOG_SERVER routing-instance mgmt_junos ; \
commit and-quit"
cli -c "configure; \
set system syslog file messages any notice ; \
set system syslog file messages authorization any ; \
commit and-quit"

# NTP
cli -c "configure; \
set system ntp server $NTP_SERVER routing-instance mgmt_junos ; \
set system ntp source-address $SOURCE_IP routing-instance mgmt_junos ; \
commit and-quit;"

# SNMP
cli -c "configure; \
set snmp name $SNMP_NAME; \
set snmp community public clients $SNMP_SERVER/32 ; \
set snmp community public routing-instance mgmt_junos ; \
set snmp routing-instance-access access-list mgmt_junos ; \
commit and-quit"

```



**CAUTION:** If you set external AAA authentication (for example authentication-order), replicate the device system agent device-user and device-user-password in the AAA system. Otherwise, the device system agent generates an authentication error.

### Restart Juniper Junos ZTP

To erase (zeroize) the device and restart Juniper Junos ZTP process:

```
root@leaf3> request system zeroize
```

## Troubleshoot Juniper Junos ZTP

When in ZTP mode, the Juniper switch downloads the `ztp.py` and `ztp.json` files to the `/var/preserve/apstra` directory. For diagnostics, take note of the `/var/preserve/apstra/aosztp.log` file.

Additional useful messages can be found in `/var/log/messages` (search for 'ztp')

## Apstra ZTP - SONiC

### IN THIS SECTION

- [Enterprise SONiC and ZTP Overview | 289](#)
- [Example: Enterprise SONiC ztp.json | 290](#)
- [Enterprise SONiC Custom Config File | 290](#)
- [Restart Enterprise SONiC ZTP | 291](#)
- [| 291](#)

## Enterprise SONiC and ZTP Overview

**NOTE:** Apstra ZTP 4.0 used with Apstra version 4.0 has support for SONiC Enterprise Distribution devices. There is no support for any SONiC devices with earlier versions of Apstra ZTP or the software.

Apstra ZTP manages the bootstrap and life-cycle of Enterprise SONiC devices with on-box agents installed. It uses a custom script to create on-box agents, create local users and set other system configuration.

As part of the ZTP process a new OS image is copied to the switch. Before installing Apstra ZTP ensure that the switch has sufficient disk space for the OS image.

**NOTE:** If you are using ONIE to install Enterprise SONiC on a device, you must copy the image to the `/containers_data/tftp` directory and rename it to `onie-installer` or another ONIE download name (`onie-installer-x86_64-dell_z9100_c2538-r0` for example). When rebooting in ONIE, the device searches for this file on the HTTP then TFTP server. If the file is not found, ZTP fails. Once ONIE SONiC installation successfully completes, the SONiC device starts ZTP automatically.

### Example: Enterprise SONiC ztp.json

SONiC On-box Agent / Apstra ZTP 4.0

```
{
  "sonic": {
    "sonic-versions": [ "SONiC-OS-3.2.0-Enterprise_Advanced" ],
    "sonic-image": "http://192.168.59.4/sonic-3.2.0-GA-adv-bcm.bin",
    "device-root-password": "root-password",
    "device-user": "admin",
    "device-user-password": "admin-password",
    "custom-config": "sonic_custom.sh",
    "system-agent-params": {
      "agent_type": "onbox",
      "job_on_create": "install"
    }
  }
}
```

**NOTE:** If you use another device-user besides admin (aosadmin for example) Apstra ZTP creates this new user, but it does not change the password for the default SONiC admin user (password set to YourPaSsWoRd by default).

### Enterprise SONiC Custom Config File

When configuring custom-config for Enterprise SONiC devices, refer to the example sonic\_custom.sh, a bash executable file executed during the ZTP process. It can set system configuration (such as Radius authentication) prior to device system agent installation.

```
#!/bin/bash

sed -i s/"#Banner.*/"Banner \etc\issue.net"/ /etc/ssh/sshd_config

cat >& /etc/issue.net << EOF
Provisioned by AOS
Date: $(date)
EOF
```

```
service ssh restart
```

## Restart Enterprise SONiC ZTP

To restart the SONiC ZTP process, use the `sudo ztp enable` and `sudo ztp run` commands.

```
admin@sonic:~$ sudo ztp enable
admin@sonic:~$ sudo ztp run
ZTP will be restarted. You may lose switch data and connectivity, continue?[yes/NO] yes
admin@sonic:~$
```

## Apstra ZTP - Cisco

### IN THIS SECTION

- [Cisco NX-OS and ZTP Disk Space | 291](#)
- [Example: Cisco NX-OS ztp.json | 292](#)
- [Cisco NX-OS Custom Config File | 292](#)
- [Cisco NX-OS Off-box Agent Custom Config File | 293](#)
- [Restart Cisco NX-OS ZTP | 293](#)

## Cisco NX-OS and ZTP Disk Space

Ensure that sufficient disk space is available on the switch. As part of the ZTP process a new OS image is copied to the switch. Before installing Apstra ZTP ensure that the switch has sufficient disk space for the OS image.

```
switch1# dir bootflash: | include free|total
1296171008 bytes free
3537219584 bytes total
```

If ZTP is installing Cisco NX-OS image, the image (nxos.7.0.3.I7.7.bin for example) must be copied to the `/containers_data/tftp` directory ensuring correct file permissions.

**Example: Cisco NX-OS ztp.json**

```
{
  "nxos": {
    "nxos-versions": [ "9.2(2)" ],
    "nxos-image": "http://192.168.0.6/nxos.9.2.2.bin",
    "device-root-password": "admin-password",
    "custom-config": "nxos_custom.sh",
    "device-user": "admin",
    "device-user-password": "admin-password",
    "system-agent-params": {
      "agent_type": "onbox",
      "job_on_create": "install"
    }
  }
}
```

This configuration enables secure off-box agent HTTPS (port 443) between the off-box agent on the server and the device API.

**Cisco NX-OS Custom Config File**

When configuring `custom-config` for Cisco NX-OS devices, refer to the example `nxos_custom.sh`, a bash executable file executed during the ZTP process. It can execute NX-OS configuration commands to set the SSH login banner or other system configuration to be set prior to device system agent installation.

**NOTE:** You must add `copp profile strict` via the NX-OS custom-config file.

```
#!/bin/sh

/isan/bin/vsh -c "conf ; copp profile strict ; banner motd ~
#####
BANNER BANNER BANNER BANNER BANNER BANNER BANNER BANNER
#####
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Donec gravida, arcu vitae tincidunt sagittis, ligula
massa dignissim blah, eu sollicitudin nisl dui at massa.
Aliquam erat volutpat. Vitae pellentesque elit at
pulvinar volutpat. Etiam lacinia derp lacus, non
```

```
pellentesque nunc venenatis rhoncus.
#####
~"
```

### Cisco NX-OS Off-box Agent Custom Config File

If using Apstra ZTP to prepare a Cisco NX-OS device for use with off-box agents, you must have the custom-config file enable the following NX-OS configuration commands.

```
feature nxapi
feature bash-shell
feature scp-server
feature evmed
copp profile strict
nxapi http port 80
```

The following `nxos_custom.sh` can be used to add these along with a banner.

```
#!/bin/sh

/isan/bin/vsh -c "conf ; feature nxapi ; nxapi http port 443 ; feature bash-shell ; feature scp-
server ; feature evmed ; copp profile strict ; banner motd ~
#####
BANNER BANNER BANNER BANNER BANNER BANNER BANNER BANNER
#####
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Donec gravida, arcu vitae tincidunt sagittis, ligula
massa dignissim blah, eu sollicitudin nisl dui at massa.
Aliquam erat volutpat. Vitae pellentesque elit at
pulvinar volutpat. Etiam lacinia derp lacus, non
pellentesque nunc venenatis rhoncus.
#####
~"
```

### Restart Cisco NX-OS ZTP

**NOTE:** If an agent is already installed on the device, before you restart the device ZTP process remove the agent either via the UI device agent installer or manually via the device CLI.

```
C9K-172-20-65-5# guestshell destroy
```

Remove remaining AOS data from system

Removing the guest-shell deletes most of the data left by AOS. Some files are still on the bootflash:/.aos folder.

```
C9K-172-20-65-5# delete bootflash:./aos no-prompt
```

See ["Cisco Device Agents" on page 207](#) for more information.

To restart Cisco NX-OS ZTP process:

```
switch# write erase
```

```
switch# reload
```

## Apstra ZTP - Arista

### IN THIS SECTION

- [Arista EOS | 294](#)
- [Example: Arista EOS ztp.json | 295](#)
- [Arista EOS Custom Config File | 295](#)
- [Restart Arista EOS ZTP | 296](#)

## Arista EOS

**NOTE:** Apstra ZTP has limited support and known issues for virtual Arista EOS (vEOS) devices.

- ZTP EOS upgrades are not supported on vEOS devices. EOS versions for vEOS device must match eos-versions set in ztp.json file.
- ZTP Logging to the controller does not work for vEOS devices due to the lack of a device serial number. This will be addressed in a future version.

As part of the ZTP process a new OS image is copied to the switch. Before installing Apstra ZTP ensure that the switch has sufficient disk space for the OS image.

```
switch1#dir flash:
Directory of flash:/

<...>

3957878784 bytes total (3074723840 bytes free)
```

If ZTP is installing Arista EOS image, the image (EOS-4.22.3M.swi for example) you must copy to the /containers\_data/tftp directory.

### Example: Arista EOS ztp.json

Arista EOS On-box Agent / Apstra ZTP 4.0

```
{
  "eos": {
    "eos-versions": [ "4.24.5M" ],
    "eos-image": "http://192.168.59.3/EOS-4.24.5M.swi",
    "custom-config": "eos_custom.sh",
    "device-root-password": "admin-password",
    "device-user": "admin",
    "device-user-password": "admin-password",
    "system-agent-params": {
      "agent_type": "onbox",
      "job_on_create": "install"
    }
  }
}
```

### Arista EOS Custom Config File

When configuring custom-config for Arista EOS devices, refer to the example eos\_custom.sh, a bash executable file executed during the ZTP process. It can execute EOS configuration commands to set the SSH login banner or other system configuration to be set prior to device system agent installation.

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

```
FastCli -p 15 -c '$conf t\n service routing protocols model multi-agent\n hardware tcam\n system
profile vxlan-routing\n banner login\n
#####
UNAUTHORIZED ACCESS TO THIS DEVICE IS PROHIBITED
#####\n EOF\n'
```

**NOTE:** During the ZTP process, the EOS banner login is set to text saying "The device is in Zero Touch Provisioning mode ...". By default, this is copied to the permanent configuration by the ZTP script.

To prevent this, you **must** configure the custom-config pointing to a script (eos\_custom.sh for example), which configures a different banner login or configure no banner login.

There must be a space after any \n.

**NOTE:** If you're using EOS 4.22, Apstra recommends adding the service routing protocols model multi-agent to the device configuration along with any other configuration during ZTP which requires a device reboot to activate (system profile vxlan-routing for example). This ensures that this configuration is applied on reboot and added to the device pristine configuration.

## Restart Arista EOS ZTP



**CAUTION:** If an agent is already installed on the device, before you restart the device ZTP process remove the agent extension either via the UI Device Agent Installer or manually via the device CLI.

```
l2-virtual-001-leaf1#sho extensions
Name                               Version/Release  Status  Extension
-----
aos-device-agent-3.1.0-0.1.205.i386.rpm  3.1.0/0.1.205    A, I    1
```

A: available | NA: not available | I: installed | NI: not installed | F: forced

```
l2-virtual-001-leaf1#delete extension:aos-device-agent-3.1.0-0.1.205.i386.rpm
```

```
l2-virtual-001-leaf1#no extension aos-device-agent-3.1.0-0.1.205.i386.rpm
```

```
l2-virtual-001-leaf1#copy installed-extensions boot-extensions
```

Copy completed successfully.

```
l2-virtual-001-leaf1#delete /recursive flash:aos*
l2-virtual-001-leaf1#
```

See ["Arista Device Agents" on page 219](#) for more information.

To restart Arista EOS ZTP process:

```
localhost# delete flash:zerotouch-config
localhost# write erase
Proceed with erasing startup configuration? [confirm]y
localhost# reload
```

## Apstra ZTP - Cumulus

### IN THIS SECTION

- [Cumulus Linux | 297](#)
- [Example: Cumulus Linux ztp.json | 298](#)
- [Cumulus Linux Custom Config File | 299](#)
- [Restart Cumulus Linux ZTP | 299](#)
- [Troubleshoot Cumulus Linux ZTP | 300](#)

## Cumulus Linux

**NOTE:** Apstra ZTP has limited support for virtual Cumulus VX (CVX) devices.

- ZTP Cumulus Linux upgrades are not supported on CVX devices. Cumulus Linux versions for CVX device must match `cumulus-versions` set in `ztp.json` file.
- ZTP Logging to the controller does not work for CVX devices due to an unsupported device serial number (MAC address). This will be addressed in a future version.

As part of the ZTP process a new OS image is copied to the switch. Before installing Apstra ZTP ensure that the switch has sufficient disk space for the OS image.

```
cumulus@cumulus:~$ df -h /dev/sda4
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda4       5.8
```

If ZTP is installing Cumulus Linux image, copy the image (for example `cumulus-linux-3.7.12-bcm-amd64.bin`) to the `/containers_data/tftp` directory. Modify the file permissions to allow for download via tftp or http, for example:

```
admin@aos-server:/containers_data/tftp$ sudo chmod a+r cl-3.7.12-mlx-amd64.bin
[sudo] password for admin:
admin@aos-server:/containers_data/tftp$
```

**NOTE:** If you're using ONIE to install Cumulus Linux on a device, you must copy the image to the `/containers_data/tftp` directory and rename it to `onie-installer` or another ONIE download name (`onie-installer-x86_64-dell_s3000_c2338-r0` for example). When rebooting in ONIE, the device searches for this file on the TFTP server. If the file is not found, ZTP fails.

### Example: Cumulus Linux ztp.json

Cumulus Linux On-box Agent / Apstra ZTP 4.0

```
{
  "cumulus": {
    "cumulus-versions": [ "3.7.12" ],
    "cumulus-image": "http://192.168.59.4/cumulus-linux-3.7.12-bcm-amd64.bin",
    "device-root-password": "root-password",
    "license": "cumulus_license_file",
    "custom-config": "cumulus_custom.sh",
    "device-user": "admin",
    "device-user-password": "admin-password",
    "system-agent-params": {
      "agent_type": "onbox",
      "job_on_create": "install"
    }
  }
}
```

```
}
}
```

### Cumulus Linux Custom Config File

When configuring `custom-config` for Cumulus Linux devices, refer to the example `cumulus_custom.sh`, a bash executable file executed during the ZTP process. It can set the SSH login banner or other system configuration to be set prior to device system agent installation.

```
#!/bin/bash

sed -i s/"#Banner.*"/"Banner \\/etc\\/issue.net"/ /etc/ssh/sshd_config

cat >& /etc/issue.net << EOF
Provisioned by AOS
Date: $(date)
EOF

service ssh restart
```

### Restart Cumulus Linux ZTP



**CAUTION:** If an agent is already installed on the device, before you restart the device ZTP process remove the agent either via the UI device agent installer or manually via the device CLI.

```
admin@cumulus:mgmt-vrf:~$ sudo dpkg -r aos-device-agent
(Reading database ... 25366 files and directories currently installed.)
Removing aos-device-agent (2.0.0-210) ...
Processing triggers for systemd (215-17+deb8u4) ...
```

See ["Cumulus Device Agents" on page 239](#) for more information.

To restart the ONIE install and Cumulus Linux ZTP process:

```
cumulus@cumulus:mgmt-vrf:~$ sudo curl -o image.bin tftp://<tftpserver-ip>/cumulus-linux-3.7.5-
bcm-amd64.bin
cumulus@cumulus:mgmt-vrf:~$ sudo onie-install -a -f -i image.bin
cumulus@cumulus:mgmt-vrf:~$ sudo reboot
```

To just do the Cumulus Linux ZTP process:

```
cumulus@cumulus:mgmt-vrf:~$ sudo echo "cumulus" >& /etc/hostname
cumulus@cumulus:mgmt-vrf:~$ sudo net del vrf mgmt && net commit
cumulus@cumulus:mgmt-vrf:~$ sudo ztp -R && sudo ztp -s
cumulus@cumulus:mgmt-vrf:~$ sudo reboot
```

## Troubleshoot Cumulus Linux ZTP

When in ZTP mode, the switch downloads the ztp.py, ztp.json, OS image and license files to the /mnt/persist directory. For diagnostics, take note of the /mnt/persist/aosztp.log file.

Additional useful messages can be found in /var/log/syslog (search for 'ztp')

## Device Profiles (Devices)

### IN THIS SECTION

- [Device Profile Overview | 300](#)
- [Create Device Profile | 308](#)
- [Edit Device Profile | 309](#)
- [Delete Device Profile | 312](#)
- [Juniper Device Profiles | 312](#)
- [SONiC Device Profile | 314](#)
- [Cumulus Device Profile | 359](#)

## Device Profile Overview

### IN THIS SECTION

- [Summary | 301](#)
- [Selector | 301](#)

- [Capabilities | 302](#)
- [Supported Features \(Cisco only\) | 303](#)
- [Ports | 304](#)
- [View Device Profiles | 308](#)

Device profiles define capabilities of supported hardware devices. Some feature capabilities have different behaviors across NOS versions and thus, capabilities can be expressed per NOS version. By default, the version matches all supported versions. As additional hardware models are qualified, they are added to the ["list of qualified devices" on page 894](#).

Device profiles are associated with ["logical devices" on page 84](#) (abstractions of physical devices) to create ["interface maps" on page 91](#).

The following sections describe device profile parameters.

Summary

Table 20: Device Profile Summary

Summary Section	Description
Name	Name of device profile. 64 characters or fewer.
Number of slots	Number of slots or modules on the device. Modular switches have multiple slots.
Start from ID	

Selector

The Selector section contains device-specific information to match the hardware device to the device profile as described below:

Table 21: Device Profile Selector

Selector Section	Description
Manufacturer	Selected from drop-down list

Table 21: Device Profile Selector *(Continued)*

Selector Section	Description
Model	Determines whether a device profile can be applied to specific hardware. Selected from drop-down list or entered as a regular expression (regex).
OS family	Defines how configuration is generated, how telemetry commands are rendered, and how configuration is deployed on a device. Selected from drop-down list.
Version	Determines whether a device profile can be applied to specific hardware. Selected from drop-down list or entered as regex.

## Capabilities

The hardware and software capabilities defined in this section can be leveraged in other parts of the Apstra environment to adapt the generated configuration, or to prevent an incompatible situation. With the exception of ECMP, hardware capabilities modify configuration rendering or deployment.

Capabilities include the following details:

Table 22: Device Profile Capabilities

Capabilities Section	Description
CPU (cpu:string)	Describes the CPU architecture of the device. For example: "x86"
Userland (bits) (userland:integer)	Type of userland (application binary/kernel) the device supports. For example: "32" or "64".
RAM (GB) (ram:integer)	Amount of memory on the device. For example: "16"
ECMP limit (ecmp_limit:integer)	Maximum number of Equal Cost Multi Path routes. For example: "64". This field changes BGP configuration on the device (ecmp max-paths).
Form factor (form_factor:string)	Number of rack units (RU)s on the device. For example: "1RU", "2RU", "6RU", "7RU", "11RU", "13RU"
ASIC (asic:string)	The switch chipset ASIC. For example: "T2", "T2(3)", "T2(6)", "Arad(3)", "Alta", "TH", "Spectrum", "XPliant XP80", "ASE2", "Jericho". Used to assist telemetry, configuration rendering and VXLAN routing semantics
LXC (lxc_support: boolean)	Selected if the device supports LXC containers.

Table 22: Device Profile Capabilities (*Continued*)

Capabilities Section	Description
ONIE (onie: boolean)	Selected if the device supports ONIE.

### Supported Features (Cisco only)

**COPP** - When Control Plane Policing is enabled (COPP), strict CoPP profile config is rendered for the specified NX-OS version resulting in the following configuration rendering:

```
terminal dont-ask
copp profile strict
```

This terminal dont-ask config is needed only when enabling the CoPP profile strict config, since we do not want NX-OS to wait for confirmation:

```
switch(config)# copp profile strict
This operation can cause disruption of control traffic. Proceed (y/n)? [no] ^C
switch(config)#
switch(config)# terminal dont-ask
switch(config)# copp profile strict
switch(config)#
```

CoPP is enabled by default, except for Cisco 3172PQ NXOS. Multiple versions can be specified.

**Breakout** - Enable breakout to indicate that ports on specified modules can be broken out to lower speed split ports.

Apstra software first un-breakouts all ports that are breakout-capable, and then applies the proper breakout commands according to intent. This is based on the assumption that the global negation command `no interface breakout module<module_number>` can always be applied successfully to a module with breakout capable ports. (This is idempotent when applied on ports that are not broken out.) However, we recognize that this assumption may be broken in future versions of NX-OS, or with a certain combination of cables / transceivers inserted into breakout-capable ports.

The example below is for the negation command for a module (1) that is set to True:

```
no interface breakout module 1
!
```

Since the negation command is always applicable per module, each module is specified individually. The advantages of this include:

- In modular systems, not all line cards have breakout capable ports.
- In non-modular systems, the breakout capable ports may not always be in module 1.

Breakout is enabled by default except for the following devices with modules incapable of breaking out ports: 3172PQ NXOS, 9372TX NXOS, C9372PX NXOS, C9396PX NXOS, NXOSv.

**Historical Context** - With a particular version of NX-OS the POAP stage would apply breakout config on those ports which are breakout capable. POAP behavior, introduced in 7.0(3)I4(1) POAP, determines which breakout map (for example, 10gx4, 50gx2, 25gx4, or 10gx2) brings up the link connected to the DHCP server. If breakout is not supported on any of the ports, POAP skips the dynamic breakout process. After the breakout loop completes, POAP proceeds with the DHCP discovery phase as normal. Apstra reverts any such breakout config that might have been rendered during the POAP stage to ensure that the ports are put back to default speed by applying the negation command.

**Sequence Numbers Support** - Applicable to autonomous system (AS) path. Enable when the device supports sequence numbers. Apstra sequences into the entry list to resequence and generate config as follows:

```
ip as-path access-list MyASN seq 5 permit ^$
ip as-path access-list Rtr seq 5 permit ^3
ip as-path access-list Srvr seq 15 permit _103$
```

The numbers 5 and 15 are sequence numbers applicable to devices that support AS sequencing.

Sequence numbers support is enabled for all Cisco device profiles by default (except Cisco 3172PQ NXOS, which does not support sequence numbers). For platforms that do not support sequence numbers, disabling this feature ensures that the AS sequence numbers are removed from the device model dictionary to avoid addition and negation in the event that something is resequenced. This scenario has no requirement to render anything on these platforms, because the entry can't be sequenced.

**Other supported features** - not available from the Apstra GUI include "vxlan", "bfd", "vrf\_limit", "vtep\_limit", "floodlist\_limit", "max\_l2\_mtu", and "max\_l3-mtu". They can be included in the backend using the following format:

key : value :: feature : feature\_properties Example: 32 vtep\_limit: 32

## Ports

The ports section defines the types of available ports, their capabilities and how they are organized.

Every port contains a collection of supported speed transformations. Each transformation represents the breakout capability (such as 1-40GBe port breaking out to 4-10GBe ports), and hence contains a collection of interfaces.

Example: If port 1 is a QSFP28 100->4x10, 100->1x40 breakout capable port, then port 1 has a collection of three transformations, one each for 4x10, 1x40 and 1x100 breakouts. The transformation element in the collection which represents the 4x10 has a collection of 4 interfaces, 1x40 and 1x100 has a collection of 1 interface.

Ports parameters include the following details:

**Table 23: Device Profile Ports**

Ports Section	Description
Port Index (port_id: integer)	Indicates a unique port in the collection of ports in the Device Profile.
Row Index (row_id: integer)	Represents the top-to-bottom dimensions of the port panel. Shows where the port is placed in the device's panel. For instance, in a panel with two rows and many columns the row index is either "1" or "2".
Column Index (column_id: integer)	Represents the left-to-right dimensions of the port panel. Shows where the port is placed in the device's panel. For instance, in a panel with thirty-two ports and two rows, the column index is in the range of "1" through "16".
Panel Index (panel_id: integer)	Indicates the panel that the port belongs to given the physical layout of ports in the device specification
Slot ID (slot_id: integer)	Represents the module that the port belongs to. A modular switch has more than one slot. In fixed function network function devices, Slot ID is usually "0".
Failure Domain (failure_domain_id: integer)	Indicates if multiple panels are relying on the same hardware components. Used when creating the cabling plan to ensure that two uplinks are not attached to the same failure domain.
Connector Type (connector_type: string)	Port transceiver type. Speed capabilities of the port are directly related to the connector type, given that certain connector types can run in certain speeds. For instance, "sfp", "sfp28", "qsfp", "qsfp28".
Transformations (transformations: list)	Possible breakouts for the port. Every entry is a specific supported speed. Each transformation has a collection of interfaces.

Table 23: Device Profile Ports *(Continued)*

Ports Section	Description
Number of interfaces (interfaces:list)	Dependent on the breakout capability of the port. For a transformation representing a certain breakout speed, the interfaces contain information about the interface names and interface settings with which the device intends to be configured. The "setting" information is crucial for configuring the interfaces correctly on the device.

Based on the OS information entered in the device profile's selector field, the Apstra GUI displays the applicable settings fields. The fields vary with the vendor OS (as found in examples below). When a device profile is created or edited, the "setting" is validated from the vendor-specific schema as listed below.:

```

eos_port_setting = Dict({
  'interface': Dict({
    'speed': Enum([
      '', '1000full', '10000full', '25gfull', '40gfull',
      '50gfull', '100gfull',
    ])),
  'global': Dict({
    'port_group': Integer(),
    'select': String()
  })
})

```

```

nxos_port_setting = Dict({
  'interface': Dict({
    'speed': Enum([
      '', '1000', '10000', '25000', '40000', '50000',
      '100000',
    ])),
  'global': Dict({
    "port_index": Integer(),
    "speed": String(),
    "module_index": Integer()
  })
})

```

```

junos_port_setting = Dict({
  'interface': Dict({

```

```

        'speed': Enum([
            '', 'disabled', '1g', '10g', '25g', '40g', '50g', '100g'
        ])),
    'global': Dict({
        'speed': Enum([
            '', '1g', '10g', '25g', '40g', '50g', '100g'
        ]),
        "port_index": Optional(Integer()),
        "fpc": Optional(Integer()),
        "pic": Optional(Integer())
    })
})

sonic_port_setting = Dict({
    'interface': Dict({
        "command": Optional(String()),
        "speed": String(),
        "lane_map": Optional(String())
    })
})

cumulus_port_setting = Dict({
    'interface': Dict({
        'speed': String(),
        'command': String()
    })
})

```

Apstra does not necessarily use all the information above for modeling. It's made available to other Apstra API orchestration tools for collection and use.

## View Device Profiles

From the left navigation menu in the Apstra GUI, navigate to **Devices > Device Profiles** to go to the device profile list view. You can create, clone, edit, and delete device profiles.

Juniper Apstra™

Devices > Device Profiles

1-25 of 187

Page Size: 25

	Manufacturer	Hardware Model	Modular?	OS Family	OS Version	ASIC	Actions
4X-O	Accton	5712-54X-O.*	no	Cumulus	((3\[5-7]\.d+)((4\[2]\.d+))(\.d+)?	T2	[edit] [clone] [delete]
2X-O	Accton	6712-32X-O.*	no	Cumulus	((3\[5-7]\.d+)((4\[2]\.d+))(\.d+)?	T2	[edit] [clone] [delete]
2-54X_SONIC_BRCM_BUZZNIK_PLUS	Edgecore Accton	5712-54X-O.*	no	SONIC	.*3\[34].*	T2	[edit] [clone] [delete]
5T_SONIC_BRCM_BUZZNIK_PLUS	Edgecore Accton	5835-54T-O.*	no	SONIC	.*3\[34].*	T3	[edit] [clone] [delete]
5X_SONIC_BRCM_BUZZNIK_PLUS	Edgecore Accton	5835-54X-O.*	no	SONIC	.*3\[34].*	T3	[edit] [clone] [delete]
6-56X_SONIC_BRCM_BUZZNIK_PLUS	Edgecore Accton	7326-56X-O.*	no	SONIC	.*3\[34].*	T3	[edit] [clone] [delete]
2-32X_SONIC_BUZZNIK_PLUS	Edgecore Accton	7712-32X-O.*	no	SONIC	.*3\[34].*	TH	[edit] [clone] [delete]
Accton-AS7726-32X_SONIC_BUZZNIK_PLUS	Edgecore Accton	7726-32X-O.*	no	SONIC	.*3\[34].*	T3	[edit] [clone] [delete]

## Create Device Profile

**NOTE:** When you upgrade the Apstra server, predefined device profile changes applicable to that version are also updated and applied to the imported interface maps in blueprints. If you create (or clone) device profiles, they are not managed or updated when you upgrade the Apstra server.

Device profiles contain extensive hardware model details. Make sure the profile accurately describes all hardware characteristics. For assistance, contact ["Juniper Support" on page 777](#).

1. From the left navigation menu, navigate to **Devices > Device Profiles** and click **Create Device Profile**.
2. If you've created a JSON payload you can import it by clicking **Import Device Profile** and selecting the file. Otherwise, continue to the next step.
3. Enter a device profile name.
4. Configure the device profile to match the characteristics of the physical device. See ["Device Profile Overview" on page 300](#) for details.
5. Click **Create** to create the device profile and return to the list view.

## Edit Device Profile

### IN THIS SECTION

- [Edit Device Profile | 309](#)
- [Edit Device Profile - Example | 309](#)

## Edit Device Profile



**CAUTION:** Editing a device profile can lead to a mismatch between the profile's stated abilities and the device's actual capabilities, potentially leading to unexpected results. Do not edit a predefined device profile. Those changes would be discarded when you upgrade the Apstra server. If you want to use a modified version of a predefined device profile, we recommend cloning a predefined one instead.

If a device profile is used in an ["interface map" on page 91](#), you may not be able to change it if it would adversely affect that interface map.

1. Either from the list view (Devices > Device Profiles) or the details view, click the **Edit** button for the device profile to edit.
2. Make your changes.
3. Click **Update** (bottom-right) to update the device profile and return to the list view.

### Edit Device Profile - Example

This example shows how to set a port on an **Arista DCS-7050TX-72Q** device to auto-negotiate its speed.













1. From the left navigation menu, navigate to **Devices > Device Profiles** and click the **Edit** button for **Arista DCS-7050TX-72Q**

☆ 🏠 > Devices > Device Profiles

[+ Create Device Profile](#)


Query: Manufacturer == "Arista" and Name == "Arista DCS-7050TX" 1-4 of 4

Page Size: 25

Name ▾	Manufacturer ⇅	Hardware Model ⓘ ⇅	Modular? ⇅	OS Family ⇅	OS Version ⓘ ⇅	ASIC ⇅	Actions
<a href="#">Arista DCS-7050TX-72Q</a>	Arista	DCS-7050TX-72Q	no	EOS	4\.(18 20 21 22 23 24 25)\..*	T2	  
<a href="#">Arista DCS-7050TX-64</a>	Arista	DCS-7050TX-64	no	EOS	4\.(18 20 21 22 23 24 25)\..*	T2	  
<a href="#">Arista DCS-7050TX-48</a>	Arista	DCS-7050TX-48	no	EOS	4\.(18 20 21 22 23 24 25)\..*	T2	  
<a href="#">Arista DCS-7050TX3-48C8</a>	Arista	DCS-7050TX3-48C8	no	EOS	4\.(21 22 23 24 25)\..*	T3	  

2. Click **Ports** and select a port in the panel to see its details.


### Edit Device Profile




 Device profiles need to accurately model various characteristics of a switch model. Make sure you update the profile to match the new switch model(s) you intend to use this profile for.

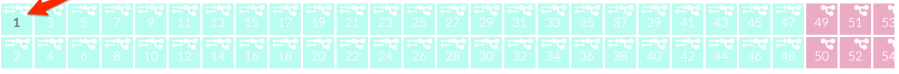
**Summary**

**Selector ⓘ**

**Capabilities**

**Ports** 

**Panel #1** *Select port(s) to fill in the details*   












**Edit 1 selected port: #1**

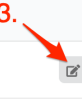
**Connector Type \***

10GBaseT

**Transformations**

Default	Speed	Interfaces	
	10 Gbps ▾	<a href="#">Ethernet1</a> +	 
	10 Gbps ▾	<a href="#">Ethernet1</a> +	 
	1 Gbps ▾	<a href="#">Ethernet1</a> +	 

[Add new transformation](#)



3. In the **Transformations** section, click the **Edit** button for the port to be edited.

### Edit Device Profile

4. Populate the port setting field with the string below. An empty string in the *speed* field signifies that the port is set to auto-negotiate. The maximum speed for the port is modeled to the speed in the fabric.

```
{"interface": {"speed": ""}, "global": {"select": "", "port_group": -1}}
```

You can edit the fields, such as "command", inside the port settings instead. Command contents are validated before the update is applied. Enabling auto-negotiation on an Arista device results in an automatically updated port setting in Apstra, because the schema for enabling auto-negotiation on an Arista device is well-defined, as shown below.

```
{ "interface":
  { "speed": "<String> example "},
  "global": "<Dict> example" {
    "select": <String> example "",
    "port_group": <Integer> example -1}
  "command": <to turn on auto neg on the port>
}
```

5. Click **Update** to save your changes and return to the list view.

## Delete Device Profile

If a device profile is used in an ["interface map" on page 91](#), you can't delete it.

1. Either from the list view (Devices > Device Profiles) or the details view, click the **Delete** button for the device profile to delete.
2. Click **Delete** to delete the device profile and return to the list view.

You can also use REST API to manage device profiles. Navigate to **Platform > Developers** for **REST API Documentation** and tools.

## Juniper Device Profiles

### IN THIS SECTION

- [Overview | 312](#)
- [Juniper QFX10002 | 312](#)

### Overview

Predefined device profiles for most qualified Juniper devices ship with Apstra software. For a complete list of qualified and recommended Juniper device series and Junos versions, see ["Qualified Device and NOS" on page 894](#). Juniper device profile constraints are specified below.

### Juniper QFX10002

The 36-port Juniper QFX10002-36Q and 72-port QFX10002-72Q are qualified by Apstra. Both of these models have a port constraint where only certain ports can be used with QSFP28 100G

transceivers.

☆ > Devices > Device Profiles > Juniper\_QFX10002-36Q

2RU
ASIC Q5(3)

Ports

Panel #1

INTERFACES CAPACITY

36 x 40 Gbps 144 x 10 Gbps 12 x 100 Gbps

PORTS Click on port to toggle the details Port breakout Autonegotiation

PORT DETAILS

ID	2
Connector type	qsfp28

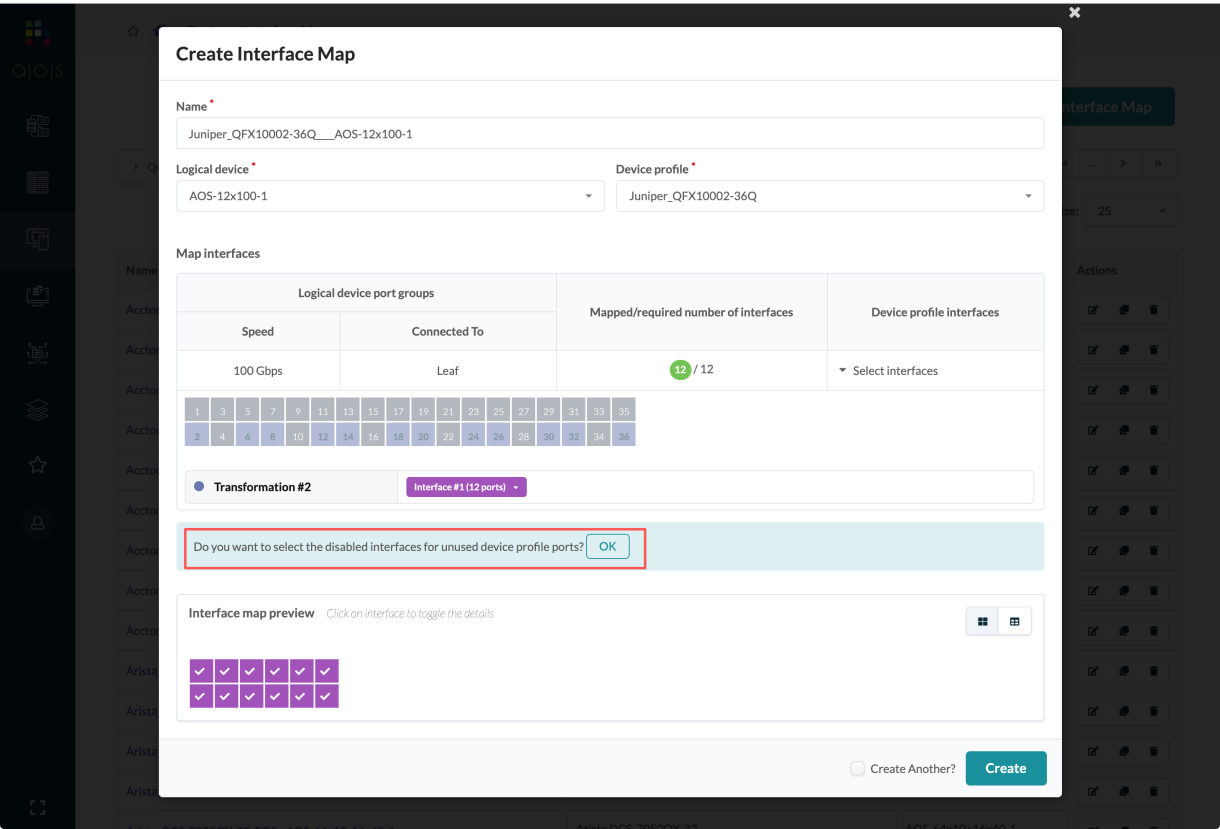
Transformations

Port #2 Tr. #1 (40 Gbps, default)	et-0/0/1
Port #2 Tr. #2 (100 Gbps)	et-0/0/1
Port #2 Tr. #3 (10 Gbps)	xe-0/0/1.0 xe-0/0/1.1 xe-0/0/1.2 xe-0/0/1.3

If these ports are used as 100G, then the adjacent QSFP 40G ports cannot be used. The device profile cannot automatically disable the adjacent QSFP 40G ports. You must create an Apstra interface map with these ports unused and disabled.

For the QFX10002 when creating the Apstra Interface Map, after selecting the 100G ports, Apstra displays an option to the user. "Do you want to select the disabled interfaces for unused device profile ports?". For 100G ports on the QFX10002, click the **OK** option so the unused QSFP ports are disabled

and cannot be used.



SONiC Device Profile

IN THIS SECTION

Background | 315

Problem Statement | 315

Solution | 315

User Interface | 315

Selector information | 316

Capabilities | 316

Interface naming conventions | 317

Troubleshooting | 317

Example: DP and port\_config.ini | 318

## Background

Device profiles are how devices are recognized and used inside Apstra. They capture device-specific semantics, which are required for being discovered by Apstra and to run network configs that work well for the datapath once inside the blueprint.

Device profiles are REST entities, which enable you to create, edit, delete, and list during the design phase. Device profiles are used to create interface maps, which get directly used inside the Apstra config rendering engine when blueprints are deployed.

This document covers the knowledge required to create (and edit) a semantically correct Sonic DP, so that not only does it pass the validations in place in Apstra which ensure the right DP is created in the database, but also honors the vendor semantic requirement applicable to the device so that it does not result in deploy failure when the generated configuration is pushed to the network device.

## Problem Statement

To create a device profile, you need the device specification (usually released by the vendor). You also need to know how to translate these specifications into Apstra DP data model so as to create the valid and config-friendly JSON. This is because the DP is a vendor semantics aware data structure.

## Solution

The high level data model is the same for all DPs, i.e. there are the same keys for every vendor's DP, just the way we get the values might differ, or might be loaded with a vendor constraint. The document enlists the following:

- The schema of the DP and the nested elements inside the DP.
- The meaning of each key value pair in the schema.
- The vendor specific recipe the values are populated.
- List any constraints, corner cases which need to be considered, especially for port configurations for certain (group of) models.
- Any lessons learnt along the way creating those DPs already in production useful in creating future ones.

## User Interface

When you create device profiles from the Apstra GUI, some of your entries are semantically validated. It's not completely capable of ensuring deep vendor-specific constraints and requirements though. With the exact vendor specification, the GUI assists you with creating a semantically valid DP which becomes part of the Apstra database data model.

Alternatively, you can write your own Python code that contains the vendor specifications, normalize it as per Apstra DP data model and generate the json to then import with the GUI.

### Selector information

Entering the correct information in all four of the selector fields is critical for the device to get matched to the device profile.

Selector Field	Value	Command to get the information on device
model	0x21	show platform syseeprom
manufacturer	If 0x2D in syseeprom, 0x2D else 0x2B	show platform syseeprom
OS family	SONiC	Show version
version	.*	Show version

### Capabilities

If you have the device specification, you can obtain its hardware and software capabilities for entry into the device profile.

The table below contains commonly found values in SONiC devices (based on qualified devices).

Selector Field	Value	Command to get the information on device
userland	64 (int)	Does not affect config
form_factor	'1RU' (string)	Does not affect config
ecmp_limit	64 (int)	Does not affect config
asic	'T2' (string)	Does not affect config
cpu	'x86' (string)	Does not affect config
ram	16 (int) (Note, the unit is in GB)	Does not affect config
onie	True (bool) (default)	Does not affect config

*(Continued)*

Selector Field	Value	Command to get the information on device
lxc	True (bool) (default)	Does not affect config

## Interface naming conventions

Sonic follows the naming conventions per the sonic port name file as found Azure SONiC on the github master. <https://github.com/Azure/SONiC/blob/master/doc/sonic-port-name.md>

To create a SONiC device profile, you must read through the device specific port\_config.ini (for example, sonic-buildimage/device/mellanox/x86\_64-mlnx\_msn2100-r0/ACS-MSN2100/port\_config.ini) file and follow the instructions in the above link to come up with the right interface names.

The interface names are explicitly mentioned in the port\_config.ini and whatever is given there is what will be used by SONiC and Apstra will need to have DP with matching interface names which will be used to generate the PORT configs in the configuration file (config\_db.json) . For this document purposes, port\_config.ini and config\_db.json should have the same interface naming standard. The user should use those interface names in their DP along with the lane numbers provided in the port\_cfg.ini file. Once a device profile has been generated based on the aforementioned steps, Apstra will use that along with the LD to generate the Interface Map (IM). Apstra as part of its validation will make sure that the IM (which describes the port and its speeds) are indeed available and supported under “/usr/share/sonic/device/ x86\_64-mlnx\_msn2100-r0/ACS-MSN2100/port\_config.ini” . This validation is performed to make sure SONiC NOS stack does not fail due to unsupported port configuration (in config\_db.json) getting wrongly generated in Apstra due to wrong DP. So it is important that the end user makes sure the DP that is generated for a SONiC platform has the correct interface names and lane maps as reflected in port\_config.ini file for that particular platform. A platform may have a few different port\_config.ini files part of different HWSKUs for that platform. Apstra will try to validate the generated port configs with any of the available options for that platform. Apstra currently does not use the Dynamic Port breakout feature which is on-going in the SONiC project.

## Troubleshooting

Device mismatch usually occurs at the beginning of a device's lifecycle. If a device profile is not being selected by the device, check the four selector fields in the device profile.

Deploy errors could arise because of incorrect port configurations. This could be either the ports were configured with incorrect speeds, or the OS specific port constraints were not handled in the device profile or in the interface map.

A possible flow for root cause would be:

- Check the DP for obvious port capabilities errors. Is the port really capable of the speeds the DP has configured. The device specific port\_config.ini Sonic open source project is a good resource to parse for ERROR messages.
- Check if the DP has configured autoneg or disabled interfaces correctly. Autoneg and disabled can both be expressed in the interface setting field.
- When debugging the interface names and lane mapping, please take a look at the corresponding port\_config.ini. As an example for AS5712-54X edgecore/accton box we can get the port\_config.ini file that has the details like lane/name/alias at [https://github.com/Azure/sonic-buildimage/tree/master/device/accton/x86\\_64-accton\\_as5712\\_54x-r0/Accton-AS5712-54X](https://github.com/Azure/sonic-buildimage/tree/master/device/accton/x86_64-accton_as5712_54x-r0/Accton-AS5712-54X)
- The naming constraints can be read from the SoNIC official documentation. For example if the user wants to generate the interface names for Accton 5712 54X running SONIC, the port\_config.ini is the authority. [https://github.com/Azure/sonic-buildimage/blob/master/device/accton/x86\\_64-accton\\_as5712\\_54x-r0/Accton-AS5712-54X/port\\_config.ini](https://github.com/Azure/sonic-buildimage/blob/master/device/accton/x86_64-accton_as5712_54x-r0/Accton-AS5712-54X/port_config.ini) Sometimes the device might have inter-port constraints. For sonic, it is kind of all laid out in the port\_config.ini file. There may be multiple port\_config.ini files for a specific platform and a specific manufacturer with each port\_config.ini file residing in their own HWSKU folders in the sonic image (like the one I had pointed above). The ability to try out different port speeds on (outside of what is listed in the port\_config.ini) will need knowledge of the chipset and also the physical switch manufacturer to see what can be achieved. This information may not be available in any white papers unless we ask the vendors.

#### Example: DP and port\_config.ini

Port\_config.ini from sonic-buildimage is below for Dell\_Z9100 (x86\_64-dell\_z9100\_c2538-r0/Force10-Z9100-C32

#	name	lanes	alias	index
	Ethernet0	49,50,51,52	hundredGigE1/1	1
	Ethernet4	53,54,55,56	hundredGigE1/2	2
	Ethernet8	57,58,59,60	hundredGigE1/3	3
	Ethernet12	61,62,63,64	hundredGigE1/4	4
	Ethernet16	65,66,67,68	hundredGigE1/5	5
	Ethernet20	69,70,71,72	hundredGigE1/6	6
	Ethernet24	73,74,75,76	hundredGigE1/7	7
	Ethernet28	77,78,79,80	hundredGigE1/8	8
	Ethernet32	37,38,39,40	hundredGigE1/9	9
	Ethernet36	33,34,35,36	hundredGigE1/10	10
	Ethernet40	45,46,47,48	hundredGigE1/11	11
	Ethernet44	41,42,43,44	hundredGigE1/12	12
	Ethernet48	81,82,83,84	hundredGigE1/13	13
	Ethernet52	85,86,87,88	hundredGigE1/14	14

Ethernet56	89,90,91,92	hundredGigE1/15	15
Ethernet60	93,94,95,96	hundredGigE1/16	16
Ethernet64	97,98,99,100	hundredGigE1/17	17
Ethernet68	101,102,103,104	hundredGigE1/18	18
Ethernet72	105,106,107,108	hundredGigE1/19	19
Ethernet76	109,110,111,112	hundredGigE1/20	20
Ethernet80	21,22,23,24	hundredGigE1/21	21
Ethernet84	17,18,19,20	hundredGigE1/22	22
Ethernet88	29,30,31,32	hundredGigE1/23	23
Ethernet92	25,26,27,28	hundredGigE1/24	24
Ethernet96	117,118,119,120	hundredGigE1/25	25
Ethernet100	113,114,115,116	hundredGigE1/26	26
Ethernet104	125,126,127,128	hundredGigE1/27	27
Ethernet108	121,122,123,124	hundredGigE1/28	28
Ethernet112	5,6,7,8	hundredGigE1/29	29
Ethernet116	1,2,3,4	hundredGigE1/30	30
Ethernet120	13,14,15,16	hundredGigE1/31	31
Ethernet124	9,10,11,12	hundredGigE1/32	32

Translate port\_config to a port-to-lane\_map data structure using parse.py script:

```

Parse.py
=====
#!/usr/bin/python
# Copyright (c) 2017 Apstrktr, Inc. All rights reserved.
# Apstrktr, Inc. Confidential and Proprietary.
#
# This source code is licensed under End User License Agreement found in the
# LICENSE file at http://apstra.com/eula

# pylint: disable=line-too-long

import sys
from pprint import pprint

# Run the program as ./parse.py <path_to_sonic_platform_port_config.ini>
# ex: ./parse.py sonic-buildimage/device/mellanox/x86_64-mlnx_msn2100-r0/ACS-MSN2100/
port_config.ini
def get_lanemap(buf):
    if not buf:
        return None
    d = {}

```

```

interface_indices = []
for line in buf.split('\n'):
    if line.startswith('#'):
        continue
    words = line.split(' ')
    words = [word for word in words if len(word)]
    if not len(words):
        continue
    intf = words[0][8:]
    lane = words[1].split(',')
    interface_indices.append(intf)
    if len(lane) > 1:
        one = 'Ethernet' + str(intf)
        two = 'Ethernet' + str(int(intf)+1)
        three = 'Ethernet' + str(int(intf)+2)
        four = 'Ethernet' + str(int(intf)+3)
        d.update({one:lane[0]})
        d.update({two:lane[1]})
        d.update({three:lane[2]})
        d.update({four:lane[3]})
    else:
        d.update({words[0]:words[1]})
return {'interface_names' : interface_indices, 'lane_mapping' : d}

def parse_portconfig(f):
    buf = ''
    with open(f, 'r') as stream:
        buf = stream.read()
    return {'<Platform>': get_lanemap(buf)}

if __name__ == '__main__':
    assert len(sys.argv) > 1, "Missing port_config.ini in cmdline"
    print "Collecting lane information from ", sys.argv[1]
    pprint(parse_portconfig(sys.argv[1]))
    print
    "=====
    print "          Substitute <Platform> with an identifier for the platform"
    print "          Append the dump into sdk/device-profile/sonic.py's sonic_device_info dictionary"
    print
    "=====

```

To run parse.py

```
parse.py <Path to the port_config.ini file from sonic_buildimage>
```

Example:

```
parse.py sonic-buildimage/device/dell/x86_64-dell_z9100_c2538-r0/Force10-Z9100-C32/
port_config.ini
```

Collecting lane information from sonic-buildimage/device/dell/x86\_64-dell\_z9100\_c2538-r0/  
Force10-Z9100-C32/port\_config.ini

```
{'<Platform>': {'interface_names': ['0',
                                     '4',
                                     '8',
                                     '12',
                                     '16',
                                     '20',
                                     '24',
                                     '28',
                                     '32',
                                     '36',
                                     '40',
                                     '44',
                                     '48',
                                     '52',
                                     '56',
                                     '60',
                                     '64',
                                     '68',
                                     '72',
                                     '76',
                                     '80',
                                     '84',
                                     '88',
                                     '92',
                                     '96',
                                     '100',
                                     '104',
                                     '108',
                                     '112',
                                     '116',
                                     '120',
                                     '124'],
```

```
'lane_mapping': {'Ethernet0': '49',  
                  'Ethernet1': '50',  
                  'Ethernet10': '59',  
                  'Ethernet100': '113',  
                  'Ethernet101': '114',  
                  'Ethernet102': '115',  
                  'Ethernet103': '116',  
                  'Ethernet104': '125',  
                  'Ethernet105': '126',  
                  'Ethernet106': '127',  
                  'Ethernet107': '128',  
                  'Ethernet108': '121',  
                  'Ethernet109': '122',  
                  'Ethernet11': '60',  
                  'Ethernet110': '123',  
                  'Ethernet111': '124',  
                  'Ethernet112': '5',  
                  'Ethernet113': '6',  
                  'Ethernet114': '7',  
                  'Ethernet115': '8',  
                  'Ethernet116': '1',  
                  'Ethernet117': '2',  
                  'Ethernet118': '3',  
                  'Ethernet119': '4',  
                  'Ethernet12': '61',  
                  'Ethernet120': '13',  
                  'Ethernet121': '14',  
                  'Ethernet122': '15',  
                  'Ethernet123': '16',  
                  'Ethernet124': '9',  
                  'Ethernet125': '10',  
                  'Ethernet126': '11',  
                  'Ethernet127': '12',  
                  'Ethernet13': '62',  
                  'Ethernet14': '63',  
                  'Ethernet15': '64',  
                  'Ethernet16': '65',  
                  'Ethernet17': '66',  
                  'Ethernet18': '67',  
                  'Ethernet19': '68',  
                  'Ethernet2': '51',  
                  'Ethernet20': '69',  
                  'Ethernet21': '70',
```

```
'Ethernet22': '71',  
'Ethernet23': '72',  
'Ethernet24': '73',  
'Ethernet25': '74',  
'Ethernet26': '75',  
'Ethernet27': '76',  
'Ethernet28': '77',  
'Ethernet29': '78',  
'Ethernet3': '52',  
'Ethernet30': '79',  
'Ethernet31': '80',  
'Ethernet32': '37',  
'Ethernet33': '38',  
'Ethernet34': '39',  
'Ethernet35': '40',  
'Ethernet36': '33',  
'Ethernet37': '34',  
'Ethernet38': '35',  
'Ethernet39': '36',  
'Ethernet4': '53',  
'Ethernet40': '45',  
'Ethernet41': '46',  
'Ethernet42': '47',  
'Ethernet43': '48',  
'Ethernet44': '41',  
'Ethernet45': '42',  
'Ethernet46': '43',  
'Ethernet47': '44',  
'Ethernet48': '81',  
'Ethernet49': '82',  
'Ethernet5': '54',  
'Ethernet50': '83',  
'Ethernet51': '84',  
'Ethernet52': '85',  
'Ethernet53': '86',  
'Ethernet54': '87',  
'Ethernet55': '88',  
'Ethernet56': '89',  
'Ethernet57': '90',  
'Ethernet58': '91',  
'Ethernet59': '92',  
'Ethernet6': '55',  
'Ethernet60': '93',
```

```

'Ethernet61': '94',
'Ethernet62': '95',
'Ethernet63': '96',
'Ethernet64': '97',
'Ethernet65': '98',
'Ethernet66': '99',
'Ethernet67': '100',
'Ethernet68': '101',
'Ethernet69': '102',
'Ethernet7': '56',
'Ethernet70': '103',
'Ethernet71': '104',
'Ethernet72': '105',
'Ethernet73': '106',
'Ethernet74': '107',
'Ethernet75': '108',
'Ethernet76': '109',
'Ethernet77': '110',
'Ethernet78': '111',
'Ethernet79': '112',
'Ethernet8': '57',
'Ethernet80': '21',
'Ethernet81': '22',
'Ethernet82': '23',
'Ethernet83': '24',
'Ethernet84': '17',
'Ethernet85': '18',
'Ethernet86': '19',
'Ethernet87': '20',
'Ethernet88': '29',
'Ethernet89': '30',
'Ethernet9': '58',
'Ethernet90': '31',
'Ethernet91': '32',
'Ethernet92': '25',
'Ethernet93': '26',
'Ethernet94': '27',
'Ethernet95': '28',
'Ethernet96': '117',
'Ethernet97': '118',
'Ethernet98': '119',
'Ethernet99': '120'}}}

```

```
=====
```

```

Substitute <Platform> with an identifier for the platform
Append the dump into sdk/device-profile/sonic.py's sonic_device_info dictionary
=====

```

The output from above will become a dictionary entry in `sonic_device_info` in the `sonic device_profile` generator python file.

Corresponding Device Profile generated in Apstra:

```

{
  "hardware_capabilities": {
    "asic": "TH",
    "cpu": "x86",
    "ecmp_limit": 64,
    "form_factor": "1RU",
    "ram": 16,
    "userland": 64
  },
  "id": "Force10-Z9100_SONiC",
  "label": "Dell Force10-Z9100_SONiC",
  "ports": [
    {
      "column_id": 1,
      "connector_type": "qsfp28",
      "failure_domain_id": 1,
      "panel_id": 1,
      "port_id": 0,
      "row_id": 1,
      "slot_id": 0,
      "transformations": [
        {
          "interfaces": [
            {
              "interface_id": 1,
              "name": "Ethernet0",
              "setting": "{\"interface\": {\"speed\": \"100000\", \"lane_map\": \"49,50,51,52\"}}",
              "speed": {
                "unit": "G",
                "value": 100
              },
            }
          ]
        }
      ]
    }
  ]
}

```

```

        "state": "active"
    }
],
    "is_default": true,
    "transformation_id": 1
},
{
    "interfaces": [
        {
            "interface_id": 1,
            "name": "Ethernet0",
            "setting": "{\"interface\": {\"speed\": \"40000\", \"lane_map\": \"49,50,51,52\"}}",
            "speed": {
                "unit": "G",
                "value": 40
            },
            "state": "active"
        }
    ],
    "is_default": false,
    "transformation_id": 2
}
]
},
{
    "column_id": 1,
    "connector_type": "qsfp28",
    "failure_domain_id": 1,
    "panel_id": 1,
    "port_id": 1,
    "row_id": 2,
    "slot_id": 0,
    "transformations": [
        {
            "interfaces": [
                {
                    "interface_id": 1,
                    "name": "Ethernet4",
                    "setting": "{\"interface\": {\"speed\": \"100000\", \"lane_map\": \"53,54,55,56\"}}",
                    "speed": {
                        "unit": "G",

```

```

        "value": 100
      },
      "state": "active"
    }
  ],
  "is_default": true,
  "transformation_id": 1
},
{
  "interfaces": [
    {
      "interface_id": 1,
      "name": "Ethernet4",
      "setting": "{\"interface\": {\"speed\": \"40000\", \"lane_map\": \"53,54,55,56\"}}",
      "speed": {
        "unit": "G",
        "value": 40
      },
      "state": "active"
    }
  ],
  "is_default": false,
  "transformation_id": 2
}
],
{
  "column_id": 2,
  "connector_type": "qsfp28",
  "failure_domain_id": 1,
  "panel_id": 1,
  "port_id": 2,
  "row_id": 1,
  "slot_id": 0,
  "transformations": [
    {
      "interfaces": [
        {
          "interface_id": 1,
          "name": "Ethernet8",
          "setting": "{\"interface\": {\"speed\": \"100000\", \"lane_map\": \"57,58,59,60\"}}",

```

```

        "speed": {
            "unit": "G",
            "value": 100
        },
        "state": "active"
    }
],
"is_default": true,
"transformation_id": 1
},
{
    "interfaces": [
        {
            "interface_id": 1,
            "name": "Ethernet8",
            "setting": "{\"interface\": {\"speed\": \"40000\", \"lane_map\": \"57,58,59,60\"}}",
            "speed": {
                "unit": "G",
                "value": 40
            },
            "state": "active"
        }
    ],
    "is_default": false,
    "transformation_id": 2
}
],
{
    "column_id": 2,
    "connector_type": "qsfp28",
    "failure_domain_id": 1,
    "panel_id": 1,
    "port_id": 3,
    "row_id": 2,
    "slot_id": 0,
    "transformations": [
        {
            "interfaces": [
                {
                    "interface_id": 1,
                    "name": "Ethernet12",

```

```

        "setting": "{\"interface\": {\"speed\": \"100000\", \"lane_map\":
\\\"61,62,63,64\\\"}}\",
        \"speed\": {
            \"unit\": \"G\",
            \"value\": 100
        },
        \"state\": \"active\"
    },
    \"is_default\": true,
    \"transformation_id\": 1
},
{
    \"interfaces\": [
        {
            \"interface_id\": 1,
            \"name\": \"Ethernet12\",
            \"setting\": \"{\\\"interface\\\": {\\\"speed\\\": \\\"40000\\\", \\\"lane_map\\\":
\\\"61,62,63,64\\\"}}\",
            \"speed\": {
                \"unit\": \"G\",
                \"value\": 40
            },
            \"state\": \"active\"
        }
    ],
    \"is_default\": false,
    \"transformation_id\": 2
}
],
{
    \"column_id\": 3,
    \"connector_type\": \"qsfp28\",
    \"failure_domain_id\": 1,
    \"panel_id\": 1,
    \"port_id\": 4,
    \"row_id\": 1,
    \"slot_id\": 0,
    \"transformations\": [
        {
            \"interfaces\": [
                {

```

```

        "interface_id": 1,
        "name": "Ethernet16",
        "setting": "{\"interface\": {\"speed\": \"100000\", \"lane_map\":
\\\"65,66,67,68\\\"}}\",
        \"speed\": {
            \"unit\": \"G\",
            \"value\": 100
        },
        \"state\": \"active\"
    }
],
    \"is_default\": true,
    \"transformation_id\": 1
},
{
    \"interfaces\": [
        {
            \"interface_id\": 1,
            \"name\": \"Ethernet16\",
            \"setting\": \"{\\\"interface\\\": {\\\"speed\\\": \\\"40000\\\", \\\"lane_map\\\":
\\\"65,66,67,68\\\"}}\",
            \"speed\": {
                \"unit\": \"G\",
                \"value\": 40
            },
            \"state\": \"active\"
        }
    ],
    \"is_default\": false,
    \"transformation_id\": 2
}
]
},
{
    \"column_id\": 3,
    \"connector_type\": \"qsfp28\",
    \"failure_domain_id\": 1,
    \"panel_id\": 1,
    \"port_id\": 5,
    \"row_id\": 2,
    \"slot_id\": 0,
    \"transformations\": [
        {

```

```

        "interfaces": [
            {
                "interface_id": 1,
                "name": "Ethernet20",
                "setting": "{\"interface\": {\"speed\": \"100000\", \"lane_map\": \"69,70,71,72\"}}",
                "speed": {
                    "unit": "G",
                    "value": 100
                },
                "state": "active"
            }
        ],
        "is_default": true,
        "transformation_id": 1
    },
    {
        "interfaces": [
            {
                "interface_id": 1,
                "name": "Ethernet20",
                "setting": "{\"interface\": {\"speed\": \"40000\", \"lane_map\": \"69,70,71,72\"}}",
                "speed": {
                    "unit": "G",
                    "value": 40
                },
                "state": "active"
            }
        ],
        "is_default": false,
        "transformation_id": 2
    }
]
},
{
    "column_id": 4,
    "connector_type": "qsfp28",
    "failure_domain_id": 1,
    "panel_id": 1,
    "port_id": 6,
    "row_id": 1,
    "slot_id": 0,

```

```

    "transformations": [
      {
        "interfaces": [
          {
            "interface_id": 1,
            "name": "Ethernet24",
            "setting": "{\"interface\": {\"speed\": \"100000\", \"lane_map\": \"73,74,75,76\"}}",
            "speed": {
              "unit": "G",
              "value": 100
            },
            "state": "active"
          }
        ],
        "is_default": true,
        "transformation_id": 1
      },
      {
        "interfaces": [
          {
            "interface_id": 1,
            "name": "Ethernet24",
            "setting": "{\"interface\": {\"speed\": \"40000\", \"lane_map\": \"73,74,75,76\"}}",
            "speed": {
              "unit": "G",
              "value": 40
            },
            "state": "active"
          }
        ],
        "is_default": false,
        "transformation_id": 2
      }
    ],
    {
      "column_id": 4,
      "connector_type": "qsfp28",
      "failure_domain_id": 1,
      "panel_id": 1,
      "port_id": 7,

```

```

    "row_id": 2,
    "slot_id": 0,
    "transformations": [
      {
        "interfaces": [
          {
            "interface_id": 1,
            "name": "Ethernet28",
            "setting": "{\"interface\": {\"speed\": \"100000\", \"lane_map\": \"77,78,79,80\"}}",
            "speed": {
              "unit": "G",
              "value": 100
            },
            "state": "active"
          }
        ],
        "is_default": true,
        "transformation_id": 1
      },
      {
        "interfaces": [
          {
            "interface_id": 1,
            "name": "Ethernet28",
            "setting": "{\"interface\": {\"speed\": \"40000\", \"lane_map\": \"77,78,79,80\"}}",
            "speed": {
              "unit": "G",
              "value": 40
            },
            "state": "active"
          }
        ],
        "is_default": false,
        "transformation_id": 2
      }
    ]
  },
  {
    "column_id": 5,
    "connector_type": "qsfp28",
    "failure_domain_id": 1,

```

```

    "panel_id": 1,
    "port_id": 8,
    "row_id": 1,
    "slot_id": 0,
    "transformations": [
      {
        "interfaces": [
          {
            "interface_id": 1,
            "name": "Ethernet32",
            "setting": "{\"interface\": {\"speed\": \"100000\", \"lane_map\": \"37,38,39,40\"}}",
            "speed": {
              "unit": "G",
              "value": 100
            },
            "state": "active"
          }
        ],
        "is_default": true,
        "transformation_id": 1
      },
      {
        "interfaces": [
          {
            "interface_id": 1,
            "name": "Ethernet32",
            "setting": "{\"interface\": {\"speed\": \"40000\", \"lane_map\": \"37,38,39,40\"}}",
            "speed": {
              "unit": "G",
              "value": 40
            },
            "state": "active"
          }
        ],
        "is_default": false,
        "transformation_id": 2
      }
    ]
  },
  {
    "column_id": 5,

```

```

    "connector_type": "qsfp28",
    "failure_domain_id": 1,
    "panel_id": 1,
    "port_id": 9,
    "row_id": 2,
    "slot_id": 0,
    "transformations": [
      {
        "interfaces": [
          {
            "interface_id": 1,
            "name": "Ethernet36",
            "setting": "{\"interface\": {\"speed\": \"100000\", \"lane_map\":
\\\"33,34,35,36\\\"}}",
            "speed": {
              "unit": "G",
              "value": 100
            },
            "state": "active"
          }
        ],
        "is_default": true,
        "transformation_id": 1
      },
      {
        "interfaces": [
          {
            "interface_id": 1,
            "name": "Ethernet36",
            "setting": "{\"interface\": {\"speed\": \"40000\", \"lane_map\":
\\\"33,34,35,36\\\"}}",
            "speed": {
              "unit": "G",
              "value": 40
            },
            "state": "active"
          }
        ],
        "is_default": false,
        "transformation_id": 2
      }
    ]
  },

```

```

{
  "column_id": 6,
  "connector_type": "qsfp28",
  "failure_domain_id": 1,
  "panel_id": 1,
  "port_id": 10,
  "row_id": 1,
  "slot_id": 0,
  "transformations": [
    {
      "interfaces": [
        {
          "interface_id": 1,
          "name": "Ethernet40",
          "setting": "{\"interface\": {\"speed\": \"100000\", \"lane_map\": \"45,46,47,48\"}}",
          "speed": {
            "unit": "G",
            "value": 100
          },
          "state": "active"
        }
      ],
      "is_default": true,
      "transformation_id": 1
    },
    {
      "interfaces": [
        {
          "interface_id": 1,
          "name": "Ethernet40",
          "setting": "{\"interface\": {\"speed\": \"40000\", \"lane_map\": \"45,46,47,48\"}}",
          "speed": {
            "unit": "G",
            "value": 40
          },
          "state": "active"
        }
      ],
      "is_default": false,
      "transformation_id": 2
    }
  ]
}

```

```

    ]
  },
  {
    "column_id": 6,
    "connector_type": "qsfp28",
    "failure_domain_id": 1,
    "panel_id": 1,
    "port_id": 11,
    "row_id": 2,
    "slot_id": 0,
    "transformations": [
      {
        "interfaces": [
          {
            "interface_id": 1,
            "name": "Ethernet44",
            "setting": "{\"interface\": {\"speed\": \"100000\", \"lane_map\": \"41,42,43,44\"}}",
            "speed": {
              "unit": "G",
              "value": 100
            },
            "state": "active"
          }
        ],
        "is_default": true,
        "transformation_id": 1
      },
      {
        "interfaces": [
          {
            "interface_id": 1,
            "name": "Ethernet44",
            "setting": "{\"interface\": {\"speed\": \"40000\", \"lane_map\": \"41,42,43,44\"}}",
            "speed": {
              "unit": "G",
              "value": 40
            },
            "state": "active"
          }
        ],
        "is_default": false,

```

```

        "transformation_id": 2
    }
]
},
{
    "column_id": 7,
    "connector_type": "qsfp28",
    "failure_domain_id": 1,
    "panel_id": 1,
    "port_id": 12,
    "row_id": 1,
    "slot_id": 0,
    "transformations": [
        {
            "interfaces": [
                {
                    "interface_id": 1,
                    "name": "Ethernet48",
                    "setting": "{\"interface\": {\"speed\": \"100000\", \"lane_map\": \"81,82,83,84\"}}",
                    "speed": {
                        "unit": "G",
                        "value": 100
                    },
                    "state": "active"
                }
            ],
            "is_default": true,
            "transformation_id": 1
        },
        {
            "interfaces": [
                {
                    "interface_id": 1,
                    "name": "Ethernet48",
                    "setting": "{\"interface\": {\"speed\": \"40000\", \"lane_map\": \"81,82,83,84\"}}",
                    "speed": {
                        "unit": "G",
                        "value": 40
                    },
                    "state": "active"
                }
            ]
        }
    ]
}

```

```

    ],
    "is_default": false,
    "transformation_id": 2
  }
]
},
{
  "column_id": 7,
  "connector_type": "qsfp28",
  "failure_domain_id": 1,
  "panel_id": 1,
  "port_id": 13,
  "row_id": 2,
  "slot_id": 0,
  "transformations": [
    {
      "interfaces": [
        {
          "interface_id": 1,
          "name": "Ethernet52",
          "setting": "{\"interface\": {\"speed\": \"100000\", \"lane_map\": \"85,86,87,88\"}}",
          "speed": {
            "unit": "G",
            "value": 100
          },
          "state": "active"
        }
      ],
      "is_default": true,
      "transformation_id": 1
    },
    {
      "interfaces": [
        {
          "interface_id": 1,
          "name": "Ethernet52",
          "setting": "{\"interface\": {\"speed\": \"40000\", \"lane_map\": \"85,86,87,88\"}}",
          "speed": {
            "unit": "G",
            "value": 40
          },

```

```

        "state": "active"
    }
],
    "is_default": false,
    "transformation_id": 2
}
]
},
{
    "column_id": 8,
    "connector_type": "qsfp28",
    "failure_domain_id": 1,
    "panel_id": 1,
    "port_id": 14,
    "row_id": 1,
    "slot_id": 0,
    "transformations": [
        {
            "interfaces": [
                {
                    "interface_id": 1,
                    "name": "Ethernet56",
                    "setting": "{\"interface\": {\"speed\": \"100000\", \"lane_map\": \"89,90,91,92\"}}",
                    "speed": {
                        "unit": "G",
                        "value": 100
                    },
                    "state": "active"
                }
            ],
            "is_default": true,
            "transformation_id": 1
        },
        {
            "interfaces": [
                {
                    "interface_id": 1,
                    "name": "Ethernet56",
                    "setting": "{\"interface\": {\"speed\": \"40000\", \"lane_map\": \"89,90,91,92\"}}",
                    "speed": {
                        "unit": "G",

```

```

        "value": 40
      },
      "state": "active"
    }
  ],
  "is_default": false,
  "transformation_id": 2
}
]
},
{
  "column_id": 8,
  "connector_type": "qsfp28",
  "failure_domain_id": 1,
  "panel_id": 1,
  "port_id": 15,
  "row_id": 2,
  "slot_id": 0,
  "transformations": [
    {
      "interfaces": [
        {
          "interface_id": 1,
          "name": "Ethernet60",
          "setting": "{\"interface\": {\"speed\": \"100000\", \"lane_map\": \"93,94,95,96\"}}",
          "speed": {
            "unit": "G",
            "value": 100
          },
          "state": "active"
        }
      ],
      "is_default": true,
      "transformation_id": 1
    },
    {
      "interfaces": [
        {
          "interface_id": 1,
          "name": "Ethernet60",
          "setting": "{\"interface\": {\"speed\": \"40000\", \"lane_map\": \"93,94,95,96\"}}",

```

```

        "speed": {
            "unit": "G",
            "value": 40
        },
        "state": "active"
    }
],
"is_default": false,
"transformation_id": 2
}
]
},
{
    "column_id": 9,
    "connector_type": "qsfp28",
    "failure_domain_id": 1,
    "panel_id": 1,
    "port_id": 16,
    "row_id": 1,
    "slot_id": 0,
    "transformations": [
        {
            "interfaces": [
                {
                    "interface_id": 1,
                    "name": "Ethernet64",
                    "setting": "{\"interface\": {\"speed\": \"100000\", \"lane_map\": \"97,98,99,100\"}}",
                    "speed": {
                        "unit": "G",
                        "value": 100
                    },
                    "state": "active"
                }
            ],
            "is_default": true,
            "transformation_id": 1
        },
        {
            "interfaces": [
                {
                    "interface_id": 1,
                    "name": "Ethernet64",

```

```

        "setting": "{\"interface\": {\"speed\": \"40000\", \"lane_map\":
\\\"97,98,99,100\\\"}}\",
        \"speed\": {
            \"unit\": \"G\",
            \"value\": 40
        },
        \"state\": \"active\"
    }
],
    \"is_default\": false,
    \"transformation_id\": 2
}
]
},
{
    \"column_id\": 9,
    \"connector_type\": \"qsfp28\",
    \"failure_domain_id\": 1,
    \"panel_id\": 1,
    \"port_id\": 17,
    \"row_id\": 2,
    \"slot_id\": 0,
    \"transformations\": [
        {
            \"interfaces\": [
                {
                    \"interface_id\": 1,
                    \"name\": \"Ethernet68\",
                    \"setting\": \"{\\\"interface\\\": {\\\"speed\\\": \\\"100000\\\", \\\"lane_map\\\":
\\\"101,102,103,104\\\"}}\",
                    \"speed\": {
                        \"unit\": \"G\",
                        \"value\": 100
                    },
                    \"state\": \"active\"
                }
            ],
            \"is_default\": true,
            \"transformation_id\": 1
        },
        {
            \"interfaces\": [
                {

```

```

        "interface_id": 1,
        "name": "Ethernet68",
        "setting": "{\"interface\": {\"speed\": \"40000\", \"lane_map\":
\\\"101,102,103,104\\\"}}\",
        \"speed\": {
            \"unit\": \"G\",
            \"value\": 40
        },
        \"state\": \"active\"
    }
],
    \"is_default\": false,
    \"transformation_id\": 2
}
],
{
    \"column_id\": 10,
    \"connector_type\": \"qsfp28\",
    \"failure_domain_id\": 1,
    \"panel_id\": 1,
    \"port_id\": 18,
    \"row_id\": 1,
    \"slot_id\": 0,
    \"transformations\": [
        {
            \"interfaces\": [
                {
                    \"interface_id\": 1,
                    \"name\": \"Ethernet72\",
                    \"setting\": \"{\\\"interface\\\": {\\\"speed\\\": \\\"100000\\\", \\\"lane_map\\\":
\\\"105,106,107,108\\\"}}\",
                    \"speed\": {
                        \"unit\": \"G\",
                        \"value\": 100
                    },
                    \"state\": \"active\"
                }
            ],
            \"is_default\": true,
            \"transformation_id\": 1
        },
        {

```

```

        "interfaces": [
            {
                "interface_id": 1,
                "name": "Ethernet72",
                "setting": "{\"interface\": {\"speed\": \"40000\", \"lane_map\": \"105,106,107,108\"}}",
                "speed": {
                    "unit": "G",
                    "value": 40
                },
                "state": "active"
            }
        ],
        "is_default": false,
        "transformation_id": 2
    }
]
},
{
    "column_id": 10,
    "connector_type": "qsfp28",
    "failure_domain_id": 1,
    "panel_id": 1,
    "port_id": 19,
    "row_id": 2,
    "slot_id": 0,
    "transformations": [
        {
            "interfaces": [
                {
                    "interface_id": 1,
                    "name": "Ethernet76",
                    "setting": "{\"interface\": {\"speed\": \"100000\", \"lane_map\": \"109,110,111,112\"}}",
                    "speed": {
                        "unit": "G",
                        "value": 100
                    },
                    "state": "active"
                }
            ],
            "is_default": true,
            "transformation_id": 1
        }
    ]
}

```

```

    },
    {
      "interfaces": [
        {
          "interface_id": 1,
          "name": "Ethernet76",
          "setting": "{\"interface\": {\"speed\": \"40000\", \"lane_map\": \"109,110,111,112\"}}",
          "speed": {
            "unit": "G",
            "value": 40
          },
          "state": "active"
        }
      ],
      "is_default": false,
      "transformation_id": 2
    }
  ],
  {
    "column_id": 11,
    "connector_type": "qsfp28",
    "failure_domain_id": 1,
    "panel_id": 1,
    "port_id": 20,
    "row_id": 1,
    "slot_id": 0,
    "transformations": [
      {
        "interfaces": [
          {
            "interface_id": 1,
            "name": "Ethernet80",
            "setting": "{\"interface\": {\"speed\": \"100000\", \"lane_map\": \"21,22,23,24\"}}",
            "speed": {
              "unit": "G",
              "value": 100
            },
            "state": "active"
          }
        ]
      }
    ]
  }
],

```

```

        "is_default": true,
        "transformation_id": 1
    },
    {
        "interfaces": [
            {
                "interface_id": 1,
                "name": "Ethernet80",
                "setting": "{\"interface\": {\"speed\": \"40000\", \"lane_map\": \"21,22,23,24\"}}",
                "speed": {
                    "unit": "G",
                    "value": 40
                },
                "state": "active"
            }
        ],
        "is_default": false,
        "transformation_id": 2
    }
]
},
{
    "column_id": 11,
    "connector_type": "qsfp28",
    "failure_domain_id": 1,
    "panel_id": 1,
    "port_id": 21,
    "row_id": 2,
    "slot_id": 0,
    "transformations": [
        {
            "interfaces": [
                {
                    "interface_id": 1,
                    "name": "Ethernet84",
                    "setting": "{\"interface\": {\"speed\": \"100000\", \"lane_map\": \"17,18,19,20\"}}",
                    "speed": {
                        "unit": "G",
                        "value": 100
                    },
                    "state": "active"
                }
            ]
        }
    ]
}

```

```

    }
  ],
  "is_default": true,
  "transformation_id": 1
},
{
  "interfaces": [
    {
      "interface_id": 1,
      "name": "Ethernet84",
      "setting": "{\"interface\": {\"speed\": \"40000\", \"lane_map\": \"17,18,19,20\"}}",
      "speed": {
        "unit": "G",
        "value": 40
      },
      "state": "active"
    }
  ],
  "is_default": false,
  "transformation_id": 2
}
]
},
{
  "column_id": 12,
  "connector_type": "qsfp28",
  "failure_domain_id": 1,
  "panel_id": 1,
  "port_id": 22,
  "row_id": 1,
  "slot_id": 0,
  "transformations": [
    {
      "interfaces": [
        {
          "interface_id": 1,
          "name": "Ethernet88",
          "setting": "{\"interface\": {\"speed\": \"100000\", \"lane_map\": \"29,30,31,32\"}}",
          "speed": {
            "unit": "G",
            "value": 100
          }
        }
      ]
    }
  ]
}

```

```

        },
        "state": "active"
    }
],
"is_default": true,
"transformation_id": 1
},
{
    "interfaces": [
        {
            "interface_id": 1,
            "name": "Ethernet88",
            "setting": "{\"interface\": {\"speed\": \"40000\", \"lane_map\": \"29,30,31,32\"}}",
            "speed": {
                "unit": "G",
                "value": 40
            },
            "state": "active"
        }
    ],
    "is_default": false,
    "transformation_id": 2
}
],
{
    "column_id": 12,
    "connector_type": "qsfp28",
    "failure_domain_id": 1,
    "panel_id": 1,
    "port_id": 23,
    "row_id": 2,
    "slot_id": 0,
    "transformations": [
        {
            "interfaces": [
                {
                    "interface_id": 1,
                    "name": "Ethernet92",
                    "setting": "{\"interface\": {\"speed\": \"100000\", \"lane_map\": \"25,26,27,28\"}}",
                    "speed": {

```

```

        "unit": "G",
        "value": 100
    },
    "state": "active"
}
],
"is_default": true,
"transformation_id": 1
},
{
    "interfaces": [
        {
            "interface_id": 1,
            "name": "Ethernet92",
            "setting": "{\"interface\": {\"speed\": \"40000\", \"lane_map\": \"25,26,27,28\"}}",
            "speed": {
                "unit": "G",
                "value": 40
            },
            "state": "active"
        }
    ],
    "is_default": false,
    "transformation_id": 2
}
]
},
{
    "column_id": 13,
    "connector_type": "qsfp28",
    "failure_domain_id": 1,
    "panel_id": 1,
    "port_id": 24,
    "row_id": 1,
    "slot_id": 0,
    "transformations": [
        {
            "interfaces": [
                {
                    "interface_id": 1,
                    "name": "Ethernet96",
                    "setting": "{\"interface\": {\"speed\": \"100000\", \"lane_map\": \"25,26,27,28\"}}",
                    "speed": {
                        "unit": "G",
                        "value": 100
                    },
                    "state": "active"
                }
            ],
            "is_default": false,
            "transformation_id": 3
        }
    ]
}
]
}

```

```

\"117,118,119,120\"}},
    "speed": {
        "unit": "G",
        "value": 100
    },
    "state": "active"
}
],
"is_default": true,
"transformation_id": 1
},
{
    "interfaces": [
        {
            "interface_id": 1,
            "name": "Ethernet96",
            "setting": "{\"interface\": {\"speed\": \"40000\", \"lane_map\": \"117,118,119,120\"}}\",
            "speed": {
                "unit": "G",
                "value": 40
            },
            "state": "active"
        }
    ],
    "is_default": false,
    "transformation_id": 2
}
]
},
{
    "column_id": 13,
    "connector_type": "qsfp28",
    "failure_domain_id": 1,
    "panel_id": 1,
    "port_id": 25,
    "row_id": 2,
    "slot_id": 0,
    "transformations": [
        {
            "interfaces": [
                {
                    "interface_id": 1,

```

```

        "name": "Ethernet100",
        "setting": "{\"interface\": {\"speed\": \"100000\", \"lane_map\":
\\\"113,114,115,116\\\"}}\",
        \"speed\": {
            \"unit\": \"G\",
            \"value\": 100
        },
        \"state\": \"active\"
    }
],
    \"is_default\": true,
    \"transformation_id\": 1
},
{
    \"interfaces\": [
        {
            \"interface_id\": 1,
            \"name\": \"Ethernet100\",
            \"setting\": \"{\\\"interface\\\": {\\\"speed\\\": \\\"40000\\\", \\\"lane_map\\\":
\\\"113,114,115,116\\\"}}\",
            \"speed\": {
                \"unit\": \"G\",
                \"value\": 40
            },
            \"state\": \"active\"
        }
    ],
    \"is_default\": false,
    \"transformation_id\": 2
}
]
},
{
    \"column_id\": 14,
    \"connector_type\": \"qsfp28\",
    \"failure_domain_id\": 1,
    \"panel_id\": 1,
    \"port_id\": 26,
    \"row_id\": 1,
    \"slot_id\": 0,
    \"transformations\": [
        {
            \"interfaces\": [

```

```

        {
            "interface_id": 1,
            "name": "Ethernet104",
            "setting": "{\"interface\": {\"speed\": \"100000\", \"lane_map\": \"125,126,127,128\"}}",
            "speed": {
                "unit": "G",
                "value": 100
            },
            "state": "active"
        }
    ],
    "is_default": true,
    "transformation_id": 1
},
{
    "interfaces": [
        {
            "interface_id": 1,
            "name": "Ethernet104",
            "setting": "{\"interface\": {\"speed\": \"40000\", \"lane_map\": \"125,126,127,128\"}}",
            "speed": {
                "unit": "G",
                "value": 40
            },
            "state": "active"
        }
    ],
    "is_default": false,
    "transformation_id": 2
}
]
},
{
    "column_id": 14,
    "connector_type": "qsfp28",
    "failure_domain_id": 1,
    "panel_id": 1,
    "port_id": 27,
    "row_id": 2,
    "slot_id": 0,
    "transformations": [

```

```

    {
      "interfaces": [
        {
          "interface_id": 1,
          "name": "Ethernet108",
          "setting": "{\"interface\": {\"speed\": \"100000\", \"lane_map\": \"121,122,123,124\"}}",
          "speed": {
            "unit": "G",
            "value": 100
          },
          "state": "active"
        }
      ],
      "is_default": true,
      "transformation_id": 1
    },
    {
      "interfaces": [
        {
          "interface_id": 1,
          "name": "Ethernet108",
          "setting": "{\"interface\": {\"speed\": \"40000\", \"lane_map\": \"121,122,123,124\"}}",
          "speed": {
            "unit": "G",
            "value": 40
          },
          "state": "active"
        }
      ],
      "is_default": false,
      "transformation_id": 2
    }
  ],
  {
    "column_id": 15,
    "connector_type": "qsfp28",
    "failure_domain_id": 1,
    "panel_id": 1,
    "port_id": 28,
    "row_id": 1,

```

```

"slot_id": 0,
"transformations": [
  {
    "interfaces": [
      {
        "interface_id": 1,
        "name": "Ethernet112",
        "setting": "{\"interface\": {\"speed\": \"100000\", \"lane_map\": \"5,6,7,8\"}}",
        "speed": {
          "unit": "G",
          "value": 100
        },
        "state": "active"
      }
    ],
    "is_default": true,
    "transformation_id": 1
  },
  {
    "interfaces": [
      {
        "interface_id": 1,
        "name": "Ethernet112",
        "setting": "{\"interface\": {\"speed\": \"40000\", \"lane_map\": \"5,6,7,8\"}}",
        "speed": {
          "unit": "G",
          "value": 40
        },
        "state": "active"
      }
    ],
    "is_default": false,
    "transformation_id": 2
  }
],
{
  "column_id": 15,
  "connector_type": "qsfp28",
  "failure_domain_id": 1,
  "panel_id": 1,
  "port_id": 29,
  "row_id": 2,

```

```

"slot_id": 0,
"transformations": [
  {
    "interfaces": [
      {
        "interface_id": 1,
        "name": "Ethernet116",
        "setting": "{\"interface\": {\"speed\": \"100000\", \"lane_map\": \"1,2,3,4\"}}",
        "speed": {
          "unit": "G",
          "value": 100
        },
        "state": "active"
      }
    ],
    "is_default": true,
    "transformation_id": 1
  },
  {
    "interfaces": [
      {
        "interface_id": 1,
        "name": "Ethernet116",
        "setting": "{\"interface\": {\"speed\": \"40000\", \"lane_map\": \"1,2,3,4\"}}",
        "speed": {
          "unit": "G",
          "value": 40
        },
        "state": "active"
      }
    ],
    "is_default": false,
    "transformation_id": 2
  }
],
{
  "column_id": 16,
  "connector_type": "qsfp28",
  "failure_domain_id": 1,
  "panel_id": 1,
  "port_id": 30,
  "row_id": 1,

```

```

    "slot_id": 0,
    "transformations": [
      {
        "interfaces": [
          {
            "interface_id": 1,
            "name": "Ethernet120",
            "setting": "{\"interface\": {\"speed\": \"100000\", \"lane_map\": \"13,14,15,16\"}}",
            "speed": {
              "unit": "G",
              "value": 100
            },
            "state": "active"
          }
        ],
        "is_default": true,
        "transformation_id": 1
      },
      {
        "interfaces": [
          {
            "interface_id": 1,
            "name": "Ethernet120",
            "setting": "{\"interface\": {\"speed\": \"40000\", \"lane_map\": \"13,14,15,16\"}}",
            "speed": {
              "unit": "G",
              "value": 40
            },
            "state": "active"
          }
        ],
        "is_default": false,
        "transformation_id": 2
      }
    ],
    {
      "column_id": 16,
      "connector_type": "qsfp28",
      "failure_domain_id": 1,
      "panel_id": 1,

```

```

    "port_id": 31,
    "row_id": 2,
    "slot_id": 0,
    "transformations": [
      {
        "interfaces": [
          {
            "interface_id": 1,
            "name": "Ethernet124",
            "setting": "{\"interface\": {\"speed\": \"100000\", \"lane_map\": \"9,10,11,12\"}}",
            "speed": {
              "unit": "G",
              "value": 100
            },
            "state": "active"
          }
        ],
        "is_default": true,
        "transformation_id": 1
      },
      {
        "interfaces": [
          {
            "interface_id": 1,
            "name": "Ethernet124",
            "setting": "{\"interface\": {\"speed\": \"40000\", \"lane_map\": \"9,10,11,12\"}}",
            "speed": {
              "unit": "G",
              "value": 40
            },
            "state": "active"
          }
        ],
        "is_default": false,
        "transformation_id": 2
      }
    ]
  },
  "selector": {
    "manufacturer": "Dell|DELL",
    "model": "Z9100-ON",

```

```

    "os": "SONiC",
    "os_version": ".*"
  },
  "slot_count": 0,
  "software_capabilities": {
    "lxc_support": false,
    "onie": true
  }
}

```

## Cumulus Device Profile

### IN THIS SECTION

- [Background | 360](#)
- [Problem Statement | 360](#)
- [Solution | 360](#)
- [User Interface | 361](#)
- [DP Data Model | 361](#)
- [Get selector information from the device | 368](#)
- [Model and manufacturer/vendor | 370](#)
- [OS Version | 371](#)
- [OS Family | 371](#)
- [Common default Values | 372](#)
- [Capabilities | 373](#)
- [Port-specific semantics | 374](#)
- [Interface Name semantics on Cumulus | 375](#)
- [Port Setting Schema on Cumulus | 375](#)
- [Edit Interface Setting per Interface on UI | 376](#)
- [Auto-negotiation for 10GBaseT ports | 380](#)
- [Inter port constraints | 381](#)
- [Dell Z9100-ON | 382](#)
- [Dell 4128F | 383](#)
- [Dell 4148T | 384](#)
- [Dell 4148F | 385](#)

- [Edgecore 6812\\_32x\\_O | 385](#)
- [Mellanox 2700 | 386](#)
- [Troubleshooting | 390](#)
- [Appendix | 390](#)
- [References | 390](#)

## Background

Device profiles are how devices are recognized and used inside Apstra. They capture device-specific semantics, which are required for being discovered by Apstra and to run network configs that work well for the datapath once inside the blueprint.

Device profiles are REST entities, which enable you to create, edit, delete, and list during the design phase. Device profiles are used to create interface maps, which get directly used inside the Apstra config rendering engine when blueprints are deployed.

This document covers the knowledge required to create (and edit) a semantically correct Cumulus DP, so that not only does it pass the validations in place in Apstra which ensure the right DP is created in the database, but also honors the vendor semantic requirement applicable to the device so that it does not result in deploy failure when the generated configuration is pushed to the network device.

## Problem Statement

To create a device profile, you need the device specification (usually released by the vendor). You also need to know how to translate these specifications into Apstra DP data model so as to create the valid and config-friendly JSON. This is because the DP is a vendor semantics aware data structure.

## Solution

The high level data model is the same for all DPs, i.e. there are the same keys for every vendor's DP, just the way we get the values might differ, or might be loaded with a vendor constraint. The document enlists the following:

- The schema of the DP and the nested elements inside the DP.
- The meaning of each key value pair in the schema.
- The vendor specific recipe the values are populated.
- List any constraints, corner cases which need to be considered, especially for port configurations for certain (group of) models.

- Any lessons learnt along the way creating those DPs already in production useful in creating future ones.

## User Interface

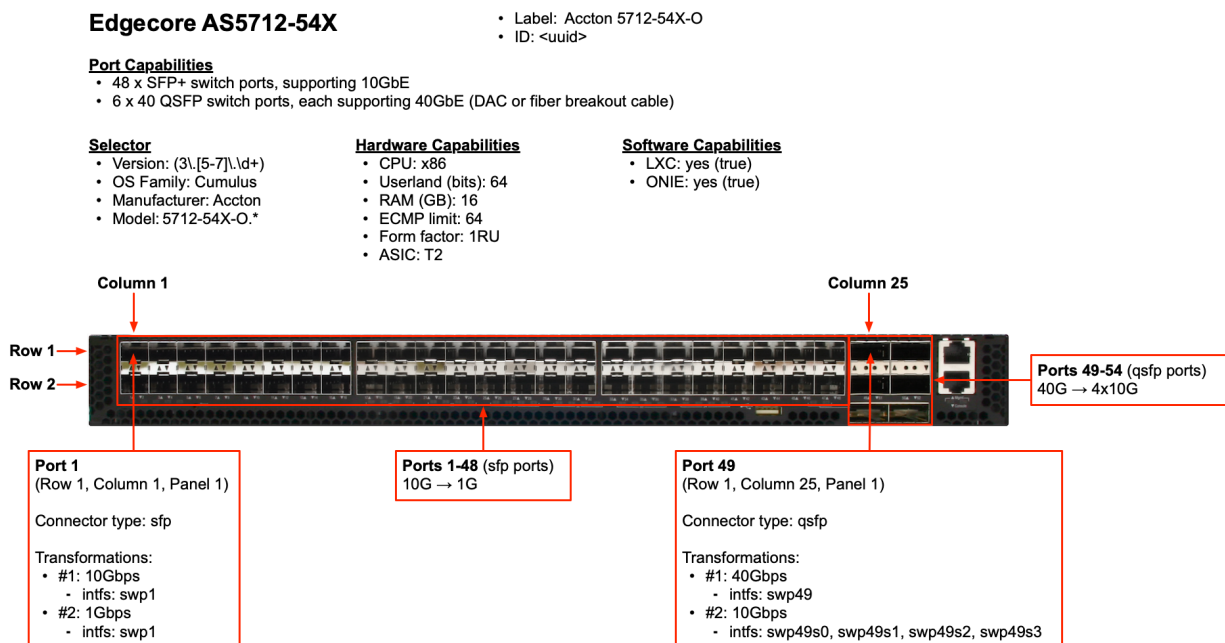
When you create device profiles from the Apstra GUI, some of your entries are semantically validated. It's not completely capable of ensuring deep vendor-specific constraints and requirements though. With the exact vendor specification, the GUI assists you with creating a semantically valid DP which becomes part of the Apstra database data model.

Alternatively, you can write your own Python code that contains the vendor specifications, normalize it as per Apstra DP data model and generate the json to then import with the GUI.

## DP Data Model

This section discusses the general DP data model in order to set the ground for Cumulus specific details. Some of the optional fields might not apply to Cumulus and we will discuss those in more detail in subsequent sections.

Here is the front panel of the device Edgecore AS5712-54X as seen on the vendor's [data specification](#). Some of the elements of the device are marked to highlight how they map in the Apstra device profile model.



Key	Value data type	Value required(R) or optional(O)	Meaning	Purpose/Examples
id	string	R	The id of the device profile REST entity, needs to be unique	The DP is a rest entity like some of the other AOS elements like logical device and IM etc. Each need a unique identifier which is also used as an index into the database when looking up DPs.
failure_domain_id	int	R	When a device has multiple failure domains, this represents the one that is currently active.	
hardware_capabilities	dict	R	Captures those specifications of the device which are purely hardware	AOS needs a way to capture some of the HW capabilities which identify the device or highlight some of its unique strengths like ram, cpu, ecmp limit etc
userland	int	R	The type of application binary/ kernel that the device supports. Usually this value is 32 or 64.	
copp_strict	list	O	when set to 'True, generate strict copp 'profile config for NXOS	<p>This field is used in the nxos jinja files to generate the config.</p> <p>Usually for most models, this value is True, except for models like Cisco 3172 and Cisco nxosv</p>

*(Continued)*

Key	Value data type	Value required(R) or optional(O)	Meaning	Purpose/Examples
breakout_capable	list	O	When set to true for a module, indicates that ports of that module are breakout capable	
ram	int	R	How much memory does the device have	The unit is GB.
asic	string	R	Type of chipset on the device	Example: "T2", "T3", "Arad (3)", "Alta"
as_seq_num_supported	list	O		
form_factor	string	R	How much rack space/units does the device need	Examples: '1RU', '2RU', '7RU', '13RU'
ecmp_limit	int	R	Maximum number of next hops for ECMP routes	This field changes the BGP configuration on the device (ecmp max-paths)
cpu	string	R	What is the cpu architecture of the device. Note: AOS can only support "x86" if users intend to use onbox device agents because of TACC limitations. If the users intend to use offbox agents, then there is no such restriction.	Example: 'x86'
label	string	R	A human readable name for the DP	
software_capabilities	dict	R	A device might have some unique software capabilities.AOS requires these fields to have been set correctly for proper config rendering and deployment.	
onie	bool	R	Does the NOS support onie	

*(Continued)*

Key	Value data type	Value required(R) or optional(O)	Meaning	Purpose/Examples
lxc_support	bool	R	Does the device support LXC containers	
slot_count	int	R	How many slots does the device have	If the device is a modular device, it will have a non-zero slot_count.
selector	dict	R	The selector information is used by the AOS to match the DP to the actual device. Each of the four fields in the selector need to exactly match the ones as derived from the device.	As part of device bring up, the device agent does an exact match on each of these fields to successfully assign the DP to the device. Unless this happens, the device cannot be made part of AOS
os_version	string	R	Regex for the OS version on the device. AOS will use this regex from the DP to match and assign a system device to a DP	
model	string	R	Regex for the Model information as seen on syseeprom that AOS will use to match and assign a system device to a DP	
os	string	R	Regex for OS name that AOS will use to match and assign a system device to a DP	
manufacturer	string	R	Regex of vendor/manufacturer that AOS will use to match and assign a system device to a DP	

*(Continued)*

Key	Value data type	Value required(R) or optional(O)	Meaning	Purpose/Examples
ports	list	R	Number of ports that the device specifies.	Port is a physical entity on the device as seen from the switch's front panel
panel_id	int	R	Being a per port field, This field is used by UI to display the rows and columns. For devices where the ports are divided between multiple panels, user will need to rightly map ports to the correct panel	panel_id is informational only at this point. The Apstra GUI uses this information to geometrically place the ports as close to how they appear on the device's physical front panel. The panel_id is not used in the config rendering engine inside AOS and is purely for cosmetic mapping between the physical appearance of the ports on the device to that on AOS UI port display.
slot_id	int	R	Being a per port field, this represents the module/slot this port belongs to. For modular devices, there will be multiple slots.	
connector_type	string	R	Being a per port field, this records the connector type for the physical port	

*(Continued)*

Key	Value data type	Value required(R) or optional(O)	Meaning	Purpose/Examples
row_id	int	R	This is used to represent the port layout. User can see the specification and figure out how many rows the layout has and the row ID for this port.	
column_id	int	R	This is used to represent the port layout. User can see the specification and figure out how many columns the layout has and the column ID for this port.	
transformations	list	R	Transformations capture the breakout capabilities of the port. If a port is 40G->4x10G, there will be 2 transformations. There is at least one transformation per port, because every port is capable of at least one speed.	
is_default	bool	R	Being a per transformation field, this represents if the speed this transformation represents is the default speed of the port. This information can be read from the spec. For example for a qsfp28 port, the default speed is 100G, so the transformation with 100G interfaces sets the is_default to True	
interfaces	list	R	This is a per transformation field. For the 4x10G transformation, there will be 4 active interfaces, each capable of 10G. Similarly for the 40G transformation, there will be one active interface with speed 40G.	Interfaces are the effective port state when put in a breakout mode.

*(Continued)*

Key	Value data type	Value required(R) or optional(O)	Meaning	Purpose/Examples
state	string	R	Per interface field. Represents if it is active or inactive interface. This field is useful when building IMs using this DP, especially for models with inter-port constraints	On certain vendors like EOS, the OS expects inactive and active interfaces in the config. For example, in a 40G->4x10G, transformation #1 will have 1 active 40G interface and 3 inactive interfaces. Transformation #2 will have 4 active 10G interfaces. Example values: "active", "inactive"
setting	string	Optional	Per interface field. This captures the vendor specific command to apply an interface speed andand to bring the port into selected breakout mode.	
speed	dict	Optional	Per interface field. This represents the speed this interface is to run once configured on the device.	
unit	string	R	Per speed field, represents the speed unit	
value	int	R	Per speed field, represents the speed value	

*(Continued)*

Key	Value data type	Value required(R) or optional(O)	Meaning	Purpose/Examples
name	string	R	Per interface field, this is the name of the interface as the vendor/ operating system requires the name when the interface is configured in the selected breakout transformation on the device. This is a semantic heavy field.	
interface_id	int	R	A unique identifier which is used to ID the interface in the database.	
transformation_id	int	R	A unique identifier which is used to ID the transformation in the database.	
port_id	int	R	A unique identifier which is used to ID the port in the database.	

### Get selector information from the device

Entering the correct information in all the 4 fields of the selector is critical for the device to get matched to the device profile.

In the following table, mostly column 4 is enough for us to get all the 4 fields. Column 5 discusses a little more in detail how Apstra gets these fields and processes in Python. Reading Column 5 helps the user gain certainty, nevertheless it is not necessary if one logs into the box and is able to read the file, type the commands and manually human-read this simple information.

Reference: Read [this link](#) to understand the Cumulus TLV codes for the sys eeprom fields.

General Example output of a typical Cumulus device syseeprom information.

```
admin@leaf-1-1:mgmt-vrf:~$ decode-syseeprom
TlvInfo Header:
  Id String:   TlvInfo
  Version:    1
  Total Length: 168
```

TLV Name	Code	Len	Value
Manufacture Date	0x25	19	10/23/2015 23:11:35
Diag Version	0x2E	7	2.0.1.4
Label Revision	0x27	4	R01I
Platform Name	0x28	27	x86_64-accton_as5712_54x-r0
ONIE Version	0x29	13	2014.08.00.05
Manufacturer	0x2B	6	Accton
Manufacture Country	0x2C	2	TW
Base MAC Address	0x24	6	CC:37:AB:62:F3:4B
Serial Number	0x23	14	571254X1541008
Part Number	0x22	13	FP1ZZ5654001A
Product Name	0x21	15	5712-54X-0-AC-F
MAC Addresses	0x2A	2	74
Vendor Name	0x2D	8	Edgecore
CRC-32	0xFE	4	0x95E4178F

(checksum valid)

admin@leaf-1-1:mgmt-vrf:~\$

Now, let's try and apply these to an example in Apstra, let's pick Accton 5712-54X.

Selector field	Value	Example	What files on the device have this information	[Extra information] Python processing to get the exact fields after we run the command/read the file
model	0x21	5712-54X-O-AC-F	decode_syseeprom_command = "/usr/cumulus/bin/decode-syseeprom"	Not much, read the syseeprom, use the 0x21 value in TLV
manufacturer	If 0x2D in syseeprom, 0x2D else 0x2B	Edgecore	decode_syseeprom_command = "/usr/cumulus/bin/decode-syseeprom"	Not much, read the syseeprom, use the (0x2D or 0x2B )value in TLV

*(Continued)*

Selector field	Value	Example	What files on the device have this information	[Extra information] Python processing to get the exact fields after we run the command/read the file
version	cat /etc/lsb-release, pick the DISTRIB_RELEASE	3.7.3	lsb_release_file = "/host_etc/lsb-release"	cumulus_info = read the lsb-release file  match = re.search(r'%s=(.+)' % re.escape('NAME'), cumulus_info)  value = match.group(1) if value[0] == value[-1] and value[0] in '"': value = value[1:-1] return value.split('"')[0]
OS family	cat /etc/os-release, pick the first part of the NAME	Cumulus	os_release_file = "/host_etc/os-release"	Read the file  cumulus_info = <file contents> match = re.search(r'%s=(S+)' % re.escape('DISTRIB_RELEASE'), cumulus_info) return match.group(1)

**Model and manufacturer/vendor**

```

admin@leaf-1-1:mgmt-vrf:~$ decode-syseeprom
TlvInfo Header:
  Id String:   TlvInfo
  Version:     1
  Total Length: 168
TLV Name      Code Len Value
-----
Manufacture Date  0x25  19 10/23/2015 23:11:35
Diag Version      0x2E   7 2.0.1.4
Label Revision    0x27   4 R01I
Platform Name     0x28  27 x86_64-accton_as5712_54x-r0
ONIE Version      0x29  13 2014.08.00.05
Manufacturer      0x2B   6 Accton

```

```

Manufacture Country  0x2C   2  TW
Base MAC Address     0x24   6  CC:37:AB:62:F3:4B
Serial Number        0x23  14  571254X1541008
Part Number          0x22  13  FP1ZZ5654001A
Product Name         0x21  15  5712-54X-0-AC-F
MAC Addresses        0x2A   2  74
Vendor Name          0x2D   8  Edgecore
CRC-32               0xFE   4  0x95E4178F
(checksum valid)
admin@leaf-1-1:mgmt-vrf:~$

```

## OS Version

```

admin@leaf-1-1:mgmt-vrf:~$ cat /etc/lsb-release
DISTRIB_ID="Cumulus Linux"
DISTRIB_RELEASE=3.7.3
DISTRIB_DESCRIPTION="Cumulus Linux 3.7.3"
admin@leaf-1-1:mgmt-vrf:~$

```


## OS Family

```

admin@leaf-1-1:mgmt-vrf:~$ cat /etc/os-release
NAME="Cumulus Linux"
VERSION_ID=3.7.3
VERSION="Cumulus Linux 3.7.3"
PRETTY_NAME="Cumulus Linux"
ID=cumulus-linux
ID_LIKE=debian
CPE_NAME=cpe:/o:cumulusnetworks:cumulus_linux:3.7.3
HOME_URL="http://www.cumulusnetworks.com/"
SUPPORT_URL="http://support.cumulusnetworks.com/"
admin@leaf-1-1:mgmt-vrf:~$

```

Using the above gotten information to Accton 5712 54X's selector as displayed in UI:


› Devices › Device Profiles › Accton 5712-54X-O

**Selector?**

<b>Manufacturer</b>	Accton
<b>Model</b>	5712-54X-O.*
<b>OS family</b>	Cumulus
<b>Version</b>	(3\[5-7]\d+)

### Common default Values

What makes the Apstra SDK generators fast tools when it comes to generating the JSONs is using some of the knowledge specific to a vendor and re-using them in Python code to generate more than one DP. Commonalities across models are set as defaults. While every field in the DP should be populated per device carefully consulting the specification, one can also use these defaults as a reference when populating the fields.

Field	Default value	Meaning
ports	-NA- (list)	Ports for a device are one of those things which mostly vary by device. There is no point in having a default for ports, more often than not it won't be used.
label	-NA- (string)	Every label needs to be unique, so a default does not apply
slot_count	0 (int)	Affects config. For non-modular devices, leave this 0. For modular device, this is the number of linecards excludes the supervisor units.

*(Continued)*

Field	Default value	Meaning
id	No default. When user is constructing json, make sure to provide a unique string as id for the DP. This is especially applicable when user builds the json and imports the payload into AOS using the REST POST/PUT API. However, if the user uses the UI to populate the fields, the UI prompts for the label but assigns an id behind the scenes. Nevertheless, this id field is required for a valid DP.	
os_version	"(3\.[5-7]\.\d+)" (string)	Usually the default reflects the versions as supported by AOS for the release. As of AOS 3.1.1 we support 3.5-3.7. AOS maintains an official feature matrix by NOS here <a href="http://docs.dc1.apstra.com/feature_matrix.html">http://docs.dc1.apstra.com/feature_matrix.html</a> The os_version value in the DP is more permissive than what is officially certified, to facilitate experiments of (not officially supported) OS versions for given hardware model.

## Capabilities

If you have the device specification, you can obtain its hardware and software capabilities for entry into the device profile.

The table below contains commonly found values in Cumulus devices (based on qualified devices).

Selector Field	Value	Description
userland	64 (int)	This does not affect config as of AOS 3.1.1
form_factor	'1RU' (string)	This does not affect config as of AOS 3.1.1
ecmp_limit	64 (int)	This does not affect config as of AOS 3.1.1

*(Continued)*

Selector Field	Value	Description
asic	'T2' (string)	AOS looks at this for certain scenarios like Cumulus release note RN 766. The RN states that on the Broadcom Trident II+, Trident 3, and Maverick platforms, in an external VXLAN routing environment, the switch does not rewrite MAC addresses and TTL, so packets are dropped by the next hop. To configure this, the asic field is looked at in conjunction with the os version field to determine if RN766 applies to the DP/device version and if yes, to handle it accordingly.
cpu	'x86' (string)	This does not affect config as of AOS 3.1.1
ram	16 (int) (Note, the unit is in GB)	This does not affect config as of AOS 3.1.1
onie	True (bool) (default)	This does not affect config as of AOS 3.1.1
lxc	True (bool) (default)	This does not affect config as of AOS 3.1.1

**NOTE:** You are not required to configure any extra configurables for "supported features" for Cumulus device profiles. Thus the GUI says "no supported features".

### Port-specific semantics

Every interface per transformation per port per device profile requires a setting.

This port setting is used by the configuration renderer (jinja files) to correctly configure the interfaces on the device. The DP allows you to configure these for the device. Thus, it is very important these setting fields are populated correctly in the DP. In this section, we will look at what is the schema Apstra enforces for Cumulus port settings.

Example port setting:

```
{
  "interface": {
    "speed": "10000",
    "lane_map": "13"
```

```
    }  
  }  
}
```

Where is this field found: If you have 32 ports, each port capable of 40G and 4x10G, you will have:

- 2 transformations per port
- 1 active interface in 1st transformation; setting can be found in each of these interfaces
- 4 active interfaces in the 2nd transformation; setting can be found in each of these interfaces

**Interface Name semantics on Cumulus**

The rules for naming an interface are as follows:

- Arguments: port\_id(required), lane\_id(optional, applies only in case of a broken out port)

Example of an interface name on Cumulus, **swp49s0**. First get the lane\_suffix:

Example: **s0**

If lane\_id: Lane\_suffix = "s%d" % lane\_id

Else: Lane\_suffix = ""

Then build the name, using port id and lane\_suffix Name = "swp%d%s" % (port\_id, lane\_suffix)

Examples: Interfaces for 4x10G transformation on port id 49: - **swp49s0** - **swp49s1** - **swp49s2** - **swp49s3** Interfaces for 40G transformation on port id 49: - **swp49**

**Port Setting Schema on Cumulus**

Cumulus port setting	dict	Required/Optional field	The high level dictionary
interface	dict	R	The interface this setting is applicable to
command	string	R. This is the speed to command mapping as required by cumulus. For example, if interface speed is 1G, then command is 'link-speed 1000'  command = { 1: 'link-speed 1000', 10: ", 20: ", 25: ", 40: ", 50: ", 100: ", }	The command if applicable when user needs to apply the speed on this interface

(Continued)

Cumulus port setting	dict	Required/Optional field	The high level dictionary
speed	string	R ()	The speed to be applied to this interface

Edit Interface Setting per Interface on UI

The setting contains two important pieces of information, the command and the speed. The "command" goes underneath the interface paragraph in /etc/network/interfaces, and "speed" is the value that goes to ports.conf.

- I know my port breakout capability
- I am ready to create ports in Apstra UI
- I am in the DP->edit->ports->port
- How do I populate the Setting?

Summary

Selector?

Capabilities

Ports

Panel #1 Select port(s) to fill in the details

Port breakoutAutonegotiation

357911131517192123252729313335373941434547495153

24681012141618202224262830323436384042444648505254

Edit selected port #1

Connector type\*

sfp28

Transformations

DefaultSpeedInterfaces

25 Gbps

Active

Name template\*

swp1

Setting\*

["interface": {"command": ""}]

10 Gbps

swp1

Add new transformation

Add Panel

Update

Interfaces applicable in this transformation	This interface speed	to_count	No breakout?	Speed to be populated in interface setting	command	Settings as found per interface on UI
40G	40G	1	Yes (Note the setting remains same even for a port which is capable of 40G breakout. For example On the Accton 6712, we have all ports capable of 40G->4x10G. Whereas on the Dell S6010, we have ports 13-16 and 29-32 which are 40G non breakout capable ports. Irrespective of whether it is breakout capable or non breakout capable, the setting for 40G interface remains same)	'40G'	"	{ "interface": { "command": "", "speed": "40G" } }
1x40G->4x10G	10G	4	No	'4x10G'	"	{ "interface": { "command": "", "speed": "4x10G" } }
1G	1G	1	Yes	'1G'	link-speed 1000'	{ "interface": { "command": "link- speed 1000", "speed": "10G" } }

*(Continued)*

Interfaces applicable in this transformation	This interface speed	to_count	No breakout?	Speed to be populated in interface setting	command	Settings as found per interface on UI
1G auto neg	1G autoneg	1	Yes	"	"	{ "interface": { "command": "link-speed 1000", "speed": "" } }
10G	10G	1	Yes	'10G'	"	{ "interface": { "command": "", "speed": "10G" } }
1x10G->1x1G	1G	1	No	'10G'	link-speed 1000'	{ "interface": { "command": "link-speed 1000", "speed": "10G" } }
1x10G->1x1G autoneg	1G	1	No	"	link-speed 1000'	"interface": { "command": "link-speed 1000", "speed": "" } }
1x10G->1x1G setting(used on Dell 4128 and 4148F)	1G	1	Yes	'1G'	link-speed 1000'	{ "interface": { "command": "link-speed 1000", "speed": "1G" } }
As per cumulus release notes rn778, this switch should use 1G (or 1000) in Ports.conf when 10G port is broken into 1G						
1x10G->1x10G autoneg	10G	1	No	"	"	{ "interface": { "command": "", "speed": "" } }

*(Continued)*

Interfaces applicable in this transformation	This interface speed	to_count	No breakout?	Speed to be populated in interface setting	command	Settings as found per interface on UI
100G	100G	1	Yes	'100G'	"	{ "interface": { "command": "", "speed": "100G" } }
1x100G->4x25G	25G	4	No	'4x25G'	"	{ "interface": { "command": "", "speed": "4x25G" } }
1x100G->4x10G	10G	4	No	'4x10G'		{ "interface": { "command": "", "speed": "4x10G" } }
1x100G->4x1G	1G	4	No	'10G'	link-speed 1000'	{ "interface": { "command": "link- speed 1000", "speed": "10G" } }
1x100G->12x10G	10G	12	No	"12x10G'	"	{ "interface": { "command": "", "speed": "12x10G" } }
1x100G->2x20G	20G	2	No	'2x20G'	"	{ "interface": { "command": "", "speed": "2x20G" } }
1x100G->1x40G	40G	1	No	'40G'	"	{ "interface": { "command": "", "speed": "40G" } }
1x100G->3x40G	40G	3	No	'3x40G'	"	{ "interface": { "command": "", "speed": "3x40G" } }

(Continued)

Interfaces applicable in this transformation	This interface speed	to_count	No breakout?	Speed to be populated in interface setting	command	Settings as found per interface on UI
1x100G->2x50G	50G	2	No	'2x50G'	"	{ "interface": { "command": "", "speed": "2x50G" } }
1x100G->1x50G	50G	1	No	'50G'	"	{ "interface": { "command": "", "speed": "50G" } }
25G	25G	1	Yes	'25G'	"	{ "interface": { "command": "", "speed": "25G" } }
1x25G->1x10G	10G	1	No	'10G'	"	{ "interface": { "command": "", "speed": "10G" } }

### Auto-negotiation for 10GBaseT ports

On the 10GBaseT, Cumulus supports auto negotiation for some devices.

To incorporate this into the DP, on that port which supports the 10G->1x1G breakout, for the 1G transformation, have the interface setting as follows:

```
{
  "interface": {
    "command": "link-speed 1000",
    "speed": ""
  }
}
```

Note the "speed" inside the "interface" setting is set to "". This allows the interface to be configured in auto-neg mode and device will auto-negotiate the interface speed.

In the IM, pick the auto-neg transformation for this port.

How to make the interface as auto neg in the DP:

1. Go to the 10GBaseT port
2. Create a 1G transformation as follows:
  - state: active
  - speed: max speed possible for interface speed
  - setting->interface->speed: empty
  - name: as per interface name rules

For example, on the Dell S4148T-ON, the ports from 1-24 and 31-54 are 10GBaseT ports. Thus, Transformation #2, configures the speed in the interface setting as "".

**Panel #1**  
 INTERFACES CAPACITY

72 x 10 Gbps    48 x 1 Gbps    4 x 100 Gbps    8 x 50 Gbps    6 x 40 Gbps    16 x 25 Gbps

PORTS Click on a port to toggle the details

1	3	5	7	9	11	13	15	17	19	21	23	25	27
2	4	6	8	10	12	14	16	18	20	22	24	26	28

PORT DETAILS
 

ID	
Connector type	
Transformations	
#1 (10 Gbps, default)	
#2 (1 Gbps)	swp31

Name	swp31
Active?	yes
Speed	1 Gbps
Setting?	<pre>{   "interface": {     "command": "link-speed 1000",     "speed": ""   } }</pre>

### Inter port constraints

To capture all the essence of ports of a particular device, not only does the user need to read the device specification, but also the OS specific port rules. Fortunately, Cumulus packages these rules in the ports.conf in the dpkg bundle.

When we are writing device profiles for new Cumulus devices, the procedure ideally to be followed is:

1. To generate a list of ports(with correct transformations and interfaces) in the DP, a good way is to have access to the ports.conf and work our way backwards to then generate the ports list in the DP

such that the Cumulus OS likes it. To know the detailed port constraints, read the ports.conf of that particular model in the dpkg.

- 2. run the following command on any Cumulus box(even VX) to get access to ports.conf:

```
dpkg -L cumulus-platform | grep ports.conf
```

Pay attention to the comments sections which lists the port breakout constraints usually. It looks something like this:

ports.conf --

This file controls port aggregation and subdivision. For example, QSFP+ ports are typically configurable as either one 40G interface or four 10G/1000/100 interfaces. This file sets the number of interfaces per port while /etc/network/interfaces and ethtool configure the link speed for each interface.

You must restart switchd for changes to take effect.

The DELL S6010 has:

32 QSFP ports numbered 1-32 These ports are configurable as 40G, split into 4x10G ports or disabled.

The X pipeline covers QSFP ports 1 through 16 and the Y pipeline covers QSFP ports 17 through 32.

The Trident2+ chip can only handle 52 logical ports per pipeline.

This means 13 is the maximum number of 40G ports you can ungang per pipeline, with the remaining three 40G ports set to "disabled". The 13 40G ports become 52 unganged 10G ports, which totals 52 logical ports for that pipeline.

QSFP+ ports <port label 1-32> = [4x10Gdisabled]

- 3. Use the comments when writing a new Cumulus DP.

Dell Z9100-ON

Port Constraint:

The Dell Z9100 has:	32 QSFP28 ports numbered 1-32	These ports are configurable as 100G, 50G, 40G, 2x50G, 4x25G, 4x10G or disabled.
	Two SFP+ ports. These ports are configurable as 10G or disabled.	The system can only handle 128 logical ports. This means that if all 32 QSFP28 ports are broken out into 4x25G or

4x10G mode, the two 10G ports (33 and 34) must be set to "disabled".

To honor this, in the DP, the ports 33-34 should provide a transformation with the interface with disabled setting.

```
{
  "interface": {
    "command": "",
    "speed": "disabled"
  }
}
```

IM constraint:

If all the 32 QSFP28 ports are broken out into 4x25G or 4x10G, then the two 33-34 need to be disabled. So explicitly pick the disabled transformation for 33-34 in this case.

**Dell 4128F**

Port Constraint:

The Dell S4128 has:	28 SFP+ ports numbered 1-24 and 27-30	These ports are configurable as 10G, or groups of four adjacent ports can be configured as 40G.
	2 QSFP28 ports numbered 25-26	These ports are configurable as 100G, 50G, 40G, or split into 2x50G, 4x25G, or 4x10G ports.

**NOTE:** SFP+ ports 13, 14, 23, and 24 are not in a group of four adjacent ports and cannot be configured as part of a ganged 40G.

Ganged ports configuration is not supported.

For the 1G broken out interface, the speed inside the interface setting should be "1G" and not "10G" (which usually is for other cumulus devices)

To honor this, in the DP, the ports 33-34 should provide a transformation with the interface with disabled setting.

```
{
  "interface": {
    "command": "",
    "speed": "disabled"
  }
}
```

IM constraint:

If all the 32 QSFP28 ports are broken out into 4x25G or 4x10G, then the two 33-34 need to be disabled. So explicitly pick the disabled transformation for 33-34 in this case.

**Dell 4148T**

Port Constraint:

<b>The Dell S4148T has:</b>	<b>48 10GT ports numbered 1-24 and 31-54</b>	These ports are not configurable.
	<b>2 QSFP+ ports numbered 27-28</b>	These ports are configurable as 40G or split into 4x10G ports.
	<b>4 QSFP28 ports numbered 25-26 and 29-30</b>	These ports are configurable as 100G, 50G, 40G, or split into 2x50G, 4x25G, or 4x10G ports.

**NOTE:** The two QSFP+ ports are DISABLED by default, and the four QSFP28 ports are configured for 100G by default. In order to enable the two QSFP+ ports for 40G or 4x10G, all four QSFP28 ports must also be configured for either 40G or 4x10G.

In the DP, to honor this constraint we need a transformation with "disabled" setting as follows for the ports 27-28

```
{
  "interface": {
    "command": "",
    "speed": "disabled"
  }
}
```

```
}
}
```

IM constraint:

In order to enable the two QSFP+ ports(27-28) for 40G or 4x10G, all four QSFP28 ports must also be configured for either 40G or 4x10G.

### Dell 4148F

Port Constraint:

<b>The Dell S4148 has:</b>	<b>48 SFP+ ports numbered 1-24 and 31-54</b>	These ports are configurable as 10G, or groups of four adjacent ports can be configured as 40G.
	<b>2 QSFP+ ports numbered 27-28</b>	These ports are configurable as 40G or split into 4x10G ports.
	<b>4 QSFP28 ports numbered 25-26 and 29-30</b>	These ports are configurable as 100G, 50G, 40G, or split into 2x50G, 4x25G, or 4x10G ports.

**NOTE:** The two QSFP+ ports are DISABLED by default, and the four QSFP28 ports are configured for 100G by default. In order to enable the two QSFP+ ports for 40G or 4x10G, all four QSFP28 ports must also be configured for either 40G or 4x10G.

The 2 QSFP+ and 4 QSFP28 have same rules as Dell 4148T. The 48 SFP+ ports have same 1G setting constraints as Dell 4128F, i.e.

```
{
  "interface": {
    "command": "link-speed 1000",
    "speed": "10G"
  }
}
```

### Edgecore 6812\_32x\_O

Port Constraint:

This file controls port aggregation and subdivision. For example, QSFP+ ports are typically configurable as either one 40G interface or four 10G/1000/100 interfaces. This file sets the number of interfaces per port while /etc/network/interfaces and ethtool configure the link speed for each interface.

You must restart switchd for changes to take effect.

**Accton AS6812\_32X** has: **32 QSFP+ ports numbered 1-32** These ports are configurable as 40G, split into 4x10G ports or disabled.

The X pipeline covers QSFP ports 1-16 and the Y pipeline covers QSFP ports 17-32.

The Trident2+ chip can only handle 52 logical ports per pipeline.

This means 13 is the maximum number of 40G ports you can ungang per pipeline, with the remaining three 40G ports set to "disabled". The 13 40G ports become 52 ungang 10G ports, which totals 52 logical ports for that pipeline.

To honor this constraint, in the DP, all the 40g->4x10G ports need to have a transformation with an interface with disabled setting as follows:

```
{
  "interface": {
    "command": "",
    "speed": "disabled"
  }
}
```

IM Constraint:

When picking 10G interfaces, make sure there are only a maximum of 52 10G interfaces (coming from the 13 ports) for this IM. Also for the rest of the 3 leftover 40G ports, explicitly select the "disabled" transformation.

This way, the port config will be generated correctly as per port constraints.

## Mellanox 2700

Port Constraint:

Odd numbered ports can do full breakout up to 4x10G and 4x25G.

Even numbered ports can only have upto 2x50G interfaces and should be disabled if immediately preceding odd-numbered port is broken out into 4 interfaces. (Source: <https://docs.mellanox.com/>)

[display/sn2000pub/Cable+Installation](#))

### SN2700 and SN2740 Splitting Options

The top QSFP28 ports marked in green are splittable to 4 SFP28 ports, each.

The bottom QSFP28 ports (gray) are blocked when the upper ports are in split mode.

All QSFP28 ports can be split to 2 QSFP28 ports.



How does this translate in the DP?

The odd numbered ports, like port 1, have following transformations:

- #1 (100 Gbps, default) swp1
- #2 (50 Gbps) swp1s0 swp1s1
- #3 (50 Gbps) swp1
- #4 (40 Gbps) swp1
- #5 (25 Gbps) swp1s0 swp1s1 swp1s2 swp1s3
- #6 (10 Gbps) swp1s0 swp1s1 swp1s2 swp1s3

#7 (1 Gbps)

swp1s0 swp1s1 swp1s2 swp1s3

Panel #1

INTERFACES CAPACITY

32 x 100 Gbps64 x 50 Gbps32 x 40 Gbps64 x 25 Gbps64 x 10 Gbps64 x 1 Gbps

PORTS Click on port to toggle the details

135791113151719212325272931

2468101214161820222426283032

PORT DETAILS

ID1

Connector typeqsf28

Transformations

Port #1 Tr. #1 (100 Gbps, default)

swp1

Port #1 Tr. #2 (50 Gbps)

swp1s0swp1s1

Port #1 Tr. #3 (50 Gbps)

swp1

Port #1 Tr. #4 (40 Gbps)

swp1

Port #1 Tr. #5 (25 Gbps)

swp1s0swp1s1swp1s2swp1s3

Port #1 Tr. #6 (10 Gbps)

swp1s0swp1s1swp1s2swp1s3

Port #1 Tr. #7 (1 Gbps)

swp1s0swp1s1swp1s2swp1s3

Name

swp1s3

Active?

yes

Speed

25 Gbps

Setting?

{  
 "interface": {  
 "command": "",  
 "speed": "4x25G"  
 }  
}

Whereas the even numbered ports, like port 2, have following transformations.

#1 (100 Gbps, default)

swp2

#2 (50 Gbps)

swp2s0 swp2s1

#3 (50 Gbps)

swp2

#4 (40 Gbps)

swp2

#5 (100 Gbps)

swp2

Note that even ports do not have the 10G and 25G transformations. Additionally, they have a "disabled" interface in transformation 5 with the following setting:

```
{
  "interface": {
    "command": "",
    "speed": "disabled"
```

}

}

Panel #1

INTERFACES CAPACITY

32 x 100 Gbps

64 x 50 Gbps

32 x 40 Gbps

64 x 25 Gbps

64 x 10 Gbps

64 x 1 Gbps

PORTS

Click on port to toggle the details

1

3

5

7

9

11

13

15

17

19

21

23

25

27

29

31

2

4

6

8

10

12

14

16

18

20

22

24

26

28

30

32

PORT DETAILS

ID

2

Connector type

Transformations

Port #2 Tr. #1 (100 Gbps, default)

Port #2 Tr. #2 (50 Gbps)

Port #2 Tr. #3 (50 Gbps)

Port #2 Tr. #4 (40 Gbps)

Port #2 Tr. #5 (100 Gbps)

Name

swp2

Active?

no

Speed

100 Gbps

Setting?

{

"interface": {

"command": "",

"speed": "disabled"

}

}

swp2

IM constraints:

If you want to include an odd numbered port which is broken out into 4 interfaces, then make sure you also pick the disabled transformation of the following even numbered port.

For example in the Interface Map Mellanox\_MSN2700\_Cumulus\_\_AOS-48x10\_8x100-1

Because we are using the ports: [1,3,5,7,9,11,13,15,17,19,21,23] to generate the 4x10G interfaces, we also pick the 5th transformation(disabled) for ports [2,4,6,8,10,12,14,16,18,20,22,24]. Thus on the UI, for the Interface Map, you see 12 unused interfaces corresponding to these disabled interfaces.

Only if these are included in the Interface Map, will Apstra generate the right ports.conf honoring the above Cumulus Mellanox 2700 constraint. Refer to the use case for Inter Port Constraints on the ["Interface Maps" on page 91](#) page for more details on Interface Map creation.

## Troubleshooting

Device mismatch usually occurs at the beginning of a device's lifecycle. If a device profile is not being selected by the device, check the four selector fields in the device profile.

Deploy errors could arise because of incorrect port configurations. This could be either the ports were configured with incorrect speeds, or the OS specific port constraints were not handled in the device profile or in the interface map.

A possible flow for root cause would be:

1. Check the DP for obvious port capabilities errors. Is the port really capable of the speeds the DP has configured. The `/var/log/switchd` logs on the device are a good resource to parse for ERROR messages.
2. Check if the DP has configured autoneg or disabled interfaces correctly. Autoneg and disabled can both be expressed in the interface setting field.
3. The port constraints can be read from the following files:

```
admin@leaf1:mgmt-vrf:~$ dpkg -L cumulus-platform | grep ports.conf
```

## Appendix

### Mapping the DP to LD in the IM

The mapping specifies how exactly the user wants to map the physical ports to the logical interfaces.

The 5 fields on the mapping are: [DP portID, DP transformationID, DP InterfaceID, LD panelID, LD portID].

These five fields allow the user to map the DP exactly to the LD. The first three fields specify which port, transformation and interface in the DP is the interface in the IM referencing:

1. (port id, transformation id) - all active interfaces in the given transformation are used; active interfaces are mapped in the order they appear in the transformation.
2. (port id, transformation id, interface id) - If the user specifies the third field in the mapping, the particular interface in the transformation is used in mapping.

## References

[Reading syseeprom on Cumulus device] <https://docs.cumulusnetworks.com/cumulus-linux/Monitoring-and-Troubleshooting/Monitoring-System-Hardware/>

[Edgecore AS5712-54X-0] <https://www.edge-core.com/productsInfo.php?cls=1&cls2=8&cls3=44&id=15>

## Resources

### IN THIS SECTION

- [ASN Pools \(Resources\) | 391](#)
- [VNI Pools \(Resources\) | 393](#)
- [IP Pools \(Resources\) | 395](#)
- [IPv6 Pools \(Resources\) | 397](#)

## ASN Pools (Resources)

### IN THIS SECTION

- [ASN Pool Overview | 391](#)
- [Create ASN Pool | 392](#)
- [Edit ASN Pool | 392](#)
- [Delete ASN Pool | 393](#)

### ASN Pool Overview

Autonomous system numbers (ASNs) are used to support BGP in the underlay. When you're building your blueprint you'll specify which resource pool to use for assigning ASNs.

**NOTE:** If you need to assign a specific ASN to a specific device, you can assign the ASN individually from the staged blueprint in the **Properties** panel of a selection.

ASN pools include the following details:

Name	Description
Pool Name	To identify the resource pool
Total Usage	Percentage of ASNs in use for all ranges in the resource pool. (Hover over the status bar to see the number of ASNs in use and the total number of ASNs in the pool.)
Range Usage	The ASNs included in the range and the percentage that are in use. (Hover over the status bar to see the number of ASNs in use and the total number of ASNs in that range.)
Status	Indicates if the pool is in use

From the left navigation menu in the Apstra GUI, navigate to **Resources > ASN Pools** to go to ASN pools in the design (global) catalog. You can create, clone, edit and delete ASN pools.

The screenshot shows the Juniper Apstra GUI interface. On the left, the navigation menu has 'Resources' highlighted with a red arrow labeled '1.' and 'ASN Pools' highlighted with a red arrow labeled '2.'. The main content area displays a table of ASN pools. The table has columns for Name, Total Usage, Range Usage, Status, and Actions. A 'Create ASN Pool' button is located in the top right corner. The table lists three pools: one in 'IN USE' status and two in 'NOT IN USE' status.

Name	Total Usage	Range Usage	Status	Actions
1 - 100	9%	9%	IN USE	Edit Clone Delete
64512 - 65534	0%	0%	NOT IN USE	Edit Clone Delete
4200000000 - 4294967294	0%	0%	NOT IN USE	Edit Clone Delete

## Create ASN Pool

1. From the left navigation menu, navigate to **Resources > ASN Pools** and click **Create ASN Pool**.
2. Enter a name and range. To add another range, click **Add a range** and enter the range.
3. Click **Create** to create the pool and return to the list view.

When you're building your blueprint, you'll "[assign resources](#)" on [page 420](#) from these pools in the **Staged > Physical** view of the blueprint.

## Edit ASN Pool

1. Either from the list view (Resources > ASN Pools) or the details view, click the **Edit** button for the pool to edit.
2. Make your changes. You can add, change and delete ranges, but you cannot remove ASNs that are in use.

3. Click **Update** to update the pool and return to the list view.

Delete ASN Pool

You can delete ASN pools as long as none of the ASNs within the pool are in use.

- 1. Either from the list view (Resources > ASN Pools) or the details view, click the **Delete** button for the pool to delete.
- 2. Click **Delete** to delete the pool and return to the list view.

VNI Pools (Resources)

IN THIS SECTION

- [VNI Pool Overview | 393](#)
- [Create VNI Pool | 394](#)
- [Edit VNI Pool | 394](#)
- [Delete VNI Pool | 395](#)

VNI Pool Overview

Virtual network identifiers (VNIs) are used in VXLAN encapsulation to provide Layer 2 separation for the overlay traffic in your data center fabric. (For more information about VNI usage, see "[Virtual Networks](#)" on page 488. When you're building your blueprint you'll specify which resource pool to use for assigning VNIs.

NOTE:

Properties

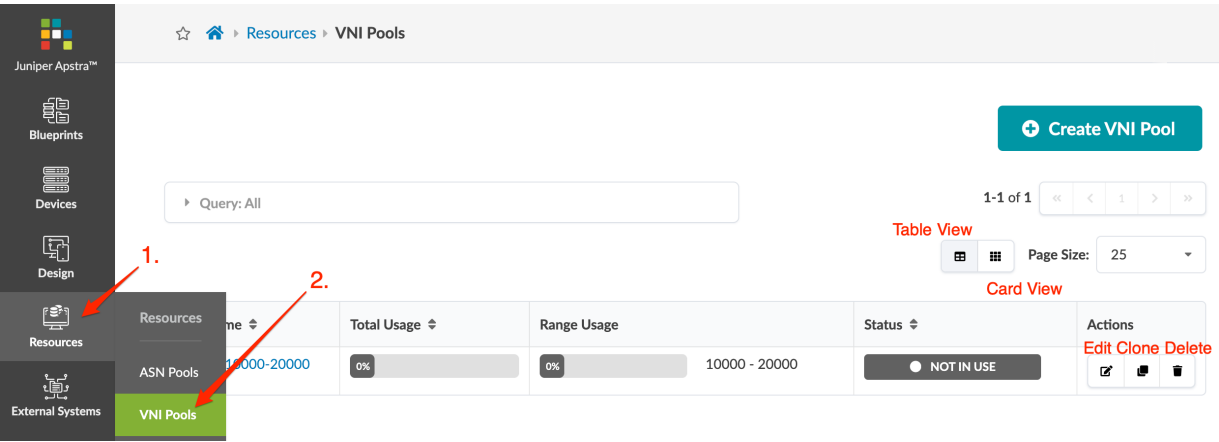
VNI pools include the following details:

Name	Description
Pool Name	To identify the resource pool

(Continued)

Name	Description
Total Usage	Percentage of VNIs in use for all ranges in the resource pool. (Hover over the status bar to see the number of VNIs in use and the total number of VNIs in the pool.)
Range Usage	The VNIs included in the range and the percentage that are in use. (Hover over the status bar to see the number of VNIs in use and the total number of VNIs in that range.)
Status	Indicates if the pool is in use

From the left navigation menu, navigate to **Resources > VNI Pools** to go to VNI pools in the design (global) catalog. You can create, clone, edit and delete VNI pools.



### Create VNI Pool

1. From the left navigation menu, navigate to **Resources > VNI Pools** and click **Create VNI Pool**.
2. Enter a name and a valid range (4096 through 16777214). To add another range, click **Add a range** and enter the range.
3. Click **Create** to create the pool and return to the list view.

When you've created your blueprint, you'll "assign resources" on page 420 from these pools in the **Staged > Virtual** view of the blueprint.

### Edit VNI Pool

1. Either from the list view (**Resources > VNI Pools**) or the details view, click the **Edit** button for the pool to edit.
2. Make your changes. You can add, change, and delete ranges, but you cannot remove any VNIs that are in use.
3. Click **Update** to update the pool and return to the list view.

## Delete VNI Pool

You can delete VNI pools as long as none of the VNIs within the pool are in use.

1. Either from the list view (Resources > VNI Pools) or the details view, click the **Delete** button for the pool to delete.
2. Click **Delete** to delete the pool and return to the list view.

## IP Pools (Resources)

### IN THIS SECTION

- [IP Pool Overview | 395](#)
- [Create IPv4 Pool | 396](#)
- [Edit IPv4 Pool | 397](#)
- [Delete IPv4 Pool | 397](#)

## IP Pool Overview

IP addresses are used in the following situations:

**Loopback IPs - Spines/Leafs/Generics** - the loopback IP is used as the BGP router ID.

**SVI Subnets - MLAG Domain** - A Switch Virtual Interfaces (SVI) subnet for an MLAG domain is used to allocate an IP address between MLAG leaf switches.

**Link IPs - Spines <-> Leafs** - Link IPs are used between spines and leafs to build the L3-CLOS fabric. These IPs are necessary for BGP peering between spines and leafs, and represent the 'fabric' of the network.

**Link IPs - Generics** - IP addresses facing generic systems are used to statically-route the generic system loopback and route across that link.

When you're building your blueprint you'll specify which resource pool to use for assigning IP addresses.

**NOTE:** If you need to assign a specific IP address to a specific device, you can assign the IP address individually from the staged blueprint in the **Properties** panel of a selection.

IP pools include the following details:

**Table 24: IPv4 Pool Parameters**

Name	Description
Pool Name	To identify the resource pool
Total Usage	Percentage of IP addresses in use for all subnets in the resource pool. (Hover over the status bar to see the number of IP addresses in use and the total number of IP addresses in the pool.)
Per Subnet Usage	The IP addresses included in the subnet and the percentage that are in use. (Hover over the status bar to see the number of IP addresses in use and the total number of IP addresses in that subnet.)
Status	Indicates if the pool is in use

From the left navigation menu, navigate to **Resources > IP Pools** to go to IP pools in the design (global) catalog. You can create, clone, edit and delete IPv4 pools.

Juniper Apstra™

☆ 🏠 > Resources > IP Pools

Query: All

1-6 of 6 << < 1 > >>

Table View Card View

Page Size: 25

Name	Total Usage	Per Subnet Usage	Status	Actions	
ASNs	2.37%	2.37%	172.16.0.0/16	IN USE	Edit Clone Delete
VNI Pools	<0.01%	<0.01%	11.1.0.0/16	IN USE	Edit Clone Delete
IP Pools	0%	0%	10.0.0.0/8	NOT IN USE	Edit Clone Delete

## Create IPv4 Pool



**CAUTION:** IP address ranges are not validated. It is your responsibility to specify valid IP addresses. If you configure a switch with an invalid IP block you may receive an **error** during the deploy phase. For example, specifying the erroneous multicast subnet 224.0.0.0/4 would be accepted, but it would result in an unsuccessful deployment. If you assign the same range (or overlapping range) of IP addresses to a blueprint, the

duplicate assignment is detected and you'll receive a **warning** in the blueprint. You can commit changes to blueprints with warnings without resolving the issues.

1. From the left navigation menu, navigate to **Resources > IP Pools** and click **Create IP Pool**.
2. Enter a name and valid subnet. To add another subnet, click **Add a Subnet** and enter a subnet.
3. Click **Create** to create the pool and return to the list view.

When you've created your blueprint, you'll ["assign resources" on page 420](#) from these pools in the **Staged > Physical** view of the blueprint.

### Edit IPv4 Pool

1. Either from the list view (Resources > IP Pools) or the details view, click the **Edit** button for the pool to edit.
2. Make your changes. You can add, change, and delete subnets, but you cannot delete any subnets if IP addresses are in use.
3. Click **Update** to update the pool and return to the list view.

### Delete IPv4 Pool

You can delete IP pools as long as none of the IP addresses within the pool are in use.

1. Either from the list view (Resources > IP Pools) or the details view, click the **Delete** button for the pool to delete.
2. Click **Delete** to delete the pool and return to the list view.

## IPv6 Pools (Resources)


### IN THIS SECTION

- [IPv6 Pool Overview | 398](#)
- [Create IPv6 Pool | 399](#)
- [Edit IPv6 Pool | 399](#)
- [Delete IPv6 Pool | 399](#)

# IPv6 Pool Overview

To use IPv6 addressing, you must update the ["fabric addressing policy" on page 603](#) to enable IPv6 in the blueprint (at Staged > Policies > Fabric Addressing Policy). IPv6 is supported on EVPN L2 deployments and L3 deployments. Full feature parity for IPv6 across vendors is not available. Refer to the applicable Apstra Feature Matrix (["4.0.2" on page 853](#), ["4.0.1" on page 866](#), or ["4.0.0" on page 880](#)) for details.

When you're building your blueprint you'll specify which resource pool to use for assigning IP addresses.



**NOTE:** If you need to assign a specific IP address to a specific device, you can assign the IP address individually from the staged blueprint in the **Properties** panel of a selection.

IP pools include the following details:

**Table 25: IPv4 Pool Parameters**

Name	Description
Pool Name	To identify the resource pool
Total Usage	Percentage of IPv6 addresses in use for all subnets in the resource pool. (Hover over the status bar to see the number of IPv6 addresses in use and the total number of IPv6 addresses in the pool.)
Per Subnet Usage	The IPv6 addresses included in the subnet and the percentage that are in use. (Hover over the status bar to see the number of IPv6 addresses in use and the total number of IPv6 addresses in that subnet.)
Status	Indicates if the pool is in use

From the left navigation menu, navigate to **Resources > IPv6 Pools** to go to IPv6 pools in the design (global) catalog. The pool fc01:a05:fab::/48 is predefined. You can create, clone, edit and delete IPv6

pools.

The screenshot displays the Juniper Apstra IPv6 Pools management interface. The left navigation menu is open, showing the 'Resources' section with a sub-menu for 'IPv6 Pools' highlighted. A red arrow labeled '1.' points to the 'Resources' menu item, and another red arrow labeled '2.' points to the 'IPv6 Pools' sub-item. The main content area shows a table of IPv6 pools with columns for Name, Total Usage, Per Subnet Usage, Status, and Actions. A 'Create IPv6 Pool' button is visible in the top right.

Name	Total Usage	Per Subnet Usage	Status	Actions
fd97:bb81:862b:dc0d::/64	0%	0%	● NOT IN USE	Edit Clone Delete
fc01:a05:fab::/48	0%	0%	● NOT IN USE	Edit Clone Delete

## Create IPv6 Pool

1. From the left navigation menu, navigate to **Resources > IPv6 Pools** and click **Create IPv6 Pool**.
2. Enter a name and valid subnet. To add another subnet, click **Add a Subnet** and enter a subnet.
3. Click **Create** to create the pool and return to the list view.

When you've created the blueprint, you'll ["assign resources" on page 420](#) from these pools in the **Staged > Virtual** view of the blueprint.

## Edit IPv6 Pool

1. Either from the list view (Resources > IPv6 Pools) or the details view, click the **Edit** button for the pool to edit.
2. Make your changes. You can add, change, and delete subnets, but you cannot delete any subnets if IP addresses are in use.
3. Click **Update** to update the pool and return to the list view.

## Delete IPv6 Pool

You can delete IP pools as long as none of the IP addresses within the pool are in use.

1. Either from the list view (Resources > IPv6 Pools) or the details view, click the **Delete** button for the pool to delete.
2. Click **Delete** to delete the pool and return to the list view.

# Blueprints

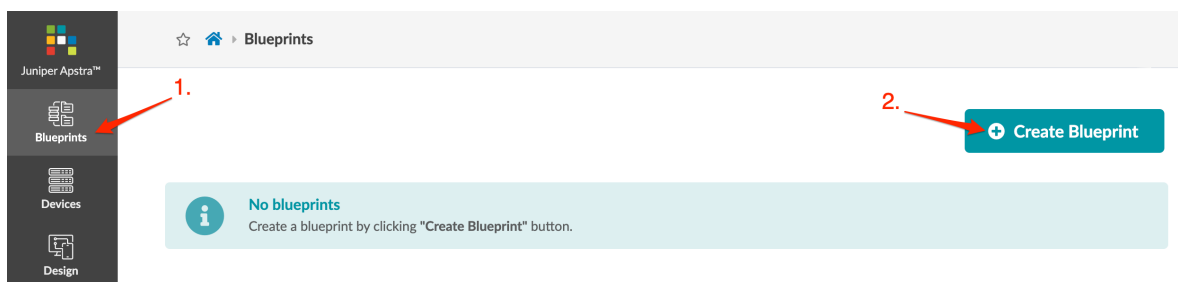
## IN THIS SECTION

- [Create Blueprint | 400](#)
- [Blueprint Dashboard | 401](#)
- [Delete Blueprint | 402](#)

## Create Blueprint

Before creating a blueprint, make sure that the global catalog includes a ["template" on page 110](#) (Design > Templates) that meets the structure and policy needs of your network.

1. From the left navigation menu in the Apstra GUI, click **Blueprints**, then click **Create Blueprint**.



2. Enter a name and select a template from the **Template** drop-down list. A preview shows template parameters, topology preview, structure, external connectivity, and policies.
3. Click **Create** to create the blueprint and return to the blueprint summary view.

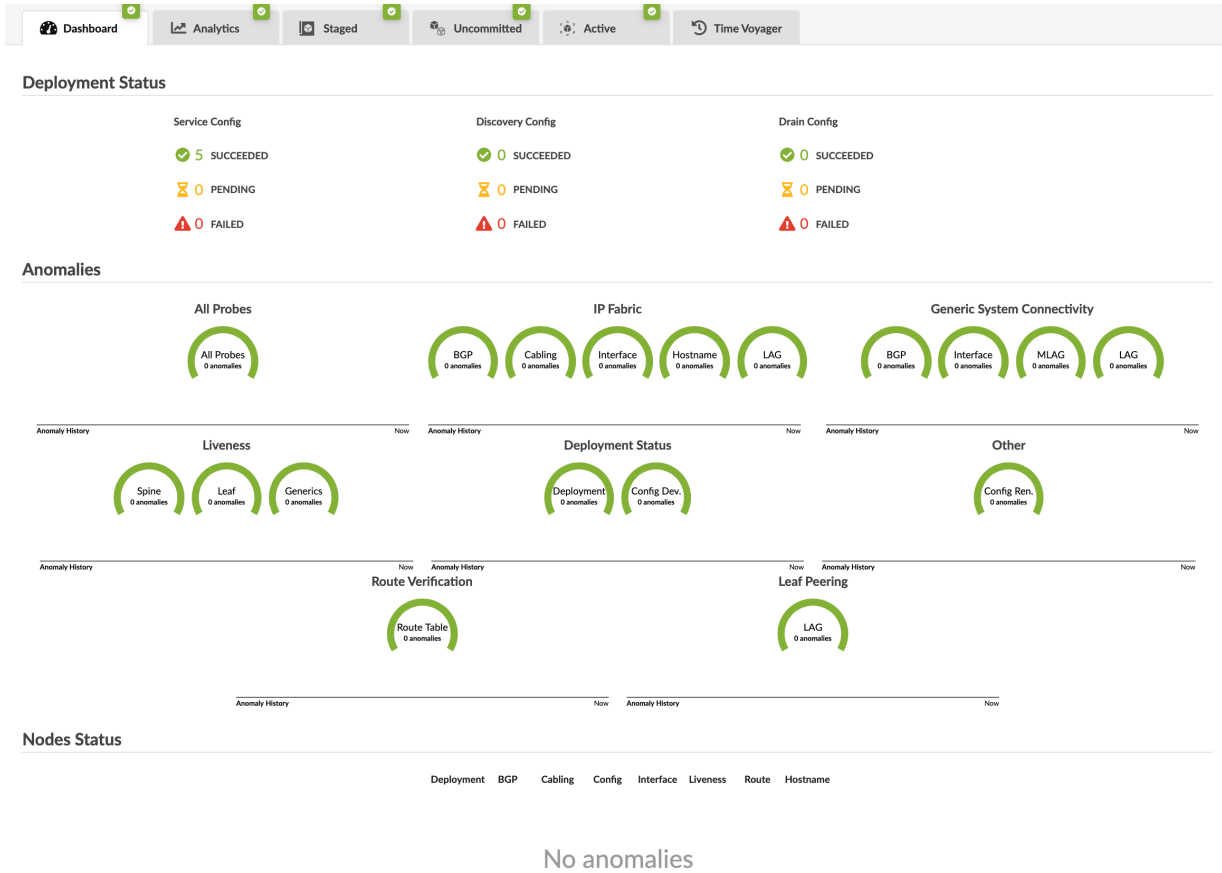
## Blueprint Dashboard

From the left navigation menu in the Apstra GUI, click **Blueprints**, then click the name of a blueprint to go to its dashboard.

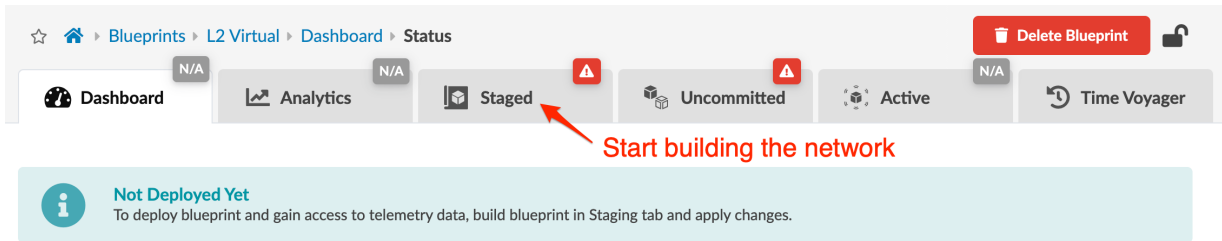
The screenshot shows the Juniper Apstra GUI interface. On the left is a dark navigation bar with icons for Blueprints, Devices, Design, Resources, External Systems, Platform, Favorites, and a user profile. The main content area has a header with a home icon and 'Blueprints'. Below this is a search bar with 'Query: All'. To the right is a 'Create Blueprint' button and pagination controls showing '1-1 of 1' and 'Page Size: 25'. The main content area displays a list of blueprints. The first entry is 'rack-based-blueprint-9c6f1fce' with a sub-label 'L3 Clos'. Below this entry is a table showing various status metrics.

rack-based-blueprint-9c6f1fce	
L3 Clos	
Structure:	2 spines, 3 leafs, 3 generic systems
Analytics	
Deployment Status	N/A
Service Anomalies	N/A
Probe Anomalies	N/A
Root Causes:	N/A
Version 60      Last modified a few seconds ago	

The blueprint dashboard shows the overall health and status of a blueprint. Statuses are indicated by color: green for succeeded, yellow for pending, and red for failed. The deployment status section includes deployment statuses for service config, discovery config, and drain config. The anomalies section includes statuses for all probes, IP fabric, generic system connectivity, liveness, deployment status, route verification, leaf peering, and other. You can have ["analytics dashboards"](#) on [page 403](#) appear on the main blueprint dashboard. The nodes status section includes statuses for deployment, BGP, cabling, config, interface, liveness, route, and hostname.



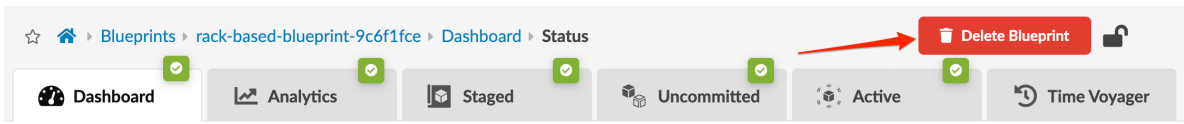
After you create your blueprint, It's time to "build" on page 420 your network in the staging area.



## Delete Blueprint

To be able to delete a blueprint you must have permission (in the user roles you are assigned).

1. From the blueprint, click **Dashboard**, then click **Delete Blueprint** (top-right).



2. Enter the blueprint name, then click **Delete** to delete the blueprint and go to the blueprint summary view.

## Analytics

### IN THIS SECTION

- [Analytics Overview | 403](#)
- [Analytics Dashboard | 404](#)
- [Configure Auto-Enabled Dashboards | 405](#)
- [Instantiate Predefined Dashboard | 405](#)
- [Create Analytics Dashboard | 406](#)
- [Edit / Delete Dashboard | 406](#)
- [Anomalies \(Analytics\) | 407](#)
- [Widgets Overview | 407](#)
- [Create Anomaly Heat Map Widget | 408](#)
- [Create Stage Widget | 409](#)
- [Edit / Delete Widget | 410](#)
- [Probes | 410](#)
- [Instantiate Predefined Probe | 416](#)
- [Create Probe | 417](#)
- [Import / Export Probe | 417](#)
- [Edit / Delete Probe | 418](#)

### Analytics Overview

Managed devices generate large amounts of data over time. On their own these data are voluminous and unhelpful. With Intent-Based Analytics (IBA) you can combine intent from the ["graph" on page 1066](#) with current and historic data from devices to reason about the network at-large.

Data generated by devices are ingested via ["agents" on page 184](#) and sent to the Apstra server. With the use of IBA ["probes" on page 410](#), data can be aggregated across devices in response to operator

configuration. Combining probes with intent from the blueprint graph generates a reduced set of data that can be more easily reasoned about. You can directly inspect advanced data from the Apstra GUI or from ["REST API" on page 755](#) to gain real-time insight about the network. It can also be streamed out with our existing streaming infrastructure. Also, based on the state of this advanced data, ["anomalies" on page 407](#) can be raised.

While operating IBA at scale, using many probes, disk usage can grow significantly within the Apstra server VM. This is expected because the system will persist at least enough samples to maintain data for the requested duration for all time-series for all existing probes. Additionally, the system will create checkpoint (backup) files up to a configured limit. Settings in the [/etc/aos/aos.conf](#) file indicate how often to rotate logs and remove old checkpoint files. Using IBA can increase disk usage to tens of gigabytes. If this is an issue, you can adjust the log rotation settings to reduce disk usage.

Additional space may be used by system snapshots and old images from any in-place Apstra server upgrades. These can be deleted or moved off the system to increase free disk space.

## Analytics Dashboard

Analytics dashboards monitor the network and raise alerts to anomalies. Specific dashboards are automatically created and enabled based on the state of the ["active" on page 654](#) (operational) blueprint. You can also instantiate predefined dashboards and create your own.

Some other characteristics of analytics dashboards include:

- You cannot configure the trigger logic that determines when dashboards are auto-created, but you can create/instantiate your own dashboards.
- Probes that you've created and not modified are reused instead of creating duplicates of those probes.
- ["Widgets" on page 407](#) within each dashboard monitor different aspects of the network and raise alerts to relevant anomalies.
- When you enable a dashboard, the required probes and widgets are instantiated. If you update or delete associated probes and/or widgets, the dashboard may enter an invalid state. Invalid dashboards are not automatically repaired.
- You can display analytics dashboards on the ["blueprint dashboard" on page 400](#) to have additional network information on one screen. To add them, turn **ON** the analytics dashboards' default toggles.
- When upgrading the controller, the auto-creation behavior of dashboards occurs on preexisting active blueprints, in the same way as for newly-created blueprints.

From the blueprint, navigate to **Analytics > Dashboards** to go to the analytics dashboard. You can create, clone, edit, and delete analytics dashboards. System-generated dashboards are labeled with **System** and user-generated (and user-modified) dashboards are labeled with the user's name. Dashboards can be shown in various levels of detail by choosing a **Display mode** (summary, preview, expanded).

1. Analytics

2. Dashboards

Configure Auto-Enabled Dashboards Create Dashboard

Display mode: Summary

1-3 of 3

Click dashboard name for details

Name	Widgets	Updated By	Default	Actions
Device Health Summary	3	System an hour ago	OFF	Edit Clone Delete
Device Traffic Hotspots	1	System an hour ago	OFF	Edit Clone Delete
Throughput Health MLAG	3	System an hour ago	OFF	Edit Clone Delete

Default analytics dashboard will be shown on the blueprint's dashboard.

## Configure Auto-Enabled Dashboards

Certain auto-enabled dashboards generate anomalies that are expected, so you may not want to see them. To suppress these anomalies, either proactively set the auto-enable toggle for the dashboard to **OFF**, or delete the dashboard after it has been enabled. Once a dashboard is disabled it won't be re-enabled unless the auto-enable toggle is set back to **ON** and the respective trigger is satisfied.

1. From the blueprint, navigate to **Analytics > Dashboards** and click **Configure Auto-Enabled Dashboards**. Dashboards are listed with their descriptions, widgets used, and toggles for auto-enablement.
2. Toggle the dashboards **ON** to auto-enable them or **OFF** to disable auto-generation.

## Instantiate Predefined Dashboard

Several predefined dashboards are available that can be instantiated and modified to show analytics in multiple ways. You can instantiate more than one instance of any predefined dashboard.

1. From the blueprint, navigate to **Analytics > Dashboards**, click **Create Dashboard**, then select **Instantiate Predefined Dashboard** from the drop-down list.
2. Select a predefined dashboard from the drop-down list. For more information about predefined dashboards, see "[Predefined Dashboards](#)" on page 919 in the References section.
3. Click **Create** to instantiate the dashboard and return to the list view.

## Create Analytics Dashboard

Some probes and dashboards are automatically created to give you immediate value. The probes auto-adjust based on the state of the blueprint (examples: undeployed or unassigned device, addition or removal of virtual infra managers). You can also create your own dashboards to display custom information from IBA probes and stages.

1. From the blueprint, navigate to **Analytics > Dashboards**, click **Create Dashboard**, then select **New Dashboard** from the drop-down list.
2. Enter a name and (optional) description.
3. Select a layout (one-column, two-column, three-column) and if you want the dashboard to appear on the blueprint **Dashboard** tab, toggle on **Default**.
4. Add and/or create "[widgets](#)" on page 407 to include in the dashboard.
5. Click **Create Dashboard** to create the dashboard and return to the list view.

A large dashboard may take some time to create. You can monitor the status at the bottom of the screen under **Active Tasks**.

## Edit / Delete Dashboard

### IN THIS SECTION

- [Edit Dashboard | 406](#)
- [Delete Dashboard | 407](#)

### Edit Dashboard

Auto-enabled dashboards can be modified, although defaults should work in most cases.

1. From the blueprint, navigate to **Analytics > Dashboards** and click the **Edit** button for the dashboard to edit.
2. Make your changes by creating, adding, editing and/or deleting widgets.
3. Click **Update** to change the dashboard and return to the list view.

## Delete Dashboard

If you delete an auto-created dashboard (because it does not apply to your network for example), the auto-creation feature is disabled so it does not reappear automatically. If you want to re-establish the dashboard you can instantiate it manually.

1. From the blueprint, navigate to **Analytics > Dashboards** and click the **Delete** button for the dashboard to delete.
2. To have all referenced widgets and probes that are exclusively referenced from this dashboard to be deleted for you so you don't have to delete them manually, check the checkbox. Deleting unnecessary widgets and probes frees up resources.
3. Click **Delete Dashboard** to delete the dashboard and return to the list view.

## Anomalies (Analytics)

From the blueprint, navigate to **Analytics > Anomalies** to go to the list of any anomalies found by IBA probes. You can search for specific anomalies by filtering **Probe Label**, **Stage Name**, and **Tags** in the **Query** box.

To display a condensed view of the anomaly count per probe/stage, check the **Group by stage** checkbox. Example: If three stages of the first of two probes are generating anomalies, and two stages of the second probe are generating anomalies, **Group by Stage** shows five entries in a table, each one representing one stage with anomalies.

**NOTE:** The "blueprint dashboard" on page 400 shows a summary of all anomalies including those generated by IBA probes. Clicking the **All Probes** gauge on the dashboard takes you to a list of anomalies (Analytics > Anomalies).

## Widgets Overview

Widgets generate data that are based on IBA "probes" on page 410. The type of widget determines whether it returns a total count of a particular type of anomaly, or displays outputs generated from

stages and processors in an IBA probe. Some widgets are created automatically (but they are not deleted automatically). Widgets can be viewed by themselves or they can be added to "analytics dashboards" on [page 403](#). You can create widgets before or during dashboard creation.

From the blueprint, navigate to **Analytics > Widgets** to go to the widgets list view. You can create, clone, edit and delete widgets.

The screenshot shows the 'Analytics > Widgets' page. At the top, there is a navigation bar with tabs: Dashboard, Analytics, Staged, Uncommitted, Active, and Time Voyager. Below this, there is a sub-navigation bar with: Dashboards, Anomalies, Widgets, and Probes. A red arrow labeled '1.' points to the 'Analytics' tab, and another red arrow labeled '2.' points to the 'Widgets' tab. On the right side, there is a 'Create Widget' button. Below the navigation bar, there are filters: 'Show widget contents' and 'Anomalies Only'. A pagination bar shows '1-7 of 7' and 'Page Size: 25'. The main content area is a table with columns: Name, Type, Properties, Updated by, and Actions. A red arrow points to the 'Name' column header with the text 'Click widget name for details'. The table contains one row with the widget name 'Fabric ECMP Imbalance', Type 'Stage', and Properties 'ECMP Imbalance (Fabric Interfaces) / system\_imbalance\_count'. The 'Updated by' column shows 'System 2 hours ago'. The 'Actions' column has links for 'Edit', 'Clone', and 'Delete'.

## Create Anomaly Heat Map Widget

Anomaly heatmap widgets count the anomalies from tagged IBA probes and stages.

1. From the blueprint, navigate to **Analytics > Widgets** and click **Create Widget**.
2. Select **Anomaly Heat Map** from the **Type** drop-down list and enter a name.
3. Enter row tags, column tags, and (optional) description.
4. Click **Create** to create the widget and return to the list view.

Creating a large widget may take some time. You can monitor the status under the **Active Tasks** section at the bottom of the screen.

## Create Stage Widget

### IN THIS SECTION

- [Create Stage Widget from Widgets View | 409](#)
- [Create Stage Widget from Probes View | 409](#)

### Create Stage Widget from Widgets View

Stage widgets contain outputs from IBA probe stages.

1. From the blueprint, navigate to **Analytics > Widgets** and click **Create Widget**.
2. Select **Stage** from the **Type** drop-down list and enter a name.
3. Select a probe and a stage, then customize the output as needed.
4. Click **Create** to create the widget and return to the list view.

Creating a large widget may take some time. You can monitor the status under the **Active Tasks** section at the bottom of the screen.

### Create Stage Widget from Probes View

You can create a widget from the details view of a probe.

1. From the blueprint, navigate to **Analytics > Probes** and select a probe.
2. Select a stage within the probe and click the **Create dashboard widget** button (right-side). The stage is preselected for you in the dialog that appears.
3. Configure the parameters as needed.
4. Click **Create** to create the widget and return to the detail view of the probe. The widget appears in the widgets list view (Analytics > Widgets) and when you create or update an analytics dashboard, the new widget appears as an option.

## Edit / Delete Widget

### IN THIS SECTION

- [Edit Widget | 410](#)
- [Delete Widget | 410](#)

### Edit Widget

Auto-created widgets can be modified, although defaults should work in most cases. When you change widgets, any dashboards that they're used in are affected.

1. From the blueprint, navigate to **Analytics > Widgets** and click the **Edit** button for the widget to edit.
2. Make your changes.
3. Click **Update** to stage the changes and return to the list view.

### Delete Widget

A widget that is used in a dashboard cannot be deleted.

1. From the list view (Analytics > Widgets) or the details view, click the **Delete** button for the widget to delete.
2. Click **Delete Widget** to stage the deletion and return to the list view.

## Probes

### IN THIS SECTION

- [IBA Probes Overview | 411](#)

## IBA Probes Overview

### IN THIS SECTION

- [Processors | 411](#)
- [Ingestion Filters | 412](#)
- [IBA Collection Filter | 412](#)
- [IBA Filter Format | 413](#)

Probes are the basic unit of abstraction in Intent-Based Analytics. Generally, a given probe consumes some set of data from the network, does various successive aggregations and calculations on it, and optionally specifies some conditions of said aggregations and calculations on which anomalies are raised.

Probes are Directed Acyclic Graphs (DAGs) where the nodes of the graph are processors and stages. Stages are data, associated with context, that can be inspected by the operator. Processors are sets of operations that produce and reduce output data from input data. The input to processors are one-or-many stages, and the output from processors are also one-or-many stages. The directionality of the edges in a probe DAG represent this input-to-output flow.

Importantly, the initial processors in a probe are special and do not have any input stage. They are notionally generators of data. We shall refer to these as source processors.

IBA works by ingesting raw telemetry from collectors into probes to extract knowledge (ex: anomalies, aggregations etc.). A given collector publishes telemetry as a collection of metrics, where each metric has identity (viz, set of key-value pairs) and a value. IBA probes, often with the use of graph queries, must fully specify the identity of a metric to ingest its value into the probe. With this feature, probes can ingest metrics with partial specification of identity using ingestion filters, thus enabling ingestion of metrics with unknown identities.

Some probes are created automatically. These probes will not be deleted automatically. This keeps things simple operationally and implementation-wise.

### Processors

The input processors of a probe handle the required configuration to ingest raw telemetry into the probe to kickstart the data processing pipeline. For these processors, the number of stage output items (one or many) is equal to the number of results in the specified graph query(s). If multiple graph queries are specified, for example. `graph_query: [A, B]`, and query A matches 5 nodes and query B matches 10

nodes, results of query A will be accessible using `query_result` indices from 0 to 4, and results of query B using indices from 5 to 14.

If a processor's input type and/or output type is not specified, then the processor takes a single input called **in**, and produces a single output called **out**.

Some processor fields are called **expressions**. In some cases, they are **graph queries** and are so noted. In other cases, they are Python **expressions** that yield a value. For example, in the Accumulate processor, duration may be specified as integer with seconds, for example 900, or as an expression, for example `60 * 15`. However, expressions could be more useful: there are multiple ways to parametrize them.

Expressions support string values. Processor configuration parameters that are strings and support expressions should use special quoting when specifying static value. For example, `state: "up"` is not valid because it'll refer to the variable "up", not a static string, so it should be: `state: "up"`.

An expression is always associated with a graph query and is run for every resulting match of that query. The execution context of the expression is such that every variable specified in the query resolves to a named node in the associated match result. For more information, see ["Service Data Collector" on page 1034](#) example.

Graph-based processors have been extended with `query_tag_filter` allowing the ability to filter graph query results by tags (new in version 4.0). In IBA probes, tags are used only as filter criteria for servers and external routers, specifically for the ECMP Imbalance (External Interfaces) probe and the Total East/West Traffic probe. For specific processor information, see ["Probe Processors" on page 993](#) in the References section.

## Ingestion Filters

With "ingestion filters" one query result can ingest multiple metrics into a probe. Table data types are used to store multiple metrics as part of a single stage output item. Table data types include `table_ns`, `table_dss`, `table_ts` - to correspond to existing types - `ns`, `dss`, `ts` -respectively.

## IBA Collection Filter

Collection filters determine the metrics that are collected from the target devices.

A collection filter for a given collector on a given device, is simply a collection of ingestion filters present in different probes. You can also specify it as part of enabling a service outside the context of IBA or probes but existing precedence rules for service enablement apply here - only filters at a given precedence level are aggregated. When multiple probes specify an ingestion filter targeting a specific service on a specific device, the metrics collected are a union - in other words, a metric is published when it matches any of the filters. This is why, the data is also filtered by the controller component prior to ingesting into the IBA probes.

This filter is evaluated by telemetry collectors, often to better control even what subset of available metrics is fetched from the underlying device operating system. For example, to fetch only a subset of routes instead of getting all routes which can be a huge number. In any case, only the metrics matching the collection filter are published as the raw telemetry.

As part of enabling a service on a device, you can now specify collection filters for services. This filter becomes an additional input provided to collectors as part of "self.service\_config.collection\_filters".

## IBA Filter Format

Following are the design/usability goals for filters (ingestion and collection)

1. Ease of authoring - given probe authors are the ones specifying it
  - Most often cases are match any, match against a given list of possible values, equality match, range check if key has numeric values.
2. Efficient evaluation - given the filters are evaluated in the hot paths of collection or ingestion.
3. Aggregatable - multiple filters are aggregated so this aggregation logic need not become the responsibility of individual collectors.
4. Programming language neutral - components operating on filters can be in Python or C++ or some other language in future.
5. Programmable - be amenable to future programmability around the filters, by the controller itself and/or collectors, to enhance things like usability, performance etc.

Considering the above goals, following is a suggested and illustrative schema for filter1. Refer to ingestion filter sections for specific examples to understand this better.

```
FILTER_SCHEMA = s.Dict(s.Object(
  'type': s.Enum(['any', 'equals', 'list', 'pattern', 'range', 'prefix']),
  'value': s.OneOf({
    'equals': s.OneOf([s.String(), s.Integer()]),
    'list': s.List(s.String(), validate=s.Length(min=1)),
    'pattern': s.List(s.String(), validate=s.Length(min=1)),
    'range': s.AnomalyRange(), validate=s.Length(min=1),
    'prefix': s.Object({
      'prefixsubnet': s.Ipv6orIpv4NetworkAddress(),
      'ge_mask': s.Optional(s.Integer()),
      'le_mask': s.Optional(s.Integer()),
      'eq_mask': s.Optional(s.Integer())
    })
  })
})
```

```
), key_type=s.String(description=
    'Name of the key in metric identity. Missing metric identity keys are '
    'assumed to match any value'))
```

One instance of filter specification is interpreted as **AND** of all specified keys (aka per-key constraints). Multiple filter specifications coming from multiple probes are considered as **OR** at the filter level.

**NOTE:** The schema presented here is only for communicating the requirements and engineering is free to choose any way that accomplishes stated use cases.

Collector Processors `additional_properties` specified in collector processors' configuration can be accessed using the special context. `namespace`. For example, if a collector defines property `system_role`, it could be used this way:

```
duration: 60 * (15 if context.system_role == "leaf" else 10)
```

**NOTE:** Items context is available as long as the items set is unchanged from the original set derived from the collector processor configuration. After data goes through a processor that changes this set, for example any grouping processor, it's no longer available.

From the blueprint, navigate to **Analytics > Probes** to go to the probes list view. To go to a probe's details, click its name. You can instantiate, create, clone, edit, delete, import, and export probes.

The screenshot shows the 'Analytics > Probes' interface. At the top, there are tabs: Dashboard, Analytics, Staged, Uncommitted, Active, and Time Voyager. Below these, there are sub-tabs: Dashboards, Anomalies, Widgets, and Probes. A red arrow labeled '1.' points to the 'Analytics' tab, and another red arrow labeled '2.' points to the 'Probes' sub-tab. To the right of the 'Probes' sub-tab, there is a 'Click for options' label pointing to a 'Create Probe' button. A dropdown menu is open below the 'Create Probe' button, showing options: 'New Probe', 'Instantiate Predefined Probe', and 'Import Probes'. Below the navigation area, there is a search bar with the text 'Query: All'. At the bottom, there is a table of probes. A red arrow labeled 'Click probe name to see details' points to the 'Device System Health' probe name. The table has columns: Name, Anomalies, State, Updated By, Tags, Enabled, and Actions. The 'Device System Health' probe is shown with 'No anomalies', 'Operational' state, 'System' updated 2 hours ago, and 'Enabled' toggle. The 'Actions' column for this probe shows 'Edit', 'Clone', and 'Delete' buttons.

Name	Anomalies	State	Updated By	Tags	Enabled	Actions
Device System Health	No anomalies	Operational	System 2 hours ago		ON	Edit Clone Delete

You can display stages in some probes in various ways. For example, when you click the probe named **Device Traffic**, you'll see the image below. Changing the data source for **Average Interface Counters**

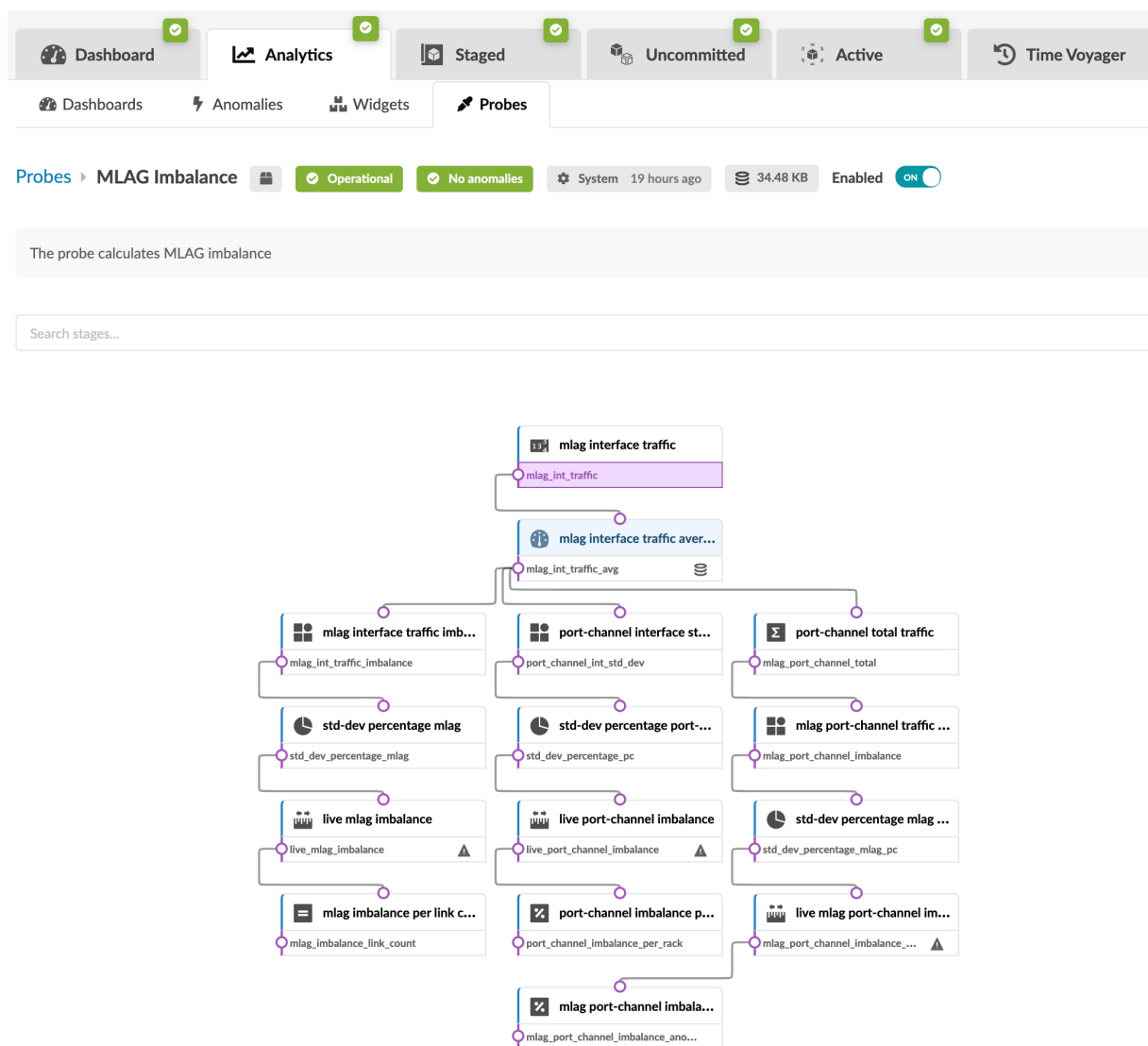
from **Real Time** to **Time Series** gives you the option to view the time series as separate graphs, combined graphs: linear or combined graphs: stacked (as of Apstra version 4.0). Also, you can see the disk space used on each probe, as applicable.



**CAUTION:** If the Apstra controller has insufficient disk space, older telemetry data files are deleted. You can increase capacity to retain older telemetry data by clustering multiple virtual machines. For more information, see ["Apstra Cluster" on page 728](#).

The structure and logic of non-linear probes with tens of processors is not easily distinguished in the standard view. You can click the expand button (top of left panel) to see an expanded representation of how the processors are inter-related (new in version 4.0). For example, the image below shows the

expanded view of the **MLAG Imbalance** probe.



## Instantiate Predefined Probe

1. From the blueprint, navigate to **Analytics > Probes**, then click **Create Probe** and select **Instantiate Predefined Probe** from the drop-down list. For information on specific ["predefined probes"](#) on page 921 see the References section.
2. Select a predefined probe from the drop-down list.
3. Configure the probe to suit your anomaly detection requirements.
4. Click **Create** to instantiate the probe and return to the list view.

## Create Probe

1. From the blueprint, navigate to **Analytics > Probes**, click **Create Probe**, then select **New Probe**.
2. Enter a name and (optional) description.
3. To be able to filter by your own defined categories, enter tag(s).
4. Probes are enabled by default. This means that data is collected and processed (potentially creating anomalies) as soon as the probe is created. To disable the probe, toggle off **Enabled**. When you are ready to start collecting and processing data, you can edit the probe to enable it.
5. Click **Add Processor**, select a processor type, then click **Add** to add the processor to the probe. For more information about individual processors, see ["Probe Processors" on page 993](#) in the References section.
6. Customize inputs and properties as appropriate, or leave defaults as is.
7. Repeat the previous two steps until you've added all required processors for the new probe.
8. Click **Create** to create the probe and return to the list view.

## Import / Export Probe

### IN THIS SECTION

- [Import Probe | 417](#)
- [Export Probe | 417](#)

### Import Probe

1. From the blueprint, navigate to **Analytics > Probes**, then click **Create Probe** and select **Import Probes** from the drop-down list.
2. Either click **Choose Files** and navigate to the file(s) on your computer, or drag and drop the file(s) from your computer into the dialog window.
3. Click **Import** to import the probe and return to the list view.

### Export Probe

1. From the blueprint, navigate to **Analytics > Probes**, then click the name of the probe to export.
2. Click the **Export** button (top-right) to see a preview of the file that will be exported.
3. To copy the contents, click **Copy**, then paste it.

4. To download the JSON file to your local computer, click **Save as File**.
5. When you've copied and/or downloaded the file, click the **X** to close the dialog.

## Edit / Delete Probe

### IN THIS SECTION

- [Edit Probe | 418](#)
- [Delete Probe | 418](#)

### Edit Probe

If a widget is using a probe, editing the probe affects those widget(s) and related dashboard(s).

1. From the list view (Analytics > Probes) or the details view, click the **Edit** button for the probe to edit.
2. Make your changes.
3. Click **Update** to stage the changes and return to the list view.

### Delete Probe

**NOTE:** You can also use REST API to work with IBA probes. Navigate to **Platform > Developers** for REST API documentation and tools.

If a widget is using a probe, you can't delete the probe.

1. From the list view (Analytics > Probes) or the details view, click the **Delete** button for the probe to delete.
2. Click **Delete Probe** to stage the deletion and return to the list view.

# Staged (Blueprints)

## IN THIS SECTION

- [Build \(Physical\) | 419](#)
- [Topology \(Staged\) | 430](#)
- [Nodes \(Staged\) | 446](#)
- [Links \(Staged\) | 468](#)
- [Racks \(Staged\) | 477](#)
- [Pods \(Staged\) | 481](#)
- [Planes \(Staged\) | 485](#)
- [Virtual | 487](#)
- [Policies | 583](#)
- [Catalog | 609](#)
- [Tasks | 624](#)
- [Connectivity Templates | 625](#)
- [Find by Tags | 647](#)

When you assign or change resources and devices in a blueprint they are visible in the staged view. You can view progressive changes here before you push them to the active blueprint. This staging area allows you to validate that the pending changes are compliant with the intent, and that they work together with available resources and devices before you deploy the network.

## Build (Physical)

## IN THIS SECTION

- [Stage Physical Resources | 420](#)
- [Stage Device Profiles | 422](#)
- [Stage Devices | 423](#)

- [Stage Configlets | 429](#)

## Stage Physical Resources

### IN THIS SECTION

- [Update Physical Resource Assignments | 420](#)
- [Reset Physical Resource Group Overrides | 421](#)

You can assign resources, release previously used resources and go to resource pool management. The resource assignment section has a convenient shortcut button, **Manage resource pools**, that takes you to resource pool management. From there, you can monitor resource usage and create additional resource pools, as needed.

### Update Physical Resource Assignments

1. From the blueprint, navigate to **Staged > Physical > Build > Resources**. (The build panel is on the right side.)

The screenshot displays the 'Staged' tab of a network management system. The interface includes a top navigation bar with tabs: Dashboard, Analytics, Staged, Uncommitted, Active, and Time Voyager. Below this is a secondary navigation bar with options: Physical, Virtual, Policies, Catalog, Tasks, and Connectivity Templates. A 'Find by tags' search bar is on the right. The main content area shows a topology view with nodes and links. A 'Layer' dropdown is set to 'Build: ASNs - Spines'. A 'Selection' panel on the right shows a list of resource pools with status indicators (green for assigned, red for not assigned). The 'ASNs - Spines' pool is highlighted, showing a red status indicator and a warning message: 'Removing existing pools may result in changes in resource assignments.' Below this, a table lists resource pools: 'ASNs - Generics' (1/1), 'ASNs - Leafs' (3/3), and 'Loopback IPs - Spines' (0/2). Red arrows and numbers 1 through 6 indicate the following steps: 1. Click the 'Staged' tab. 2. Click the 'Physical' tab. 3. Click the 'Build' button. 4. Click the 'Resources' button. 5. Click the 'Update assignments' button. 6. Click the 'Manage resource pools' button. A red text box at the bottom of the screenshot states: 'When resources are staged, status indicator turns green'.

2. Red status indicators mean that resources need to be assigned. Click a red status indicator, then click the **Update assignments** button.
3. Select a pool from which to pull the resources, then click the **Save** button. The required number of resources are automatically assigned to the resource group. When the red status indicator turns green, the resource assignment has been successfully staged.

**NOTE:** You can also assign resources on a per-device basis (especially useful if you have a predefined resource mapping). Select the device from the topology view or nodes view, then assign the resource from the **Properties** section of the **Selection** panel (right-side). (This is also where you can see the specific resource that was assigned from a resource pool.)

### Reset Physical Resource Group Overrides

Certain blueprint operations require resource allocations to be retained even when you've removed a device from a blueprint. Overridden resource groups re-use previously allocated resources when a device is re-used. For example, if you've deleted a rack, then you rollback to a version with that rack, the same resources must be used. Otherwise, the topology would change (for example, it might have different IP addresses). In the case of a revert operation, the originally assigned resources appear in the



3. Select the appropriate interface map from the drop-down list for each node. Or, to assign the same interface map to multiple nodes, you can select the ones that use the same interface map (or all of them with one click), then select the interface map from the drop-down list located above the selections, and click **Assign Selected**.

Query: All 1-5 of 5 Page Size: 25

Interface Map  
Arista\_vEOS\_\_slicer-4x10-1 ✕ **Assign Selected**

2 selected

Name	Interface Map	Device Profile
<input checked="" type="checkbox"/> rack_1_001_leaf1	Select...	N/A
<input checked="" type="checkbox"/> rack_1_001_leaf2	Select...	N/A
<input type="checkbox"/> rack_2_001_leaf1	Select...	N/A
<input type="checkbox"/> spine1	Select...	N/A
<input type="checkbox"/> spine2	Select...	N/A

Or select one at a time from these drop-down lists

**Update Assignments**

4. Click **Update Assignments**. When the red status indicator turns green, the device profile assignments have been successfully staged.

## Stage Devices

### IN THIS SECTION

- [Stage Device Overview | 423](#)
- [Assign \(Multiple\) System IDs and Deploy Modes | 424](#)
- [Assign \(Single\) System ID | 427](#)
- [Set Deploy Mode \(Single Device\) | 428](#)
- [Change Hostname - One Device | 429](#)

## Stage Device Overview

Devices must have interface maps associated with them before they can have system IDs assigned to them. See the previous section for details. When a device is assigned to a blueprint, it performs

discovery configuration. During this phase all interfaces are changed to L3-only mode allowing interfaces to be *up*. There is no BGP configuration, no routing expectations, nothing that can influence the network. A device in *discovery* mode is benign; it does not participate in the datacenter fabric, and it does not forward any packets through it. You can then perform critical validations of network health including viewing statistics for cabling, LLDP, transceivers and more. Any issues, such as miscabling or physical link errors, cause a telemetry alarm. You can address and correct the anomalies *before* deploying the device.

**NOTE:** When resetting system IDs (serial number) Discovery-1 is re-applied. Before physically uninstalling the agent, it is good practice to fully erase the device configuration and uninstall the device agent.

### Assign (Multiple) System IDs and Deploy Modes

**NOTE:** You can also use AOS CLI to bulk-assign system IDs to devices either with a CSV text file or the blueprint `set-serial-numbers` command.

1. From the blueprint, navigate to **Staged > Physical > Build > Devices**, and click the status indicator for **Assigned System IDs** (if the nodes list is not already displayed). Unassigned devices are indicated in

yellow.

The screenshot shows a network management interface with a topology diagram and a table of assigned system IDs. Red arrows and numbers 1-6 highlight specific actions and elements:

- 1. Arrow pointing to the **Staged** tab in the top navigation bar.
- 2. Arrow pointing to the **Physical** tab in the left sidebar.
- 3. Arrow pointing to the **Build** button in the top right corner.
- 4. Arrow pointing to the **Build: System IDs** dropdown menu.
- 5. Arrow pointing to the **Assigned System IDs** table.
- 6. Arrow pointing to the **Change System ID assignments** button (pencil icon) below the table.

The topology diagram shows a network structure with nodes labeled **sys001**, **spine1**, **spine2**, and racks **rack\_1\_001** and **rack\_2\_001**. The **rack\_2\_001** rack contains a node labeled **rack\_2\_001\_leaf1** which is highlighted in green.

The **Assigned System IDs** table shows the following data:

Node	System ID
spine1	Not assigned
spine2	Not assigned
rack_1_001_leaf1	Not assigned
rack_1_001_leaf2	Not assigned
rack_2_001_leaf1	525400477465
rack_1_001_sys001	Not assigned
rack_2_001_sys001	Not assigned
sys001	Not assigned


2. Click the **Change System IDs assignments** button (below Assigned System IDs) and, for each node, select system IDs from the drop-down list. (If you don't see an expected serial number (system ID),

you may still need to ["acknowledge"](#) on [page 132](#) the device (Devices > Managed Devices.)

Assign Systems

▶ Query: All

1-8 of 8 < >

Name ↕	Role ↕	Hostname ↕	System ID ↕	Deploy Mode ↕
spine1	Spine	spine-1	<div>525400C50088 (10.28.11.13) - some_location ✖ </div>	<div><input checked="" type="radio"/> Deploy</div> <div><input type="radio"/> Ready</div> <div><input type="radio"/> Drain</div> <div><input type="radio"/> Undeploy</div>
spine2	Spine	spine-2	<div>Select... ▼</div>	<div><input type="radio"/> Deploy</div> <div><input type="radio"/> Ready</div> <div><input type="radio"/> Drain</div> <div><input type="radio"/> Undeploy</div>
rack_1_001_leaf1	Leaf	leaf-1-1	<div><div>505400B41BBC (10.28.11.11) - some_location ▼ 505400B41BBC (10.28.11.11) - some_location 5054004B6645 (10.28.11.9) - some_location</div></div>	<div><input type="radio"/> Deploy</div> <div><input type="radio"/> Ready</div> <div><input type="radio"/> Drain</div> <div><input type="radio"/> Undeploy</div>
rack_1_001_leaf2	Leaf	leaf-1-2	<div>Select... ▼</div>	<div><input type="radio"/> Deploy</div> <div><input type="radio"/> Ready</div> <div><input type="radio"/> Drain</div> <div><input type="radio"/> Undeploy</div>

Update Assignments

- 3. When you select a system ID, the deploy mode changes to **Deploy** by default. If you don't want to stage the device to be deployed yet, change the deploy mode here. When you're ready to deploy the device, return here to set the deploy mode back to **Deploy**.
- 4. Click **Update Assignments** to stage the changes. Before the task is completed you can click **Active Tasks** at the bottom of the screen to see its progress.

## Assign (Single) System ID

1. From the blueprint, navigate to **Staged > Physical > Build > Devices**; if you don't see the nodes list, click the status indicator for **Assigned System IDs**.

The screenshot shows the network management interface with the following elements:

- Top Navigation:** Dashboard, Analytics, Staged, Uncommitted, Active, Time Voyager.
- Left Sidebar:** Physical, Virtual, Policies, Catalog, Tasks, Connectivity Templates.
- Filters:** Nodes: All, Links: All, Selection, Build.
- Topology View:** 2D, 3D, Layer: Build: System IDs, Selected Rack: All, Selected Node: All, Expand Nodes?, Show Links?.
- Topology Diagram:** A 2D diagram showing a network topology with nodes labeled sys001, spine1, spine2, rack\_1\_001, and rack\_2\_001.
- Assigned System IDs Panel:** A table listing nodes and their system IDs. The table has columns for Node and System ID. The status 'Not assigned' is shown for all nodes.

Red arrows indicate the navigation path:

1. Staged tab
2. Physical tab
3. Build button
4. Devices icon
5. Assigned System IDs panel
6. Select an unassigned node

2. From the **Assigned System IDs** list, click the name of the node that you want to assign. Device details are displayed (deploy mode, serial number, hostname rendered, incremental and pristine config, as

applicable).

The screenshot displays the network management interface. At the top, there are tabs for Dashboard, Analytics, Staged, Uncommitted, Active, and Time Voyager. Below these are filters for Physical, Virtual, Policies, Catalog, Tasks, and Connectivity Templates. A search bar labeled 'Find by tags' is on the right. The main area shows a topology view with 'Nodes: All' and 'Links: All' filters. A 'Selection' and 'Build' button is present. The 'Topology' tab is active, showing a 2D view. A 'Selected Rack' dropdown shows 'rack\_2\_001' and a 'Selected Node' dropdown shows 'rack\_2\_001\_leaf1 (Leaf)'. A red arrow points to the 'Selected Node' dropdown with the text 'Alternative method for accessing device details'. Below the dropdowns are 'Neighbors' and 'Links' buttons, and 'Add links' and 'Edit links' buttons. A diagram shows 'rack\_2\_001\_leaf1' connected to 'spine1', 'spine2', 'rack\_2\_001\_sys001', and 'sys001'. A 'Show Unused Ports' checkbox is present. On the right, a 'Device' details panel for 'rack\_2\_001\_leaf1' is shown. It includes a 'Deploy Mode' section with a 'not set' value and a 'Deploy' button. A red arrow points to the 'Deploy' button with the text 'Click to assign ID'. Below this is an 'S/N' section with a 'Not assigned' value and an 'Edit' button. Below that is a 'Hostname' section with a value '525400477465' and an 'Edit' button.

**NOTE:** You can also go to these device details by selecting a node name in the **Selected Nodes** drop-down list (left-middle).

3. To assign a system ID, click the **Edit** button for **S/N**, select the system ID from the drop-down list, and click the **Save** button to stage the change. (If you don't see the expected serial number (system ID), you may still need to "[acknowledge](#)" on [page 132](#) the device (Devices > Managed Devices).
4. To remove an existing S/N instead of assigning one, click the **Edit** button for **S/N**, then click the red square to stage the change.

### Set Deploy Mode (Single Device)

**NOTE:** You can also change one or more deploy modes in the same transaction from **Staged > Physical > Nodes**. See "[Set Deploy Mode \(Multiple Devices\)](#)" on [page 464](#) for more information.

1. From the blueprint, navigate to **Staged > Physical > Build > Devices**; if you don't see the nodes list, click the status indicator for **Assigned System IDs**.
2. Click a node name to see device details.
3. Click the **Edit** button for **Deploy Mode** and select a deploy mode.

- **Deploy** - Adds service configuration and puts the device fully in service.
- **Ready** - Adds discovery 2 configuration (hostnames, interface descriptions, port speeds / breakouts). Changing from deploy to ready removes service configuration.
- **Drain** - Takes a device out of service gracefully for maintenance. See ["Draining Device Traffic" on page 152](#) for more information.

**NOTE:** While TCP sessions drain (which could take some time, especially for EVPN blueprints) BGP anomalies are expected. When configuration deployment is complete the temporary anomalies are resolved. To ensure switches are completely drained before undeploying them, you could ["instantiate" on page 403](#) the drain validation dashboard to monitor progress.

- **Undeploy** - Removes Apstra-rendered configuration. If a device is carrying traffic it is best to first put the device into drain mode (and commit the change). When the device is completely drained, proceed to undeploy the device.

#### 4. Click the **Save** button to stage the change.

When you're ready to activate changes, ["Commit" on page 648](#) them from the **Uncommitted** tab.

### Change Hostname - One Device

1. From the blueprint, navigate to **Staged > Physical > Build > Devices**; if you don't see the nodes list, click the status indicator for **Assigned System IDs**.
2. Click a node name to see device details.
3. Click the **Edit** button for **Hostname**, change the name, and click the **Save** button to stage the change.

When you're ready to activate changes, ["Commit" on page 648](#) them from the **Uncommitted** tab.

### Stage Configlets

Configlets are vendor-specific. Apstra software automatically ensures that configlets of a specific vendor are not assigned to devices from a different vendor.

**NOTE:** If you're using a version prior to 4.0, refer to version 3.3.0 documentation for information about staging external routers, which was deprecated in version 4.0.

1. Make sure that the appropriate configlets have been ["imported" on page 615](#) into the blueprint catalog from the global catalog.
2. From the blueprint, navigate to **Staged > Physical > Build > Configlets**.

3. If the configlet has not been imported yet, you can click **Manage Configlets** to import it .
4. Click the status indicator for the configlet. If the configlet uses a property set, click the **Import Property Set** button, select the property set from the drop-down list, then click **Import Property Set**.

## Topology (Staged)

### IN THIS SECTION

- [2D Topology View \(Staged\) | 430](#)
- [3D Topology View \(Staged\) | 432](#)
- [Neighbors View \(Staged\) | 434](#)
- [Links View \(Staged Topology\) | 436](#)
- [Virtual Networks Endpoints \(Staged\) | 437](#)
- [Add Links | 437](#)
- [Add Leaf Peer Links | 441](#)
- [Update Logical Links and LAG Mode | 442](#)
- [Change Link Speeds | 444](#)
- [Delete Link | 445](#)

Topologies can be displayed as 2D or 3D. You can view selections within topologies as neighbors, links, or (as of Apstra version 4.0.1) virtual network endpoints, as applicable.

You can perform the following tasks from the staged topology view:

- ["Add links" on page 437](#)
- ["Add leaf peer links" on page 441](#)
- ["Update logical links and LAG mode" on page 442](#)
- ["Change link speeds" on page 444](#)
- ["Delete links" on page 445](#)

### 2D Topology View (Staged)

From the blueprint, navigate to **Staged > Physical > Topology**. The default view is **2D**.

- To make topology elements larger, click the **Expand Nodes** check box.
- To show the links between elements, click the **Show Links** check box.
- To show node name, hostname (and role and tags as of Apstra version 4.0.1) as applicable, hover over an element.
- To display a different label (name, hostname, S/N), select a different label from the **Topology Label** drop-down list.
- To show rack details, select a rack by either clicking its element or by selecting it from the **Selected Rack** drop-down list.

- To show node details, select the node by either clicking its element in the topology or by selecting it from the **Selected Node** drop-down list.

The screenshot shows the Juniper Apstra interface in the 'Staged' view. The top navigation bar includes 'Dashboard', 'Analytics', 'Staged' (selected), and 'Uncommitted'. Below this is a secondary navigation bar with 'Physical' (selected), 'Virtual', 'Policies', 'Catalog', 'Tasks', and 'Connectivity Templates'. The main area displays a topology view with filters for 'Nodes: All' and 'Links: All'. The 'Topology' tab is active, showing a 2D view. A red arrow labeled '1.' points to the 'Staged' tab. A red arrow labeled '2.' points to the 'Physical' tab. A red arrow labeled '3.' points to the 'Topology' tab. A red arrow points to the 'Selected Rack' dropdown menu, which is set to 'All'. A red arrow points to the 'Selected Node' dropdown menu, which is set to 'All'. A red arrow points to the 'spine2' node in the topology. A red arrow points to the 'rack\_1\_001\_leaf1' node in the 'rack\_1\_001' rack. A red arrow points to the 'rack\_2\_001\_sys001' node in the 'rack\_2\_001' rack. A red arrow points to the 'sys001' node in the topology. A red arrow points to the 'Topology Label' dropdown menu, which is set to 'Name'. A red arrow points to the 'Has Uncommitted Changes' indicator. A red arrow points to the 'Expand Nodes?' and 'Show Links?' checkboxes. A red arrow points to the 'spine1' node in the topology. A red arrow points to the 'spine2' node in the topology. A red arrow points to the 'rack\_1\_001\_sys001' node in the 'rack\_1\_001' rack. A red arrow points to the 'rack\_2\_001\_sys001' node in the 'rack\_2\_001' rack. A red arrow points to the 'rack\_1\_001\_leaf2' node in the 'rack\_1\_001' rack. A red arrow points to the 'rack\_2\_001\_leaf1' node in the 'rack\_2\_001' rack. A red arrow points to the 'sys001' node in the topology. A red arrow points to the 'Topology Label' dropdown menu, which is set to 'Name'. A red arrow points to the 'Has Uncommitted Changes' indicator. A red arrow points to the 'Expand Nodes?' and 'Show Links?' checkboxes. A red arrow points to the 'spine1' node in the topology. A red arrow points to the 'spine2' node in the topology. A red arrow points to the 'rack\_1\_001\_sys001' node in the 'rack\_1\_001' rack. A red arrow points to the 'rack\_2\_001\_sys001' node in the 'rack\_2\_001' rack. A red arrow points to the 'rack\_1\_001\_leaf2' node in the 'rack\_1\_001' rack. A red arrow points to the 'rack\_2\_001\_leaf1' node in the 'rack\_2\_001' rack.

Make selection from drop-down lists, or...

... click selection directly for more info

Rollover an element to see info

### 3D Topology View (Staged)

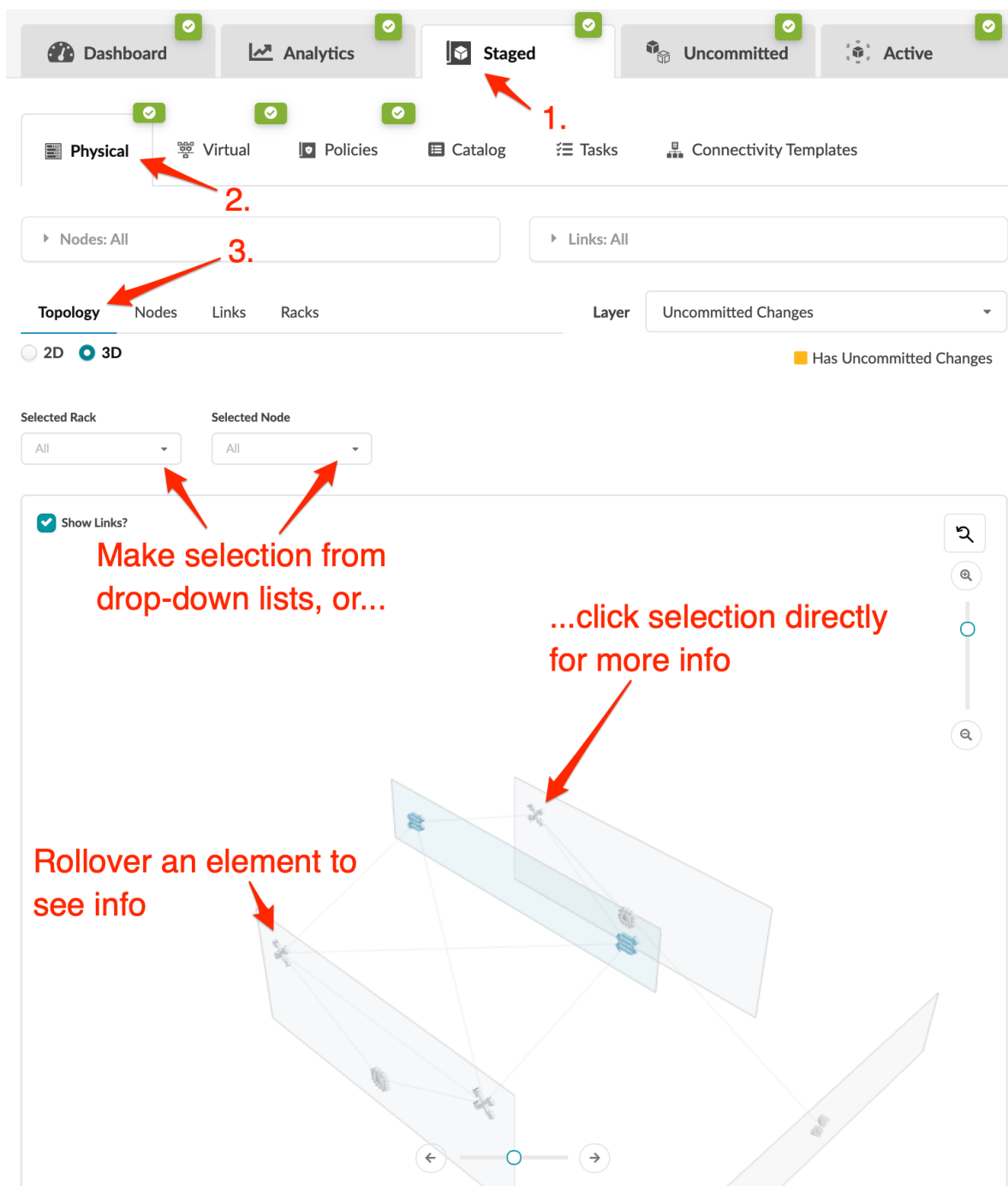
**NOTE:** This feature has been classified as a Juniper Apstra Technology Preview feature. These features are "as is" and voluntary use. Juniper Support will attempt to resolve any issues that customers experience when using these features and create bug reports on behalf of support cases. However, Juniper may not provide comprehensive support services to Tech Preview features.

For additional information, refer to the ["Juniper Apstra Technology Previews" on page 1082](#) page or contact ["JuniperSupport" on page 777](#).

From the blueprint, navigate to **Staged > Physical > Topology** and click **3D**.

- You can zoom in and out, move left and right, and reset to the default size and orientation.
- To show node name (and hostname as applicable) hover over an element.
- To show rack details in 2D view, select a rack by either clicking its element or by selecting it from the **Selected Rack** drop-down list.

- To show node details in 2D view, select a node by either clicking its element or by selecting it from the **Selected Node** drop-down list.



### Neighbors View (Staged)

- To show aggregate links, click the **Show Aggregate Links** check box.
- To show unused ports, click the **Show Unused Ports** check box.

- To show a different label (name, hostname, S/N), select a different label from the **Topology Label** drop-down list (right side).
- To show a different layer, select a different layer from the **Layer** drop-down list.
- Choose to show all neighbors or only specific types (generic, leaf, spine, and so on).

The interface displays a network topology management dashboard. At the top, there are tabs for Dashboard, Analytics, Staged, Uncommitted, and Active. Below these are tabs for Physical, Virtual, Policies, Catalog, Tasks, and Connectivity Templates. Two filters are present: 'Nodes: All' and 'Links: All'. A navigation bar includes 'Topology', 'Nodes', 'Links', 'Racks', 'Pods', and 'Planes'. View options for '2D' and '3D' are available. Selection filters include 'Selected Plane' (All), 'Selected Pod' (pod1), 'Selected Rack' (evpn\_mlag\_001\_001), 'Selected Node' (leaf1 (Leaf)), and 'Topology Label' (Name). A red arrow points to the 'Neighbors' tab in the 'Links' section. Below this are buttons for '+ Add links', '+ Add leaf peer links', and 'Edit links'. At the bottom, there are checkboxes for 'Show Aggregate Links' (checked), 'Show Unused Ports' (unchecked), and a dropdown for 'Show All Neighbors'. The main diagram shows 'leaf1' connected to several other nodes: rack1-server1, leaf2, rtr\_leaf1\_leaf2, spine2, spine1, and switch1-server1. Each connection is labeled with the specific Ethernet ports involved.

**Topology Label**

Selected Plane: All | Selected Pod: pod1 | Selected Rack: evpn\_mlag\_001\_001 | Selected Node: leaf1 (Leaf) | Topology Label: Name

**Neighbors** | Links

+ Add links | + Add leaf peer links | Edit links

☒ Show Aggregate Links | ☐ Show Unused Ports | Show All Neighbors ▼

**leaf1**

- Ethernet1/7
- Ethernet1/6
- Ethernet1/5
- Ethernet1/1
- Ethernet1/2
- Ethernet1/3
- Ethernet1/4

**Connections:**

- rack1-server1 (n/a)
- leaf2 (Ethernet1/7, Ethernet1/6)
- rtr\_leaf1\_leaf2 (eth1)
- spine2 (Ethernet1/1)
- spine1 (Ethernet1/2)
- switch1-server1 (n/a)

# Links View (Staged Topology)

Dashboard

Analytics

Staged

Uncommitted

Active

Time Voyager

Physical

Virtual

Policies

Catalog

Tasks

Connectivity Templates

Nodes: All

Links: All

Topology

Nodes

Links

Racks

Pods

Planes

2D

3D

Selected Plane

Selected Pod

Selected Rack

Selected Node

Topology Label

All

pod1

evpn\_single\_001\_001

leaf3 (Leaf)

Name

Neighbors

Links

+ Add links

Edit links

1-3 of 3

Page Size: 25

Filter selected by

all

selected only

unselected only

<input type="checkbox"/>	Name	Role	Tags	Speed	Link label	Port Channel ID	Endpoint 1				Endpoint 2				Actions
							Name	Role	Interface	Lag Mode	Name	Role	Interface	Lag Mode	
<input type="checkbox"/>	spine001_001_1<->leaf001_002_1[1]	Spine to Leaf		10G	N/A	N/A	leaf3	Leaf	Ethernet1/1	N/A	spine1	Spine	Ethernet1/1	N/A	>>
<input type="checkbox"/>	spine001_001_2<->leaf001_002_1[1]	Spine to Leaf		10G	N/A	N/A	leaf3	Leaf	Ethernet1/2	N/A	spine2	Spine	Ethernet1/2	N/A	>>



## Virtual Networks Endpoints (Staged)

New in Apstra version 4.0.1.

Selected Plane: All | Selected Pod: pod1 | Selected Rack: evpn\_single\_001\_001 | Selected Node: switch3-server1 (Generic System) | Topology Label: Name

Neighbors | Links | **Virtual Networks Endpoints**

+ Add links | Edit links

Query: All | 1-7 of 7 | Page Size: 25

Virtual Network	Tag Type	Leaf(s)	Port Channel ID	Interface Name(s)
<a href="#">vlan_30_leaf3_v4</a>	Untagged	leaf3	N/A	Ethernet1/3
<a href="#">blue_300_leaf3_v4</a>	VLAN Tagged	leaf3	N/A	Ethernet1/3

### Add Links

You can add generic systems (such as servers and external routers) to a running blueprint by adding links. You can add a link to new systems or existing ones, and you can add a link to an existing system to make it dual-homed. In an MLAG configuration, links must be symmetrical to both switches in the MLAG. When changing a system from single- to dual-homed, both links must be of the same speed.

1. From the blueprint, navigate to **Staged > Physical > Topology**, make a selection and click **Add links**.

The screenshot shows the 'Staged > Physical > Topology' view in a network management interface. The top navigation bar includes 'Dashboard', 'Analytics', 'Staged', 'Uncommitted', 'Active', and 'Time Voyager'. The left sidebar shows 'Physical', 'Virtual', 'Policies', 'Catalog', 'Tasks', and 'Connectivity Templates'. The main area has filters for 'Nodes: All' and 'Links: All'. Below these are tabs for 'Topology', 'Nodes', 'Links', 'Racks', 'Pods', and 'Planes'. The 'Topology' tab is active, showing a 2D/3D view selector. There are dropdowns for 'Selected Plane' (All), 'Selected Pod' (pod1), 'Selected Rack' (evpn\_single\_001\_001), and 'Selected Node' (leaf3 (Leaf)). A 'Topology Label' dropdown is also present. Below these are 'Neighbors' and 'Links' tabs. The 'Links' tab is active, showing a table of links. A red arrow points to the 'Add links' button. Below the table are pagination controls (1-3 of 3) and a 'Page Size' dropdown (25). At the bottom, there is a filter section with radio buttons for 'all', 'selected only', and 'unselected only'.

	Name	Role	Tags	Speed	Link label	Port Channel ID	Endpoint 1				Endpoint 2				Actions
							Name	Role	Interface	Lag Mode	Name	Role	Interface	Lag Mode	
<input type="checkbox"/>	spine001_001_1<->leaf001_002_1[1]	Spine to Leaf		10G	N/A	N/A	leaf3	Leaf	Ethernet1/1	N/A	spine1	Spine	Ethernet1/1	N/A	>>
<input type="checkbox"/>	spine001_001_2<->leaf001_002_1[1]	Spine to Leaf		10G	N/A	N/A	leaf3	Leaf	Ethernet1/2	N/A	spine2	Spine	Ethernet1/2	N/A	>>

2. In the dialog that opens, select the source (existing system, copy generic, or create new system) and click **Next**.

If you're creating a new system select one of the following:

- **Generic** - same rack, connects to leaf only.
- **Access switch** - See ["Access Layer Switches" on page 805](#) for details.
- **External generic** - different rack, connects to leaf, spine or superspine (5-stage).

Choose action

Create links

✕

Select source \*

☒ Existing system

Create links towards existing system.

☐ Copy generic

Copy existing generic system including its links.

☐ New system

Create new system and add links to it.

System info

Logical device: Leaf-LD

Device profile: Juniper vQFX

Interface map: Juniper vQFX\_\_Leaf-LD

System group label: switch

Leaf-LD

2 x 10 Gbps Spine

6 x 10 Gbps Generic

Next

### 3. Create links.

For existing systems:

- Select a port and transformation.
- Select a system type and system.
- Select a logical link, LAG mode (and optional tags).
- Click **Add link**.
- Click **Create** to stage the link addition and return to the topology view.

Choose action

Create links

✕

Select devices and their interfaces to create a link:

Leaf: leaf3

Device profile: Arista vEOS

1 2 3 4 5 6 7

Port #4 Tr: #1 (10 Gbps, default)

Ethernet4

Select system type:

☒ Generic ☐ Access switch ☐ External generic

Select Generic: \*

switch3-server1

Generic System: switch3-server1

Device profile: N/A

Link properties

Links

1-1 of 1

Type	Speed	Leaf		Link label	LAG Mode	Generic		Tags	Actions
		Name	Interface			Name	Interface		
Existing	10G	leaf3	Ethernet3	single-link	No LAG	switch3-server1	N/A		

Add Link →

!

Back Create

For copying generics:

- Select a port and transformation.
- Select an existing generic system.

- Click the **Interface** field in the table on the right to add interface.
- Select port and transformation and click **Confirm**.
- Click **Add link**.
- Click **Submit** to stage the link addition and return to the topology view.

✓ Choose action

Copy existing generic

✕

Leaf: leaf3  
Device profile: Arista vEOS

1

2

3

4

5

6

7

1. Select available port

2. Select transformation

Port #4 Tr. #1 (10 Gbps, default)

Ethernet4

2. Select transformation

Select existing generic

switch3-server1
✕

3. Select existing generic system

Logical device: AOS-1x10-1  
 Device profile: N/A  
 Interface map: N/A  
 System group label: single-server  
 Port Channel ID min: 0  
 Port Channel ID max: 0

Back

Submit

**Links**

Select ports in the table below to connect existing system links

Speed	Leaf		LAG Mode	Generic		Actions
	Name	Interface		Name	Interface	
10G	leaf3	N/A	No LAG	New Generic	N/A	✕

4. Click to select interface

For new systems:

- Select a representation type, representation and system group label.
- Click **Next**, select a port, transformation, logical link and LAG mode (same as section above).
- Click **Create** to stage the link addition and return to the topology view.

✓ Choose action

✓ Create new system

Create links

✕

Choose a representation for a new device \*

☐ None <sup>?</sup>
☒ AOS Logical Device
 ☐ AOS Logical Device With an Interface Map

☐ Show whole catalog

AOS-1x10-1

1. Select representation

AOS-1x10-1

1 x 10 Gbps

Leaf • Access

System Group Label <sup>?</sup> \*

evpn-single

2. Select system group label

Port Channel ID min

0

Port Channel ID max

0

Tags

Select...

3.

Back Next

**NOTE:** When adding a link, you can review the config changes in the incremental config to evaluate the changes. For information about incremental config, see ["Device Configuration Lifecycle"](#) on page 140.

## Add Leaf Peer Links



**CAUTION:** Do not use leaf peer links if the platform used does not support it. Currently, Junos devices do not support any peer links, and SONiC devices do not support L3 peer links.

- From the blueprint, navigate to **Staged > Physical > Topology** and select a leaf that is an MLAG member.
- Click **Add leaf peer links** and select the link type.
  - Peer Link - adds a link between the two leafs and adds a BGP session (between the two leafs) in all VRFs.

- L3 Peer Link - adds a link between the two leafs and adds a BGP session (between the two leafs) in all VRFs.
3. Select an unused port for each leaf peer member. (Only unused ports are selectable). When a port has been selected for each leaf peer member, the **Add Link** button becomes clickable.

### Add Leaf Peer Links ✕

Link Type

☒ Peer Link ☐ L3 Peer Link

Select ports and interfaces to create a link:

Leaf: dual\_rack\_001\_leaf2  
Device profile: Juniper vQFX

1

2

3

4

5

6

7

8

9

10

11

12

Port #8 Tr. #1 (10 Gbps, default)

xe-0/0/7

Leaf: dual\_rack\_001\_leaf1  
Device profile: Juniper vQFX

1

2

3

4

5

6

7

8

9

10

11

12

Port #8 Tr. #1 (10 Gbps, default)

xe-0/0/7

Tags

Select...

New Links

< >

Speed	Leaf 1		Leaf 2		Link Type	Actions
	Name	Interface	Name	Interface		
No new links						

Add Link →

Add

4. Click **Add Link**.
5. Click **Add** to stage the addition.

## Update Logical Links and LAG Mode

You can update **Link Labels** of existing leaf-generic system links.

You can also change the **LAG mode** between leafs and generic systems. For example, you may have a dual-connected generic system configured with LACP, and you want it to have separate, individually configured Layer 2 links. You can change the LAG mode (such as LACP (Active)) to No LAG. When you change the LAG mode to No LAG, you must change the Link Label for all affected links, or you will receive an error.



**CAUTION:** Changing Link Labels results in the loss of all virtual network assignments for the port(s). You'll need to re-assign them. If you want to change a dual-connected generic system to No LAG, we recommend that you only change the second Link Label value to retain virtual network assignments on the first port.

You can also change to a LAG mode from No LAG. With the same procedure as above set all the links in the same group to the same Link Label, and change the LAG mode.

To update logical links and LAG modes follow the steps below:

1. From the blueprint, navigate to **Staged > Physical > Topology** and make a selection.
2. Click **Edit links**, then click **Update logical links and LAG Mode**.

The screenshot shows the 'Staged > Physical > Topology' view. The 'Edit links' button is highlighted with a red arrow. Below it, a dropdown menu shows options: 'Update logical links and LAG Mode' and 'Change link speeds'. The interface also includes a table of links with columns for Name, Role, Tags, Speed, Link label, Port Channel ID, Endpoint 1, Endpoint 2, and Actions.

0 selected	Name	Role	Tags	Speed	Link label	Port Channel ID	Endpoint 1				Endpoint 2				Actions
							Name	Role	Interface	Lag Mode	Name	Role	Interface	Lag Mode	
1-7 of 7															

3. Change the link labels, LAG modes and/or tags, as needed.

- To change one link, change the Link Label and/or select a different LAG mode from the drop-down list for that link.

## Update Logical Links and LAG Mode

Query: All 1-1 of 1 Page Size: 10

0 selected	Name	Role	Speed	Link label <sup>®</sup>	Port Channel ID	LAG Mode	Tags	Endpoint 1			Endpoint 2		
								Name	Role	Interface	Name	Role	Interface
<input type="checkbox"/>	rack_2_001_leaf1<->rack_2_001_sys001(link)[1]	To Generic System	10G	link	N/A	No LAG	Select...	rack_2_001_sys001	Generic System	n/a	rack_2_001_leaf1	Leaf	swp3

Update

- To change multiple links, select one or more links, make your changes in the sections that appear above the table, then click **Apply to selection**.

## Update Logical Links and LAG Mode

Link label<sup>®</sup>  Apply to selection

LAG Mode  
☒ LACP (Active)<sup>®</sup> ☐ LACP (Passive)<sup>®</sup> ☐ Static LAG (no LACP)<sup>®</sup> ☐ No LAG<sup>®</sup> Apply to selection

Query: All 1-2 of 2 Page Size: 10

1. Select one or more links...

2 selected	Name	Role	Speed	Link label <sup>®</sup>	Port Channel ID	LAG Mode	Tags	Endpoint 1			Endpoint 2		
								Name	Role	Interface	Name	Role	Interface
<input checked="" type="checkbox"/>	rack_2_001_leaf1<->rack_2_001_sys001(link)[1]	To Generic System	10G	link	N/A	No LAG	Select...	rack_2_001_leaf1	Leaf	swp3	rack_2_001_sys001	Generic System	n/a
<input checked="" type="checkbox"/>	rack_2_001_leaf1<->sys001(ext-link-1)[1]	To Generic System	10G	ext-link-1	N/A	No LAG	Select...	rack_2_001_leaf1	Leaf	swp4	sys001	Generic System	eth1

Update

- Click **Update** to stage the change and return to the links view.

## Change Link Speeds

You can change link speeds from **Staged > Physical > Links**, or from **Staged > Physical > Topology** as shown below.

- From the blueprint, navigate to **Staged > Physical > Topology** and make a selection.

- 2. From the neighbors view or the links view of a selection, click **Edit links**, then click **Change link speeds**.

### Change Link Speeds

► Query: All

1-1 of 1 < > Page Size: 10 ▼

<input type="checkbox"/> 0 selected	Name	Role	Speed	Link label ⓘ	Port Channel ID	Endpoint 1			Endpoint 2		
						Name	Role	Interface	Name	Role	Interface
<input type="checkbox"/>	rack_2_001_leaf1<->rack_2_001_sys001(link)[1]	To Generic System	10 Gbps ▼	link	N/A	rack_2_001_sys001	Generic System	n/a	rack_2_001_leaf1	Leaf	swp3

Update

- 3. Change link speeds as needed.
- 4. Click **Update** to stage the change and return to the links view.

Delete Link

- 1. From the blueprint, navigate to **Staged > Physical > Topology** and make a selection.

2. From the links view of a selection, click the **Delete** button for the link to remove.

The screenshot shows the network management interface with the following components:

- Navigation Bar:** Dashboard, Analytics, Staged, Uncommitted, Active, Time Voyager.
- Section Bar:** Physical, Virtual, Policies, Catalog, Tasks, Connectivity Templates.
- Filters:** Nodes: All, Links: All.
- Topology View:** Topology (selected), Nodes, Links, Racks, Pods, Planes. 2D (selected), 3D.
- Selection Fields:**
  - Selected Plane: All
  - Selected Pod: pod1
  - Selected Rack: evpn\_mlag\_001\_001
  - Selected Node: leaf1 (Leaf)
  - Topology Label: Name
- Actions:** Neighbors, Links (selected), Add links, Add leaf peer links, Edit links.
- Table:**

	Name	Role	Tags	Speed	Link label	Port Channel ID	Endpoint 1				Endpoint 2				Actions
							Name	Role	Interface	Lag Mode	Name	Role	Interface	Lag Mode	
<input type="checkbox"/>	leaf001_001_1<->leaf001_001_2(i3_peer_link)	Leaf L3 Peer Link		10G	i3_peer_link	3	leaf1	Leaf	Ethernet1/5	N/A	leaf2	Leaf	Ethernet1/6	N/A	>>
- Footer:** 1-7 of 7, Page Size: 25.

3. Click **Delete** to stage the deletion and return to the links view.

## Nodes (Staged)

### IN THIS SECTION

- [Add Generic System | 448](#)
- [Add External Generic System | 455](#)
- [Edit Server Names and Hostnames | 463](#)
- [Edit Port Channel ID Min Max | 463](#)
- [Set Deploy Mode \(Multiple Devices\) | 464](#)
- [Edit Device Details | 464](#)
- [Edit Device Properties | 465](#)

- [Add/Remove Tags \(Single Node\) | 466](#)
- [Add/Remove Tags \(Multiple Nodes\) | 466](#)
- [View Node's Static Routes | 467](#)

You can perform the following tasks from the staged nodes view:

- ["Add external generic systems" on page 455](#)
- ["Edit server names and hostnames" on page 463](#)
- ["Edit port channel ID min max" on page 463](#)
- Set deploy modes - single device or ["multiple devices" on page 464](#)
- ["Edit device details" on page 464](#)
- ["Edit device properties" on page 465](#)
- Add and remove tags on a ["single node" on page 466](#) or ["multiple nodes" on page 466](#)
- ["View a node's static routes" on page 467](#)

From the blueprint, navigate to **Staged > Physical > Nodes** to go to nodes in the blueprint. You can view nodes in table view or card view. In table view, you can select which details to display (from the drop-down list). You can click the name of a node in the table to display information in the right panel (such as

telemetry, properties, and tags).

The screenshot shows the Apstra GUI interface. At the top, there are tabs for Dashboard, Analytics, Staged (selected), Uncommitted, and Active. Below these are sub-tabs for Physical, Virtual, Policies, Catalog, Tasks, and Connectivity Templates. The main view is the Staged view, showing a topology diagram. Annotations with red arrows and numbers indicate the following steps:

- 1. Click on the Staged tab.
- 2. Click on the Physical sub-tab.
- 3. Click on the Nodes tab in the topology view.

Below the topology view, there are filters for Selected Plane, Selected Pod, and Selected Rack, all set to 'All'. A 'Topology Label' dropdown is also present. A legend indicates 'Has Uncommitted Changes' with a yellow square. Below this, there are buttons for adding, editing, and deleting nodes. Annotations indicate:

- 'Add external generic systems' points to the add button.
- 'Edit server names and hostnames' points to the edit button.
- 'Edit port channel id min max' points to the edit button.

A table of nodes is displayed below the filters. The table has columns for Name, Tags, Role, External?, and Deploy M. The table shows three rows of nodes: sspine1, sspine2, and spine1. A dropdown menu is open over the table, showing options for Columns (9/17), Tags, Role, External?, Pod, and Rack. An annotation 'Select what to display in table' points to the dropdown menu.

Name	Tags	Role	External?	Deploy M
sspine1		Superspine	N/A	Deploy
sspine2		Superspine	N/A	Deploy
spine1		Spine	N/A	Deploy

## Add Generic System

### IN THIS SECTION

- [Add Generic System \(from Topology View\) | 449](#)
- [Copy Existing Generic \(from Topology View\) | 453](#)

Generic systems and external generic systems are systems that are not managed in the Apstra environment. They can be external routers, firewalls, or whatever else you want; you specify their roles with tags. If you add the system to a leaf in a rack topology, we call it a generic system. If the system is

not part of a rack topology, we call it an external generic system. This page shows you a couple ways that you can add *generic* systems.

**NOTE:** You can also create generic systems during the **Design** phase before creating your blueprint. For more information, see ["Rack Types" on page 101](#).

## Add Generic System (from Topology View)

1. From the blueprint, navigate to **Staged > Physical > Topology** and select the leaf to link to the new generic system.

The screenshot displays the 'Topology' view in the Network Configuration UI. The top navigation bar includes tabs for Dashboard, Analytics, Staged, Uncommitted, Active, and Time Voyager. Below this, a sub-navigation bar shows Physical, Virtual, Policies, Catalog, Tasks, and Connectivity Templates. The main area features a topology diagram with nodes and links. A red arrow points to a node labeled 'rack\_1\_001\_leaf1', which has a tooltip showing its role as 'Leaf', hostname as 'leaf-1-1', and tags as 'n/a'. A red text label 'Select node' is positioned next to the arrow. The diagram also shows spine nodes (spine1, spine2) and other rack nodes (rack\_1\_001, rack\_2\_001). The interface includes various filters and controls, such as 'Nodes: All', 'Links: All', 'Selection', 'Build', 'Topology Label', and 'Expand Nodes? Show Links?'.

**NOTE:** You can also get to the selection page from the **Nodes** view. From the blueprint, navigate to **Staged > Physical > Nodes**, click the leaf name in the table (not the check box), then click the leaf name that appears at the top of the **Selection** panel (on the right side of the page).

2. Click **Add links**.

3. For the source, select **New System** and for the new system type, select **Generic**, then click **Next**.

4. Select how to represent the new system, then select the logical device or interface map from the drop-down list.

- **Apstra Logical Device** - specifies interfaces, speed, and telemetry from the leaf node towards the new system.
- **Apstra Logical Device with an Interface Map** - also includes device profile information.

✓ Choose action

✓ Create new system

Create links

✕

Choose a representation for a new device \*

☐ None
 ☐ Apstra Logical Device
 ☒ Apstra Logical Device With an Interface Map

☐ Show whole catalog

Generic\_Server\_1RU\_1x10G\_\_AOS-1x10-1

Generic System  
Device profile: Generic\_Server\_1RU\_1x10G

1

System Group Label

generic

Port Channel ID min

0

Port Channel ID max

0

Tags

Select...

Back

Next

- Select a system group label from the drop-down list or create a new one. A group label is a way to group things together that share certain characteristics. For example, if a leaf and a server have two physical links between them and these physical links have the same group label, then these two links are considered to be bonded and configuration is rendered accordingly. Grouped systems have the same logical devices and identical uplinks (the same number of links with the same speed bonded the same way).
- Optionally, enter a port channel ID range and tags, then click **Next**.

✓ Choose action

✓ Create new system

Create links

✕

Select devices and their interfaces to create a link:

Leaf: rack\_1\_001\_leaf1  
Device profile: Arista vEOS

1 2 3 4 5 6 7

Leaf: rack\_1\_001\_leaf2  
Device profile: Arista vEOS

1 2 3 4 5 6 7

Generic System  
Device profile: Generic\_Server\_1RU\_1x10G

1

Link properties

Logical Link

Select from dropdown or type a new name...

Lag mode

Select...

Tags

Select...

Add Link

Links

1-2 of 2

Type	Speed	Leaf	Link label	LAG Mode	Generic	Tags	Actions
		Name	Interface		Name	Interface	
Existing	10G	rack_1_001_leaf1	Ethernet4	link	LACP (Active)	rack_1_001_sys001	N/A
Existing	10G	rack_1_001_leaf2	Ethernet4	link	LACP (Active)	rack_1_001_sys001	N/A

Back

Create

- Select available port(s) and transformation(s). The gray **Add Link** button turns green.

Choose action Create new system Create links

Select devices and their interfaces to create a link:

Leaf: rack\_1\_001\_leaf1  
Device profile: Arista vEOS

Port #5 Tr: #1 (10 Gbps, default) Ethernet5

Leaf: rack\_1\_001\_leaf2  
Device profile: Arista vEOS

Port #5 Tr: #1 (10 Gbps, default) Ethernet5

Generic System  
Device profile: Generic\_Server\_1RU\_1x10G

Port #1 Tr: #1 (10 Gbps, default) eth0

Link properties

Logical Link <sup>ⓘ</sup> link Lag mode <sup>ⓘ</sup> No LAG

Tags  
Select...

Back Create

Links

1-2 of 2

Type	Speed	Leaf Name	Leaf Interface	Link label	LAG Mode	Generic Name	Generic Interface	Tags	Actions
Existing	10G	rack_1_001_leaf1	Ethernet4	link	LACP (Active)	rack_1_001_sys001	N/A		
Existing	10G	rack_1_001_leaf2	Ethernet4	link	LACP (Active)	rack_1_001_sys001	N/A		

1. Select available ports/transformations

2. Add Link

8. Click **Add Link**. The link is added to the link table.

Choose action Create new system Create links

Select devices and their interfaces to create a link:

Leaf: rack\_1\_001\_leaf1  
Device profile: Arista vEOS

Port #5 Tr: #1 (10 Gbps, default) Ethernet5

Leaf: rack\_1\_001\_leaf2  
Device profile: Arista vEOS

Port #5 Tr: #1 (10 Gbps, default) Ethernet5

Generic System  
Device profile: Generic\_Server\_1RU\_1x10G

Port #1 Tr: #1 (10 Gbps, default) eth0

Link properties

Logical Link <sup>ⓘ</sup> link Lag mode <sup>ⓘ</sup> No LAG

Tags

Back Create

Links (1 will be added)

1-3 of 3

Type	Speed	Leaf Name	Leaf Interface	Link label	LAG Mode	Generic Name	Generic Interface	Tags	Actions
New	10G	rack_1_001_leaf2	Ethernet5	link	No LAG	N/A	eth0		
Existing	10G	rack_1_001_leaf1	Ethernet4	link	LACP (Active)	rack_1_001_sys001	N/A		
Existing	10G	rack_1_001_leaf2	Ethernet4	link	LACP (Active)	rack_1_001_sys001	N/A		

New link is added

9. Click **Create** to stage the change and return to the **Topology** view.

When you're ready to activate your changes, commit them from the **Uncommitted** tab.

## Copy Existing Generic (from Topology View)

1. From the blueprint, navigate to **Staged > Physical > Topology** and select the leaf to link to the new generic system.

The screenshot displays the network management interface in the 'Staged > Physical > Topology' view. The top navigation bar includes 'Dashboard', 'Analytics', 'Staged', 'Uncommitted', 'Active', and 'Time Voyager'. The sub-navigation bar shows 'Physical', 'Virtual', 'Policies', 'Catalog', 'Tasks', and 'Connectivity Templates'. The main area features a topology diagram with nodes and links. A red arrow points to a node labeled 'rack\_1\_001\_leaf1' with a tooltip showing its role as 'Leaf', hostname as 'leaf-1-1', and tags as 'n/a'. A red text label 'Select node' is next to the arrow. The interface also includes filters for Nodes and Links, a Layer dropdown set to 'Uncommitted Changes', and a 'Nothing selected yet' message box.

**NOTE:** You can also get to the selection page from the **Nodes** view. From the blueprint, navigate to **Staged > Physical > Nodes**, click the leaf name in the table, then click the leaf name that appears at the top of the **Selection** panel (on the right side of the page).

2. Click **Add Links**.

The screenshot displays the Cisco DNA Center interface for configuring a network topology. The top navigation bar includes tabs for Dashboard, Analytics, Staged, Uncommitted, Active, and Time Voyager. Below this, a filter bar allows selection between Physical, Virtual, Policies, Catalog, Tasks, and Connectivity Templates, with a 'Find by tags' search option. The main workspace is divided into two sections: 'Nodes: All' and 'Links: All'. The 'Nodes: All' section shows a 2D topology view with a selected rack (rack\_1\_001) and a selected node (rack\_1\_001\_leaf1). A red arrow points to the 'Add links' button. The right sidebar displays the 'Deploy Mode' (deploy) and 'S/N' (50540FC6429) for the selected node. The bottom section shows the 'Device Info' for rack\_1\_001\_leaf1, including Management IP (10.28.1.9), OS (EOS 4.24.5M), and Operation Mode (FULL CONTROL).

- For the source, select **Copy generic**, then click **Next**.
- Select an existing generic system from the drop-down list. The links table appears.

✓

Choose action

✕

Copy existing generic

Select existing generic

rack\_1\_001\_sys001✕

Logical device: AOS-2x10-1

Device profile: N/A

Interface map: N/A

System group label: generic

Port Channel ID min: 0

Port Channel ID max: 0

Links

Select ports in the table below to connect existing system links

Speed	Leaf		LAG Mode	Generic		Actions
	Name	Interface		Name	Interface	
10G	rack_1_001_leaf1	N/A	LACP (Active)	New Generic	N/A	
10G	rack_1_001_leaf2	N/A	LACP (Active)	New Generic	N/A	

Back

Submit

5. Select an interface (previous screenshot), select a port and transformation, then click **Confirm**.

Select interface

Leaf: rack\_1\_001\_leaf1

Device profile: Arista vEOS

1

2

3

4

5

6

7

Port #5 Tr. #1 (10 Gbps, default)

Ethernet5

Cancel

Confirm

6. The interface turns green in the links table. Repeat the steps to assign the port and transformation for the other interface, as applicable.

Choose action

Copy existing generic

Select existing generic

rack\_1\_001\_sys001

Logical device: AOS-2x10-1

Device profile: N/A

Interface map: N/A

System group label: generic

Port Channel ID min: 0

Port Channel ID max: 0

Links

Select ports in the table below to connect existing system links

Speed	Leaf		LAG Mode	Generic		Actions
	Name	Interface		Name	Interface	
10G	rack_1_001_leaf1	Ethernet5	LACP (Active)	New Generic	N/A	
10G	rack_1_001_leaf2	N/A	LACP (Active)	New Generic	N/A	

Back

Submit

7. Click **Submit** to stage your changes and return to the **Topology** view.

When you're ready to activate your changes, commit them from the **Uncommitted** tab.

### Add External Generic System

IN THIS SECTION

Add External Generic (from Topology View) | 456

Add External Generic (from Nodes View) | 459

Generic systems and external generic systems are systems that are not managed in the Apstra environment. They can be external routers, firewalls, or whatever else you want; you specify their roles

with tags. If you add the system to a leaf in a rack topology, we call it a generic system. If the system is not part of a rack topology, we call it an external generic system. This page shows you a couple ways that you can add *external* generic systems.

### Add External Generic (from Topology View)

1. From the blueprint, navigate to **Staged > Physical > Topology** and select the node to add the new external generic system to.

The screenshot shows the 'Staged > Physical > Topology' view. The interface includes a top navigation bar with tabs like Dashboard, Analytics, Staged, Uncommitted, Active, and Time Voyager. Below this is a sub-navigation bar with Physical, Virtual, Policies, Catalog, Tasks, and Connectivity Templates. The main area shows a topology diagram with nodes like sys001, spine1, spine2, and various rack\_1\_001\_\*. A red arrow points to the 'rack\_1\_001\_leaf1' node with the text 'Select node'. A tooltip for this node shows details: Role: Leaf, Hostname: leaf-1-1, Tags: n/a. On the right, a 'Nothing selected yet' message is displayed.

**NOTE:** You can also get to the selection page from the **Nodes** view. From the blueprint, navigate to **Staged > Physical > Nodes**, click the node name in the table, then click the node name that appears at the top of the **Selection** panel (on the right side of the page).

2. Click **Add links**.

- For the source, select **New System** and for the new system type, select **External Generic**, then click **Next**.

- Select how to represent the new system, then select the logical device or interface map from the drop-down list.
  - Apstra Logical Device** - specifies interfaces, speed, and telemetry from the leaf node towards the new system.

- **Apstra Logical Device with an Interface Map** - also includes device profile information.

Choose action

Create new system

Create links

Choose a representation for a new device \*

None

Apstra Logical Device

Apstra Logical Device With an Interface Map

Show whole catalog

Generic\_Server\_1RU\_1x10G\_AOS-1x10-1

Device profile: Generic\_Server\_1RU\_1x10G

1

System Group Label® \*

generic

Tags

Select...

Back

Next

5. Select a system group label from the drop-down list or create a new one, then click **Next**. A group label is a way to group things together that share certain characteristics. For example, if a leaf and a server have two physical links between them and these physical links have the same group label, then these two links are considered to be bonded and configuration is rendered accordingly. Grouped systems have the same logical devices and identical uplinks (the same number of links with the same speed bonded the same way).

Choose action

Create new system

Create links

Select devices and their interfaces to create a link:

Leaf: rack\_2\_001\_leaf1

Device profile: Cumulus VX

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

Device profile: Generic\_Server\_1RU\_1x10G

1

Link properties

Logical Link® \*

Select from dropdown or type a new name...

Lag mode \*

Select...

Add Link →

Links

1-1 of 1

Type	Speed	Leaf	Link label	LAG Mode	External Generic	Tags	Actions
		Name	Interface		Name	Interface	
Existing	10G	rack_2_001_leaf1	swp4	ext-link-1	No LAG	sys001 eth1	

Back

Create

6. Select available port(s) and transformation(s), then select a logical link and LAG mode from the drop-down lists. The gray **Add Link** button turns green.

Choose action Create new system Create links

Select devices and their interfaces to create a link:

Leaf: rack\_2\_001\_leaf1  
Device profile: Cumulus VX

Port #5 Tr. #1 (10 Gbps) swp5

Port #5 Tr. #2 (1 Gbps, default) swp5

Device profile: Generic\_Server\_1RU\_1x10G

Port #1 Tr. #1 (10 Gbps, default) eth0

1. Select available ports/transformations

2. Add Link →

Link properties

Logical Link ext-link-1

Lag mode No LAG

Tags

Select...

Back Create

7. Click **Add Link**. The link is added to the link table.

Choose action Create new system Create links

Select devices and their interfaces to create a link:

Leaf: rack\_2\_001\_leaf1  
Device profile: Cumulus VX

Port #5 Tr. #1 (10 Gbps) swp5

Port #5 Tr. #2 (1 Gbps, default) swp5

Device profile: Generic\_Server\_1RU\_1x10G

Port #1 Tr. #1 (10 Gbps, default) eth0

New link is added.

Add Link →

Type	Speed	Leaf	Link label	LAG Mode	External Generic	Tags	Actions
		Name	Interface		Name	Interface	
New	10G	rack_2_001_leaf1	swp5	ext-link-1	No LAG	N/A	eth0
Existing	10G	rack_2_001_leaf1	swp4	ext-link-1	No LAG	sys001	eth1

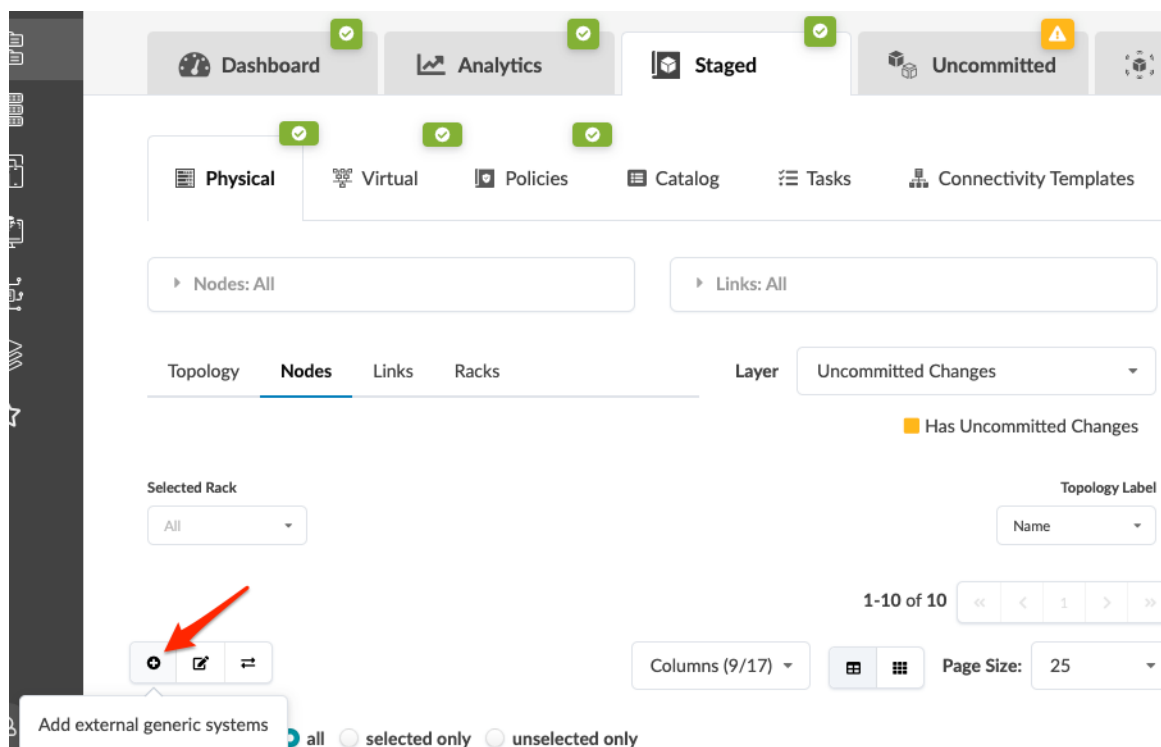
Back Create

8. Click **Create** to stage the change and return to the **Topology** view.

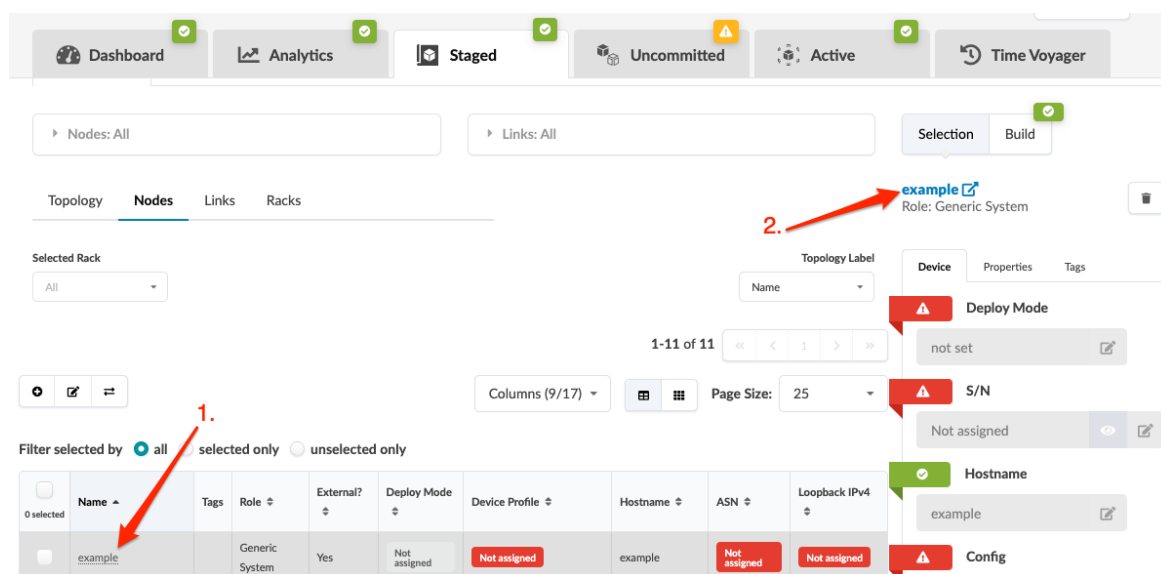
When you're ready to activate your changes, commit them from the **Uncommitted** tab.

### Add External Generic (from Nodes View)

- From the blueprint, navigate to **Staged > Physical > Nodes** and click the **Add external generic systems** button.



2. Enter a hostname, and if you want to define port roles, select a logical device from the drop-down list.
3. To make the system easier to find later, add (optional) tags.
4. Click **Create** to stage your changes and return to the Nodes view. The new system appears in the list. It is not yet linked to any Apstra-managed device. You can link it now or later. To link it now, proceed to the next step.
5. Click the external generic name in the list (not the check box), then click the external generic name that appears at the top of the **Selection** panel (on the right side of the page).



6. Click **Add links**.

The screenshot shows a network configuration interface. At the top, there are two tabs: 'Staged' (active) and 'Active'. Below these, there is a 'Physical' tab. Under the 'Physical' tab, there are two sub-tabs: 'Neighbors' and 'Links'. Below the 'Links' sub-tab, there is a button labeled '+ Add links' which is highlighted with a red arrow. Below this button, there is a section labeled 'example'.

7. Click **Next**.

The screenshot shows the 'Add Link' form. At the top, there are two buttons: 'Choose action' (with a green checkmark) and 'Create links'. Below these, there is a section for 'Select system type' with radio buttons for 'Leaf' (selected), 'Spine', and 'Superspine'. Below this, there is a 'Select Leaf:' dropdown menu. Below the dropdown, there is a section for 'Select devices and their interfaces to create a link:' with 'Generic System: example' and 'Device profile: N/A'. Below this, there is a section for 'Link properties' with 'Logical Link' and 'Lag mode' dropdown menus. Below the 'Link properties' section, there is a 'Tags' dropdown menu. To the right of the 'Link properties' section, there is a red warning icon. Below the 'Link properties' section, there is a section for 'Links' with a table showing 'No new links'. At the bottom right, there are 'Back' and 'Create' buttons.

Type	Speed	External Generic		Link label	LAG Mode	Leaf		Tags	Actions
		Name	Interface			Name	Interface		
No new links									

- Select the system type (leaf, spine, superspine) to connect the new external generic system to, then select the system from the drop-down list. Add (optional) tags.
- Select available port(s) and transformation(s), then select a logical link and LAG mode from the drop-down lists. The gray **Add Link** button turns green.

Choose action **Create links**

Select system type: ☒ Leaf ☐ Spine ☐ Superspine

Select Leaf:

Select devices and their interfaces to create a link:

Leaf: rack\_1\_001\_leaf1  
Device profile: Arista vEOS

Port #6 Tr. #1 (10 Gbps, default)

Generic System: example  
Device profile: N/A

Link properties

Logical Link

Lag mode

Tags

**Add Link**

Links

Type	Speed	External Generic	Link label	LAG Mode	Leaf	Tags	Actions
		Name	Interface		Name	Interface	
No new links							

Back Create

10. Click **Add Link**. The link is added to the link table.

Choose action **Create links**

Select system type: ☒ Leaf ☐ Spine ☐ Superspine

Select Leaf:

Select devices and their interfaces to create a link:

Leaf: rack\_1\_001\_leaf1  
Device profile: Arista vEOS

Port #6 Tr. #1 (10 Gbps, default)

Generic System: example  
Device profile: N/A

Link properties

Logical Link

Lag mode

Tags

**Add Link**

Links (1 will be added)

Type	Speed	External Generic	Link label	LAG Mode	Leaf	Tags	Actions
		Name	Interface		Name	Interface	
New	10G	example	N/A	ext-link-1	No LAG	rack_1_001_leaf1	Ethernet6

New link is added.

Back Create

11. To create additional links, repeat the above steps.

12. Click **Create** to stage your changes and return to the **Nodes** view.

When you're ready to activate your changes, commit them from the **Uncommitted** tab. When external generic systems are connected to the fabric, IP addresses are automatically assigned to the links from the pool assigned under **Routing Zones**. You can also assign custom IP addresses to the links under a specific routing zone.



1. From the blueprint, navigate to **Staged > Physical > Nodes** and click the **Edit port channel ID min max** button (third of three buttons above the nodes list).
2. Change min and max values, as needed.
3. Click **Update** to stage the changes and return to the nodes view.

## Set Deploy Mode (Multiple Devices)

You can change the deploy mode for one or more nodes at the same time from the nodes view.

1. From the blueprint, navigate to **Staged > Physical > Nodes** and check one or more check boxes for the node(s) to change. (You can narrow your search with the drop-down lists for planes, pods, and racks as applicable, as of Apstra version 4.0.)
2. Click the **Set Deploy Mode** button (fourth of five buttons above the nodes list) and select the deploy mode. (To filter selection before changing deploy mode, you can use the query.)
3. Click **Set Deploy Mode** to stage the change and return to the nodes view.

## Edit Device Details

1. From the blueprint, navigate to **Staged > Physical > Nodes** and select a node name (not the check box). (You can narrow your search with the drop-down lists for planes, pods, and racks as applicable, as of Apstra version 4.0.)
2. Click the **Device** tab in the right panel (if it's not already selected).

The screenshot shows the Apstra GUI interface for managing nodes. The top navigation bar includes tabs for Dashboard, Analytics, Staged, Uncommitted, Active, and Time Voyager. Below this, the Physical tab is selected, showing a list of nodes. The left sidebar has filters for Nodes and Links. The main area displays a table of nodes with columns for Name, Tags, Role, External?, Deploy Mode, Device Profile, Hostname, ASN, and Loopback IPv4. The right sidebar shows the details for the selected node, leaf1, with tabs for Device, Properties, Tags, and Virtual. The Device tab is active, showing the Deploy Mode and S/N. Annotations with red arrows point to the 'leaf1' node in the table, the 'Table View' button, and the 'Edit' button in the right sidebar.

Click a node name... ... to see its details

Table View

Device

Deploy Mode

S/N

525400EF54F3

Device Info

Management IP 10.29.46.20

OS NXOS 9.3(7)

Operation Mode FULL CONTROL

Hostname

leaf1

Config

Rendered Incremental Pristine

Name	Tags	Role	External?	Deploy Mode	Device Profile	Hostname	ASN	Loopback IPv4
leaf1		Leaf	N/A	Deploy	Cisco NXOSv	leaf1	64515	10.0.0.6/32
leaf2		Leaf	N/A	Deploy	Cisco NXOSv	leaf2	64516	10.0.0.7/32
leaf3		Leaf	N/A	Deploy	Cisco NXOSv	leaf3	64517	10.0.0.8/32
leaf_pair001_001_1		Leaf Pair	N/A	N/A	N/A	N/A	N/A	N/A
rack1-server1		Generic System	No	Not assigned	Not assigned	rack1-server1	Not assigned	Not assigned
spine1		Spine	N/A	Deploy	Cisco NXOSv	spine1	64513	10.0.0.2/32
spine2		Spine	N/A	Deploy	Cisco NXOSv	spine2	64513	10.0.0.3/32
switch1-server1		Generic System	No	Not assigned	Not assigned	switch1-server1	Not assigned	Not assigned
switch2-server1		Generic System	No	Not assigned	Not assigned	switch2-server1	Not assigned	Not assigned

3. Change device details (such as deploy mode, S/N, hostname), as applicable. (You can also access configuration files from here: rendered, incremental, pristine).
4. Click the **Save** button to stage the changes.

## Edit Device Properties

1. From the blueprint, navigate to **Staged > Physical > Nodes** and select a node name (not the check box). (You can narrow your search with the drop-down lists for planes, pods, and racks as applicable, as of Apstra version 4.0.)
2. Click the **Properties** tab in the right panel.

The screenshot shows the Apstra interface for editing device properties. The top navigation bar includes 'Dashboard', 'Analytics', 'Staged', 'Uncommitted', 'Active', and 'Time Voyager'. Below this is a sub-navigation bar with 'Physical', 'Virtual', 'Policies', 'Catalog', 'Tasks', and 'Connectivity Templates'. The main area displays a list of nodes under the 'Nodes: All' tab. A table lists nodes with columns for 'Name', 'Role', and 'Group label'. The 'leaf1' node is selected. A red arrow points to the 'leaf1' node name with the text 'Click a node name...'. Another red arrow points to the 'Properties' tab in the right panel with the text '... to see its properties'. A third red arrow points to the 'Edit' button in the right panel with the text 'Edit'. A fourth red arrow points to the 'Card View' button with the text 'Card View'.

3. You can change device properties such as name, interface map, ASN, and loopback IP, depending on the node chosen. The attributes that can be edited have an **Edit** button associated with them. Change properties as applicable.

**NOTE:** If you changed leaf names in a leaf pair, the leaf pair name does not change. You can manually change the leaf pair name to correspond with the new leaf names. This is especially useful when assigning leaf pairs when you create virtual networks.

4. Click the **Save** button to stage the changes.

## Add/Remove Tags (Single Node)

1. From the blueprint, navigate to **Staged > Physical > Nodes** and select a node name (not the check box). (You can narrow your search with the drop-down lists for planes, pods, and racks as applicable, as of Apstra version 4.0.)
2. Click the **Tags** tab in the right panel.

The screenshot shows the Apstra interface with the following elements:

- Navigation Bar:** Dashboard, Analytics, Staged, Uncommitted, Active, Time Voyager.
- Sub-Menu:** Physical, Virtual, Policies, Catalog, Tasks, Connectivity Templates.
- Filters:** Nodes: All, Links: All, Selection, Build.
- Topology View:** Topology, Nodes, Links, Racks, Pods, Planes.
- Selected Plane:** All, Selected Pod: pod1, Selected Rack: All.
- Table:**

Name	Tags	Role	External?	Deploy Mode	Device Profile	Hostname	ASN	Loopback IPv4
leaf1		Leaf	N/A	Deploy	Cisco NXOSv	leaf1	64515	10.0.0.6/32
- Right Panel:** leaf1 (Role: Leaf, Group label: evpn-mlag), Add/Remove Tags button, No tags assigned message.

3. Click **Add/Remove Tags** to see tags that are in the blueprint catalog.
4. Select existing tag(s) to tag the node or create new one(s) that will be tagged to the node and added to the blueprint catalog.
5. Click **Update Tags** to stage the tagged nodes and return to the nodes view.

## Add/Remove Tags (Multiple Nodes)

1. From the blueprint, navigate to **Staged > Physical > Nodes** and check one or more check boxes for the node(s). (You can narrow your search with the drop-down lists for planes, pods, and racks as applicable, as of Apstra version 4.0.) The **Add/Remove Tags** button appears to the right of the other

buttons.

1. Select one or more nodes

2. Click to add/remove tags

1-10 of 10

Columns (9/17)

Page Size: 25

<input type="checkbox"/>	Name ▲	Tags	Role ▾	External? ▾	Deploy Mode ▾	Device Profile ▾	Hostname ▾	ASN ▾	Loopback IPv4 ▾
<input checked="" type="checkbox"/>	leaf1		Leaf	N/A	Deploy	Cisco NXOSv	leaf1	64515	10.0.0.6/32
<input checked="" type="checkbox"/>	leaf2		Leaf	N/A	Deploy	Cisco NXOSv	leaf2	64516	10.0.0.7/32
<input type="checkbox"/>	leaf3		Leaf	N/A	Deploy	Cisco NXOSv	leaf3	64517	10.0.0.8/32

2. Click the **Add/Remove Tags** button (fifth of five buttons above the nodes list). (To filter selection before adding tags, you can use the query.)
3. Select existing tag(s) to tag the node or create new one(s) that will be tagged to the node and added to the blueprint catalog.
4. Click **Add/Remove Tags** to stage the change and return to the nodes view.

## View Node's Static Routes

1. From the blueprint, navigate to **Staged > Physical > Nodes** and select a node name (not the check box). (You can narrow your search with the drop-down lists for planes, pods, and racks as applicable, as of Apstra version 4.0.)

2. Click the **Nodes** tab in the right panel.

The screenshot shows the network management interface with the **Nodes** tab selected. The interface includes a top navigation bar with tabs like Dashboard, Analytics, Staged, Uncommitted, Active, and Time Voyager. Below this is a secondary navigation bar with Physical, Virtual, Policies, Catalog, Tasks, and Connectivity Templates. The main area displays a table of nodes. A red arrow points to the 'leaf1' node in the table. Another red arrow points to the 'Node's Static Routes' link in the right panel.

1. Click a node name

2. Access static routes

Name	Tags	Role	External?	Deploy Mode	Device Profile	Hostname	ASN	Loopback IPv4
leaf1		Leaf	N/A	Deploy	Cisco NXOSv	leaf1	64515	10.0.0.6/32

3. Click **Node's Static Routes** to go to **Staged > Virtual > Static Routes** where you can see that node's static routes.

## Links (Staged)

### IN THIS SECTION

- Import Cabling Map | 470
- Export Cabling Map | 470
- Override Cabling (GUI) | 471
- Override Cabling (JSON) | 472
- Change Link Speeds | 473
- Fetch Discovered LLDP Data | 473
- Edit Link Properties | 474
- Add/Remove Tags on One Link | 475
- Add/Remove Tags (Multiple Links) | 476

**NOTE:** You can perform various tasks from the staged links view of the blueprint. You can perform additional links-related tasks, such as adding, editing and deleting links, from the ["staged topology" on page 430](#) view of the blueprint.

You can perform the following tasks from the staged links view:

- ["Import cabling map" on page 470](#)
- ["Export cabling map" on page 470](#)
- Override cabling using the ["Apstra GUI" on page 471](#) or ["JSON" on page 472](#)
- ["Change link speeds" on page 473](#)
- ["Fetch discovered LLDP data" on page 473](#)
- ["Edit link properties" on page 474](#)
- Add and remove tags on a ["single link" on page 475](#) or ["multiple links" on page 476](#)

From the blueprint, navigate to **Staged > Physical > Links** to go to staged links.

The screenshot shows the Apstra GUI interface. At the top, there are tabs for Dashboard, Analytics, Staged (selected), Uncommitted, Active, and Time Voyager. Below these, there are sub-tabs for Physical (selected), Virtual, Policies, Catalog, Tasks, and Connectivity Templates. The Physical tab is further divided into Topology, Nodes, Links (selected), Racks, Pods, and Planes. The Links tab shows a table of links with columns for Name, Role, Speed, Tags, Endpoint 1, and Endpoint 2. A toolbar above the table contains icons for Import cabling map, Export cabling map, Edit cabling map, Change link speeds, and Fetch discovered LLDP data. Red arrows point to these icons and the navigation path: Staged > Physical > Links. The table shows one link with the name 'leaf001\_001\_1<->leaf001\_001\_2((l3\_peer\_link)[1])' and a role of 'Leaf L3 Peer Link'.

1. Staged

2. Physical

3. Links

Fetch discovered LLDP data

Change link speeds

Edit cabling map

Export cabling map

Import cabling map

Filter selected by ☒ all ☐ selected only ☐ unselected only

0 selected	Name	Role	Speed	Tags	Endpoint 1				Endpoint 2			
					Name	Role	Interface	IPv4	Name	Role	Interface	IPv4
<input type="checkbox"/>	leaf001_001_1<->leaf001_001_2((l3_peer_link)[1])	Leaf L3 Peer Link	10G		leaf1	Leaf	Ethernet1/6	N/A	leaf2	Leaf	Ethernet1/7	N/A

## Links Example

In this example, each link is assigned a unique /31 subnet from the IP Pool, the smaller /31 IP is assigned to the spine interface, the larger /31 IP is assigned to the leaf interface. Subnets are assigned in increasing order in a spine-major order, that is, the links between spine1 and all leaves (in ascending order) are assigned subnets first, followed by links between spine2 and all leaves, and so on.

Physical

Virtual

Policies

Catalog

Settings

Nodes: All

Links: All

Topology

Nodes

Links

Layer

Build: Link IPs - Spines<>Leaves

Assigned

Not Assigned

1-18 of 18

Page Size: 25

Name	Role	Speed	Endpoint 1				Endpoint 2					
			Name	Role	Interface	IPv4	IPv6	Name	Role	Interface	IPv4	IPv6
I2_virtual_001_leaf1<->I2_virtual_001_server001[link][1]	Leaf to L2 Server	10G	I2_virtual_001_server001	L2 server	Not assigned	N/A	N/A	I2_virtual_001_leaf1	Leaf	Not assigned	N/A	N/A
I2_virtual_001_leaf1<->I2_virtual_001_server002[link][1]	Leaf to L2 Server	10G	I2_virtual_001_server002	L2 server	Not assigned	N/A	N/A	I2_virtual_001_leaf1	Leaf	Not assigned	N/A	N/A
I2_virtual_002_leaf1<->I2_virtual_002_server001[link][1]	Leaf to L2 Server	10G	I2_virtual_002_server001	L2 server	Not assigned	N/A	N/A	I2_virtual_002_leaf1	Leaf	Not assigned	N/A	N/A
I2_virtual_002_leaf1<->I2_virtual_002_server002[link][1]	Leaf to L2 Server	10G	I2_virtual_002_server002	L2 server	Not assigned	N/A	N/A	I2_virtual_002_leaf1	Leaf	Not assigned	N/A	N/A
I2_virtual_003_leaf1<->I2_virtual_003_server001[link][1]	Leaf to L2 Server	10G	I2_virtual_003_server001	L2 server	Not assigned	N/A	N/A	I2_virtual_003_leaf1	Leaf	Not assigned	N/A	N/A
I2_virtual_003_leaf1<->I2_virtual_003_server002[link][1]	Leaf to L2 Server	10G	I2_virtual_003_server002	L2 server	Not assigned	N/A	N/A	I2_virtual_003_leaf1	Leaf	Not assigned	N/A	N/A
I2_virtual_004_leaf1<->I2_virtual_004_server001[link][1]	Leaf to L2 Server	10G	I2_virtual_004_server001	L2 server	Not assigned	N/A	N/A	I2_virtual_004_leaf1	Leaf	Not assigned	N/A	N/A
I2_virtual_004_leaf1<->I2_virtual_004_server002[link][1]	Leaf to L2 Server	10G	I2_virtual_004_server002	L2 server	Not assigned	N/A	N/A	I2_virtual_004_leaf1	Leaf	Not assigned	N/A	N/A
spine1<->I2_virtual_001_leaf1[1]	Spine to Leaf	10G	I2_virtual_001_leaf1	Leaf	Not assigned	192.168.0.1/31	N/A	spine1	Spine	Not assigned	192.168.0.0/31	N/A
spine1<->I2_virtual_002_leaf1[1]	Spine to Leaf	10G	I2_virtual_002_leaf1	Leaf	Not assigned	192.168.0.3/31	N/A	spine1	Spine	Not assigned	192.168.0.2/31	N/A
spine1<->I2_virtual_003_leaf1[1]	Spine to Leaf	10G	I2_virtual_003_leaf1	Leaf	Not assigned	192.168.0.5/31	N/A	spine1	Spine	Not assigned	192.168.0.4/31	N/A
spine1<->I2_virtual_004_leaf1[1]	Spine to Leaf	10G	I2_virtual_004_leaf1	Leaf	Not assigned	192.168.0.7/31	N/A	spine1	Spine	Not assigned	192.168.0.6/31	N/A
spine2<->I2_virtual_001_leaf1[1]	Spine to Leaf	10G	I2_virtual_001_leaf1	Leaf	Not assigned	192.168.0.9/31	N/A	spine2	Spine	Not assigned	192.168.0.8/31	N/A
spine2<->I2_virtual_002_leaf1[1]	Spine to Leaf	10G	I2_virtual_002_leaf1	Leaf	Not assigned	192.168.0.11/31	N/A	spine2	Spine	Not assigned	192.168.0.10/31	N/A
spine2<->I2_virtual_003_leaf1[1]	Spine to Leaf	10G	I2_virtual_003_leaf1	Leaf	Not assigned	192.168.0.13/31	N/A	spine2	Spine	Not assigned	192.168.0.12/31	N/A
spine2<->I2_virtual_004_leaf1[1]	Spine to Leaf	10G	I2_virtual_004_leaf1	Leaf	Not assigned	192.168.0.15/31	N/A	spine2	Spine	Not assigned	192.168.0.14/31	N/A
spine1<->router001	To External Router	10G	Not assigned	External Router	N/A	Not assigned	N/A	spine1	Spine	Not assigned	Not assigned	N/A
spine2<->router002	To External Router	10G	Not assigned	External Router	N/A	Not assigned	N/A	spine2	Spine	Not assigned	Not assigned	N/A

## Import Cabling Map

1. From the table view or card view (Staged > Physical > Links) click the **Import cabling map** button (first of five buttons above the links list).
2. Either click **Choose File** and navigate to the file on your computer, or drag and drop the file onto the dialog window.
3. Click **Import** to import the cabling map and return to the links view.

## Export Cabling Map

Data center technicians may find a printed cabling map useful when wiring in switches, or remote network operators may find it useful for viewing IP assignments. It's available in CSV or JSON format, and you can copy the contents or download the file to your local computer.

1. From the blueprint, navigate to **Staged > Physical > Links** and click the **Export cabling map** button (second of five buttons above the links list) and select **JSON** or **CSV**.
2. Click **Copy** to copy the contents or click **Save As File** to download the file.
3. When you've copied or downloaded the cabling map, close the dialog to return to the **Links** view.

**NOTE:** You can also export cabling maps from the **Active > Physical > Links** view.

## Override Cabling (GUI)

Situations when you might want to edit the cabling map include:

- to use existing network cabling instead of recabling to the Apstra-prescribed cabling
- to change interface names or IP addresses in the existing network cabling map
- to specify a different port from the one that the Apstra cabling algorithm selected
- to avoid the use of a defective interface

Device profiles must be assigned to blueprint nodes.



**CAUTION:** Overriding Apstra-generated cabling can be disruptive to the network. Use with extreme caution. For assistance with production networks, please contact ["Juniper Support" on page 777](#).

1. From the blueprint, navigate to **Staged > Physical > Links** and click the **Edit cabling map** button (third of five buttons above the links list).
2. Change interface names and/or IP addresses, as applicable.
3. You can use **Batch clear override** to clear all Interface and IPv4/IPv6 values for a specific device type.

**NOTE:** To drop the override for either an interface name or IPv4/IPv6 address, submit an empty value in the corresponding field.

**Cabling Map Editor**

Fields to be cleared: ☒ Interface ☐ IPv4 ☐ IPv6

System roles: ☐ Spine ☒ Leaf

Query: Role = Spine to Leaf

1-6 of 6 Page Size: 25

6 selected	Role	Logical Link	Port Channel ID	Endpoint 1				Endpoint 2					
				Name	Role	Interface	IPv4	IPv6	Name	Role	Interface	IPv4	IPv6
<input checked="" type="checkbox"/>	Spine to Leaf	N/A	N/A	spine1	Spine	Ethernet2	172.16.0.0/31		leaf1	Leaf	Ethernet3	172.16.0.1/31	
<input checked="" type="checkbox"/>	Spine to Leaf	N/A	N/A	spine1	Spine	Ethernet3	172.16.0.2/31		leaf2	Leaf	Ethernet7	172.16.0.3/31	
<input checked="" type="checkbox"/>	Spine to Leaf	N/A	N/A	spine1	Spine	Ethernet1	172.16.0.4/31		leaf3	Leaf	Ethernet1	172.16.0.5/31	
<input checked="" type="checkbox"/>	Spine to Leaf	N/A	N/A	spine2	Spine	Ethernet2	172.16.0.6/31		leaf1	Leaf	Ethernet2	172.16.0.7/31	
<input checked="" type="checkbox"/>	Spine to Leaf	N/A	N/A	spine2	Spine	Ethernet3	172.16.0.8/31		leaf2	Leaf	Ethernet6	172.16.0.9/31	
<input checked="" type="checkbox"/>	Spine to Leaf	N/A	N/A	spine2	Spine	Ethernet1	172.16.0.10/31		leaf3	Leaf	Ethernet2	172.16.0.11/31	

Update

- Click **Update** to stage the changes.
- Click **Uncommitted** to see the diffs between **Staged** and **Active**.
- Click the **Commit** button to save the changes to the **Active** Blueprint.

## Override Cabling (JSON)

Situations when you might want to edit the cabling map include:

- to use existing network cabling instead of recabling to the Apstra-prescribed cabling
- to change interface names or IP addresses in the existing network cabling map
- to specify a different port from the one that the Apstra cabling algorithm selected
- to avoid the use of a defective interface

To change the cabling map with a JSON file, you'll export the JSON file, edit the file, then import it back into Apstra.



**CAUTION:** Overriding Apstra-generated cabling can be disruptive to the network. Use with extreme caution. For assistance with production networks, please contact ["Juniper Support" on page 777](#).

- From the blueprint, navigate to **Staged > Physical > Links** and click the **Export cabling map** button to see the dialog for exporting a cabling map.
- Select **JSON** and click **Save As File** to download the file.
- Change interface names (if\_name) and/or IP addresses (ipv4\_addr or ipv6\_addr) in the file, as applicable. Do not change any other fields. If you do, the changes will be ignored or they will result in an error message.

4. From the cabling map (Staged > Physical > Links) click the **Import cabling map** button to see the dialog for importing a cabling map.
5. Either click **Choose File** and navigate to the revised file on your computer, or drag and drop the file onto the dialog window.
6. Click **Import**.
7. Click **Uncommitted** to see the diffs between **Staged** and **Active**.
8. Click the **Commit** button to save the changes to the **Active** Blueprint, or click the **Revert** button to discard changes and return to the previous cabling map.

## Change Link Speeds

You can change link speeds on leaf-generic system, generic systems and MLAG peer links. To change link speeds on spine-leaf and superspine-spine you must ["change the rack" on page 480](#).

1. From the table view or card view (Staged > Physical > Links) click the **Change link speeds** button (fourth of five buttons above links list).
2. To search for specific links, click the query box, enter search criteria and click **Apply** to see results.
3. From the **Speed** drop-down list for the link to be changed, select the new speed.
4. Click **Update** to update the link and return to the cabling map.

**NOTE:** You can also change link speeds from the **Staged > Physical > Topology** view.

## Fetch Discovered LLDP Data

If you've already cabled up your devices, you can have Apstra discover your existing cabling instead of using the cabling map prescribed by Apstra. All system nodes in the blueprint must have system IDs assigned to them.



**CAUTION:** This is a disruptive operation. All links can potentially be renumbered.

1. From the blueprint, navigate to **Staged > Physical > Links** and click the **Fetch discovered LLDP data** button (fifth of five buttons above links list).
2. If staged data is *identical* to LLDP discovery results, you will see a message with that statement. Your actual cabling matches the Apstra cabling map. No further action is needed.
3. If staged data is *different* from LLDP discovery results, the message includes the number of links that are different.
4. Scroll to see details of the diffs (in red), or check the **Show only links with LLDP diff?** checkbox to see only the differences.

5. To accept the changes and update the map to match LLDP data, click **Update Stated Cabling Map from LLDP**. You might also need to reset resource group overrides.

## Edit Link Properties

If you have ["changed server names and/or hostnames" on page 463](#) for switches, any associated link names do not automatically update to match. This may cause confusion when reviewing an updated cabling map in the **Uncommitted** tab. You can change link names to match your other name changes. You can also change link IP for endpoints from here.

1. From the blueprint, navigate to **Staged > Physical > Links** and click the name of the link to change.
2. Go to the **Properties** tab in the right panel.
3. Depending on the link chosen, you can change link properties such as name and Link IP for endpoints. The attributes that can be edited have an **Edit** button associated with them. Change properties as applicable.

When you change link IP for an endpoint, you must remove link IP from the other endpoint first. Otherwise you will get validation error "User-specified link IPv4 addresses not in the same subnet".


spine1<->single\_rack\_001\_leaf1[1]  
Role: Spine to Leaf

Properties

Tags


✓

Name

spine1<->single\_rack\_0... 


✓

Link IP - single\_rack\_001\_leaf1

10.1.0.11/31 




⚠

Link IP (IPv6) - single\_rack\_001\_leaf1




✓

Link IP - spine1

10.1.0.10/31   

⚠

Link IP (IPv6) - spine1



When you assign new link IP to an endpoint, the link IP for the other endpoint is automatically assigned from the same subnet.

4. Click the **Save** button to stage the changes.

### Add/Remove Tags on One Link

1. From the blueprint, navigate to **Staged > Physical > Links** and select a link name (not the check box). (You can narrow your search with the drop-down lists for planes, pods, and racks as applicable, new in version 4.0.)

2. Click the **Tags** tab in the right panel.

The screenshot shows the 'Links' view in the network management interface. The top navigation bar includes 'Dashboard', 'Analytics', 'Staged', 'Uncommitted', 'Active', and 'Time Voyager'. Below this, there are tabs for 'Physical', 'Virtual', 'Policies', 'Catalog', 'Tasks', and 'Connectivity Templates'. A 'Find by tags' button is on the right. The main content area shows a list of links. The selected link is 'leaf001\_001\_1<->leaf001\_001\_2(i3\_peer\_link)[1]'. The right panel shows the 'Tags' tab with the 'Add/Remove Tags' button. A red arrow points from the link name in the table to this button. Another red arrow points from the 'Tags' tab to the same button.

1. Click a link name...

2. ...to access tags

Name	Role	Speed	Tags	Endpoint 1				Endpoint 2		
				Name	Role	Interface	IPv4	Name	Role	Interface
leaf001_001_1<->leaf001_001_2(i3_peer_link)[1]	Leaf L3 Peer Link	10G		leaf1	Leaf	Ethernet1/6	N/A	leaf2	Leaf	Ethernet1/6

3. Click **Add/Remove Tags** to see tags that are in the blueprint catalog.

4. Select existing tag(s) to tag the link or create new one(s) that will be tagged to the link and added to the blueprint catalog.

5. Click **Update Tags** to stage the tagged links and return to the links view.

### Add/Remove Tags (Multiple Links)

1. From the blueprint, navigate to **Staged > Physical > Nodes** and check one or more check boxes for the node(s). (You can narrow your search with the drop-down lists for planes, pods, and racks as applicable, new in version 4.0.) The **Add/Remove Tags** button appears to the right of the other five

buttons.

2. Click to add/remove tags

1. Select one or more links

	Name	Role	Speed	Tags	Endpoint 1				Endpoint 2			
					Name	Role	Interface	IPv4	Name	Role	Interface	IPv4
<input checked="" type="checkbox"/>	leaf001_001_1<->leaf001_001_2[3_peer_link][1]	Leaf L3 Peer Link	10G		leaf1	Leaf	Ethernet1/6	N/A	leaf2	Leaf	Ethernet1/7	N/A
<input checked="" type="checkbox"/>	leaf001_001_1<->leaf001_001_2[1]	Leaf Peer Link	10G		leaf2	Leaf	Ethernet1/6	N/A	leaf1	Leaf	Ethernet1/5	N/A

2. Click the **Add/Remove Tags** button (sixth of six buttons above the nodes list). (To filter selection before adding tags, you can use the query.)
3. Select existing tag(s) to tag the link or create new one(s) that will be tagged to the link and added to the blueprint catalog.
4. Click **Add/Remove Tags** to stage the change and return to the links view.

## Racks (Staged)

### IN THIS SECTION

- Change Rack Name | 479
- Add Rack | 479
- Export Rack Type | 479

- [Edit Rack | 480](#)
- [Delete Rack | 480](#)

You can control the growth of your network by adding, editing and deleting complete racks in a running blueprint. This flexible fabric expansion (FFE) feature is supported on both 3-stage and 5-stage Clos networks. (In 5-stage topologies, you can also ["add and remove pods" on page 481](#), and (as of version 4.0.1) ["increase the number of superpines per plane" on page 485](#). Although, you cannot add or remove planes themselves.) You can also ["change rack names" on page 479](#).

Rack types are *embedded* into blueprints from the global catalog. The rack type in the global catalog and the blueprint are initially the same. When you use FFE operations (for example to change link speeds, add generic systems or add/remove links) the rack type is modified and its timestamp is updated. The rack type name in the global catalog and the blueprint are still the same, but their contents are now different from each other.

From the blueprint, navigate to **Staged > Physical > Racks** to go to staged racks. You can change the default table view (as of Apstra version 4.0.1) to list view. You can filter racks to show **all**, **selected only**, or **unselected only** (as of Apstra version 4.0.1). The image below shows **all** racks in the list view.

1. Staged

2. Physical

3. Racks

4. Name

5. evpn\_mlag\_001\_001

6. Timestamp

7. 2021-09-18 12:38

8. Export rack type to global catalog

9. Edit rack

10. Delete rack

11. Rack Properties

12. Change rack name

13. Table List

14. Add Racks

15. Filter by: all selected only unselected only

Name	Pod Name	Rack Type	Leaf Count	Generic Systems Capacity	Actions
evpn_mlag_001_001	pod1	evpn-mlag 2021-09-18 12:38	1 MLAG pair	3 of 6 available	[Export] [Edit] [Delete]
evpn_single_001_001	pod1	evpn-single	1 single leaf	4 of 5 available	[Export] [Edit] [Delete]
evpn_single_002_001	pod2	evpn-single	1 single leaf	4 of 5 available	[Export] [Edit] [Delete]
evpn_single_002_002	pod2	evpn-single	1 single leaf	4 of 5 available	[Export] [Edit] [Delete]

To change rack name click it, then change name in rack properties in right panel

## Change Rack Name

You may want to use your own rack naming schema (for example, your rack names could be based on their physical locations). In these cases you can modify the existing rack names.

1. From the blueprint, navigate to **Staged > Physical > Racks** and select the rack that you want to change.
2. In **Rack Properties** (right panel) click the **Edit** button for the rack name.
3. Change the name and click the **Save** button to stage the change.

**NOTE:** You can also change rack names from the active blueprint.

## Add Rack

The easiest and fastest way to expand your network is to add a rack.

1. From the blueprint, navigate to **Staged > Physical > Racks** and click the **Add Racks** button (+).
2. If your blueprint is for a 5-stage topology, select the pod that needs a rack.
3. From the **Rack Type** drop-down list, select a rack type to preview and validate. (To go to a different preview, select a different rack type.)
4. Enter the number of racks to add.
5. If you uncheck **Keep existing cabling in the fabric after change**, port assignments are re-calculated and you may need to re-cable. When in doubt, leave this box checked.
6. Click **Add** to stage the rack addition and return to the list view.
7. ["Assign device profiles" on page 422](#) and ["system IDs" on page 423](#) (serial numbers) to the new rack(s).
8. Commit the changes to your blueprint to configure the rack(s) and complete the fabric expansion.

## Export Rack Type

For changes that cannot be made directly in the blueprint rack, you can export the rack type to the global catalog and update it there. Then from the blueprint, edit the rack to use the modified rack type from the global catalog. For more information, see ["Edit Rack" on page 480](#).

1. To export the rack type from the blueprint, navigate to **Staged > Physical > Racks** and click the **Export rack to global catalog** button (first of three buttons).
2. Enter a unique **Rack Type** name.
3. Click **Export** to export the rack type to the global catalog.

Navigate to **Design > Rack Types** and edit the rack type from there.

## Edit Rack

You can change running racks while preserving many rack characteristics (such as leaf/server/link names and virtual network (VN) endpoints if labels have not changed). To edit a rack, you export its rack type to the global catalog with a unique name, update that rack type in the global catalog, then, in the blueprint, select the updated rack type to replace the one in the blueprint.

VN endpoints remain as long as the server and link labels between the old and new rack type are the same.



**CAUTION:** If it's not possible to retain VN endpoints, you must re-assign them. Review pending changes on the **Uncommitted** tab before committing. If you don't want to commit the changes, you can **revert** them.

**NOTE:** If you don't need to retain rack details, we recommend that you ["delete the rack" on page 480](#) and ["add a replacement rack" on page 479](#), instead of editing the rack.

Typically, a rack edit operation involves the following steps:

1. Ensure that the global catalog or the blueprint includes a suitable ["rack type" on page 101](#) for replacement.
2. From the blueprint, navigate to **Staged > Physical > Racks** and click the **Edit** button for the rack to edit (second of three buttons).
3. From the **New Rack Type** drop-down list, select the required rack type.
4. If you added new devices, ["assign device profiles" on page 422](#) and ["system IDs" on page 423](#) (serial numbers) to them.



**CAUTION:** This action is service-impacting since it requires a full config push.

5. You have the option of reviewing the **Incremental Config** to see the changes that will be pushed to the device(s). If devices were assigned, a full config push is performed.
6. Commit the changes to the blueprint to push all required configuration changes to the devices in the modified rack.

## Delete Rack

Before deleting a rack that has live traffic on it, you may want to take its devices out-of-service gracefully by draining them. For information, see ["Drain Device Traffic" on page 152](#).

1. To delete a rack from the blueprint, navigate to **Staged > Physical > Racks** and click the **Delete** button for the rack to delete (third of three buttons).
  - If you will be adding a rack back into your system, leave the **Keep existing cabling in the fabric after change** box checked.
  - If you will *not* be replacing the rack in your system, uncheck the **Keep existing cabling in the fabric after change** box. Otherwise, the intent will not match the actual topology anymore, and you will encounter anomalies, such as for cabling and BGP.
2. Click **Delete Rack** to stage the deletion and return to the list view.
3. Commit the changes to the blueprint. Configuration on any running devices will be erased and the devices will be ready to be decommissioned.

## Pods (Staged)

### IN THIS SECTION

- [Change Pod Name | 483](#)
- [Add Pod | 483](#)
- [Delete Pod | 484](#)

If you're building a 5-stage Clos network, the physical view has a tab for **Pods**. These details come from the ["pod-based template" on page 110](#) that you used to create your blueprint. From the **Pods** tab, you can change pod names, and add and remove entire pods (new in version 4.0). The ability to add pods to your running blueprint allows for organic growth of large networks without having to pre-design every pod. For more information about building 5-stage topologies, see the ["5-stage Clos Architecture" on page 802](#).

From the blueprint, navigate to **Staged > Physical > Pods** to go to staged pod details. You can search for specific nodes or links and select a layer to see build details, deploy modes, and uncommitted changes.

1. Staged

2. Physical

3. Pods

Select layer to see build details, deploy modes, and uncommitted changes

Click rack type name to see preview

Layer: Uncommitted Changes

1-2 of 2 Page Size: 25

pod1

Capacity:

Query: All 1-5 of 12

Name	Type	Used	Available
evpn-mlag	global	0	1
evpn-mlag	embedded	1	1
evpn-single	global	0	2
evpn-single	embedded	1	2
L2 MLAG 1x access	global	0	1

pod2

Capacity:

Query: All 1-5 of 14

Name	Type	Used	Available
evpn-mlag	global	0	1
evpn-mlag	embedded	0	1
evpn-single	global	0	3
evpn-single	embedded	2	3
L2 ESI 2x Links	global	0	1

## Change Pod Name

1. From the blueprint, navigate to **Staged > Physical > Pods** and click the pod name to change.

The screenshot shows the 'Staged > Physical > Pods' view. The 'Pod Properties' panel on the right shows the 'Name' field for 'pod1' with an edit icon. Red arrows indicate the steps: 1. Click pod name... (pointing to 'pod1' in the list) and 2. ...to change pod name (pointing to the edit icon).

**Pod 1 Resources:**

Name	Type	Used	Available
evpn-mlag	global	0	1
evpn-mlag	embedded	1	1
evpn-single	global	0	2
evpn-single	embedded	1	2
L2 MLAG 1x access	global	0	1

**Pod 2 Resources:**

Name	Type	Used	Available
evpn-mlag	global	0	1
evpn-mlag	embedded	0	1
evpn-single	global	0	3
evpn-single	embedded	2	3
L2 ESI 2x Links	global	0	1

2. In **Pod Properties** (right panel) click the **Edit** button for the name.
3. Change the name and click the **Save** button to stage the change.
4. ["Commit" on page 648](#) the changes to your blueprint to activate the name change.

## Add Pod

1. From the blueprint, navigate to **Staged > Physical > Pods**, and click the **Add Pods** button (+) (center-left).

- From the **Pod Type** drop-down list, select a pod type to preview and validate. To go to a different preview, select a different pod type.

## Add Pods

### Parameters

Pod Type \*

☒ Show available only

Select pod type from drop-down list

evpn\_pod\_rbt\_pod1 (embedded) ✕

evpn_pod_rbt_pod1	embedded
evpn_pod_rbt_pod2	embedded
L2 Pod Mlag	global 2021-10-22 11:53
L2 Pod Single	global 2021-10-22 11:53

### Pod Preview

Expanded View

Compact View

Template Parameters

Topology Preview

Structure

Superspine Connectivity

Policies

Name	evpn_pod_rbt_pod1
Type	RACK BASED

Click for details

- Enter the number of pods to add.
- Click **Add** to stage the pod addition and return to the list view.
- "Commit" on page 648 the changes to your blueprint to complete the fabric expansion.

## Delete Pod

When you delete a pod all of its devices are removed from the blueprint; this is potentially highly impacting. Before deleting a pod that has live traffic on it, you may want to take its devices out-of-service gracefully by draining them. For more information, see the ["device draining guide" on page 152](#).

- From the blueprint, navigate to **Staged > Physical > Pods**.

2. Using the check boxes, select the pod(s) to delete. (You cannot delete all pods in a blueprint.)

Dashboard

Analytics

Staged

Uncommitted

Active

Physical

Virtual

Policies

Catalog

Tasks

Connectivity Templates

Nodes: All

Links: All

Topology

Nodes

Links

Racks

Pods

Planes

Layer

Uncommitted Changes

Has Uncommitted Changes

1-2 of 2

Page Size: 25

☒

pod1

Capacity:

Query: All1-5 of 12

Name	Type	Used	Available
evpn-mlag	global	0	1
evpn-mlag	embedded	1	1
evpn-single	global	0	2
evpn-single	embedded	1	2
L2 MLAG 1x access	global	0	1

☐

pod2

Capacity:

Query: All1-5 of 14

Name	Type	Used	Available
evpn-mlag	global	0	1
evpn-mlag	embedded	0	1
evpn-single	global	0	3
evpn-single	embedded	2	3
L2 ESI 2x Links	global	0	1

2. Click Delete button

1. Select pod to delete

3. Click the **Delete** button (trash can) for the pod to delete.
4. Click **Delete Pod** to stage the deletion and return to the list view.
5. **"Commit "** on page 648 the changes to your blueprint. Configuration on any running devices is erased and the devices are ready to be decommissioned.

## Planes (Staged)

### IN THIS SECTION

Planes Overview | 486

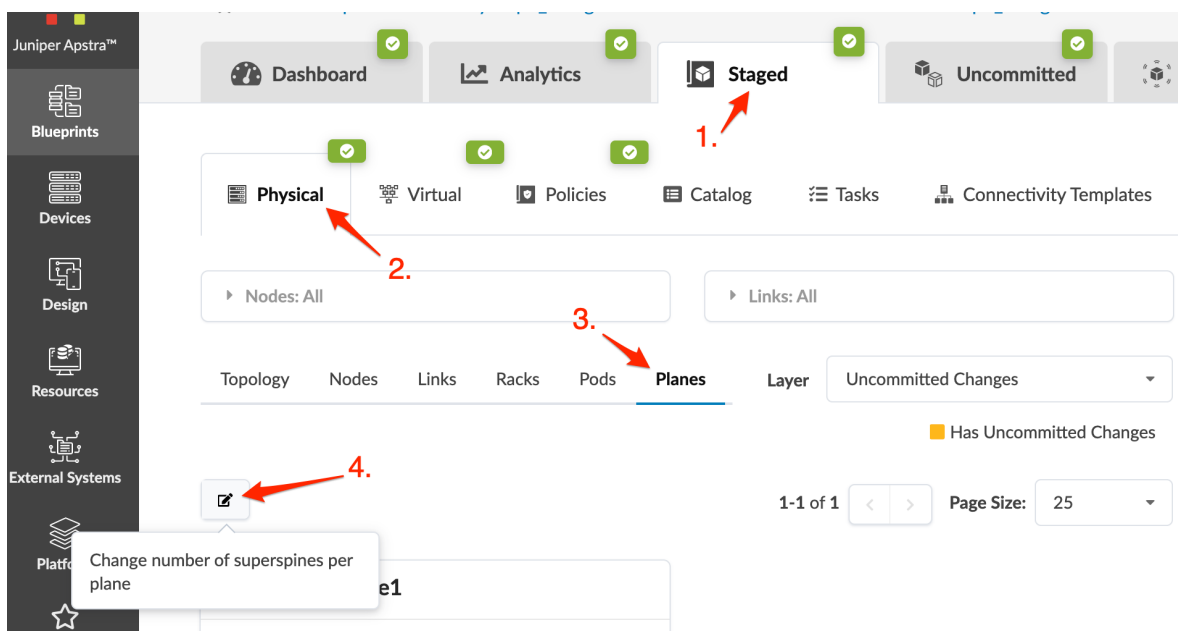
## Planes Overview

Planes are groups of superspines. Each 5-stage topology consists of one or more planes

You can add superspines to planes in 5-stage Clos networks (as of Apstra version 4.0.1). The maximum number of super spines is limited by the number of available spine ports of type **superspine**. When you add superspines, additional superspine nodes are created with the same logical devices that are used in the existing blueprint template. You must manually ["assign the interface maps for the device profiles"](#) on [page 422](#) of each new node. When the devices are physically ready, you can ["assign"](#) on [page 423](#) each node with their corresponding system IDs (serial numbers). When you ["commit pending changes"](#) on [page 648](#), the superspines are configured and they become part of the control and data plane, taking part of forward traffic between pods.

## Add Superspines per Plane

1. From the 5-stage blueprint, navigate to **Staged > Physical > Planes** and click the **Change number of superspines per plane** button.



2. Enter a number higher than the existing one.
3. Click **Update** to increase the number of superspines and return to the **Planes** view.

Stage the superspine as described in the overview, then commit the changes to activate them.

## Virtual

### IN THIS SECTION

- [Stage Virtual Resources | 487](#)
- [Virtual Networks | 488](#)
- [Routing Zones \(Virtual\) | 496](#)
- [Protocol Sessions \(Virtual\) | 501](#)
- [Data Center Interconnect \(DCI\) / Remote EVPN Gateways \(Virtual\) | 502](#)
- [Virtual Infra \(Virtual\) | 514](#)
- [Endpoints Overview \(Virtual\) | 578](#)

## Stage Virtual Resources

### IN THIS SECTION

- [Update Virtual Resources Assignments | 487](#)
- [Reset Virtual Resource Group Overrides | 488](#)

You can assign resources, release previously used resources and go to resource pool management from the virtual build panel. The resource assignment section has a convenient shortcut button, **Manage resource pools**, that takes you to resource pool management. From there, you can monitor resource usage and create additional resource pools, as needed.

### Update Virtual Resources Assignments

A red status indicator in the build panel means that resources need to be assigned. Resources may include virtual network SVI subnets for routing zones, SVI subnets for MLAG domain, SVI subnet for virtual networks, VNI Virtual Network IDs, and VTEP IPs.

1. From the blueprint, navigate to **Staged > Virtual > Virtual Networks > Build**. (The build panel is on the right side.)

1. Staged

2. Virtual

3. Virtual Networks

4. Build

5. SVI Subnets - Virtual Networks

6. Update assignments

When resources are staged, status indicator turns green

Reset resource group overrides

Manage resource pools

Name	Routing Zone	Type	VN ID	Assigned to	IPv4 Connectivity	IPv4 Subnet	IPv6 Connectivity	IPv6 Subnet	Actions
vnet_10_on_rack_1_001_leaf_pai	default	VLAN	10	1 nodes	Enabled	Not assigned	Disabled	N/A	
vnet_10_on_rack_2_001_leaf1	default	VLAN	10	1 nodes	Enabled	Not assigned	Disabled	N/A	

- Red status indicators mean that resources need to be assigned. Click a red status indicator, then click the **Update assignments** button.
- Select a pool from which to pull the resources, then click the **Save** button. The required number of resources are automatically assigned to the resource group. When the red status indicator turns green, the resource assignment has been successfully staged.

### Reset Virtual Resource Group Overrides

Certain blueprint operations require resource allocations to be retained even when a device has been removed from a blueprint. For example, if you decide to reuse a device, previously allocated resources need to be re-used as well. If resources were not retained, build errors may occur because the expected resources would no longer be available to the device. To minimize build errors, resource allocations persist by default. If you know that a device won't be re-instated, you don't need to keep its resources allocated to it. Click the **Reset resource group overrides** button to reset the resource group and release the resources.

## Virtual Networks

### IN THIS SECTION

- Virtual Networks Overview | 489
- Create Virtual Network | 494
- Edit Virtual Network | 496
- Delete Virtual Network | 496

## Virtual Networks Overview

You can create an overlay network in an Apstra blueprint by creating virtual networks (VN)s to group physically separate endpoints into logical groups. These collections of Layer 2 forwarding domains can be either VLANs or VXLANs.

VLANs have the following characteristics:

- Single rack (rack-local)
- Single leafs or leaf pairs
- Can be deployed in Layer 2-only mode (for example, isolated cluster networks for database replication)
- Can be deployed with Layer 3 gateway (SVI) IP address on rack leaf, hosted with or without first-hop redundancy

VXLANs have the following characteristics:

- Fabric-wide for ubiquitous Layer 2 (inter-rack)
- Combination of single rack leafs or leaf pairs (MLAG)
- Can be deployed in Layer 2-only mode
- Can be deployed with Layer 3 gateway functionality
- The control plane selected (Static VXLAN Routing or MP-EBGP EVPN) when configuring the template for your blueprint determines what is configured in the VN. (MP-EBGP EVPN provides a control plane for VXLAN routing.)
- VXLAN-EVPN capabilities for VXLAN VNs are dependent on network device makes and models. For more information see the `evpn_support_addendum:Apstra EVPN Support Addendum`.

For complete VN feature compatibility for supported Network Operating Systems (NOS), see the Apstra Feature Matrix for the applicable release (in the Reference <reference> section). For detailed capability information for a device, contact your network device vendor or ["Juniper Support" on page 777](#).

VNs contain the following details:

Table 26: Virtual Network Parameters

Name	Description
Type	<ul style="list-style-type: none"><li>• VLAN (rack-local VN)</li><li>• VXLAN (EVPN) (inter-rack VN)</li></ul>
Name	32 characters or fewer. Underscore, dash, and alphanumeric characters only.
Routing Zone	<ul style="list-style-type: none"><li>• VLAN - default routing zone only</li><li>• VXLAN - default routing zone or user-defined routing zone</li><li>• Default routing zone is used for the underlay network.</li></ul>

Table 26: Virtual Network Parameters *(Continued)*

Name	Description
Default VLAN ID (VLAN only)	<ul style="list-style-type: none"> <li>Layer 2 VLAN ID on the switch that the VN is assigned to.</li> <li>If left blank, it's auto-assigned from static pool (2-4094).</li> <li>If you assign it, we don't recommend assigning VLAN ID 1 for active VNs.</li> <li>Cisco NX-OS reserves VLAN IDs 3968-4094.</li> <li>Cumulus VLAN-aware Bridge Mode reserves: <ul style="list-style-type: none"> <li>for Cumulus 3.7: 3000-3999</li> <li>for Cumulus 4.1: 3600-3999</li> </ul> </li> <li>Arista reserves 1006-4094 for internal VLANs for routed ports. You can modify "reserved" VLAN ID range with the EOS <code>vlan internal allocation policy</code> configuration command. You can apply it to all EOS devices using a <b>SYSTEM</b> configlet before configuring and deploying VNs. <pre> 12-virtual-ext-002-leaf1(config)#vlan internal allocation policy ascending range 3001 3999 12-virtual-ext-002-leaf1(config)#exit 12-virtual-ext-002-leaf1#show vlan internal allocation policy Internal VLAN Allocation Policy: ascending Internal VLAN Allocation Range: 3001-3999 12-virtual-ext-002-leaf1# </pre> </li> <li>Using reserved VLAN IDs may cause deployment errors, but not build errors.</li> </ul>
VNI(s) (VXLAN only)	Layer 2 VXLAN ID on the switch that the VN is assigned to. If left blank, it's auto-assigned from resource pools. Create up to 40 VNs at once by entering ranges or individual VNI IDs separated by commas (for example: 5555-5560, 7777). Commit the first 40 VNs before creating additional ones.
Set same VLAN ID on all leafs (VXLAN only)	Option to use same VLAN ID on all leafs
DHCP server	Enabled/Disabled - DHCP relay forwarder configuration on SVI. Implies L3 routing on SVI

Table 26: Virtual Network Parameters *(Continued)*

Name	Description
IPv4 Connectivity	Enabled/Disabled - for SVI routing
IPv4 subnet (if connectivity is enabled)	<ul style="list-style-type: none"> <li>IPv4 subnet - (for example: 192.168.100.0/24) (can't use batching VLANs)</li> <li>IPv4 CIDR length - automatically assigns a subnet with the specified length (for example: /26)</li> <li>If left blank, it's auto-assigned a /24 subnet network from resource pools</li> </ul>
Virtual Gateway IPv4	The IPv4 address, if enabled
IPv6 Connectivity	Enabled/Disabled - IPv6 connectivity for SVI routing. IPv6 must be enabled in blueprint. If template used IPv4 spine-to-leaf link types, IPv6 can't be used in default routing zone and for VLAN type VNs.
IPv6 subnet (if connectivity is enabled)	<ul style="list-style-type: none"> <li>IPv6 subnet (for example: 2001:4de0::/64)</li> <li>IPv6 CIDR length - automatically assigns a subnet with the specified length (for example: /56)</li> <li>If left blank, it's auto-assigned a /64 subnet network from resource pools.</li> <li>If assigned automatically, the IP is derived from the assigned VNs SVI pools.</li> <li>To assign multiple VLAN networks, leave blank or specify CIDR length.</li> </ul>
Virtual Gateway IPv6	The IPv6 address, if enabled
Create connectivity templates for	<ul style="list-style-type: none"> <li>Tagged</li> <li>Untagged</li> </ul>
Assigned to	The racks that the VN is assigned to. For more information, see table below.

Table 27: Virtual Network Rack (or Pod) Details

Assigned To Details	Description
Pod Name (5-stage)	5-stage Clos networks include pods, and leaf devices within each pod can be selected to extend VN to those devices.
Bound to	The racks assigned. For MLAG racks, the leaf pair is shown. For VLANs, if more than one rack is selected, multiple rack-local VLAN-based VNs are created.
Link Labels	Label assigned to rack (for example, ext-link-1, single-link, single-link, ext-link-0)
VLAN ID	Can be used for batch creating VNs
IPv4 mode / IPv6 Mode (aka Secondary IP allocation mode)	<p>This is in the context of virtual network secondary IP allocation only. IRB/SVI rendering is not related, and Virtual gateway IP is controlled by a different flag on its own.</p> <ul style="list-style-type: none"> <li>• <b>Disabled</b> - The secondary IP address is not rendered on the virtual network. If a connectivity template requires it, you'll get a build error.</li> <li>• <b>Enabled</b> - The secondary IP address is rendered if Apstra determines it's needed (based on connectivity templates)</li> <li>• <b>Forced</b> - The secondary IP address is rendered regardless of whether a connectivity template needs it.</li> <li>• <b>Link Local</b> (IPv6 only) - used when an IPv6 link-local address is required on the SVI. Usually for a BGP unnumbered scenario without explicit IPv6 address allocation for the SVI.</li> </ul>
IPv4 Address / IPv6 Address	Can be specified to set the first-hop-redundancy IP address for the SVI (VRRP, VARP and so on). If left blank, the SVI IP address is assigned from the selected pool. When you bind an EVPN connectivity template to a Layer 2 application point, the SVI IP address is used as the source / destination for the BGP session, static routes and so on.

From the blueprint, navigate to **Staged > Virtual > Virtual Networks** to go to the VN list view. You can create, edit and delete VNs.

1. Staged

2. Virtual

3. Virtual Networks

Click VN name for details

Name	Routing Zone	Type	VN ID	Assigned to	IPv4 Connectivity	IPv4 Subnet	IPv6 Connectivity	IPv6 Subnet	Actions
vnet_10_on_rack_1_001_leaf1	default	VLAN	10	1 nodes	Enabled	172.16.1.0/24	Disabled	N/A	
vnet_10_on_rack_2_001_leaf1	default	VLAN	10	1 nodes	Enabled	172.16.2.0/24	Disabled	N/A	

## Create Virtual Network

1. From the blueprint, navigate to **Staged > Virtual > Virtual Networks** and click **Create Virtual Networks**.
2. Select the VN type (VLAN, VXLAN) and enter a name.

Create Virtual Network

Virtual Network Parameters

Type  
☐ VLAN ☒ VXLAN

Will create single VXLAN for all selected nodes

Name \*

Routing Zone

VNI(s)

Set same VLAN ID on all leafs? ☐

DHCP Service ☒ Disabled ☐ Enabled

IPv4 Connectivity ☐ Disabled ☒ Enabled

IPv4 Subnet

Virtual Gateway IPv4 Enabled? ☒

Virtual Gateway IPv4

Create Connectivity Templates for ☐ Tagged ☐ Untagged

☐ Create Another?

3. Select the "routing zone" on page 496 to associate with the VN(s). (VLANs must use the default routing zone.)
4. If you're creating VLANs, specify the default VLAN ID(s). If you're creating VXLANs, specify VNIs and VLAN ID (on leafs). See overview above for details.

5. If you enable **DHCP Service**, enter a subnet. A DHCP relay forwarder is configured on the SVI. This option also implies Layer 3 routing on this SVI. (You assign the DHCP server in the routing zone.)
6. If you enable **IPv4 Connectivity**, enter a subnet, unless you're batch creating VNs. Then enter an IPv4 CIDR length, or leave subnet blank to allow auto-assignment.
7. If you enable **Virtual Gateway IPv4**, enter an IPv4 address.
8. If IPv6 is enabled in the blueprint (Policies > Fabric Addressing Policy), and you enable **IPv6 Connectivity**, enter a subnet, unless you're batch creating VNs. Then enter an IPv6 CIDR length, or leave subnet blank to allow auto-assignment.
9. If you enable **Virtual Gateway IPv6**, enter an IPv6 address.
10. To create connectivity templates for the VN(s), check the box for **Tagged** and/or **Untagged**, as applicable.
11. Select and configure racks that the VN is to be assigned to. See overview above for details.

**Create Virtual Network**
✕

---

Assigned To

▸ Query: All

1-2 of 2

Page Size: 25

☐	Bound To	Link Labels	VLAN ID	IPv4 Mode	IPv4 Address
<input checked="" type="checkbox"/>	rack_1_001_leaf_pair1	link	From resource pool	<div style="border: 1px solid #ccc; padding: 2px;">           rack_1_001_leaf1            Enabled ▾  <b>Enabled</b>            Forced         </div>	<div style="border: 1px solid #ccc; padding: 2px;">           rack_1_001_leaf1            From resource pool         </div> <div style="border: 1px solid #ccc; padding: 2px; margin-top: 5px;">           rack_1_001_leaf2            From resource pool         </div>
<input type="checkbox"/>	rack_2_001_leaf1	ext-link-1, link	From resource pool	Enabled ▾	From resource pool

☐ Create Another?
 

Create

12. Click **Create** to stage the VN and return to the list view.
13. Assign IPv4 (IPv6) resources for SVI subnets. Navigate to **Staged > Virtual > Virtual Networks** and ["assign resources" on page 420](#) in the **Build** panel (right-side).
14. For VXLAN only: Assign VTEP IPs. Navigate to **Staged > Virtual > Virtual Networks** and assign resources in the **Build** panel (right-side). (You can display the VTEPs list in the nodes table (Staged > Physical > Nodes). Select the type of VTEP to display from the **Columns** drop-down list (above the table).)
  - **Single Leaf Nodes** require one VTEP IP and an anycast VTEP IP for all switches in the VN.
  - **MLAG Leaf-pair Nodes** require a common VTEP IP for the leaf-pair and an anycast VTEP IP for all switches in the VN.
15. To deploy changes to the active blueprint, click the ["Uncommitted" on page 648](#) tab to review and commit (or discard) changes.

## Edit Virtual Network

1. From the blueprint, navigate to **Staged > Virtual > Virtual Networks** and click the name of the VN to edit.
2. Click the **Edit** button (on the right) and make your changes.
3. Click **Update** to stage the changes and return to the list view.

## Delete Virtual Network

1. From the blueprint, navigate to **Staged > Virtual > Virtual Networks** and click the **Delete** button (trash can) for the VN to delete.
2. Click **Delete** to stage the deletion and return to the list view.

## Routing Zones (Virtual)

### IN THIS SECTION

- [Routing Zone Overview | 496](#)
- [Create Routing Zone | 498](#)
- [Assign Resources to Routing Zone | 499](#)
- [Assign DHCP to Routing Zone | 500](#)
- [Edit Routing Zone | 500](#)
- [Delete Routing Zone | 501](#)

## Routing Zone Overview

A routing zone is an L3 domain, the unit of tenancy in multi-tenant networks. You create routing zones for tenants to isolate their IP traffic from one another, thus enabling tenants to re-use IP subnets. In addition to being in its own VRF, each routing zone can be assigned its own DHCP relay server and external system connections. You can create one or more virtual networks within a routing zone, which means a tenant can stretch its L2 applications across multiple racks within its routing zone. For virtual networks with Layer 3 SVI, the SVI is associated with a Virtual Routing and Forwarding (VRF) instance for each routing zone isolating the virtual network SVI from other virtual network SVIs in other routing zones. If you're using multiple routing zones, external system connections must be from leaf switches in the fabric. Routing between routing zones must be accomplished with external systems. All SVIs configured for virtual networks in this zone are in the default VRF. This is the same VRF used for the underlay or fabric network routing between network devices. All blueprints include a default routing policy. The number of routing zones is limited only by the network devices being used.

Routing zones include the following details:

VRF Name	15 characters or fewer. Underscore, dash and alphanumeric characters only
Type	L3 Fabric or EVPN
VLAN ID	Used for VLAN tagged Layer 3 links on external connections. Leave this field blank to have it automatically assigned from a static pool in the range of 2-4094), or enter a specific value.
VNI	VxLAN VNI associated with the routing zone. Leave this field blank to have it automatically assigned from a resource pool, or enter a specific value.
Route Target	Only EVPN routing zones use route targets. The rendered EVPN L3-VNI route target represents the built-in, automatic route target that is associated with the EVPN routing zone VRF. When using EVPN remote gateway features for Datacenter Interconnect, this route target must be imported by the EVPN fabric external to this fabric. This route target is composed of "<VNI_ID>:1" where "1" is hard-coded. If route target is not assigned, then a VNI must be assigned.
DHCP Servers	
Routing Policies	Non-EVPN blueprints must use the default policy. EVPN blueprints can use non-default policies. For more information, see <a href="#">"Routing Policies" on page 599</a> .
Route Target Policies	<ul style="list-style-type: none"> <li>• Import Route Targets</li> <li>• Export Route Targets</li> </ul>
Resources	
Virtual Networks	
Interfaces	

From the blueprint, navigate to **Staged > Virtual > Routing Zones** to go to the routing zones list view. You can create, edit, and delete routing zones and assign DHCP servers to them.

The screenshot shows the network management interface. The top navigation bar includes 'Dashboard', 'Analytics', 'Staged', 'Uncommitted', 'Active', and 'Time Voyager'. The left sidebar has 'Physical', 'Virtual', 'Policies', 'Catalog', 'Tasks', and 'Connectivity Templates'. Under 'Virtual', there are 'Virtual Networks', 'Routing Zones', 'Floating IPs', 'Static Routes', 'Protocol Sessions', 'Remote EVPN Gateways', 'Virtual Infra', and 'Endpoints'. The 'Routing Zones' section is highlighted. A 'Create Routing Zone' button is visible in the top right. Below the navigation, there is a search bar with 'Query: All', a page size dropdown set to 25, and a table of existing routing zones. To the right of the table is a 'Build' panel with a list of resources to be assigned to the new routing zone.

VRF Name	Type	VLAN ID	Route Target	VNI	DHCP Servers
blue	EVPN	201	20002:1	20002	198.51.100.2 fc01:a05:198:51:100::2
default	L3 Fabric	N/A	N/A	N/A	198.51.100.2 fc01:a05:198:51:100::2
red	EVPN	200	20001:1	20001	198.51.100.2 fc01:a05:198:51:100::2

**Build**

- 2/2 blue: Leaf L3 Peer Links
- 5/5 blue: Leaf Loopback IPs
- 4/4 blue: To Generic Link IPs
- 2/2 red: Leaf L3 Peer Links
- 5/5 red: Leaf Loopback IPs
- 4/4 red: To Generic Link IPs
- 2/2 EVPN L3 VNIs

## Create Routing Zone

If your blueprint is using **MP-EBGP EVPN** overlay control protocol, you can create routing zones. If it's using **Static VXLAN**, you must use the default routing zone. (Overlay control protocol is specified in ["templates" on page 110.](#))

1. From the blueprint, navigate to **Staged > Virtual > Routing Zones** and click **Create Routing Zone**.
2. Enter a VRF name (15 characters or fewer).
3. You can leave the remaining fields as is to use default values and have resources assigned from pools, or you can configure them manually. See the routing zone overview for details.
4. Click **Create** to create the routing zone and return to the list view.

Assign resources (leaf loopback IPs, leaf L3 peer links) to the new routing zone.

## Assign Resources to Routing Zone

Each leaf network device in each routing zone requires a loopback IP. If IPv6 is enabled on the blueprint, you must also assign IPv6 addresses to the routing zone. After you've assigned connectivity templates to your external generic systems, you'll also need to assign IP addresses.

1. From the blueprint, navigate to **Staged > Virtual > Routing Zones**.
2. Red status indicators in the **Build** panel (on the right) indicate that resources need to be assigned. Click a red indicator and click the **Update assignments** button.

The screenshot shows the 'Routing Zones' section of the interface. The 'Build' panel on the right lists resources and their assignment status:

- blue: Leaf L3 Peer Links (2/2, green)
- blue: Leaf Loopback IPs (5/5, green)
- blue: To Generic Link IPs (4/4, green)
- red: Leaf L3 Peer Links (2/2, green)
- red: Leaf Loopback IPs (5/5, green)
- red: To Generic Link IPs (0/4, red)

A red arrow points to the 'Update assignments' button (a square with a pencil icon) next to the 'red: To Generic Link IPs' item, which has a red status indicator and '0/4' assigned.

VRF Name	Type	VLAN ID	Route Target	VNI	DHCP Servers
blue	EVPN	201	20002:1	20002	198.51.100.2 fc01:a05:198:51:100::2
default	L3 Fabric	N/A	N/A	N/A	198.51.100.2 fc01:a05:198:51:100::2
red	EVPN	200	20001:1	20001	198.51.100.2 fc01:a05:198:51:100::2

3. Select a pool from which to pull the resources, then click the **Save** button. (For information about IP address pools, see ["IP Pools" on page 395.](#)) When the red status indicator turns green, the required resources were successfully assigned.
4. Repeat the steps to assign resources from pools until all required resources have been assigned.

**NOTE:** You can also assign individual IP addresses to links by clicking the name of the routing zone in the list view, scrolling down to the **Interfaces** section, clicking the **Edit IP addresses**

button, and entering them from there.

Interfaces 4

1-4 of 4 < >

Page Size: 25

Edit IP Addresses by ☒ all ☐ selected only ☐ unselected only

	Routing Zone	VLAN ID	Name	Role	Interface	IPv4 Address	IPv4 Address Type	Endpoint 2	Interface 2
<input checked="" type="checkbox"/>	red	2	leaf1	Leaf	Ethernet1/1.3	Not assigned	Numbered	rtr_leaf1_leaf2	Generic System n/a
<input checked="" type="checkbox"/>	red	2	leaf2	Leaf	Ethernet1/5.3	Not assigned	Numbered	rtr_leaf1_leaf2	Generic System n/a
<input type="checkbox"/>	red	200	leaf1	Leaf	port-channel3.1	172.16.0.40/31	Numbered		
<input type="checkbox"/>	red	200	leaf2	Leaf	port-channel3.1	172.16.0.41/31	Numbered		

### Assign DHCP to Routing Zone

1. From the blueprint, navigate to **Staged > Virtual > Routing Zones** and click the name of the routing zone that needs a DHCP server assigned to it.
2. Click the **Assign DHCP Servers** button (upper-right).

Dashboard Analytics Staged Uncommitted Active Time Voyager

Physical Virtual Policies Catalog Tasks Connectivity Templates

Virtual Networks Routing Zones Floating IPs Static Routes Protocol Sessions Remote EVPN Gateways Virtual Infra Endpoints

Expanded View Compact View

Assign DHCP Servers

3. Enter the IPv4 address (or IPv6 address) for one or more DHCP servers.
4. Click **Update** to stage the assignment and return to the routing zone.

### Edit Routing Zone

1. From the blueprint, navigate to **Staged > Virtual > Routing Zones** and click the name of the routing zone to edit.
2. Click the **Edit** button (upper-right) and make your changes.
3. Click **Confirm** to stage the change and return to the routing zone.

### Delete Routing Zone

1. From the blueprint, navigate to **Staged > Virtual > Routing Zones** and click the name of the routing zone to delete.
2. Click the **Delete** button (upper-right). All virtual networks that were created under that routing zone will also be deleted.
3. Click **Confirm** to stage the deletion and return to the list view.

### Protocol Sessions (Virtual)

Protocol sessions are BGP sessions that are created when you create connectivity templates. (As of Apstra version 4.0, protocol sessions replace security zone external connectivity points.) From the blueprint, navigate to **Staged > Virtual > Protocol Sessions** to go to protocol sessions.

1. Click 'Staged' in the top navigation bar.

2. Click 'Virtual' in the left sidebar.

3. Click 'Protocol Sessions' in the bottom navigation bar.

Endpoint 1					Endpoint 2					Protocol	Routing Zone	Protocol Session ID
Name	Tags	Peer IPv4 Address	Peer IPv6 Address	ASN	Name	Tags	Peer IPv4 Address	Peer IPv6 Address	ASN			
leaf-2		172.16.0.2/32	N/A	3	sys001		10.0.0.1/32	N/A	65533	BGP	default	<a href="#">9ca27bf1-9e36-4200-9bd9-2fc54b20a27d</a>

Click for details

To see details including peer configuration, click the **Protocol Session ID**.

Dashboard

Analytics

Staged

Uncommitted

Active

Time Voyager

Physical

Virtual

Policies

Catalog

Tasks

Connectivity Templates

Virtual Networks

Routing Zones

Floating IPs

Static Routes

Protocol Sessions

Virtual Infra

Endpoints

Expanded View

Compact View

Parameters

Protocol	BGP
IPv4 AFI	Enabled
IPv6 AFI	Disabled
Keepalive Timer	60
Holdtime Timer	180
TTL	10
BFD	Disabled
Deploy Mode	N/A
Routing Zone	default

Peer Configurations

Query: All

1-2 of 2

Page Size: 25

Systems	Peer Interface	Peer Type	ASN	Routing Policy	IPv4				IPv6			
					Addressing	Peer Address	ASN Type	Prefix Neighbor	Addressing	Peer Address	ASN Type	Prefix Neighbor
<div>1 system(s)</div> <div>rack_2_001_leaf1</div>	loopback0	Loopback	3	N/A	Addressed	172.16.0.2/32	Static	N/A	N/A	N/A	Static	N/A
<div>1 system(s)</div> <div>ty001</div>	N/A	Loopback	65533	N/A	Addressed	10.0.0.1/32	Static	N/A	N/A	N/A	Static	N/A

**NOTE:** Beginning with Apstra version 4.0, protocol sessions replace security zone external connectivity points.

Data Center Interconnect (DCI) / Remote EVPN Gateways (Virtual)

IN THIS SECTION

●

DCI / EVPN Gateway Overview | 503

●

DCI Deployment Options | 504

●

Implementation | 506

## DCI / EVPN Gateway Overview

Historically, enterprises have leveraged Data Center Interconnect (DCI) technology as a building block for business continuity, disaster recovery (DR), or Continuity of Operations (COOP). These service availability use cases primarily relied on the need to connect geographically separated data centers with Layer 2 connectivity for application availability and performance.

With the rise of highly virtualized Software-Defined Data Centers (SDDC), cloud computing, and more recently, edge computing, additional use cases have emerged:

- **Colocation Expansion:** Share compute and storage resources to colocation data center facilities.
- **Resource Pooling:** Share and shift applications between data centers to increase efficiency or improved end-user experience.
- **Rapid Scalability:** Expand capacity from a resource-limited location to another facility or data center.
- **Legacy Migration:** Gracefully move applications and data off older and inefficient equipment and architecture to more efficient, higher-performing, and cost-effective architecture.

With Apstra software, you can deploy and manage a vendor inclusive DCI solution that is simple, flexible, and Intent-Based. Apstra utilizes the standards-based MP-BGP EVPN with VXLAN, which has achieved broad software and hardware adoption in the networking industry. You can choose from a vast selection of cost-effective commodity hardware from traditional vendors to white-box ODMs and software options ranging from conventional vendor integrated Network Operating Systems (NOS) to disaggregated open source options.

EVPN VXLAN is a standards-based (RFC-7432) approach for building modern data centers. It incorporates both data plane encapsulation (VXLAN) and a routing control plane (MP-BGP EVPN Address Family) for extending Layer 2 broadcast domains between hosts as well as Layer 3 routed domains in spine-leaf networks. Relying on a pure Layer 3 underlay for routing of VXLAN tunneled traffic between VXLAN Tunnel Endpoints (VTEPs), EVPN introduces a new address family to the MP-BGP protocol family and supports the exchange of MAC/IP addresses between VTEPs. The advertisement of endpoint MACs and IPs, as well as "ARP/ND-suppression", eliminates the need for a great majority of Broadcast/Unknown/Multicast (BUM) traffic and relies upon ECMP unicast routing of VXLAN, from Source VTEP to Destination VTEP. This ensures optimal route selection and efficient load-sharing of forwarding paths for overlay network traffic.

Just as EVPN VXLAN works within a single site for extending Layer 2 between hosts, the DCI feature enables Layer 2 connectivity between sites. The Apstra DCI feature enables the extension of Layer 2 or

Layer 3 services between data centers for disaster recovery, load balancing of active-active sites, or even for facilitating the migration of services from one data center to another.

Limitation: EVPN-GW (DCI) between different vendors' EVPN fabric is not supported.

## DCI Deployment Options

### IN THIS SECTION

- [Over the Top | 504](#)
- [Gateway \(GW\) | 505](#)
- [Autonomous System Border Router \(ASBR\) | 505](#)

You can implement Data Center Interconnect using the following methods:

- Over the Top
- Gateways (GW)
- Autonomous System Border Router (ASBR)

For assistance with selecting the best option for your organization, consult your Apstra Solutions Architect (SA) or Systems Engineer (SE).

The following characteristics apply to all deployment options:

- You can extend Apstra DCI to other Apstra-managed data centers, non-Apstra managed data centers, or even to legacy non-spine-leaf devices.
- Apstra implementation and behavior is the same in all three cases.
- Whether the remote end is another DCI GW or an ASBR, it is transparent to Apstra.
- Apstra manages neither the GWs nor ASBRs.

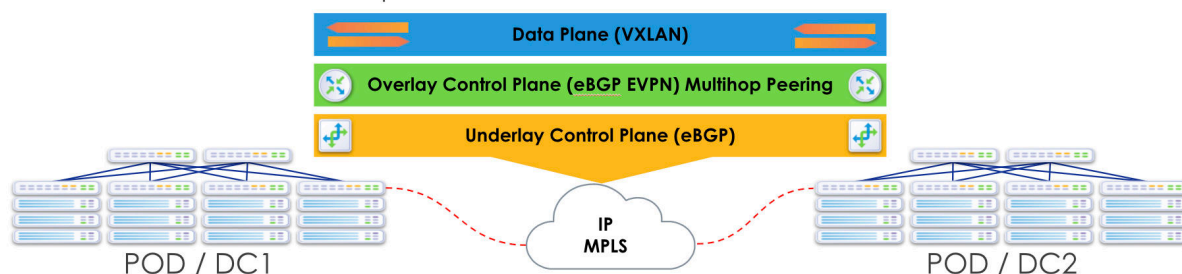
### *Over the Top*

DCI "Over the Top" is a transparent solution, meaning EVPN routes are encapsulated into standard IP and hidden from the underlying transport. This makes the extension of services simple and flexible and is often chosen because it can be implemented by data center teams with little to no coordination with WAN or Service Provider groups. This reduces the implementation times and internal company friction.

However, the tradeoff is scalability and resilience.

## DCI - Over the Top

Similar to RFC 4364 - InterAS option C

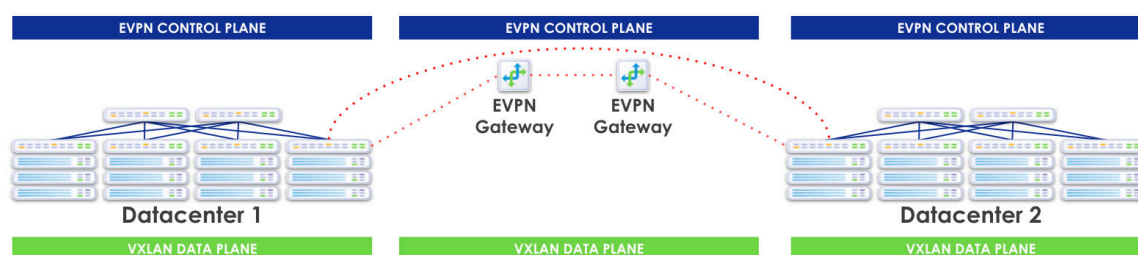


### Gateway (GW)

Building upon the Apstra **Remote EVPN Gateway** capability, you can optionally specify the **Remote EVPN Gateway** to be an external generic system (tagged as an external router) in the same site, thus extending the EVPN attributes to said gateway. This solution creates a fault domain per site, preventing failures from affecting convergence in remote sites and creating multiple fault domains. IP/MAC endpoint tables for remote sites are processed and held in state on a generic system (tagged as external router) gateway. You can also implement WAN QoS and security, along with optimizations made available by the transport technology (for example, MPLS TE). However, this solution is more operationally complex, requiring additional hardware and cost.

## DCI using Gateway

Independent Control Planes - described in RFC 8365, section-10.1



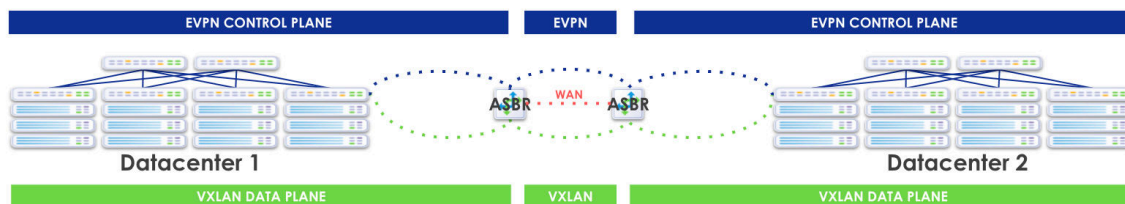
### Autonomous System Border Router (ASBR)

Using the Apstra **Remote EVPN Gateway** capability, you can optionally specify the **Remote EVPN Gateway** to be an ASBR WAN Edge Device. This end-to-end EVPN enables uniform encapsulation and removes the dedicated GW requirement. It is operationally complex but has greater scalability as

compared to both "DCI Using Gateway" and "Over the Top".

## DCI using ASBR

Described in RFC 8365, section-10.2 (Similar to RFC4364 - InterAS option B)



### Implementation

#### IN THIS SECTION

- [EVPN Gateways Use Cases | 506](#)
- [Over the Top | 507](#)
- [Data Plane Extension: Layer 3 | 508](#)
- [Data Plane Extension: Layer 2 | 509](#)

You can extend routing zones and virtual networks (VN) to span across Apstra-managed blueprints (across pods) or to remote networks that are not managed by Apstra (across data centers). This feature introduces the EVPN Gateway (GW) role, which could be a switch that participates in the fabric or RouteServer(s) on a generic system (tagged as a server) that is connected to the fabric.

#### *EVPN Gateways Use Cases*

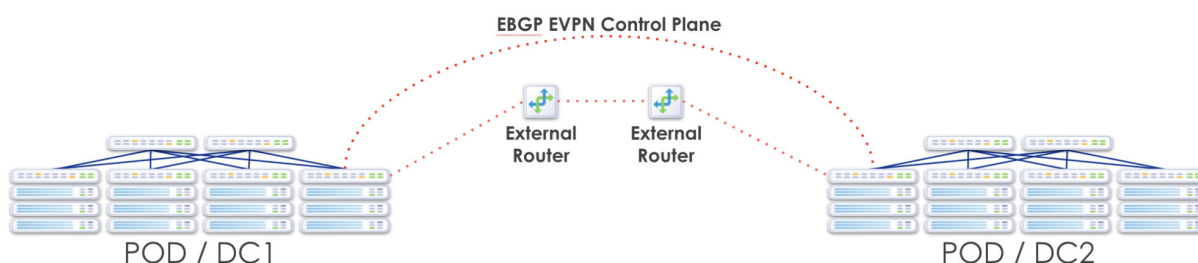
- Span Layer 3 isolation domains (VRFs / routing zones) to multiple Apstra-managed pods (blueprints) or extend to remote EVPN domains.
- Provide Layer 2 domain extensions for L2VNI / virtual networks.
- Help extend EVPN domain from Apstra to Apstra-managed and Apstra to unmanaged pods.
- No VXLAN traffic termination on the spines. You can achieve this by connecting external generic systems (tagged as external routers) on spines. This is to support IPv4 (underlay) external connectivity. Here spines don't need to terminate VXLAN traffic, unlike border leafs, when connected to external generic systems (tagged as external routers). In a nutshell, using this can exchange IPv4 routes to remote VTEPs (in the default routing zone/VRF) and only Layer 3 connectivity is required:

### Over the Top

When BGP EVPN peering is done "over the top", the Data Center Gateway (DC-GW) is a pure IP transport function and BGP EVPN peering is established between gateways in different data centers.

The next sections describes the procedures for interconnecting two or more BGP-based Ethernet VPN (EVPN) sites in a scalable fashion over an IP network. The motivation is to support extension of EVPN sites without having to rely on typical Data Center Interconnect (DCI) technologies like MPLS/VPLS, which are often difficult to configure, sometimes proprietary, and likely legacy in nature.

"Over the Top" is a simple solution that only requires IP routing between data centers and an adjusted MTU to support VXLAN encapsulation between gateway endpoints. In such an implementation, EVPN routes are extended end-to-end via MP-BGP between sites. Multi-hop BGP is enabled with the assumption that there will be multiple Layer 3 hops between sites over a WAN. Otherwise the default TTL decrements to 0 and packets are discarded and don't make it to the remote router. Apstra automatically renders the needed configuration to address these limitations.



This design merges the separate EVPN-VXLAN domains and VXLAN tunnels between sites. Merging of previously separate EVPN domains in different sites realizes the benefit of extending Layer 2 and Layer 3 (VRF) services between sites, but also renders the sites as a single fault domain. So a failure in one site is necessarily propagated. Also, anytime you stretch Layer 2 across the WAN between sites, you are also extending the flood domain and along with it, all broadcast traffic over your costly WAN links. At this time, this solution does not offer any filtering or QoS.

**NOTE:** When individual sites are managed by separate Apstra blueprints, or only one site is Apstra-managed, extended routing zones (VRFs) and virtual networks (Layer 2 and/or Layer 3 defined VLANs/subnets) must be created and managed independently in each site. This creates administrative overhead and requires manual mapping of VRFs and VNs between sites.

**NOTE:** If you're setting up P2P connections between two data centers (blueprints) in the same Apstra controller, each blueprint must pull resources from different IP pools to avoid build errors. To do this, create two IP pools with the same IP subnet, but with different names.

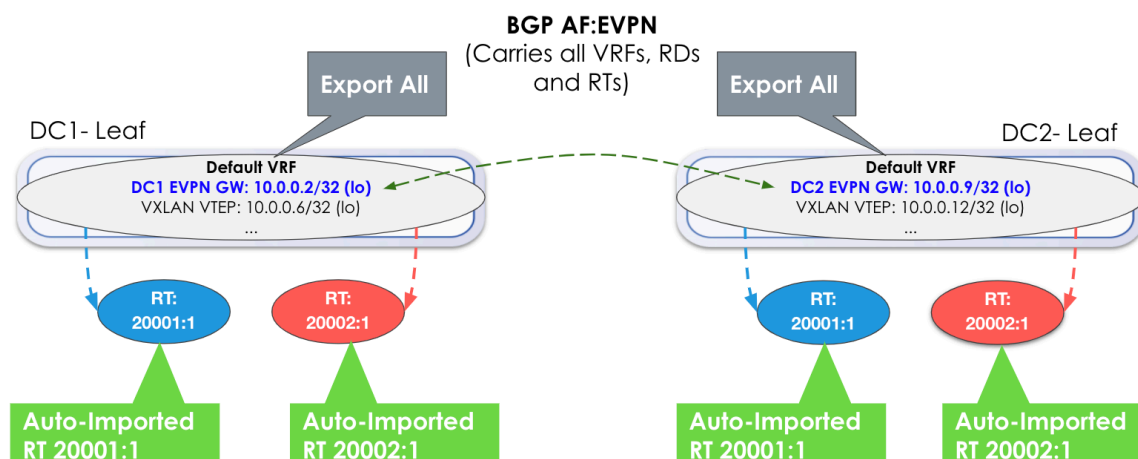
This "Over the Top" solution is the easiest to deploy, requires no additional hardware and introduces no additional WAN config other than increasing the MTU. It is the most flexible and has the lowest barrier to entry. However, the downside is that there is a single EVPN control plane and a routing anomaly in one site will affect convergence and reachability in the other site(s). The extension of Layer 2 flood domains also implies that a broadcast storm in one site extends to the other site(s).

With any DCI implementation, careful resource planning and coordination is required. Adding more sites requires an exponential increase in such planning and coordination. VTEP loopbacks in the underlay need to be leaked. VNIDs must match between sites and in some cases, additional Route Targets (RTs) must be imported. This is covered in detail later in this document.

### *Data Plane Extension: Layer 3*

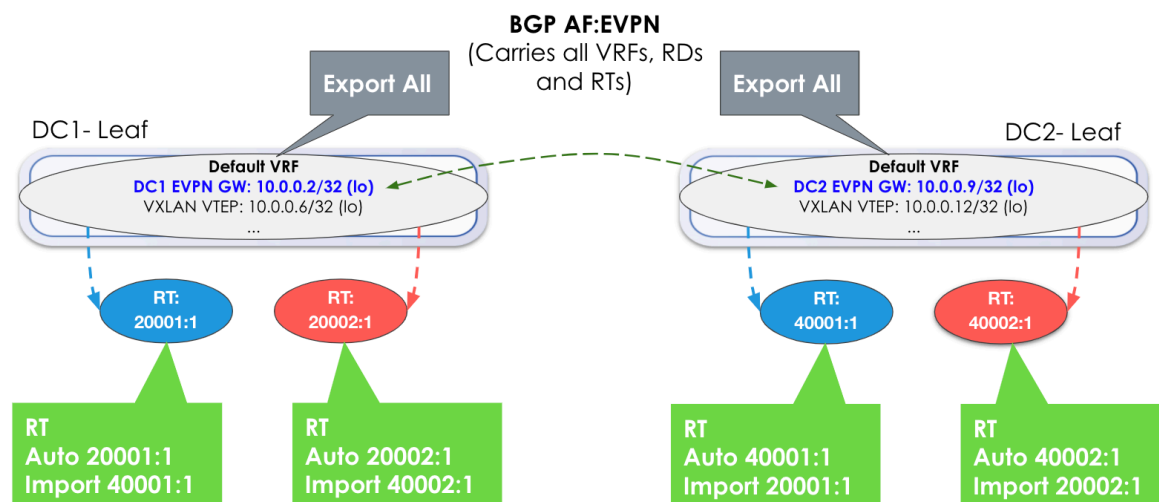
VXLAN Network IDs (VNIDs) are a part of the VXLAN header that identify unique VXLAN tunnels, each of which are isolated from the other VXLAN tunnels in an IP network. Layer 3 packets can be encapsulated into a VXLAN packet or Layer 2 MAC frames can be encapsulated directly into a VXLAN packet. In both cases, a unique VNID is associated with either the Layer 3 subnet, or the Layer 2 domain. When extending either Layer 3 or Layer 2 services between sites, you are essentially stitching VXLAN tunnels between sites. VNIDs therefore need to match between sites.

It is important to understand that a particular VNID will be associated with only one VRF (or routing zone in Apstra terminology). VNIDs exist within a VRF. They are tied to a VRF. For Layer 3 services, the stitching, or extending, of each VNID is done with the export and import of RTs within a routing zone (VRF). Layer 3 subnets (routes) are identified via RTs. All VNIDs are exported automatically at the EVPN gateway (edge) towards the WAN. Conversely, RTs of the same value are automatically imported at the EVPN gateway (edge) coming into the fabric. So if you coordinate the Layer 3 VNIDs at one site to match the other, no additional configuration is needed.



In the image above, no additional export or import is required. Everything is automatically exported (Export All) and because the RTs match, they are automatically imported.

However, if a VNID in DC1 is different from a VNID in DC2, then you must import the RTs respectively. Each respective gateway still automatically imports RTs of the same value. In the example below, an additional step of manually adding the RTs from the other site is required.



### Data Plane Extension: Layer 2

When you create a virtual network in Apstra, you can create it to be a pure Layer 2 service, meaning no Layer 3 anycast gateway is instantiated. This is for either a rack local Layer 2 (VLAN on server facing ports contained within a rack), or you can extend the Layer 2 flood and broadcast domain between racks by selecting VXLAN and then selecting the racks you want the Layer 2 domain extended to. This Layer 2 domain has its own VNID, and the MAC frames (as opposed to IP packets) are encapsulated into the VXLAN header with the VNID of the Layer 2 domain.

The same principles apply in that all VNIDs are exported at the EVPN gateway (in this case Type-2 routes/MAC addresses), and matching RTs are automatically imported. However, the location of importing and exporting RTs is not at the routing zone level, but instead at the virtual network itself.

### Apstra Workflow

#### IN THIS SECTION

- [Control Plane Extension: EVPN Gateway | 510](#)
- [Underlay VTEP Route Advertisements | 510](#)
- [Create Remote EVPN Gateways | 510](#)
- [Enhanced Routing Zone | 512](#)

- [Enhanced Virtual Networks | 513](#)
- [Remote Gateway Topology Representation | 514](#)

### ***Control Plane Extension: EVPN Gateway***

Apstra uses the concept of an "EVPN Gateway". This device can theoretically be a leaf, spine or superspine fabric node, as well as the DCI device. EVPN Gateways separate the fabric-side from the network that interconnects the sites and masks the site-internal VTEPs.

In Apstra, an EVPN Gateway is a device that belongs to and resides at the edge of an EVPN fabric which is also attached to an external IP network. In an Apstra EVPN blueprint, this is always a border-leaf device. The EVPN Gateway of one data center, establishes BGP EVPN peering with a reciprocal EVPN gateway, or gateways, in another data center. The "other" EVPN gateway is the "Remote EVPN Gateway" in Apstra terminology. The Local EVPN Gateway is assumed to be one of the Apstra-managed devices in the blueprint, and is selected during the creation of the "Remote EVPN Gateway". The Local EVPN Gateway will be the border-leaf switch with one or more external routing connections for traffic in and out of the EVPN Clos fabric.

Due to this capability, a Local EVPN Gateway (always an Apstra-managed switch) can be configured to peer with a non Apstra-managed, or even a non Spine-Leaf device, in another DC. The EVPN Gateway BGP peering is used to carry all EVPN attributes from inside a pod, to outside the pod. In Apstra, each data center is represented by an autonomous blueprint and if two or more sites are under Apstra management, you still must configure each site to point to the "Remote EVPN Gateway(s)" in other sites. Apstra recommends that you create multiple, redundant EVPN Gateways for each data center. There is also currently a full mesh requirement between EVPN gateways, although in future releases this requirement will be removed.

### ***Underlay VTEP Route Advertisements***

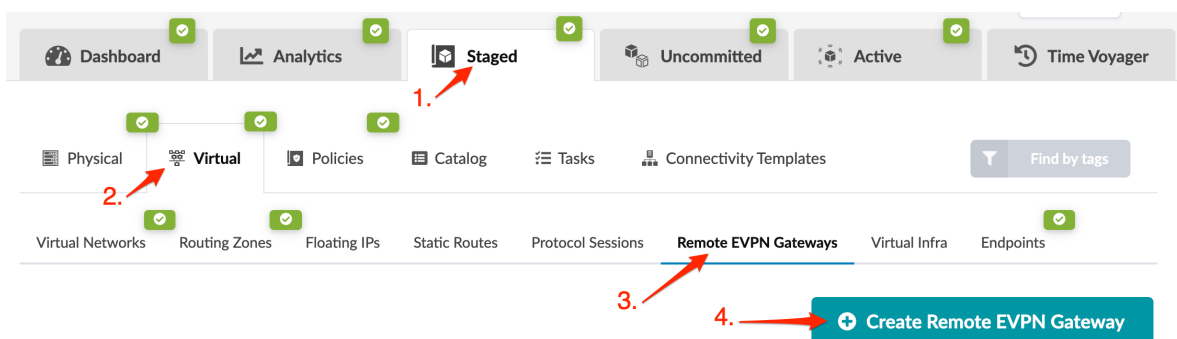
The underlay reachability to VTEP IP addresses, or an equivalent summary route, must be established reciprocally. Each site must advertise these VTEP loopbacks from within the default routing zone into the exported BGP (IPv4) underlay advertisements. Loopbacks in the routing policy are enabled by default.

### ***Create Remote EVPN Gateways***

Remote EVPN Gateway is a logical function that could be instantiated anywhere and on any device, and requires support for BGP in general, L2VPN/EVPN AFI/SAFI specifically. To establish a BGP session with an EVPN gateway, IP connectivity, as well as connectivity to TCP port 179 (BGP TCP port allocated by IANA), should be available.

**NOTE:** For resilience, we recommend having at least two remote gateways for the same remote EVPN domain.

1. From the blueprint, navigate to **Staged > Virtual > Remote EVPN Gateways** and click **Create Remote EVPN Gateway**.



2. In the dialog that opens, fill in the following information for the remote EVPN gateway.

#### Create Remote EVPN Gateway

##### Parameters

##### Name \*

BP2 EVPN Gateway 1. remote EVPN gateway name

##### IP Address \*

10.0.0.9 2. IP address for remote EVPN gateway

##### ASN \*

64515 3. BGP autonomous system number for remote EVPN gateway

##### TTL

15 4. BGP multi-hop time-to-live (max number of L3 hops) - optional  
If not specified, default is used

##### Password

..... 5. Optional BGP TCP authentication password

##### Keep-alive Timer

3 6. BGP keep-alive timer - optional. If not specified, default is used

##### Hold-time Timer

20 7. BGP hold-time timer - optional. If not specified, default is used

##### Local Gateway Nodes

3. Select the **Local Gateway Nodes**. These are the devices in the blueprint that will be configured with a Local EVPN Gateway. You can select one or more devices to peer with the configured remote EVPN gateway. You can use the query function to help you locate the appropriate nodes. We recommend using multiple border-leafs which have direct connections to external generic systems (tagged as

external routers).

### Create Remote EVPN Gateway

Local Gateway Nodes

1-11 of 11

▼ Query: All

Label<sup>?</sup>

Role

Group Label<sup>?</sup>

Hostname<sup>?</sup>

Apply Clear

Page Size: 25

Filter selected by ☒ all ☐ selected only ☐ unselected only

	Label	Role	Group Label	ASN	Hostname
0 selected	sspine1	Superspine	N/A	64512	sspine1

Create Another? Create

You can filter the nodes list below to find them easier.

Select local gateway nodes

4. Click **Create** to stage the gateway and return to the list view.

5. When you are ready to deploy the devices in the blueprint, ["commit" on page 648](#) your changes.

We recommend using multiple remote EVPN gateways. To configure additional remote EVPN gateways, repeat the steps above.

If you are configuring the Remote EVPN Gateway(s) to another Apstra blueprint, you must configure and deploy the remote EVPN gateway(s) separately in that blueprint.

Once the change is deployed, Apstra monitors the BGP session for the remote EVPN gateways. To see any anomalies from the blueprint, navigate to **Active > Anomalies**.

### Enhanced Routing Zone

The installation of EVPN routes is governed mainly by RT (route-target) import/export policies on devices that are part of extended service. You can add import and export route-targets used by Apstra for routing zones/VRFs by specifying route target policies. You do this when you create routing zones. Navigate to **Staged > Virtual > Routing Zones** and click **Create Routing Zone** in the section as shown in

the **Create Routing Zone** dialog below. For more information, see ["Routing Zones" on page 496](#).

## Create Routing Zone

VRF Name \*

VLAN ID ⓘ

VNI

Routing Policies

Route Target Policies

Import Route Targets

+ Add Import Route Target

Export Route Targets

+ Add Export Route Target

**NOTE:** The default route-target generated by Apstra for routing zones is **<L3 VNI>:1**. This can't be altered.

To confirm that correct routes are received at VTEP make sure L3VNIs and route target are identical between the blueprint and remote EVPN domains.

### Enhanced Virtual Networks

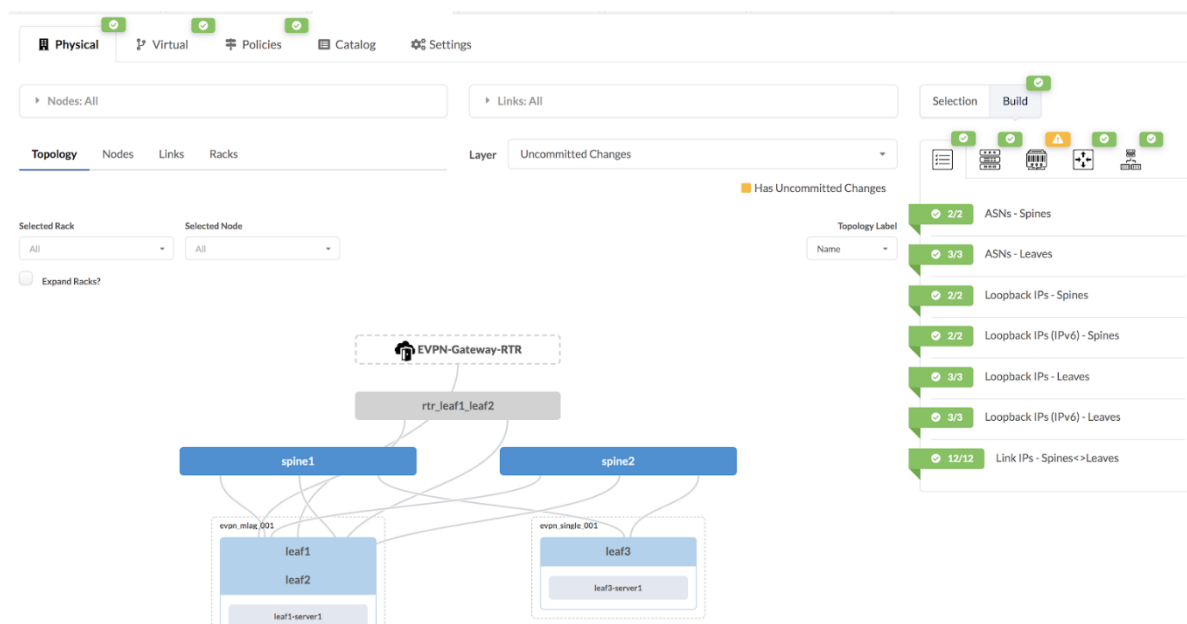
You can add additional import and export route-targets that Apstra uses for virtual networks.

**NOTE:** The default route target that Apstra generates for virtual networks is <L2 VNI>:1. You can't alter this.

For Intra-VNI communication L2VNI specific RT is used. The import RT is used to determine which received routes are applicable to a particular VNI. For connectivity to be established, Layer 2 VNIs must be the same between the blueprint and the remote domains. SVI subnets must be identical across domains.

### *Remote Gateway Topology Representation*

Remote EVPN gateways are represented on the topology view as cloud elements with dotted line connections to the blueprint elements with which BGP sessions are established as shown in the image below. (Image below is slightly different from more recent versions.)



## Virtual Infra (Virtual)

### IN THIS SECTION

- [VMware vCenter/vSphere Virtual Infra | 515](#)
- [NSX-T Integration | 521](#)
- [NSX-T 3.0 Edge and Connectivity Templates | 531](#)

- [VMware NSX-T Inventory Mapping to Apstra Virtual Infrastructure | 542](#)

## VMware vCenter/vSphere Virtual Infra

### IN THIS SECTION

- [VMware vSphere Integration Overview | 515](#)
- [Enable vCenter Integration | 516](#)
- [VM Visibility | 517](#)
- [Validate Virtual Infra Integration | 518](#)
- [Auto-Remediation Overview | 519](#)
- [Enable Auto-Remediation | 519](#)
- [Remediate Probe Anomalies | 520](#)
- [Disable Virtual Infra Integration | 520](#)

## *VMware vSphere Integration Overview*

### IN THIS SECTION

- [Supported Versions | 516](#)
- [Limitations | 516](#)

With Apstra vCenter integration, you have VM visibility of your virtualized environments. This feature helps to troubleshoot various VM connectivity issues. Inconsistencies between virtual network settings (VMware Port Groups) and physical networks (Apstra Virtual Networks) that might affect VM connectivity are flagged.

To do this, the Apstra software identifies the ESX/ESXi hosts and thereby the VMs connected to Apstra-managed leaf switches. LLDP information transmitted by the ESX/ESXi hosts is used to associate host interfaces with leaf interfaces. For this feature to work, LLDP transmit must be enabled on the VMware distributed virtual switch.

The Apstra software also connects to vCenter to collect information about VMs, ESX/ESXi hosts, port groups and VDS. Apstra extensible telemetry collectors collect this information. The collector runs in an off-box agent and uses pyVmomi to connect to vCenter. On first connect, it downloads all of the necessary information and thereafter polls vCenter every 60 seconds for new updates. The collector updates the discovered data into the Apstra Graph Datastore allowing VM queries and alerts to be raised on physical/virtual network mismatch.

### Supported Versions

VMware vSphere/vCenter integration is available for the following versions of VMware:

- vCenter Server/vSphere 6.7
- vCenter Server/vSphere 6.5

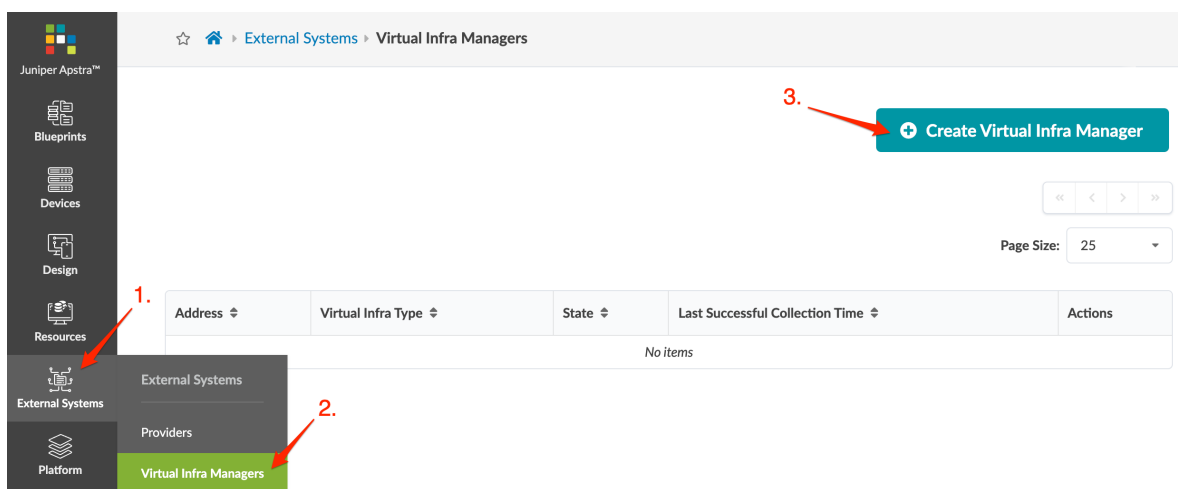
### Limitations

vCenter integration does not support DVS port group with VLAN type Trunking.

### Enable vCenter Integration

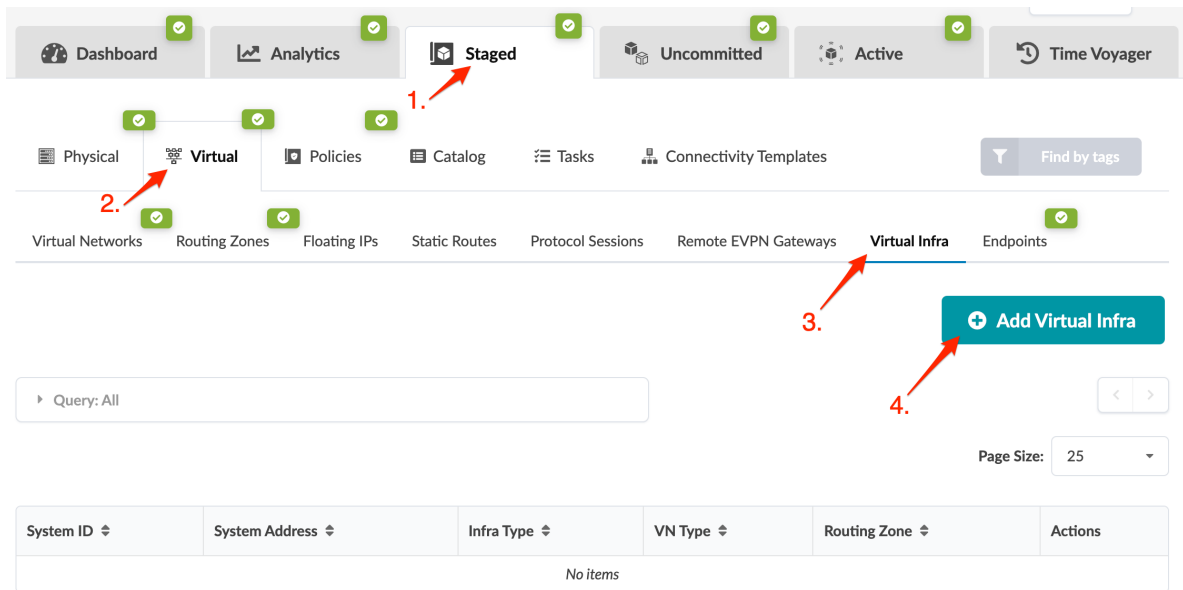
You only need **Read** permissions to enable vSphere Integration.

1. From the left navigation menu, navigate to **External Systems > Virtual Infra Managers** and click **Create Virtual Infra Manager**.



2. Enter the vCenter IP address (or DNS name), select **VMware vCenter Server**, then enter a username and password.
3. Click **Create** to launch an off-box container running vCenter. While the container is connecting, the state is DISCONNECTED. When the container successfully connects, the state changes to CONNECTED.

- When vCenter is connected, from the blueprint, navigate to **Staged > Virtual > Virtual Infra** and click **Add Virtual Infra**.



- Select the vCenter Server from the **Virtual Infra Manager** drop-down list, then click **Create** to stage the change.

When you are ready to deploy, commit the changes from the **Uncommitted** tab.

### VM Visibility

When Apstra software manages virtual infra, you can query VMs by name. From the blueprint, navigate to **Active > Query > VMs** and enter search criteria. VMs include the following details:

Hosted On	The ESX host that the VM is on
VM IP	The IP address as reported by vCenter after installation of VM tools. If the IP address is not available this field is empty. If the IP address is available, the VM IP address is displayed.
Leaf:Interface	The leaf and the interface ESX host is connected to
Port Group Name:VLAN ID	The VNIC's portgroup and the VLAN ID associated with the portgroup
MAC Addresses	MAC address of the VNIC
Virtual Infra Address	IP address of the vCenter the VM is part of

### Validate Virtual Infra Integration

You can validate virtual infra with intent-based analytics. Apstra validates BGP session towards NSX-T Edge. In case BGP neighborhood in NSX-T Manager is deleted then respective anomalies can be seen in Apstra dashboard.

#### Set BGP Neighbors

Tier-0 Gateway test\_vanillaB... #Neighbors 2

ADD BGP NEIGHBOR						
EXPAND ALL						
Search						
	IP Address	BFD	Remote AS number	Route Filter	Allowas-in	Status
⋮	10.100.150.1	Disabled	1	1	Disabled	In Progress
	Source Addresses	Not Set		Graceful Restart	Helper Only	
	Max Hop Limit	1		Description	Not Set	
TIMERS & PASSWORD						
>	10.100.160.1	Disabled	1	1	Disabled	Success

### Generic System Connectivity



			Source				Destination					Expected	Actual				
Rule #	VRF Name #	Address Family #	ASN #	IP #	Hostname #	Interface #	Name #	ASN #	IP #	Hostname #	Interface #	State #	State #	Intent status #	Last fetched #	Last modified #	BGP Peer State #
generic	default	IPv4	1	10.100.150.1	leaf-1-5254005A6FFD	nsxt_vlan150	sys002	65000	10.100.150.2			up	down	mismatch	a minute ago	a minute ago	active
generic	default	IPv4	1	10.100.160.1	leaf-1-5254005A6FFD	nsxt_vlan160	sys001	65000	10.100.160.2			up	up	ok	a minute ago	3 days ago	established
Spine to Leaf	default	evpn	1	1.1.0.0	leaf-1-5254005A6FFD	lo0.0	spine-1	4	1.1.0.3	spine-1	lo0.0	up	up	ok	a minute ago	11 days ago	established
Spine to Leaf	default	IPv4	1	172.10.0.1	leaf-1-5254005A6FFD	xe-0/0/1	spine-1	4	172.10.0.0	spine-1	xe-0/0/0	up	up	ok	a minute ago	11 days ago	established
Spine to Leaf	default	IPv4	1	172.10.0.7	leaf-1-5254005A6FFD	xe-0/0/0	spine-2	5	172.10.0.6	spine-2	xe-0/0/0	up	up	ok	a minute ago	11 days ago	established
Spine to Leaf	default	evpn	1	1.1.0.0	leaf-1-5254005A6FFD	lo0.0	spine-2	5	1.1.0.4	spine-2	lo0.0	up	up	ok	a minute ago	11 days ago	established

Two predefined analytics dashboards (as listed below) are available that instantiate predefined virtual infra probes.

### Virtual Infra Fabric Health Check Dashboard

- "Hypervisor MTU Mismatch Probe" on page 963
- "Hypervisor MTU Threshold Check Probe" on page 964
- "Hypervisor & Fabric LAG Config Mismatch Probe" on page 955
- "Hypervisor & Fabric VLAN Config Mismatch Probe" on page 956
- "Hypervisor Missing LLDP Config Probe" on page 965

- ["VMs without Fabric Configured VLANs Probe" on page 988](#)

### Virtual Infra Redundancy Check Dashboard

- ["Hypervisor Redundancy Checks Probe" on page 965](#)

For more information, see ["Analytics Dashboard" on page 403](#) and ["Probes" on page 410](#).

### *Auto-Remediation Overview*

Automatic remediation of virtual network anomalies is available without user intervention. This can reduce operational cost when network operators don't need to investigate each anomaly and check for details and intervene to mitigate anomalies. VxLAN auto-remediation is a policy configured while adding vCenter/NSX-T to a blueprint. Anomaly remediation is done in accordance with this policy.

A policy-based auto-remediation approach automatically notifies you if there is a mismatch between vSphere DPG (VMware Port Groups) and VN in a particular blueprint, or if there is a VLAN mismatch between virtual infra and the Apstra fabric, or if there is a mismatch in LAG configuration on hypervisors and the corresponding leaf ports. Apstra software provides automatic guided remediation of such anomalies.

Some of the constraints and validations that take place before the remediation happens are listed below:

- When remediation policy is set to VLAN, that is rack-local, routing zone can only be the default one.
- If VLAN ID for virtual network spanning multiple hypervisors is the same, a single layer 2 broadcast domain is assumed. For such scenarios, the VLAN remediation policy must be set to VXLAN as for missing VLAN anomalies it is checked on all the ToR leafs connected to different hypervisors having virtual network with the same VLAN ID. If this is mistakenly chosen as VLAN type, validation errors are generated.
- Errors are flagged for different types of remediation policies (For example, if one is VXLAN type and other is VLAN type) are found attached to different virtual infras (such as two different vCenter servers) having the same VLAN ID in anomalies.
- If two different virtual infra servers are mapped in a blueprint and they have the same VLAN IDs then it is checked as two separate virtual networks by VXLAN auto-remediation policy.

### *Enable Auto-Remediation*

1. From the blueprint, navigate to **Staged > Virtual > Virtual Infra** and click **Add Virtual Infra**.
2. Select the **Virtual Infra Manager** from the drop-down list.
3. Click **VLAN Remediation Policy** to see the attributes to configure.
4. Select the **VN Type** from the drop-down list.
  - **VXLAN** (inter-rack) (default) Assumes VXLAN virtual network and looks for VN mismatch in all of the related ToRs in the Apstra fabric.

- **VLAN** (rack-local) Select VLAN if the VLAN footprint on local vSphere does not extend to other ToR leafs in a fabric.

5. Select the **Routing zone**. (If VN type is **rack-local** only the default routing zone is allowed.)

6. Click **Create**.

After enabling the VLAN remediation policy as inter-rack, Apstra software searches for matching local VLANs in all ToRs connecting any member host (hypervisor for example) participating in the virtual infra virtual network. If such a VN is found, it simply extends that VN to also be bound to the ToR in question with the same local VLAN. If it's not found, a new inter-rack VN is created in the specified routing zone.

### *Remediate Probe Anomalies*

Apstra policy-based remediation has the following features:

- VLAN mismatch anomalies create one virtual network for one vCenter Distributed Virtual Switch (vDS) port group that is attached to hypervisors connected to leaf ports of ToRs in Apstra fabric.
- You cannot delete a routing zone that is being referenced in remediation policy.

**NOTE:** For an EVPN-enabled fabric, we recommend that you have VN type as inter-rack or VXLAN in a specific routing zone.

1. From the blueprint, navigate to **Analytics > Probes** and click one of the instantiated predefined probe names.
2. Click **Remediate Anomalies** on a given stage. The Apstra software automatically updates the staged blueprint by **adding/removing/updating VN endpoints** and **VNs** to resolve the anomalies.
3. Review the staged configuration in terms of virtual network parameters, then commit the configuration. The Apstra software indicates if there are no detected changes. This could happen if you invoke remediation more than once.
4. Review and commit the changes on the **Uncommitted** tab.
5. Return to the predefined probe to view any remaining anomalies.

### *Disable Virtual Infra Integration*

Virtual infra integrations are disabled by deleting them from the blueprint and external systems.

1. From the blueprint, navigate to **Staged > Virtual > Virtual Infra** and click the **Delete** button for the virtual infra to disable.
2. Click **Uncommitted** (top menu) and commit the deletion.
3. From the left navigation menu, navigate to **External Systems > Virtual Ingra Managers** and click the **Delete** button for the virtual infra to disable.

## NSX-T Integration

### IN THIS SECTION

- [VMware NSX-T Integration Overview | 521](#)
- [Enable NSX-T Integration | 522](#)
- [Virtual Infrastructure Visibility | 526](#)
- [Validate Virtual Infra Integration | 529](#)
- [Disable Virtual Infra Integration | 531](#)

### *VMware NSX-T Integration Overview*

#### IN THIS SECTION

- [Supported Versions | 521](#)
- [Limitations | 521](#)

You can integrate NSX-T with Apstra software to help deploy fabric VLANs that are needed for deploying NSX-T in the data center or for providing connectivity between NSX-T overlay networks and fabric underlay networks. You can accelerate NSX-T deployments by making sure the fabric is ready in terms of LAG, MTU and VLAN configuration as per NSX-T transport node requirements. This feature also helps network operators with fabric visibility in terms of seeing all the NSX-T VMs, VM ports, and physical gateway ports. NSX-T integration helps identify issues on the fabric and on the virtual infrastructure. It eliminates manual config validation tasks between the NSX-T Nodes side and the ToR switches.

#### ***Supported Versions***

As of Apstra version 4.0, VMware NSX-T integration is available for both VMware NSX-T Data Center 3.0 and 2.5 version.

#### ***Limitations***

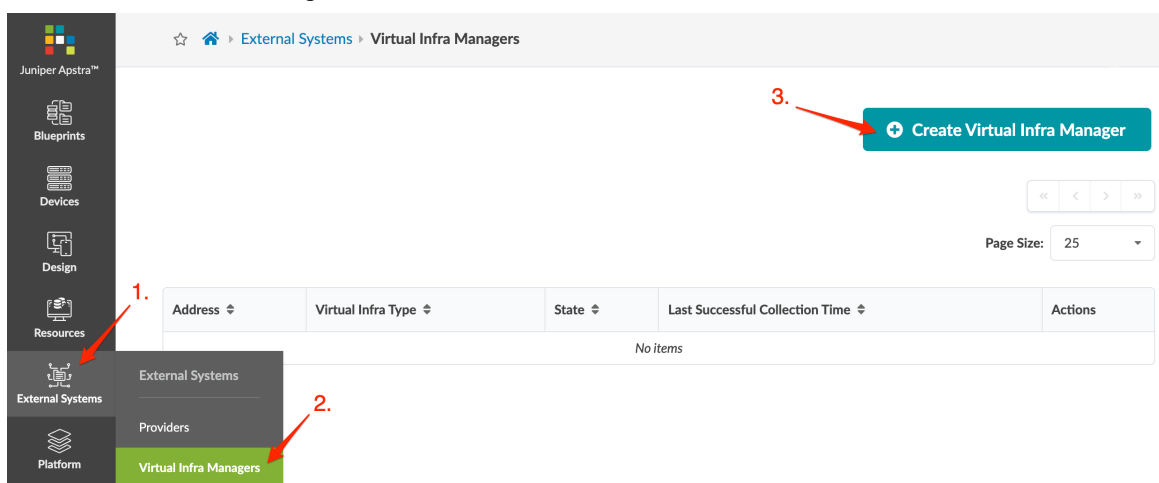
- Having more than one NSX-T virtual infra in a blueprint is not supported. We recommend only one virtual infra per blueprint.

- NSX-T integration does not support DVS port group with VLAN-type trunking.
- Please note that migration of NSX-T Edge VM is supported only within a Rack. In case it is attempted between the Racks it will result in BGP disruption. We can migrate the NSX-T Edge VM from ESXi host connected to Leaf Pair(i.e ToR-Leaf and ToR-Right) to the other ESXi host which is connected to single Leaf with the Rack.

### Enable NSX-T Integration

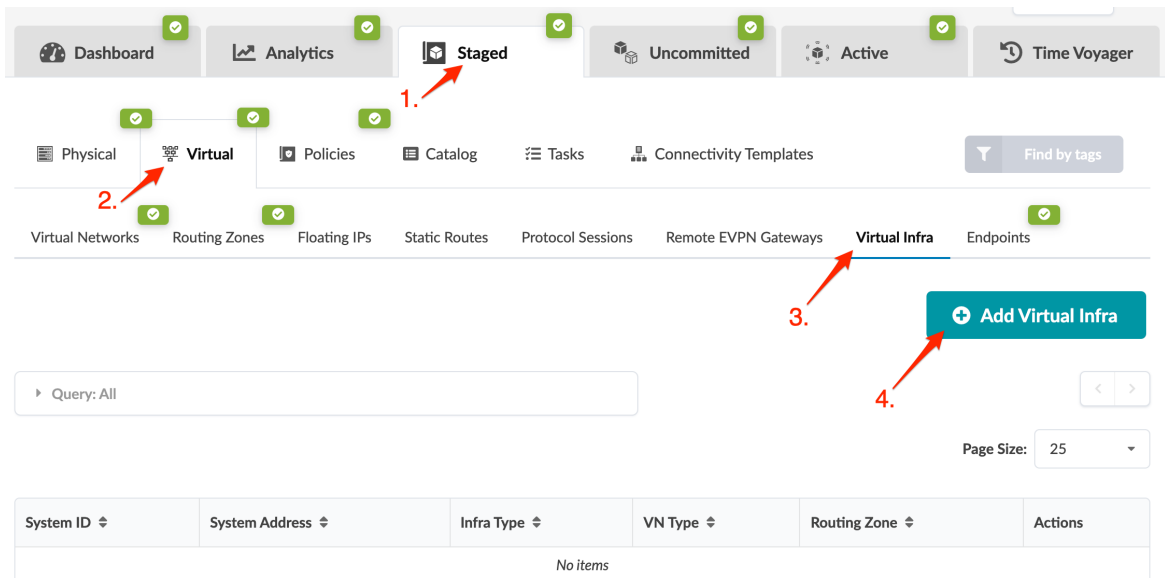
We recommend that you ["create a user profile" on page 706](#) dedicated to managing NSX-T integration activities.

1. From the left navigation menu, navigate to **External Systems > Virtual Infra Managers** and click **Create Virtual Infra Manager**.



2. Enter the NSX-T manager IP address (or DNS name), select **VMware NSX-T Manager** and enter a username and password.
3. Click **Create** to create the virtual infra manager and return to the list view. When the connection is successful, the connection state changes from DISCONNECTED to CONNECTED.

4. When NSX-T is connected, from the blueprint, navigate to **Staged > Virtual > Virtual Infra** and click **Add Virtual Infra**.



5. Select the NSX-T manager from the **Virtual Infra Manager** drop-down list, then click **VLAN Remediation Policy** to expose additional fields. The information entered here is used in Intent-based analytics (IBA) probes <remediate\_anomalies> that can remediate anomalies.
6. Select the VN type and routing zone.
  - If VLAN (rack-local) is selected you can't change the default routing zone.
  - If VXLAN (inter-rack - when VN extends to different ToRs in the fabric) is selected you can select a different routing zone.
7. Click **Create** to stage the virtual infra manager and return to the list view. The new virtual infra manager appears in the list.
8. Click **Uncommitted** (top menu) to review changes, then click **Commit** (top-right) to add the NSX-T manager to the active blueprint.

9. Create a **Routing Zone** in the blueprint and specify the **VLAN ID**, **VNI** and **Routing Policies**. Routing Zone maps to a VRF on which BGP peering towards NSX-T Edge node is established.

Edit Routing Zone

VRF Name

NSX\_VRF

VLAN ID

111

VNI

11011

Route Target<sup>?</sup>

11011:1

Routing Policies

imp\_all

Name	imp_all
Import Policy <sup>?</sup>	All
Extra Import Routes <sup>?</sup>	Not provided
Spine Leaf Links <sup>?</sup>	yes
L3 Edge Server Links <sup>?</sup>	yes
L2 Edge Subnets <sup>?</sup>	yes
Loopbacks <sup>?</sup>	yes
Static routes <sup>?</sup>	yes
Extra Export Routes <sup>?</sup>	Not provided
Aggregate Prefixes <sup>?</sup>	Not provided

Update

10. For the GENEVE Tunnels to come up between the Transport Nodes in NSX-T, will need to have connectivity established via Juniper Apstra Fabric. This will be ensured by creating VXLAN VN in Apstra and assigning correct port mapping in ToR leafs towards Transport Node. Please note that VLAN ID for Overlay VXLAN VN defined in Apstra should match the one mapped in Overlay Profile

in NSX-T for Transport Nodes. Also, same IP subnet as that of the TEP Pool in NSX will be used.

Remember to check the box to Create Connectivity Templates automatically.

Create Virtual Network

Virtual Network Parameters

Type

VLAN

VXLAN

Will create single VLAN for all selected nodes

Name

overlay-tep-pool-vn

Routing Zone

NSX\_VRF

VNI

11050

Do Same VLAN ID on all hosts?

☒

VLAN ID (per host)

50

Route Target

11050:1

DHCP Service

Enabled

Disabled

IPv4 Connectivity

Enabled

Disabled

IPv4 Subnet

10.10.10.0/24

Virtual Gateway (IPv4 Enabled?)

☒

Virtual Gateway IPv4

10.10.10.1

Create Connectivity Templates for

Tagged

Untagged

Assigned To

Query All

1-4 of 4

Page Size: 25

	Bound To	Link Labels	VLAN ID	IPv4 Mode	IPv4 Address
	mac_host_5100_001_host_001	5100-Server-LINK	50	Enabled	mac_host_5100_001_host_001
				Enabled	mac_host_5100_001_host_002

Create Anyway

Create

Name	overlay-tep-pool-vn
Type	VXLAN
Routing Zone	NSX_VRF
VNI	11050
DHCP Service	Disabled
IPv4 Connectivity	Enabled
IPv4 Subnet	10.10.10.0/24
Virtual Gateway IPv4	10.10.10.1
Route Target	11050:1

11. Since we checked the box to **Create Connectivity Template for** in last step during VXLAN VN creation in Apstra a **Connectivity Template** of type **Virtual Network** is automatically created under **Blueprints > Staged > Connectivity Templates** as shown below:

Blueprints > MUC\_DC1 > Staged > Connectivity Templates > Connectivity Templates

Dashboard

Analytics

Staged

Uncommitted

Active

Time Voyager

Physical

Virtual

Policies

Catalog

Settings

Tasks

Connectivity Templates

Find by tags

Tagged VLAN overlay-tep-pool-vn

Automatically created by ACS at VN creation time

Tagged VLAN overlay-tep-pool-vn

Tag: Virtual Network (Single)

Application Point

Tagged VLAN overlay-tep-pool-vn

12. Assign the interfaces to the **Connectivity Template** created above towards Transport nodes in NSX-T side.

Assign Tagged VLAN 'overlay-tep-pool-vn'

Query: All

All bulk actions (0) will be applied only to the loaded connectivity templates.

Fabric	Tags	Tagged VLAN 'overlay-tep-pool-vn'
port1 (Port)		<input type="checkbox"/>
mac_leaf_5100_001 (Rack)		<input type="checkbox"/>
mac_leaf_5100_001_leaf1 / mac_leaf_5100_001_leaf2 (Leaf pair)		<input type="checkbox"/>
ae1 -> mac_leaf_5100_001_svy001 (Interface)		<input type="checkbox"/>
ae2 -> mac_leaf_5100_001_svy002 (Interface)		<input checked="" type="checkbox"/>
ae3 -> mac_leaf_5100_001_svy003 (Interface)		<input type="checkbox"/>
ae4 -> mac_leaf_5100_001_svy004 (Interface)		<input type="checkbox"/>
mac_leaf_5110_001 (Rack)		<input type="checkbox"/>
mac_leaf_5110_001_leaf1 / mac_leaf_5110_001_leaf2 (Leaf pair)		<input type="checkbox"/>
ae1 -> mac_leaf_5110_001_svy001 (Interface)		<input type="checkbox"/>
ae2 -> mac_leaf_5110_001_svy002 (Interface)		<input type="checkbox"/>
ae3 -> mac_leaf_5110_001_svy003 (Interface)		<input type="checkbox"/>
ae4 -> mac_leaf_5110_001_svy004 (Interface)		<input checked="" type="checkbox"/>
mac_leaf_5120_001 (Rack)		<input type="checkbox"/>
mac_leaf_5120_001_leaf1 / mac_leaf_5120_001_leaf2 (Leaf pair)		<input type="checkbox"/>
ae1 -> mac_leaf_5120_001_svy001 (Interface)		<input type="checkbox"/>
ae2 -> mac_leaf_5120_001_svy002 (Interface)		<input type="checkbox"/>
ae3 -> mac_leaf_5120_001_svy003 (Interface)		<input type="checkbox"/>
ae4 -> mac_leaf_5120_001_svy004 (Interface)		<input checked="" type="checkbox"/>
mac_rack_border_001 (Rack)		<input type="checkbox"/>
mac_rack_border_001_leaf1 (Leaf)		<input type="checkbox"/>
ae0/0/0/3 -> mac_rack_border_001_svy004 (Interface)		<input type="checkbox"/>
ae0/0/0/0 -> mac_rack_border_001_svy001 (Interface)		<input type="checkbox"/>
mac_rack_border_001_leaf1 / mac_rack_border_001_leaf2 (Leaf pair)		<input type="checkbox"/>
ae2 -> mac_rack_border_001_svy002 (Interface)		<input type="checkbox"/>
ae3 -> mac_rack_border_001_svy003 (Interface)		<input type="checkbox"/>

Assign

13. Once the configuration as per above steps is rendered towards devices we can observe GENEVE Tunnels between Transport and Edge nodes are UP in NSX-T Manager.

edge01

Overview Monitor **Tunnels** Related ▾

Tunnel Endpoint

Show Status: ALL 4 UP 0 DOWN

Filter by BFD Status: ALL ▾

Source IP	Remote IP	Status	BFD Diagnostic Code	Remote Transport Node	Encap Interface	Encap	Tunnel Name
10.10.10.14	10.10.10.10	Up	0 - No Diagnostic	10.6.1.31	fp-eth0	GENEVE	geneve168430090
10.10.10.14	10.10.10.11	Up	0 - No Diagnostic	10.6.1.35	fp-eth0	GENEVE	geneve168430091
10.10.10.14	10.10.10.12	Up	0 - No Diagnostic	10.6.1.42	fp-eth0	GENEVE	geneve168430092
10.10.10.14	10.10.10.13	Up	0 - No Diagnostic	10.6.1.37	fp-eth0	GENEVE	geneve168430093

REFRESH

1 - 4 of 4 Tunnel Endpoints

GENEVE Tunnels from the EDGE Node to the Transport Nodes

**NOTE:** When you install the NSX Edge as a virtual appliance or host Transport Node, use the default uplink profile. If the Failover teaming policy is configured for an uplink profile, then you can only configure a single active uplink in the teaming policy. Standby uplinks are not Supported and must not be configured in the failover teaming policy.

### Virtual Infrastructure Visibility

When you've successfully integrated NSX-T, you have visibility of NSX-T VMs and transport nodes in the virtual infrastructure. You can query the status of the VMware fabric health.

When you have successfully integrated NSX-T, you have visibility of NSX-T VMs and transport nodes in the virtual infrastructure. You can query the status of the VMware fabric health.

To see a list of the VMs connected to the hypervisor, navigate to the dashboard and scroll to fabric health for VMware option.

## Fabric Health for VMware NSX-T

VMs on hypervisors connected to Fabric

Hypervisor	Virtual Machine	Virtual Machine Ip	Vnic	Vnet	Vlan
nsxtcomputehost01	webtier010	192.168.1.10	Network adapter 1	benefitswebtier	No va
nsxtcomputehost01	webtier011	192.168.1.30	Network adapter 1	benefitswebtier	No va
nsxtedgehost01	webtier020	192.168.1.20	Network adapter 1	benefitswebtier	No va
View stage					

You can also query VMs that are hosted on hypervisors connected to ToR leafs. From the blueprint, navigate to **Active > Query > VMs**.

The screenshot shows the VMware NSX-T Fabric Health dashboard. The top navigation bar includes links for Blueprints, MUC\_DC1, Active, Query, and VMs. The main content area displays a table of VMs hosted on hypervisors connected to Fabric. The table has columns for VM Name, Hosted On, Hypervisor Hostname, Hypervisor Version, VM IPs, Leaf/Interface, Port Group Name/VLAN ID, MAC Addresses, Virtual Infra Address, and Virtual Infra Type. The table lists several VMs, including Calico-VM, Embedded-vCenter-Server-Appliance, MUC\_DC1\_NSX-T, NSX-template, centos-vm-template, edge01, vCLS (1), and vCLS (11).

VM Name	Hosted On	Hypervisor Hostname	Hypervisor Version	VM IPs	Leaf/Interface	Port Group Name/VLAN ID	MAC Addresses	Virtual Infra Address	Virtual Infra Type
Calico-VM	10.6.1.31 (muc_leaf_5120_001_sys004)	R5-U14-Dell	7.0.2	10.6.1.44				10.6.1.33	vcenter
Embedded-vCenter-Server-Appliance	10.6.1.29	localhost	7.0.0	10.6.1.33				10.6.1.33	vcenter
MUC_DC1_NSX-T	10.6.1.38	R5-U21-Dell	7.0.2	10.6.1.39				10.6.1.33	vcenter
NSX-template	10.6.1.145 (muc_rack_border_001_sys001)	localhost	7.0.0					10.6.1.33	vcenter
centos-vm-template	10.6.1.40 (muc_rack_border_001_sys005)	R5-U23-Dell	7.0.2					10.6.1.33	vcenter
edge01	10.6.1.40 (muc_rack_border_001_sys005)	R5-U23-Dell	7.0.2	10.6.1.46			00:50:56:85:58:e6 00:50:56:85:18:03 00:50:56:85:69:7e	10.6.1.33	vcenter
vCLS (1)	10.6.1.29	localhost	7.0.0					10.6.1.33	vcenter
vCLS (11)	10.6.1.145 (muc_rack_border_001_sys001)	localhost	7.0.0					10.6.1.33	vcenter

VMs include the following details:

VM Name	The Virtual Machine name which is hosted on NSX managed hypervisor.
Hosted On	The ESXi host on which Virtual Machine is hosted.

Hypervisor Hostname	The hypervisor hostname on which Virtual Machine is hosted and is connected to the leaf TORs in a fabric.
Hypervisor Version	The software version of OS running on the hypervisor.
VM IP	The IP address as reported by NSX-T after the installation of VM tools. If the IP address is not available this field is empty. Apstra displays VM IP if the IP address is available on installation VM tools on the VM.
Leaf:Interface	System ID for the interface on the leaf to which ESXi host is connected and on which VM resides.
Port Group Name:VLAN ID	The VLAN ID which NSX-T port groups are using. Overlay VM to VM traffic in a NSX-T enabled Data Center tunnels between transport nodes over this Virtual network.
MAC Addresses	MAC address of the VM connected to the Apstra Fabric.
Virtual Infra address	IP address of the NSX-T infra added to a Blueprint

To search for nodes in the physical topology that have VMs, navigate to **Active > Physical** and select **Has VMs?** from the **Nodes** drop-down list.

The screenshot displays the Apstra GUI interface. At the top, the breadcrumb navigation shows 'Blueprints > Menlo HQ NSX-T Lab'. Below this, a series of tabs are visible: Dashboard, Analytics, Staged, Uncommitted, and Active (selected). Under the 'Active' tab, there are sub-tabs: Physical (selected), Virtual, Policies, Settings, Query, Anomalies, and Root Causes. The 'Physical' sub-tab is active, showing a list of nodes filtered by 'Nodes: Has VMs?=yes'. The filter is applied, and the results are displayed in a table. The table has columns for 'Selection' and 'Status'. The 'Status' column shows a list of anomalies with their respective counts. The anomalies listed are: Anomalies: All Services (1), Anomalies: BGP (0), Anomalies: Cabling (0), Anomalies: Hostname (0), Anomalies: Interface (0), Anomalies: LAG (0), Anomalies: Liveness (0), Anomalies: MLAG (0), Anomalies: Probes (0), Anomalies: Route (0), Deployment Mode (3/0/0), Deployment Status: Discovery (0/0/0), and Deployment Status: Service (3/0/0).

In case the VM is moved from one Transport node to another in NSX-T it can be visualized in Apstra under **Active > Physical > Nodes > Generic System (Node\_name)** and selecting **VMs** option as below:

Selection

Status

nsx\_compute\_001\_sys001

Role: Generic System

Group label: server

Telemetry

Device

Properties

Tags

VMs

Hypervisor

test\_v1bgp.nsxt\_edge.vanilla\_bgp.vqfx-TN-1

Hosted VMs

1-1 of 1

VM	Part of Port Group
test_v1bgp.nsxt_edge.vanilla_bgp.vqfx_vm1	test_v1bgp.nsxt_edge.vanilla_bgp.vqfx-ls

**Validate Virtual Infra Integration**

You can validate virtual infra with intent-based analytics. Apstra validates BGP session towards NSX-T Edge. In case BGP neighborship in NSX-T Manager is deleted then respective anomalies can be seen in Apstra dashboard.

## Set BGP Neighbors



Tier-0 Gateway test\_vanillaB... #Neighbors 2

ADD BGP NEIGHBOR

EXPAND ALL

Search

	IP Address	BFD	Remote AS number	Route Filter	Allows-in	Status
⋮	10.100.150.1	Disabled	1	1	Disabled	In Progress ⓘ
	Source Addresses	Not Set		Graceful Restart	Helper Only ⓘ	
	Max Hop Limit	1		Description	Not Set	
	TIMERS & PASSWORD					
⋮	10.100.160.1	Disabled	1	1	Disabled	Success ⓘ ⓘ

## Generic System Connectivity



			Source				Destination					Expected	Actual				
Rule #	VRF Name #	Address Family #	ASN #	IP #	Hostname #	Interface #	Name #	ASN #	IP #	Hostname #	Interface #	State #	State #	Intent status #	Last fetched #	Last modified #	BGP Peer State #
generic	default	IPv4	1	10.100.150.1	leaf-1-5254005A6FF0	msr_vlan150	sys002	65000	10.100.150.2			up	down	mismatch	a minute ago	a minute ago	active
generic	default	IPv4	1	10.100.160.1	leaf-1-5254005A6FF0	msr_vlan160	sys001	65000	10.100.160.2			up	up	ok	a minute ago	3 days ago	established
Spine to Leaf	default	evpn	1	1.1.0.0	leaf-1-5254005A6FF0	lo0/0	spine-1	4	1.1.0.3	spine-1	lo0/0	up	up	ok	a minute ago	11 days ago	established
Spine to Leaf	default	IPv4	1	172.10.0.1	leaf-1-5254005A6FF0	xe-0/0/1	spine-1	4	172.10.0.0	spine-1	xe-0/0/0	up	up	ok	a minute ago	11 days ago	established
Spine to Leaf	default	IPv4	1	172.10.0.7	leaf-1-5254005A6FF0	xe-0/0/0	spine-2	5	172.10.0.6	spine-2	xe-0/0/0	up	up	ok	a minute ago	11 days ago	established
Spine to Leaf	default	evpn	1	1.1.0.0	leaf-1-5254005A6FF0	lo0/0	spine-2	5	1.1.0.4	spine-2	lo0/0	up	up	ok	a minute ago	11 days ago	established

Two predefined analytics dashboards (as listed below) are available that instantiate predefined virtual infra probes.

## Virtual Infra Fabric Health Check Dashboard

- "Hypervisor MTU Mismatch Probe" on page 963
- "Hypervisor MTU Threshold Check Probe" on page 964
- "Hypervisor & Fabric LAG Config Mismatch Probe" on page 955
- "Hypervisor & Fabric VLAN Config Mismatch Probe" on page 956
- "Hypervisor Missing LLDP Config Probe" on page 965
- "VMs without Fabric Configured VLANs Probe" on page 988

## Virtual Infra Redundancy Check Dashboard

- ["Hypervisor Redundancy Checks Probe" on page 965](#)

For more information, see ["Analytics dashboard" on page 403](#) and ["Instantiate Predefined Probe" on page 416](#).

### ***Disable Virtual Infra Integration***

Virtual infra integrations are disabled by deleting them from the blueprint and external systems.

1. From the blueprint, navigate to **Staged > Virtual > Virtual Infra** and click the **Delete** button for the virtual infra to disable.
2. Click **Uncommitted** (top menu) and commit the deletion.
3. From the left navigation menu, navigate to **External Systems > Virtual Infra Managers** and click the **Delete** button for the virtual infra to disable.

## **NSX-T 3.0 Edge and Connectivity Templates**

### **IN THIS SECTION**

- [Overview | 531](#)
- [Set Up NSX-T Tier-0 Router BGP peering | 531](#)
- [Set Up NSX-T VRF Lite | 537](#)
- [Set Up Default Static Route towards NSX-T Edge | 540](#)
- [Set Up BGP IPv6 towards NSX-T Edge | 541](#)
- [Un-assign BGP on VXLAN VN towards NSX-T Edge | 542](#)

### ***Overview***

Juniper Apstra supports NSX-T 3.0 Edge connectivity requirements using connectivity templates (new in version 4.0). Connectivity templates can be used both where NSX-T Edge is hosted on Bare Metal or when used as a virtual machine.

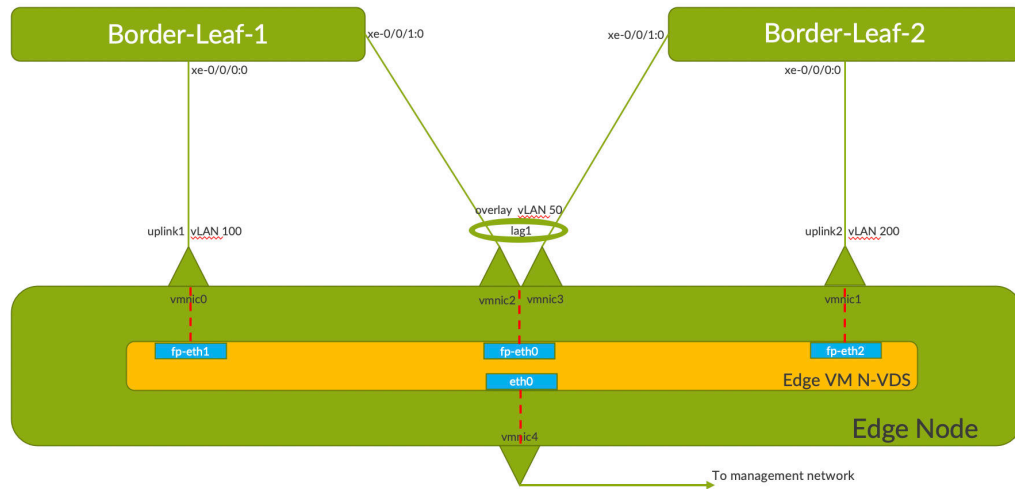
We support VRF lite enabled Tier-0 edge Gateway using connectivity templates.

The use cases below relate to connectivity templates for NSX-T 3.0 Edge:

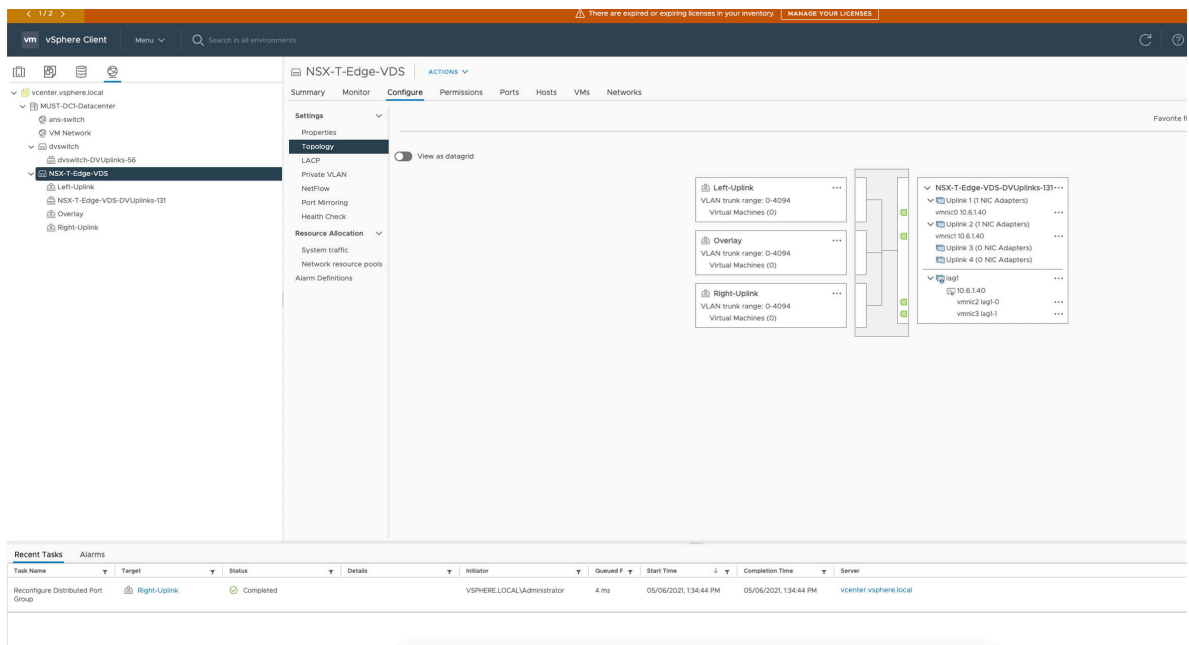
### ***Set Up NSX-T Tier-0 Router BGP peering***

Let's say NSX-T Edge VM uplinks are connected to ToR leafs via VLAN Transport Zone which provides uplink network connectivity to physical infrastructure. Then Edge VM will also have vmnics as per below screenshot which will help for tunnelling traffic between Transport Nodes. This is called Overlay

Transport Zone.



- Create three Distributed Port Groups for respective vmnics and VLAN Trunking to be enabled on all the Nodes as per the networking depicted in previous screenshot.



- Create respective Uplink profiles for Overlay and VLAN Transport Zones in NSX Manager(UI).

The screenshot shows the NSX Manager UI with the 'Uplink Profiles' tab selected. The table lists various uplink profiles for different transport zones.

Uplink Profile	ID	Teaming Policy	Active Uplinks	Standby Uplinks	Transport VLAN	MTU
nsx-default-uplink-profile	0a26_dcf9	Follower Order	uplink1	uplink2	0	1600 (Global MTU)
overlay-profile	8224_b823	Follower Order	lag1		50	1600 (Global MTU)
edge-right-uplink	b8f2_2364	Follower Order	uplink2		200	1600 (Global MTU)
edge-left-uplink	b672_e06a	Follower Order	uplink1		100	1600 (Global MTU)
nsx-edge-lag-uplink-profile	c352_3733	Follower Order	lag		0	1600 (Global MTU)
nsx-edge-multiple-teege-uplink-pr...	ce82_1007	Load Balance Source	uplink-Luplink-2		0	1600 (Global MTU)
nsx-edge-single-nc-uplink-profile	c732_430c	Follower Order	uplink1		0	1600 (Global MTU)
nsx-default-loadbalance-uplink-pr...	f638_240d	Load Balance Source	uplink-Luplink-2,uplink-3,uplink-4		0	1600 (Global MTU)

- After NSX-T is configured on the Transport nodes, a Tunnel endpoint(TEP) IP pool is created in the NSX UI as below:

The screenshot shows the NSX Manager UI with the 'Host Transport Nodes' tab selected. The table lists the nodes and their TEP IP addresses. A callout box highlights the TEP IP addresses, stating 'IPs from the TEP Pool'.

Node	ID	IP Addresses	OS Type	NSX Configuration	NSX Version	Host Switches	Tunnels	TEP IP Addresses	Node Status	Alarms
MUC_Cluster1 (4)	MoRef ID: d...								4 Hosts Up	
10.6.1.35	9843..576e	10.6.1.35, 10.10.1...	ESXi 7.0.2	Success	3.0.0.0.0.1...	1	Not Available	10.10.10.11	Up	0
10.6.1.37	678a..52dd	10.6.1.37, 10.10.1...	ESXi 7.0.2	Success	3.0.0.0.0.1...	1	Not Available	10.10.10.13	Up	0
10.6.1.31	e131..7cd2	10.6.1.31, 10.10.10...	ESXi 7.0.2	Success	3.0.0.0.0.1...	1	Not Available	10.10.10.10	Up	0
10.6.1.42	0924..af40	10.6.1.42, 10.10.1...	ESXi 7.0.2	Success	3.0.0.0.0.1...	1	Not Available	10.10.10.12	Up	1

- Now create the NSX-T Edge VM in NSX Manager UI as below. It is used as the device for north-south communication and BGP peering with Juniper Apstra Fabric. Also configure VDS on the Edge Nodes under NSX Manager(UI) for respective overlay and Uplink interfaces.

Add Edge VM

1 Name and Description

2 Credentials

3 Configure Deployment

4 Configure Node Settings

5 Configure NSX

Name and Description

Name \*

edge01

Host name/FQDN \*

edge01.localhost

Enter Fully Qualified Domain Name (FQDN)  
e.g. subdomain.example.com

Description

Form Factor \*

☐ Small

☒ Medium

☐ Large

☐ Extra Large

2 vCPU

4 vCPU

8 vCPU

16 vCPU

4 GB RAM

8 GB RAM

32 GB RAM

64 GB RAM

200 GB Storage

200 GB Storage

200 GB Storage

200 GB Storage

> Advanced Resource Reservations

CANCEL

NEXT

Add Edge VM

1 Name and Description

2 Credentials

3 Configure Deployment

4 Configure Node Settings

5 Configure NSX

Configure NSX

+ ADD SWITCH

New Node Switch

Edge Switch Name

nvds-overlay

Transport Zone \*

Overlay-TZ

OR Create New Transport Zone

Uplink Profile \*

overlay-profile

OR Create New Uplink Profile

IP Assignment \*

Use IP Pool

IP Pool \*

TEP-Pool

Teaming Policy Switch Mapping

Uplinks

DPDK Fastpath Interfaces

lag1 (active)

Overlay (Distributed Virtu...

CANCEL

PREVIOUS

FINISH

# Overlay

- Tier-0 Gateway in the NSX-T Edge cluster provides a gateway service between the logical and physical network. In NSX Manager create T0 Gateway which connects to the ToR Leaf via BGP to

communicate with the rest of Juniper Apstra Fabric.

Specify the edge-cluster created previously

- Add External interfaces to the T0 GW which maps to the Uplink segments

Used for the link between T0-GW and ToR

Specify the edge VM

One uplink per segment

- Configure BGP peering on NSX T0 GW towards Juniper Apstra Fabric in NSX Manager.

- For NSX-T integration with Juniper Apstra, see ["NSX-T Integration" on page 521](#)

First create a **Routing Zone** in Juniper Apstra UI which maps to a VRF. Then need to setup **IP Link Primitive** based connectivity template to establish BGP peering from the NSX-T Edge node to Fabric as below:

Specify the routing zone on which the IP link will be added and respective VLAN ID.

### Set Up NSX-T VRF Lite

With NSX-T VRF Lite we are able to configure per tenant data isolation. Each VLAN can be considered as a separate channel for data plane under VRF gateways.

BGP peering can be built over these VLANs in VRF gateways for route exchange with the upstream Juniper Apstra fabric. Inter-VRF traffic is routed through the physical Juniper Apstra fabric.

- In NSX-T Manager create the VLAN Segments for the Uplink networks for the tenants.

⋮	▼	🔗	seg-red-TOR-L-testing_rfe1729.nsx_t_edge.vrf_lite.vqfx	None	testing_rfe1729.nsx_t_edge.vrf_lite.vqfx_vlan   VLAN	Not Set	0
<hr/>							
L2 VPN			Not Set		VPN Tunnel ID	Not Set	
VLAN			10		Uplink Teaming Policy	TOR-LEFT	
			11				
			12				
			<a href="#">View Less</a>				
Domain Name			Not Set		IP Address Pool	Not Set	
Edge Bridges			0		Metadata Proxy	0	
Address Bindings			Not Set		Replication Mode	Hierarchical Two-Tier replication	
Connectivity			● On		Tags	0	
Description			Not Set				
<hr/>							
⋮	▼	🔗	seg-red-TOR-R-testing_rfe1729.nsx_t_edge.vrf_lite.vqfx	None	testing_rfe1729.nsx_t_edge.vrf_lite.vqfx_vlan   VLAN	Not Set	0
<hr/>							
L2 VPN			Not Set		VPN Tunnel ID	Not Set	
VLAN			20		Uplink Teaming Policy	TOR-RIGHT	
			21				
			22				
			<a href="#">View Less</a>				
Domain Name			Not Set		IP Address Pool	Not Set	
Edge Bridges			0		Metadata Proxy	0	
Address Bindings			Not Set		Replication Mode	Hierarchical Two-Tier replication	
Connectivity			● On		Tags	0	
Description			Not Set				

- In NSX-T Manager create the VRF-enabled Tier-0 Gateway for the tenants and add the uplink interfaces on the VRF enabled Gateways. Thereafter add the BGP neighbors.

## Interfaces

Tier-O Gateway blue-testing\_... #Interfaces 2

EXPAND ALL

Search

	Name	Type	IP Address / Mask	Connected To(Segment)	Status
>	blue-uplink1-testing_rfe1729.nsxt_edge.vrf_lite.vqfx	External	10.100.30.2/24	seg-blue-TOR-L-testing_rfe1729.nsxt_edge.vrf_lite.vqfx	Success  
>	blue-uplink2-testing_rfe1729.nsxt_edge.vrf_lite.vqfx	External	10.100.40.2/24	seg-blue-TOR-R-testing_rfe1729.nsxt_edge.vrf_lite.vqfx	Success  

## BGP Neighbors

Tier-O Gateway blue-testing\_... #Neighbors 2

EXPAND ALL

Search

	IP Address	BFD	Remote AS number	Route Filter	Allowas-in	Status
>	10.100.30.1	Disabled	2	1	Disabled	Down  
>	10.100.40.1	Disabled	3	1	Disabled	Down  

- From the Apstra GUI, setup the **Routing Zone** and the respective VNs on which BGP session will be established towards ToR Leafs as below:

Expanded View
Compact View

### Parameters

VRF Name	nsxt
Type	EVPN
VLAN ID	2
VNI	20000
Route Target <sup>?</sup>	20000:1
DHCP Servers	DHCP Relay not configured

### Routing Policy

Name	Default Immutable
Import Policy <sup>?</sup>	All

Query: All

1-4 of 4

Columns (10/11)
Page Size: 25

Build

Selected	Name	Routing Zone	Type	VNI	Assigned to	IPv4 Connectivity	IPv4 Subnet	IPv6 Connectivity	IPv6 Subnet	Actions
<input type="checkbox"/>	nsxt_host_edge_traffic	nsxt	VLAN	30000	<div>3 nodes</div> <div> nsx.edge.compute_3001_sw1  nsx.edge.compute_3001_sw2  nsx.compute_3001_sw1 </div>	Enabled	10.10.200.0/24	Disabled	N/A	
<input type="checkbox"/>	nsxt_host_traffic	nsxt	VLAN	10000	<div>3 nodes</div> <div> nsx.edge.compute_3001_sw1  nsx.edge.compute_3001_sw2  nsx.compute_3001_sw1 </div>	Enabled	10.10.100.0/24	Disabled	N/A	
<input type="checkbox"/>	vn150	default	VLAN	150	<div>1 nodes</div> <div>nsx.edge.compute_3001_sw1</div>	Enabled	10.100.150.0/24	Disabled	N/A	
<input type="checkbox"/>	vn160	default	VLAN	160	<div>1 nodes</div> <div>nsx.edge.compute_3001_sw2</div>	Enabled	10.100.160.0/24	Disabled	N/A	

Create Virtual Networks

2/2 nsxt: Virtual Network SVI Subnets  
2/2 SVI Subnets - Virtual Networks  
2/2 VNI Virtual Network IDs

- Create connectivity template under Staged option for the VNs created before and assign the respective interfaces towards NSX-T Edge VM.

## Application Endpoints

Query: All		All bulk actions (🔑) will be applied only to the loaded connectivity templates.	
Fabric	Tags	Templates Applied	
pod1 (Pod)		N/A	
nsx_compute_001 (Rack)		N/A	
nsx_compute_001_leaf1 (Leaf)		N/A	
xe-0/0/2 -> nsx_compute_001_syn001 (Interface)		nsxt vlan vn100 nsxt vlan vn200	
nsx_edge_compute_001 (Rack)		N/A	
nsx_edge_compute_001_leaf1 (Leaf)		N/A	
xe-0/0/0 -> nsx_edge_compute_001_syn001 (Interface)		nsxt vlan150 nsxt vlan vn100 nsxt vlan vn200	
nsx_edge_compute_001_leaf2 (Leaf)		N/A	
xe-0/0/2 -> nsx_edge_compute_001_syn001 (Interface)		nsxt vlan vn100 nsxt vlan160 nsxt vlan vn200	

Type	Action	Name
Connectivity Point	ADDED	nsxt vlan150
Connectivity Point	ADDED	nsxt vlan160
Connectivity Point	ADDED	nsxt vxlan vn100
Connectivity Point	ADDED	nsxt vxlan vn200
Virtual Network	CHANGED	vn160
Virtual Network	CHANGED	nsxt host edge traffic
Virtual Network	CHANGED	nsxt host traffic
Virtual Network	CHANGED	vn150

**Set Up Default Static Route towards NSX-T Edge**

Static default could be required in NSX-T edge setup to provide Internet connectivity. It can be taken care of by adding a default route(0/0) with the next hop pointing towards uplink ToR leaf using a connectivity template.

In the connectivity templates, assign the correct uplink:

## Assign Internet-Server-link

Query: All		All bulk actions (🔑) will be applied only to the loaded connectivity templates.	
Fabric	Tags	Internet-Server-link	
pod1 (Pod)		<input type="checkbox"/>	
msc_leaf_5110_001 (Rack)		<input type="checkbox"/>	
msc_leaf_5110_001_leaf1 / msc_leaf_5110_001_leaf2 (Leaf pair)		<input type="checkbox"/>	
ae1 -> msc_leaf_5110_001_syn001 (Interface)		<input type="checkbox"/>	
ae2 -> msc_leaf_5110_001_syn002 (Interface)		<input type="checkbox"/>	
ae3 -> msc_leaf_5110_001_syn003 (Interface)		<input checked="" type="checkbox"/>	
ae4 -> msc_leaf_5110_001_syn004 (Interface)		<input type="checkbox"/>	

Assign

Navigate to **Staged > Connectivity Templates > Add Template > Primitives > Custom Static Route** to inject default route:

The screenshot shows the 'Edit Connectivity Template' window with the 'Primitives' tab selected. The 'Custom Static Route' configuration is as follows:

- Routing Zone:** NSX\_VRF
- Network:** 0.0.0.0/0
- Next Hop IP Address:** 9.9.9.100

Callouts from the configuration fields:

- Same Routing Zone (points to NSX\_VRF)
- Static Route to 0/0 (points to 0.0.0.0/0)
- NH is the uplink we specified earlier (points to 9.9.9.100)

### Set Up BGP IPv6 towards NSX-T Edge

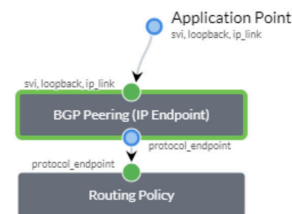
We can enable IPv6-based BGP neighborship between T0 Gateway and ToR leaf using connectivity templates.

See "Set up NSX-T VRF Lite" section for details on creating uplink VLAN interfaces on T0 Gateway. This VLAN should be IPv6-enabled.

Create a connectivity template for each of the VXLAN VN and enable BGP towards IPv6 neighbor on NSX-T Edge as below:




The screenshot shows the 'TTL' configuration section with the following settings:




- TTL:** 2
- Single-hop BFD:** OFF
- IPv6 Address:** 2021:310:310::2/64



*Un-assign BGP on VXLAN VN towards NSX-T Edge*

Let's say BGP neighborship from Tier-0 Gateway in NSX-T needs to be torn down towards ToR Leaf. In this case we need to unassign the interfaces in the **Virtual Network** based Connectivity Template used for BGP peering so that it is in the **Ready** state, and then delete the connectivity template:

0 selected	Name ↕	Description	Tags	Status	Actions
	BGP vlan 150			Ready	 

Type ↕	Action ↕	Name ↕
Floating IPs	 REMOVED	5362661d-6e64-41c3-937b-ac974f20b5c0
Protocol Sessions	 REMOVED	9005b70b-67e5-4db1-aa4c-70e408614726
Virtual Network	 CHANGED	nsxt_vlan150

VMware NSX-T Inventory Mapping to Apstra Virtual Infrastructure

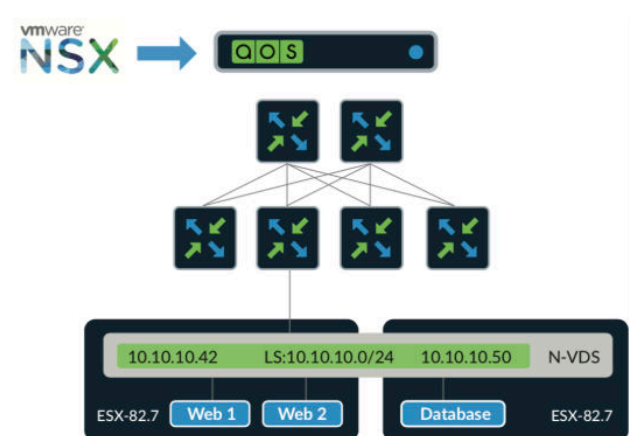
IN THIS SECTION

- [Overview | 542](#)
- [NSX-T Networking Terminology and correlation | 543](#)
- [NSX Inventory Model | 551](#)
- [Model Details and Relationship | 552](#)

*Overview*

Apstra software can connect to the NSX-T API to gather information about the inventory in terms of hosts, clusters, VMs, portgroups, vDS/N-vDS, and NICs within the NSX-T environment. Apstra can integrate with NSX-T to provide Apstra admins visibility into the application workloads (aka VMs) running and alert them about any inconsistencies that would affect workload connectivity. **Apstra Virtual Infrastructure visibility** helps provide underlay/overlay correlation visibility and use IBA analytics for overlay/underlay.

You cannot view the NSX Inventory in Apstra until the NSX-T manager is associated to a blueprint.



As per above screenshot inventory collection for NSX-T is done via Apstra extensible telemetry collector.

### ***NSX-T Networking Terminology and correlation***

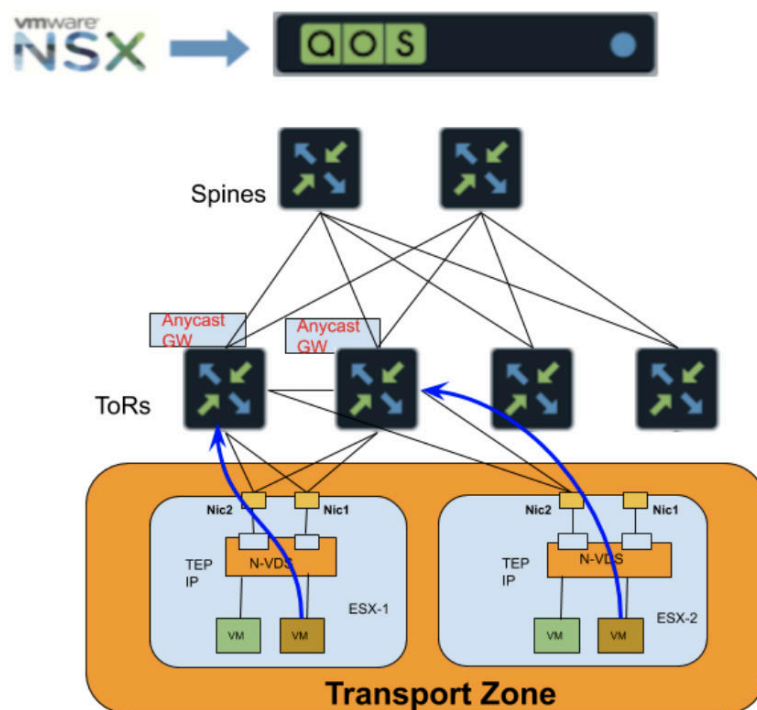
#### **IN THIS SECTION**

- [Transport Zones | 544](#)
- [N-VDS | 546](#)
- [Transport Node | 549](#)
- [NSX Edge Node | 550](#)
- [NSX Controller Cluster | 550](#)
- [NSX Manager | 550](#)

NSX-T uses the following terminology for their control plane and data plane components. Also please find respective correlation with respect to Apstra.

## Transport Zones

Transport Zones (TZ) define a group of ESXi hosts that can communicate with one another on a physical network.



There are two types of Transport Zones:

1. **Overlay Transport Zone:** This transport zone can be used by both transport nodes or NSX edges. When an ESXi host or NSX-T Edge transport node is added to an Overlay transport zone, an N-VDS is installed on the ESXi host or NSX Edge Node.
2. **VLAN Transport Zone:** It can be used by NSX Edge and host transport nodes for its VLAN uplinks.

Each Hypervisor Hosts can only belong to one Transport Zone at a given point of time.

A newly created VLAN VN tagged towards an interface in Apstra fabric corresponds to a VLAN based transport zone as per the screenshots below:

Create Virtual Network

vn3000

default

✕

VNI ID

30000

DHCP Service

☒ Disabled

☐ Enabled

IPv4 Connectivity

☐ Disabled

☒ Enabled

IPv4 Subnet

172.16.5.0/24

Virtual Gateway IPv4

172.16.5.1

Default Endpoint Types

Link Label	Tag Type
link	<div><div><input type="radio"/> Unassigned</div><div><input type="radio"/> Untagged</div><div><input checked="" type="radio"/> VLAN Tagged</div></div>

Assigned To

▸ Query: All

1-2 of 2

<

>

Page Size: 25

▼

<input checked="" type="checkbox"/>	Bound To	Link Labels	VLAN ID	IPv4 Address
<input checked="" type="checkbox"/>	rack1_001_leaf1	link	3000	172.16.5.2/24

☐ Create Another?

Create

Here tagged VLAN VN is mapped to the respective Transport Zone in NSX-T with traffic type as VLAN.

**New Transport Zone** ⓘ ✕

Name \*

Description

N-VDS Name \*

N-VDS Mode

- ☒ Standard
- ☐ Enhanced Datapath

Traffic Type

- ☐ Overlay
- ☒ VLAN

Uplink Teaming Policy Names

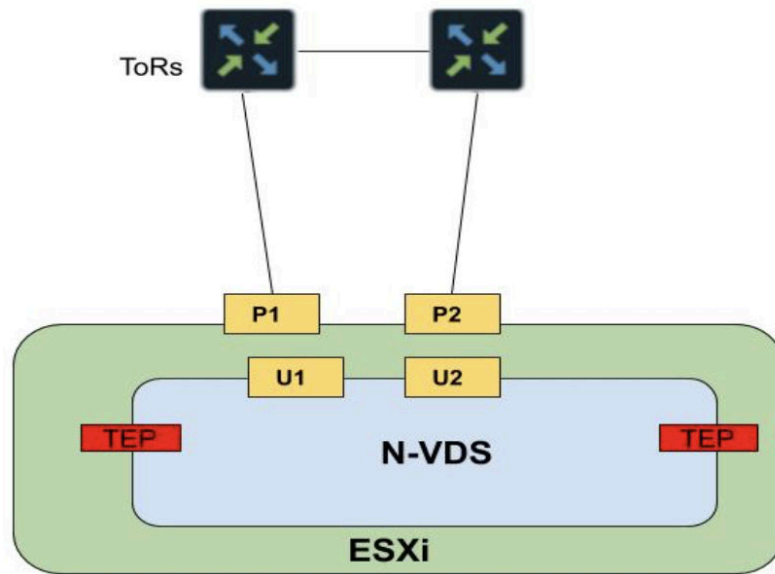
### **N-VDS**

An NSX-managed virtual distributed switch provides the underlying forwarding and is the data plane of the transport nodes.

A few notables about N-VDS virtual switches include:

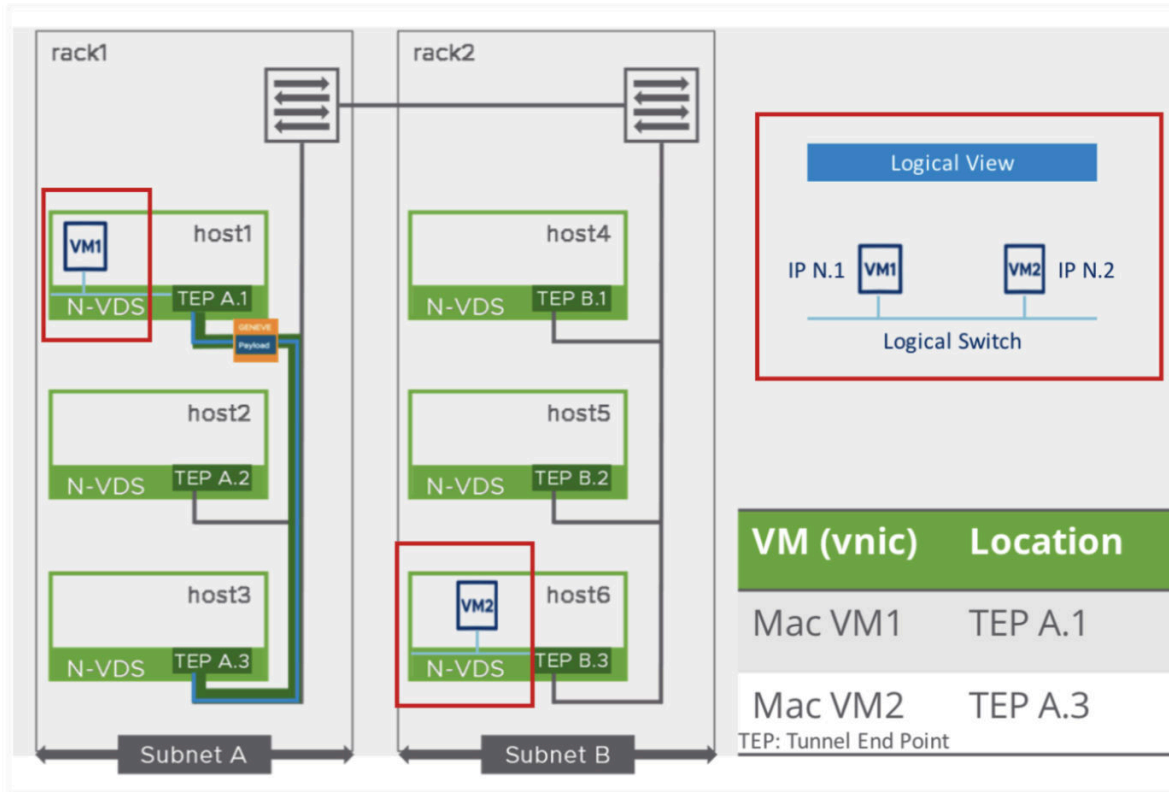
- pnics are physical ports on the ESXi host
- pnics can be bundled to form a link aggregation (LAG)
- uplinks are logical interfaces of an N-VDS

- uplinks are assigned pnic's or LAGs



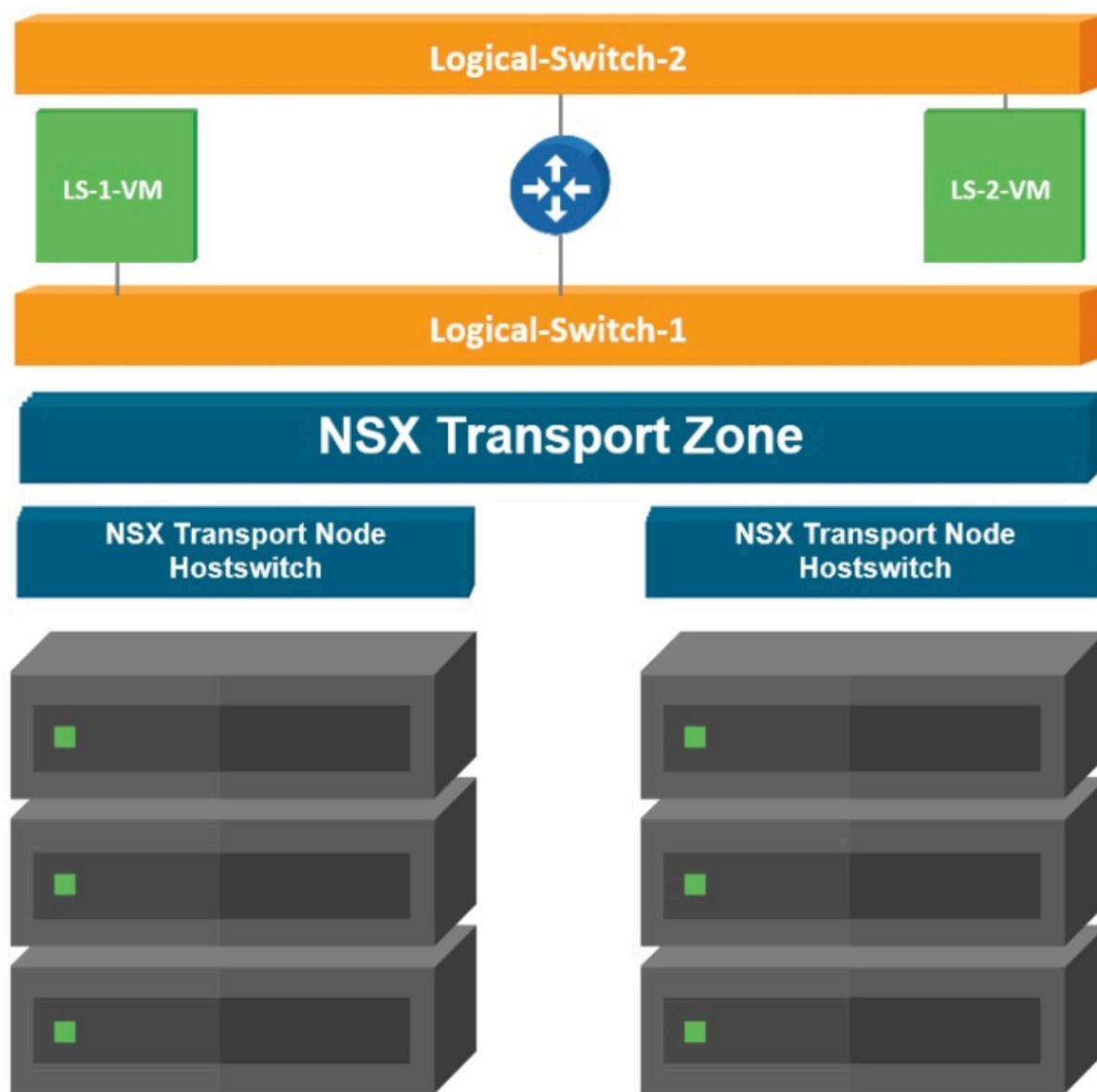
Here TEP are Tunnel Endpoints used for the NSX overlay networking (geneve encapsulation/decapsulation). P1/P2 are pNICs mapped to the uplink profile(U1/U2).

N-VDS are instantiated at the Hypervisor level and can be thought of Virtual switch connected to the ToR physical leafs as below:



### Transport Node

It is a node capable of participating in an NSX-T Data Center overlay or VLAN networking.



VMs hosted on different Transport nodes communicate seamlessly across the overlay network. A transport node can belong to:

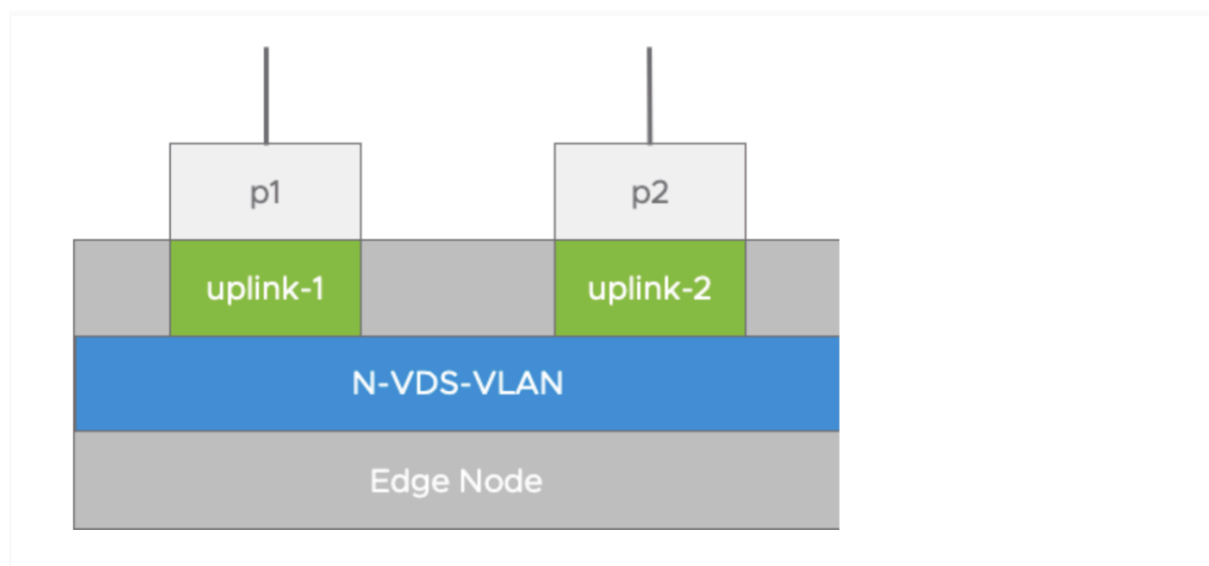
- Multiple VLAN transport zones.
- At most one overlay transport zone with a standard N-VDS.

This can be compared to setting end hosts(servers) in an Apstra blueprint to be part of VLAN (leaf-local) or VXLAN (inter-leaf) Virtual Network.

### ***NSX Edge Node***

The NSX Edge provides routing services and connectivity to networks that are external to the NSX-T deployment. It is required for establishing external connectivity from the NSX-T domain, through a Tier-0 router via BGP or static routing.

NSX Edge VMs have uplinks towards ToR leaves needing a separate VLAN transport zone. Apstra fabric must be configured with the corresponding VLAN Virtual Network.



**NOTE:** NSX-T Edge Bare Metal or VM form factors are Transport nodes and discovered as hypervisors in Apstra. However, VM edge Transport nodes can't be correlated to the connected ToR Leaf.

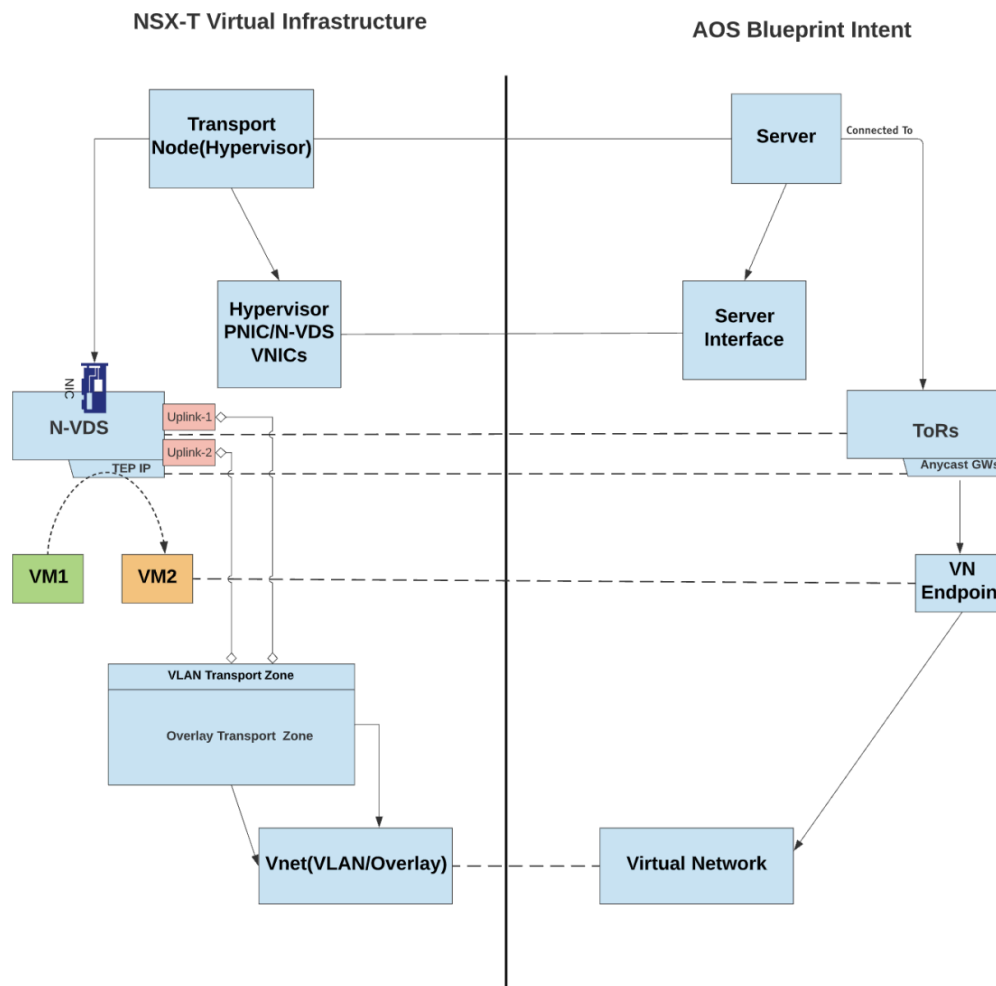
### ***NSX Controller Cluster***

It provides control plane functions for NSX-T Data Center logical switching and routing components.

### ***NSX Manager***

It is a node that hosts the API services, the management plane, and the agent services.

## NSX Inventory Model



- In NSX-T Transport nodes are hypervisor hosts and they can be correlated to server nodes in a Blueprint connected to the ToR leafs. In NSX-T Data Center, ESXi hosts are prepared as Transport Node which allows nodes to exchange traffic for virtual networks on Apstra Fabric or amongst network on nodes. You must ensure hypervisors (ESXi) networking stack is sending LLDP packets to aid the correlation of ESXi hosts with server nodes in the blueprint.
- PNIC is the actual physical network adapter on ESXi or hypervisor host. Hypervisor PNICs can be correlated to the server interface on the Blueprint. LAG or Teaming configuration is done on the links mapped to these physical NICs. This can be correlated to bond configuration done on the ToR leafs towards the end servers.
- In NSX-T integration with Apstra VM virtual networks are discovered. These can be correlated to blueprint virtual networks. In case VMs need to communicate with each other over tunnels between hypervisors VMs are connected to the same logical switch in NSX-T(called N-VDS). Each logical switch has a virtual network identifier (VNI), like a VLAN ID. This corresponds to VXLAN VNIs as in Apstra fabric physical infrastructure.

- The NSX-T Uplink Profile defines the network interface configuration facing the fabric in terms of LAG and LACP config on PNIC interfaces. The uplink profile is mapped in Transport node for the links from the hypervisor/ESXi towards top-of-rack switches in Apstra Fabric.
- VNIC defines Virtual Interface of transport nodes or VMs. N-VDS switch does mapping of physical NICs to such uplink virtual interfaces. These Virtual Interfaces can be correlated to server interface ports of Apstra Fabric.

### *Model Details and Relationship*

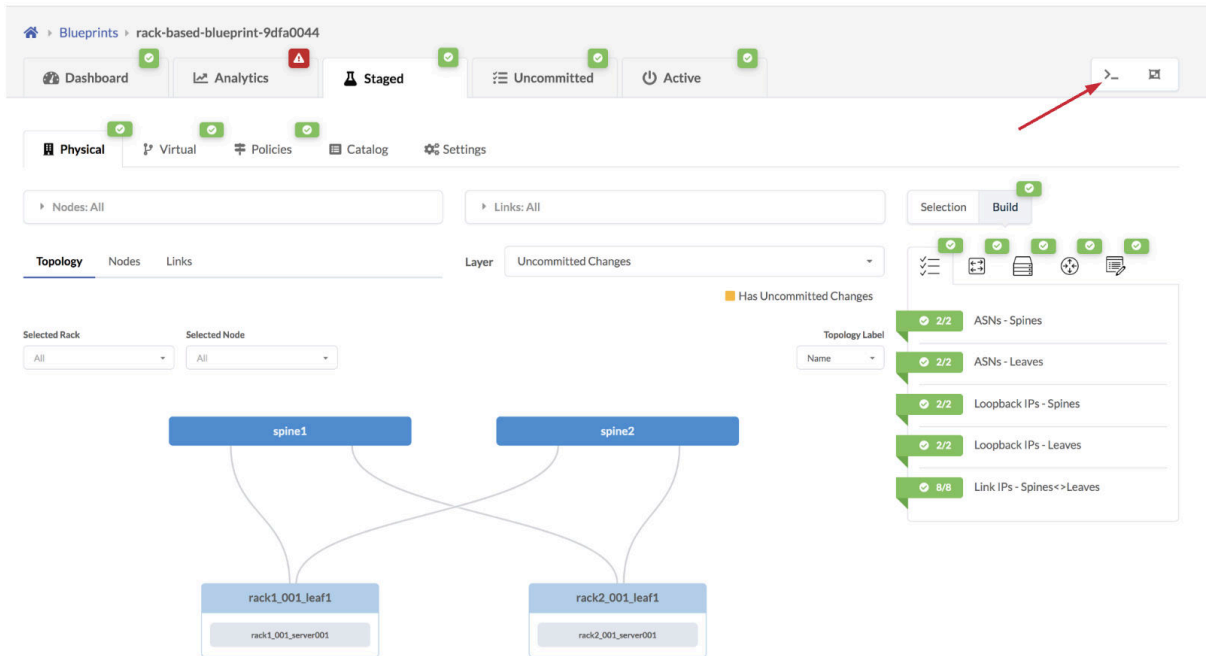
#### IN THIS SECTION

- [Hypervisor | 552](#)
- [Hypervisor PNIC | 555](#)
- [VNIC | 562](#)
- [Port Channel Policy | 568](#)
- [Vnet | 573](#)

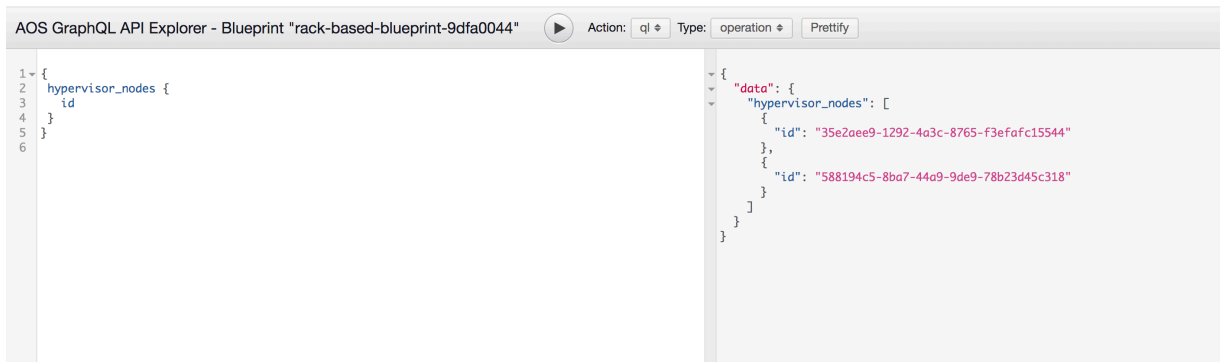
### *Hypervisor*

- **Hostname:** FQDN attribute of transport node
- **Hypervisor\_id:** Id attribute of transport node
- **Label:** Display name attribute of transport node
- **version:** NSX-T version installed on the transport node

To obtain NSX-T API response for respective hypervisor hosts and understand the correlation you can use graph query. To open the GraphQL Explorer, click the ">\_" button



After that in the graph explorer we can type a graph query on the left as per the screenshot below using GraphQL:



To check for respective Label for the transport nodes below query can be used:

Request:

```
{
  hypervisor_nodes{
    label
  }
}
```

Response:

```
{
  "data": {
    "hypervisor_nodes": [
      {
        "label": "zz-karun-nsxt.cvx.2485377892354-357746820-TN-2"
      },
      {
        "label": "zz-AndyF-nsxt.cvx.2485377892354-4240714876-TN-2"
      }
    ]
  }
}
```

Hypervisors which act as Transport Nodes can be visualized in Apstra under **Active** tab with **Has Hypervisor = Yes** option as below:

The screenshot shows the Apstra interface with the 'Active' tab selected. The 'Nodes: Has Hypervisor? = yes' filter is applied. The table displays two nodes, both with 'Has VMs?' set to 'Yes'.

Name	Role	Deploy Mode	Device Profile	S/N	Hostname	ASN	Loopback IPv4	Loopback IPv6	Deploy Status	Hypervisor	VMs Count
rack1_001_server001	L2 server	Deploy	Generic_Server_1RU_2x10G	Not assigned	server-1	N/A	N/A	N/A	N/A	zz-cvx-nsxt.cvx.2485377892354-2902673742-TN-1	0
rack2_001_server001	L2 server	Deploy	Generic_Server_1RU_2x10G	Not assigned	server-2	N/A	N/A	N/A	N/A	zz-cvx-nsxt.cvx.2485377892354-2902673742-TN-2	0

On the right side, the 'Anomalies' panel shows a list of anomalies with counts:

- Anomalies: All Services: 0
- Anomalies: BGP: 0
- Anomalies: Cabling: 0
- Anomalies: Hostname: 0
- Anomalies: Interface: 0
- Anomalies: LAG: 0
- Anomalies: Liveness: 0
- Anomalies: MLAG: 0
- Anomalies: Probes: 30
- Anomalies: Route: 0
- Deploy Mode: 4/0/0

To obtain respective hostname for the transport nodes below query can be used:

Request:

```
{
  hypervisor_nodes {
    hostname
  }
}
```

Response:

```
{
  "data": {
    "hypervisor_nodes": [
      {
        "hostname": "localhost"
      },
      {
        "hostname": "ubuntu-bionic-nsxt"
      }
    ]
  }
}
```

### ***Hypervisor PNIC***

- **MAC address:** Physical address attribute of transport node's interface
- **Switch\_id:** Switch name attribute of transport node's transport zone
- **Label:** Interface id attribute of transport node's interface
- **Neighbor\_name:** System name attribute of transport node's interface lldp neighbor
- **Neighbor\_intf:** Name attribute of transport node's interface lldp neighbor
- **MTU:** MTU attribute of transport node's interface

Physical NICs are selected for uplink profile dedicated for the Overlay Network. NSX-T Uplink Profile defines the network interface configuration for the PNIC interfaces facing the Apstra fabric in terms of

LAG and LACP config.

**Add Transport Node**

General \* **Host Switches \***

Host Switch Type ☒ Standard ☐ Preconfigured

**+ ADD HOST SWITCH**

**New Node Switch**

Host Switch Name \* overlay-hostswitch

Uplink Profile \* nsx-default-uplink-hostswitch-profile

IP Assignment \* Use IP Pool

IP Pool \* TEP IPs

Physical NICs

vmnic2	vmnic3	uplink-1	uplink-2
--------	--------	----------	----------

vmnic0: Up (00:50:56:87:2c:4e)  
**vmnic3: Up (00:50:56:87:b9:24)**  
 vmnic2: Up (00:50:56:87:9f:b9)  
 vmnic1: Up (00:50:56:87:2b:3b)

**SAVE CANCEL**

So the uplink profile is mapped in Transport node for the links from the NSX-T logical switch of the hypervisor/ESXi hosts. It points towards top-of-rack switches in Apstra Fabric.

NSX-API Request/Response to check MAC address for the Transport node interfaces.

Request:

```
{
  pnic_nodes {
    id mac_address
```

```
}
}
```

Response:

```
{
  "data": {
    "pnics": [
      {
        "id": "1e2162c3-9ce6-4f35-afc2-217bb48ced49",
        "mac_address": "52:54:00:88:41:28"
      },
      {
        "id": "9752a438-1939-4648-bc8e-0494addf7c7e",
        "mac_address": "52:54:00:04:d5:4f"
      }
    ]
  }
}
```

The MAC address shown in above example is learned on a LAG interface in Apstra Fabric towards the NSX-T Transport Node. It is the MAC address of the ESXi host pNICs having LAG bond towards ToR leafs in Apstra fabric.

The NSX-API Request/Response below checks the switch name attribute of transport node's transport zone.

Request:

```
{
  pnics {
    id switch_id
  }
}
```

Response:

```
{
  "data": {
    "pnics": [
      {
```

```

    "id": "82586be7-2998-401f-82ba-11afa5bb9730",
    "switch_id": "zz-cvx-nsxt.cvx.2485377892354-2902673742"
  },
  {
    "id": "0043d742-405a-454f-9e9b-695d5dd14608",
    "switch_id": "zz-cvx-nsxt.cvx.2485377892354-2902673742"
  }
]
}
}

```

Switch ID attribute of the respective transport zone are read by NSX-T API from NSX manager as below:

Transport Zones							
+ ADD EDIT DELETE ACTIONS View							
Transport Zone	ID	Traffic Type	N-VDS Name	Status	Host Membership Criteria	Where Used	
<input type="checkbox"/> DEMO NSX-T Transport zone	b9d4...960f	Overlay	zz-clarie-nsxt.cvx.24853...	Unknown	Standard	Where Used	
<input type="checkbox"/> DEMO-NEW-VLAN142	c9f9...f9dc	VLAN	zz-clarie-nsxt.cvx.24853...	Unknown	Standard	Where Used	
<input type="checkbox"/> chiahui-82-tz	9ff3...23a7	Overlay	chiahui-82-nvds	Unknown	Standard	Where Used	
<input type="checkbox"/> mahi-nsxt-kvm-debug_OVERLAY	d39a...6dbf	Overlay	mahi-nsxt-kvm-debug	Unknown	Standard	Where Used	
<input type="checkbox"/> mahi-nsxt-kvm_OVERLAY	d860...eaf5	Overlay	mahi-nsxt-kvm	Unknown	Standard	Where Used	
<input type="checkbox"/> rags-76-test	e53f...68da	Overlay	rags-76-test	Unknown	Standard	Where Used	
<input type="checkbox"/> zz-cvx-nsxt.cvx.2485377892354-2...	6bff...adb4	Overlay	zz-cvx-nsxt.cvx.248537...	Unknown	Standard	Where Used	
<input checked="" type="checkbox"/> zz-cvx-nsxt.cvx.2485377892354-2...	fd04...37aa	VLAN	zz-cvx-nsxt.cvx.248537...	Unknown	Standard	Where Used	
<input type="checkbox"/> zz-naman-nsxt.cvx.248537789235...	c005...1007	Overlay	zz-naman-nsxt.cvx.2485...	Unknown	Standard	Where Used	
<input type="checkbox"/> zz-naman-nsxt.cvx.248537789235...	8654...5bff	VLAN	zz-naman-nsxt.cvx.2485...	Unknown	Standard	Where Used	

NSX-API Request/Response to check Transport node's interface.

Request:

```

{
  pnic_nodes {
    id label
  }
}

```

Response:

```

{
  "data": {
    "pnic_nodes": [
      {

```

```

      "id": "82586be7-2998-401f-82ba-11afa5bb9730",
      "label": "eth2"
    },
    {
      "id": "0043d742-405a-454f-9e9b-695d5dd14608",
      "label": "eth1"
    },
    {
      "id": "b91a5725-7500-489b-a454-e05d7c311525",
      "label": "eth0"
    }
  ]
}

```

Transport nodes has the mapping of physical NICs which can be seen returned as labels according to above NSX-T API response.

zz-cvx-nsxt.cvx.2485377892354-2902673742-TN-2

Overview

Monitor

Physical Adapters

N-VDS Visualization

Related ▾

Interface Id	Admin Status	Link Status	MTU	Interface Details	Stats
nsx-switch.0	<div><div></div>Down</div>	<div><div></div>Down</div>	1600		1 <div><div></div></div>
ovs-greTap0	<div><div></div>Down</div>	<div><div></div>? Unknown</div>	1462		1 <div><div></div></div>
lo	<div><div></div>Up</div>	<div><div></div>Up</div>	65536		1 <div><div></div></div>
gre0	<div><div></div>Down</div>	<div><div></div>? Unknown</div>	1476		1 <div><div></div></div>
ovs-ip6gre0	<div><div></div>Down</div>	<div><div></div>? Unknown</div>	1448		1 <div><div></div></div>
ovs-system	<div><div></div>Down</div>	<div><div></div>Down</div>	1500		1 <div><div></div></div>
nsx-vtep0.0	<div><div></div>Up</div>	<div><div></div>Up</div>	1600		1 <div><div></div></div>
nsx-managed	<div><div></div>Down</div>	<div><div></div>Down</div>	1500		1 <div><div></div></div>
eth2	<div><div></div>Up</div>	<div><div></div>Up</div>	1600		1 <div><div></div></div>
eth1	<div><div></div>Up</div>	<div><div></div>Up</div>	1600		1 <div><div></div></div>
eth0	<div><div></div>Up</div>	<div><div></div>Up</div>	1500		1 <div><div></div></div>
hyperbus	<div><div></div>Up</div>	<div><div></div>Up</div>	1500		1 <div><div></div></div>
ovs-ip6tnl0	<div><div></div>Down</div>	<div><div></div>? Unknown</div>	1452		1 <div><div></div></div>
erspan0	<div><div></div>Down</div>	<div><div></div>? Unknown</div>	1450		1 <div><div></div></div>

Please find below NSX-API Request/Response to check Transport node's LLDP neighbor System name attribute.

Request:

```
{
  pnic_nodes {
    id neighbor_name
  }
}
```

Response:

```
{
  "data": {
    "pnice_nodes": [
      {
        "id": "82586be7-2998-401f-82ba-11afa5bb9730",
        "neighbor_name": "leaf-2-525400C6DD2B"
      },
      {
        "id": "0043d742-405a-454f-9e9b-695d5dd14608",
        "neighbor_name": "leaf-2-525400C6DD2B"
      },
      {
        "id": "b91a5725-7500-489b-a454-e05d7c311525",
        "neighbor_name": "spine-1"
      },
      {
        "id": "f77575fb-44ea-4ec7-9913-1c75b7af87bc",
        "neighbor_name": "leaf-1-5254004D5560"
      },
      {
        "id": "628d0f86-4bc1-4faf-8f3f-f1deb92ceee2",
        "neighbor_name": "leaf-2-525400C6DD2B"
      },
      {
        "id": "1e2162c3-9ce6-4f35-afc2-217bb48ced49",
        "neighbor_name": "leaf-1-5254004D5560"
      }
    ]
  }
}
```

Here Leaf1/2 are LLDP neighbors to the Transport nodes.

To obtain respective transport node's LLDP neighbor interface name attribute below query can be used:

Request:

```
{
  pnic_nodes {
    id neighbor_intf
  }
}
```

Response:

```
{
  "data": {
    "pnic_nodes": [
      {
        "id": "82586be7-2998-401f-82ba-11afa5bb9730",
        "neighbor_name": "leaf-2-525400C6DD2B"
      },
      {
        "id": "0043d742-405a-454f-9e9b-695d5dd14608",
        "neighbor_name": "leaf-2-525400C6DD2B"
      },
      {
        "id": "b91a5725-7500-489b-a454-e05d7c311525",
        "neighbor_name": "spine-1"
      },
      {
        "id": "f77575fb-44ea-4ec7-9913-1c75b7af87bc",
        "neighbor_name": "leaf-1-5254004D5560"
      },
      {
        "id": "628d0f86-4bc1-4faf-8f3f-f1deb92ceee2",
        "neighbor_name": "leaf-2-525400C6DD2B"
      },
      {
        "id": "1e2162c3-9ce6-4f35-afc2-217bb48ced49",
        "neighbor_name": "leaf-1-5254004D5560"
      }
    ]
  }
}
```

```
}
}
```

NSX-API Request/Response to check the MTU attribute of Transport node's interface.

Request:

```
{
  pnic_nodes {
    id neighbor_intf
  }
}
```

Response:

```
{
  "data": {
    "pnic_nodes": [
      {
        "id": "82586be7-2998-401f-82ba-11afa5bb9730",
        "neighbor_intf": "swp4"
      },
      {
        "id": "0043d742-405a-454f-9e9b-695d5dd14608",
        "neighbor_intf": "swp3"
      },
      {
        "id": "b91a5725-7500-489b-a454-e05d7c311525",
        "neighbor_intf": "eth0"
      }
    ]
  }
}
```

MTU size of 1600 or greater is needed on any network that carries Geneve overlay traffic must. Hence in the NSX-T reply we can notice MTU value 1600 on network interfaces towards Transport nodes.

### **VNIC**

- **MAC address:** Physical address attribute of transport node's or VM's Virtual interface
- **Label:** VNIC label attribute of transport node

- **Ipv4\_addr:** IP address attribute of transport node's virtual interface
- **Traffic\_types:** It is derived from transport node's virtual interface type
- **MTU:** MTU attribute of transport node's virtual interface

You can check the VNIC mac address attribute with the below NSX-API Request/Response. This can be of transport node's interface Virtual Interface or can be for the Virtual Interface of the VMs. For transport nodes under Host Switches select the Virtual NIC that matches the MAC address of the VM NIC attached to the uplink port group.

Request:

```
{
  vnic_nodes{
    id mac_address
  }
}
```

Response:

```
{
  "data": {
    "vnic_nodes": [
      {
        "id": "c84d8636-c28b-4db3-8747-37fadca4c7aa",
        "mac_address": "1e:5c:3b:a2:ea:c3"
      },
      {
        "id": "7d5826d8-0622-4a45-88d7-6b1e88bac62f",
        "mac_address": "ca:0f:93:24:24:43"
      }
    ]
  }
}
```

NSX-API Request/Response to check VNIC label which signifies interface id attribute of transport node's virtual interface or device name attribute of virtual machine's virtual interface.

Request:

```
{
  vnic_nodes{
    id label
  }
}
```

Response:

```
{
  "data": {
    "vnic_nodes": [
      {
        "id": "c84d8636-c28b-4db3-8747-37fadca4c7aa",
        "label": "hyperbus"
      },
      {
        "id": "7d5826d8-0622-4a45-88d7-6b1e88bac62f",
        "label": "nsx-switch.0"
      },
      {
        "id": "473c2b7d-ab2f-41cd-9a4b-fcf2eb248fd6",
        "label": "nsx-switch.0"
      },
      {
        "id": "9553390b-754e-45ef-8976-e63396d554ee",
        "label": "nsx-vtep0.0"
      },
      {
        "id": "a00bb649-5032-462f-97e7-b6c4f5f1ac86",
        "label": "nsx-vtep0.0"
      }
    ]
  }
}
```

Below is the NSX-API Request/Response to check VNIC Ipv4 address which signifies ip address attribute of transport node's virtual interface or for the virtual interface of logical port.

Request:

```
{
  vnic_nodes{
    id ipv4_addr
  }
}
```

Response:

```
{
  "data": {
    "vnic_nodes": [
      {
        "id": "9553390b-754e-45ef-8976-e63396d554ee",
        "ipv4_addr": "192.168.1.13"
      },
      {
        "id": "a00bb649-5032-462f-97e7-b6c4f5f1ac86",
        "ipv4_addr": "192.168.1.12"
      }
    ]
  }
}
```

Host Transport Nodes   Edge Transport Nodes   Edge Clusters   ESXi Bridge Clusters					
Managed by: None: Standalone Hosts					
<div> <div> <div>+</div> <div>✎</div> <div>✕</div> <div>⚙</div> <div>▼</div> </div> <div> <div>Node</div> <div> <div>☑</div> <div>zz-cvx-nsxt.cvx.2485377892354-29...</div> </div> <div> <div>☐</div> <div>zz-cvx-nsxt.cvx.2485377892354-29...</div> </div> <div> <div>☐</div> <div>zz-gmat-nsxt.veos.2485377892355...</div> </div> <div> <div>☐</div> <div>zz-leblon-dep-nsxt.veos.248537789...</div> </div> <div> <div>☐</div> <div>zz-virt-nsxt.cvx.2485377892354-26...</div> </div> <div> <div>☐</div> <div>zz-virt-nsxt.cvx.2485377892354-34...</div> </div> <div> <div>☐</div> <div>zz-virt-nsxt.nxosv.2485377892354-2...</div> </div> <div> <div>☐</div> <div>zz-virt-nsxt.nxosv.2485377892354-2...</div> </div> <div> <div>☐</div> <div>zz-virt-nsxt.veos.2485377892354-2...</div> </div> <div> <div>☐</div> <div>zz-virt-nsxt.veos.2485377892354-2...</div> </div> <div> <div>☐</div> <div>zz-virt-speci-nsxt.cvx.24853778923...</div> </div> <div> <div>☐</div> <div>zz-virt-speci-nsxt.cvx.24853778923...</div> </div> <div> <div>☐</div> <div>zz-virt-speci-nsxt.cvx.24853778923...</div> </div> <div> <div>☐</div> <div>zz-virt-speci-nsxt.veos.2485377892...</div> </div> <div> <div>☐</div> <div>zz-virt-speci-nsxt.veos.2485377892...</div> </div> <div> <div>☐</div> <div>zz-virt-speci-nsxt.veos.2485377892...</div> </div> <div> <div>☐</div> <div>zz-virt-speci-nsxt.veos.2485377892...</div> </div> <div> <div>☐</div> <div>zz-virt-speci-nsxt.veos.2485377892...</div> </div> </div> </div>					
zz-cvx-nsxt.cvx.2485377892354-2902673742-TN-1					
Overview   Monitor <b>Physical Adapters</b> N-VDS Visualization   Related					
Interface Id	Admin Status	Link Status	MTU	Interface Details	Stats
nsx-switch0	Down	Down	1600	1	
ovs-greTap0	Down	? Unknown	1462	1	
lo	Up	Up	65536	1	
gre0	Down	? Unknown	1476	1	
ovs-ip6gre0	Down	? Unknown	1448	1	
ovs-system	Down	Down	1500	1	
nsx-vtep0.0	Up	Up	1600	1	
nsx-managed	Down	Down	1500	1	
eth2	Up	Up	1600	1	
eth1	Up	Up	1600	1	
eth0	Up	Up	1500	1	
hyperbus	Up	Up	1500	1	
ovs-ip6tni0	Down	? Unknown	1452	1	
erspan0	Down	? Unknown	1450	1	

Here "192.168.1.13" and "192.168.1.12" are ipv4 addresses for the bridge interface of the host transport nodes i.e **"nsx-vtep0.0"** which acts as a virtual tunnel endpoint (VTEP) of the transport node. Each hypervisor has a Virtual Tunnel Endpoint (VTEP) responsible for encapsulating the VM traffic inside a VLAN header and routing the packet to a destination VTEP for further processing. This can be compared to VXLAN Virtual Network anycast GW VTEP IP.

```
nsx-vtep0.0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1600
    inet 192.168.1.12 netmask 255.255.255.224 broadcast 192.168.1.31
    inet6 fe80::c8ec:50ff:fe69:536 prefixlen 64 scopeid 0x20<link>
    ether ca:ec:50:69:05:36 txqueuelen 1000 (Ethernet)
    RX packets 60312 bytes 3975194 (3.9 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 31215 bytes 2675310 (2.6 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
admin@localhost:~$
```

NSX-API Request/Response to check traffic types for the transport node's virtual interface. Traffic type for the transport node can be overlay type as per the example below or it can be of VLAN type. One can add both the VLAN and overlay NSX Transport Zones to the Transport Nodes.

VLAN based Transport zone is mainly for uplink based traffic. In case VMs on different Hypervisor hosts need to communicate to each other then overlay network should be used. It can be compared to VXLAN Virtual network in Apstra Fabric.

Request:

```
{
  vnic_nodes{
    id traffic_types
  }
}
```

Response:

```
{
  "data": {
    "vnic_nodes": [
      {
        "id": "9553390b-754e-45ef-8976-e63396d554ee",
        "traffic_types": [
          "overlay"
        ]
      },
      {
        "id": "a00bb649-5032-462f-97e7-b6c4f5f1ac86",
        "traffic_types": [
          "overlay"
        ]
      }
    ]
  }
}
```

NSX-API Request/Response to obtain the mtu size for the transport node. MTU size for networks that carry overlay traffic must be size of 1600 or greater as it carries Geneve overlay traffic. N-VDS and TEP kernel interface all should have the same jumbo frame MTU size(i.e 1600 or greater).

Request:

```
{
  vnic_nodes{
    id mtu
  }
}
```

Response:

```
{
  "data": {
    "vnic_nodes": [
      {
        "id": "9553390b-754e-45ef-8976-e63396d554ee",
        "mtu": 1600
      },
      {
        "id": "a00bb649-5032-462f-97e7-b6c4f5f1ac86",
        "mtu": 1600
      }
    ]
  }
}
```

Host Transport Nodes   Edge Transport Nodes   Edge Clusters   ESXi Bridge Clusters

Managed by: None: Standalone Hosts

Node

zz-cvx-nsxt.cvx.2485377892354-2902673742-TN-1

Overview   Monitor   **Physical Adapters**   N-VDS Visualization   Related

Interface Id	Admin Status	Link Status	MTU	Interface Details	Stats
nsx-switch0	Down	Down	1600	1	
ovs-greTap0	Down	? Unknown	1462	1	
lo	Up	Up	65536	1	
gre0	Down	? Unknown	1476	1	
ovs-ip6gre0	Down	? Unknown	1448	1	
ovs-system	Down	Down	1500	1	
<b>nsx-vtep0.0</b>	<b>Up</b>	<b>Up</b>	<b>1600</b>	<b>1</b>	<b></b>
nsx-managed	Down	Down	1500	1	
eth2	Up	Up	1600	1	
eth1	Up	Up	1600	1	
eth0	Up	Up	1500	1	
hyperbus	Up	Up	1500	1	
ovs-ip6tni0	Down	? Unknown	1452	1	
erspan0	Down	? Unknown	1450	1	

So Virtual Interface i.e NSX VTEP and vswitch should have mtu of 1600 as per screenshot above.

### Port Channel Policy

- **Label:** Name attribute of the host switch uplink lag profile

- **Mode:** Mode attribute of host switch uplink lag profile
- **Hashing\_algorithm:** Load balance algorithm attribute of host switch uplink lag profile

An uplink profile is mapped in a Transport node on the NSX-T side with policies for the links from the hypervisor hosts to NSX-T logical switches.

Edit Transport Node -  
zz-karun-  
nsxt.cvx.2485377892354  
357746820-TN-2

1 Host Details
2 **Configure NSX**

Configure NSX

Transport Zone \*

zz-karun-nsxt.cvx.2485377892354-357746820\_VLAN
zz-karun-nsxt.cvx.2485377892354-357746820\_OVERLAY

N-VDS Creation \*

☒ NSX Created
☐ Preconfigured

+ ADD N-VDS

New Node Switch

N-VDS Name \*
zz-karun-nsxt.cvx.2485377892354-357746820

Associated Transport Zones
zz-karun-nsxt.cvx.2485377892354-357746820\_VLAN, zz-karun-nsxt.cvx.2485377892354-357746820\_OVERLAY

Uplink Profile \*
zz-karun-nsxt.cvx.2485377892354-357746820\_VLAN-1

LLDP Profile \*
LLDP [Send Packet Enabled]

IP Assignment \*
Use IP Pool

CANCEL
PREVIOUS
FINISH

The links from the Hypervisor hosts to NSX-T logical switches can comprise of the LAG or Teaming configuration which must be tied to physical NICs.

NSX-API Request/Response to check the logical switch uplink LAG profile attribute.

Request:

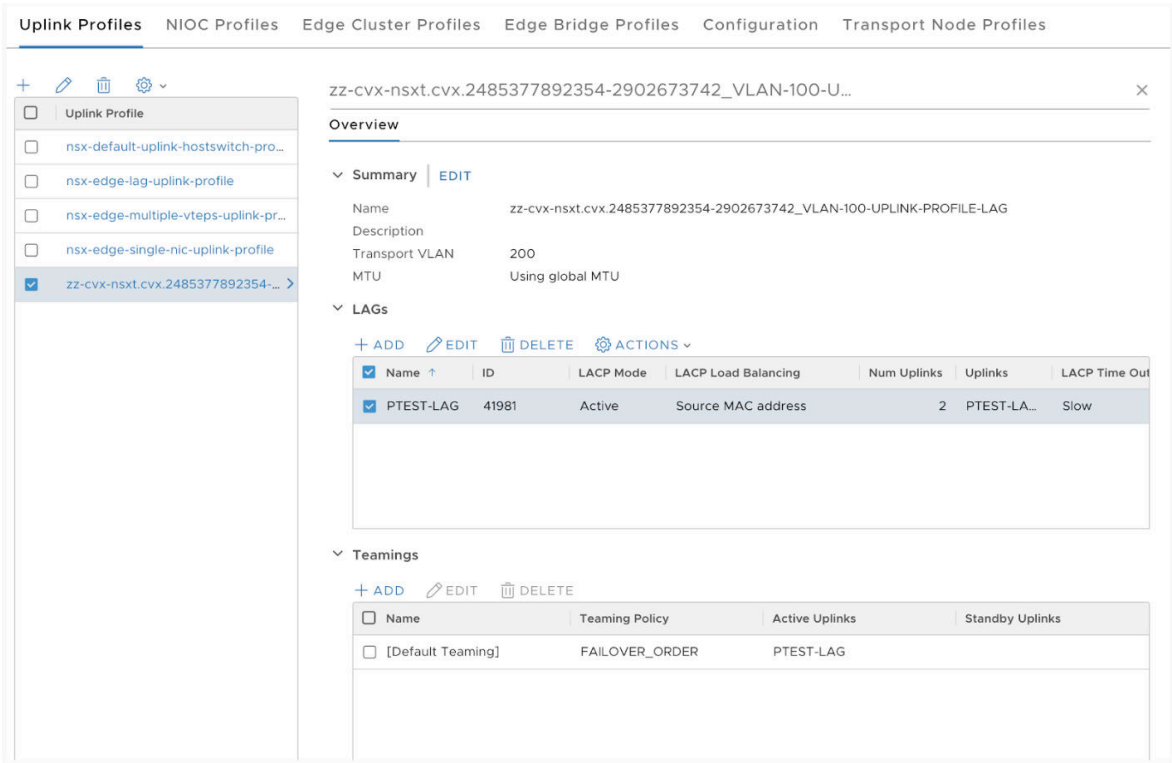
```
{
  port_channel_nodes {
    id label
  } id port_channel_policy_nodes {
    id label
  }
}
```

```
}
}
```

Response:

```
{
  "data": {
    "port_channel_nodes": [
      {
        "id": "bd86666b-239d-4baa-8715-d73ca40d7100",
        "label": null
      },
      {
        "id": "ff5a5b6b-a103-471a-bbfd-ee3dc8c6e1c7",
        "label": null
      }
    ],
    "id": "rack-based-blueprint-9dfa0044",
    "port_channel_policy_nodes": [
      {
        "id": "59f60d47-ca48-441d-a4a4-e570af7bdb72",
        "label": "PTEST-LAG"
      }
    ]
  }
}
```

Uplink profile label can also be matched with one retrieved from the GUI in NSX-T Manager as below:



Below is NSX-API Request/Response to check the LACP mode attribute for the uplink LAG profile.

Request:

```
{
  port_channel_nodes {
    id
  } id port_channel_policy_nodes {
    id mode
  }
}
```

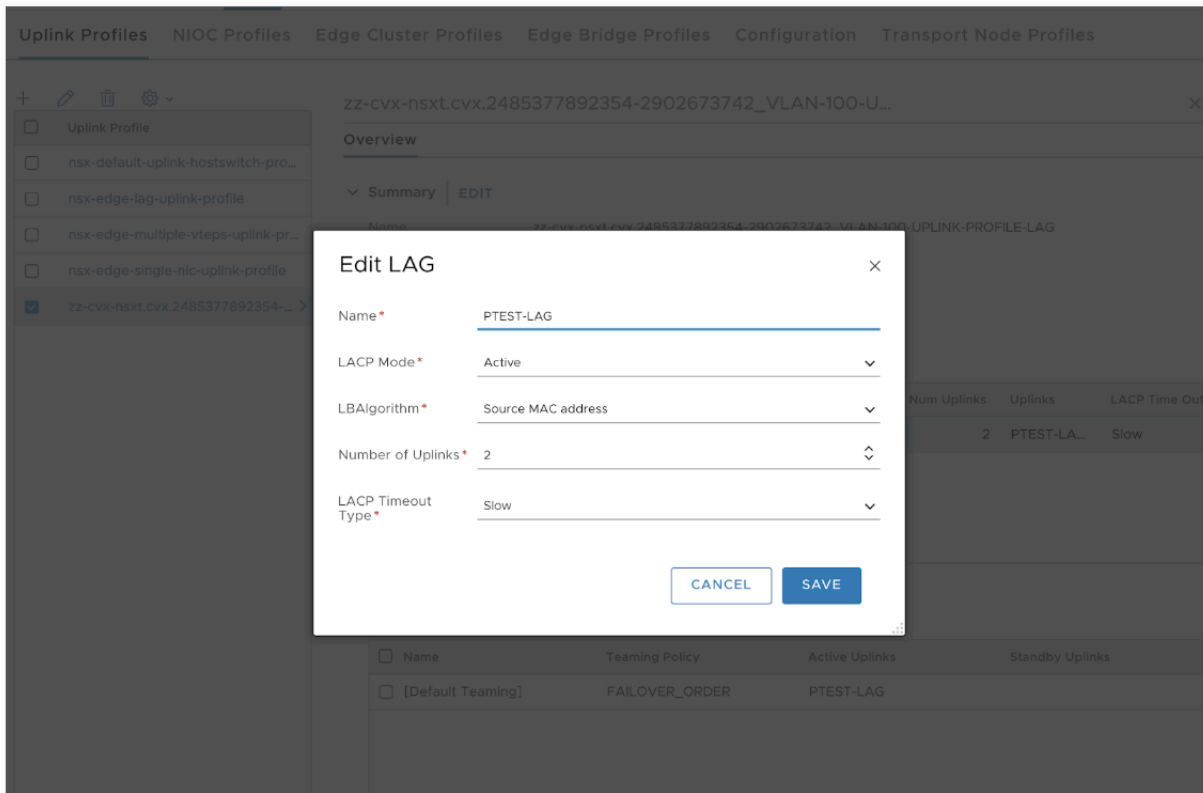
Response:

```
{
  "data": {
    "port_channel_nodes": [
      {
        "id": "bd86666b-239d-4baa-8715-d73ca40d7100"
      },
    ],
  },
}
```

```

    {
      "id": "ff5a5b6b-a103-471a-bbfd-ee3dc8c6e1c7"
    }
  ],
  "id": "rack-based-blueprint-9dfa0044",
  "port_channel_policy_nodes": [
    {
      "id": "59f60d47-ca48-441d-a4a4-e570af7bdb72",
      "mode": "active"
    }
  ]
}
}

```



NSX-API Request/Response to check load balancing algorithm attribute of host switch uplink profile.

Request:

```

{
  port_channel_nodes {

```

```

id
} id port_channel_policy_nodes {
id hashing_algorithm
}
}

```

Response:

```

{
  "data": {
    "port_channel_nodes": [
      {
        "id": "bd86666b-239d-4baa-8715-d73ca40d7100"
      },
      {
        "id": "ff5a5b6b-a103-471a-bbfd-ee3dc8c6e1c7"
      }
    ],
    "id": "rack-based-blueprint-9dfa0044",
    "port_channel_policy_nodes": [
      {
        "id": "59f60d47-ca48-441d-a4a4-e570af7bdb72",
        "hashing_algorithm": "srcMac"
      }
    ]
  }
}

```

From the LAG profile screenshot above it can be validated that it is using Source MAC Address based load balancing algorithm.

### **Vnet**

- **Vn\_type:** Transport type attribute of transport zone
- **Label:** Display name attribute of logical switch
- **switch\_label:** Switch name attribute of transport zone
- **Vlan:** Vlan attribute of logical switch for vlan transport zone
- **Vni:** vni attribute of logical switch for overlay transport zone

To obtain respective transport type attribute of the transport zone below query can be used. This mainly signifies the type of traffic for a transport zone which can be Overlay or VLAN type.

Request:

```
{
  vnet_nodes {
    id vn_type
  } id
}
```

Response:

```
{
  "data": {
    "vnet_nodes": [
      {
        "id": "a3320cc6-601e-4a81-abe9-8464ae054f18",
        "vn_type": "overlay"
      },
      {
        "id": "6bdd7cd9-82eb-433d-8360-076d9daddd1b",
        "vn_type": "vlan"
      }
    ],
    "id": "rack-based-blueprint-9dfa0044"
  }
}
```

Traffic type can also be identified in NSX-T Manager GUI as below:

**New Transport Zone** ⓘ ×

Name \* OVERLAY-TZ

Description

N-VDS Name \* OVERLAY-N-VDS

Host Membership Criteria

- ☒ Standard (For all hosts)
- ☐ Enhanced Datapath (For ESXi hosts with version 6.7 or above)

Traffic Type

- ☒ Overlay
- ☐ VLAN

Uplink Teaming Policy Names

CANCEL ADD

NSX-API Request/Response to check the display name of the N-VDS logical switch.

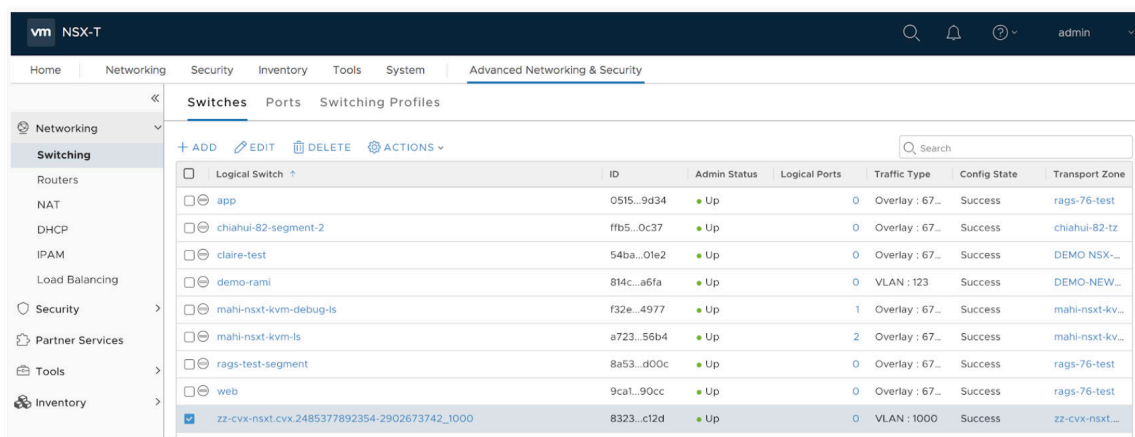
Request:

```
{
  vnet_nodes {
    id label
  } id
}
```

Response:

```
{
  "data": {
    "vnet_nodes": [
      {
        "id": "241ce8e1-b31d-4093-a1a3-2f99a29ac2f9",
        "label": "mahi-nsxt-kvm-ls"
      },
      {
        "id": "fef41435-ac20-4c4d-81c0-b7f3059d977b",
        "label": "zz-cvx-nsxt.cvx.2485377892354-2902673742_1000"
      },
      {
        "id": "6bdd7cd9-82eb-433d-8360-076d9dadd1b",
        "label": "zz-cvx-nsxt.cvx.2485377892354-2902673742_VLAN-100-UPLINK-PROFILE-LAG"
      }
    ],
    "id": "rack-based-blueprint-9dfa0044"
  }
}
```

Here as per API response above “zz-cvx-nsxt.cvx.2485377892354-2902673742\_1000” is the respective logical switch associated with the transport zone.



Logical Switch	ID	Admin Status	Logical Ports	Traffic Type	Config State	Transport Zone
app	0515...9d34	Up	0	Overlay : 67...	Success	rag-76-test
chiahui-82-segment-2	ffb5...0c37	Up	0	Overlay : 67...	Success	chiahui-82-tz
claire-test	54ba...01e2	Up	0	Overlay : 67...	Success	DEMO NSX...
demo-rami	814c...a6fa	Up	0	VLAN : 123	Success	DEMO-NEW...
mahi-nsxt-kvm-debug-ls	f32e...4977	Up	1	Overlay : 67...	Success	mahi-nsxt-kv...
mahi-nsxt-kvm-ls	a723...56b4	Up	2	Overlay : 67...	Success	mahi-nsxt-kv...
rag-76-test-segment	8a53...d00c	Up	0	Overlay : 67...	Success	rag-76-test
web	9ca1...90cc	Up	0	Overlay : 67...	Success	rag-76-test
zz-cvx-nsxt.cvx.2485377892354-2902673742_1000	8323...c12d	Up	0	VLAN : 1000	Success	zz-cvx-nsxt...

Below is the NSX-API Request/Response to check VLAN ID attribute of a VLAN based logical switch for the transport zone.

Request:

```
{
  vnet_nodes {
    id vlan
  } id
}
```

Response:

```
{
  "data": {
    "vnet_nodes": [
      {
        "id": "e0b29951-7739-4ecb-8c87-5725a61f669a",
        "vlan": 123
      },
      {
        "id": "cdd0c6d5-fecb-44d8-84c4-06c685e8ef14",
        "vlan": 2000
      },
      {
        "id": "fef41435-ac20-4c4d-81c0-b7f3059d977b",
        "vlan": 1000
      },
      {
        "id": "6bdd7cd9-82eb-433d-8360-076d9dadd1b",
        "vlan": 200
      }
    ],
    "id": "rack-based-blueprint-9dfa0044"
  }
}
```

Here in Apstra Fabric VNI IDs 1000 and 2000 represent such VXLAN Virtual network for east-west L2 stretched traffic. Bridge backed logical switch on NSX-T should have the same VLAN IDs defined.

NSX-API Request/Response to check the VNI attribute of logical switch of NSX-T

Request:

```
{
  vnet_nodes {
    id vni
  } id
}
```

Response:

```
{
  "data": {
    "vnet_nodes": [
      {
        "id": "a3320cc6-601e-4a81-abe9-8464ae054f18",
        "vni": 67595
      },
      {
        "id": "b7923224-659b-4075-b69b-3edeb5726a32",
        "vni": 67589
      },
      {
        "id": "18b81c81-8ae1-46b1-83ca-05cd5b364a1c",
        "vni": 67584
      }
    ],
    "id": "rack-based-blueprint-9dfa0044"
  }
}
```

## Endpoints Overview (Virtual)

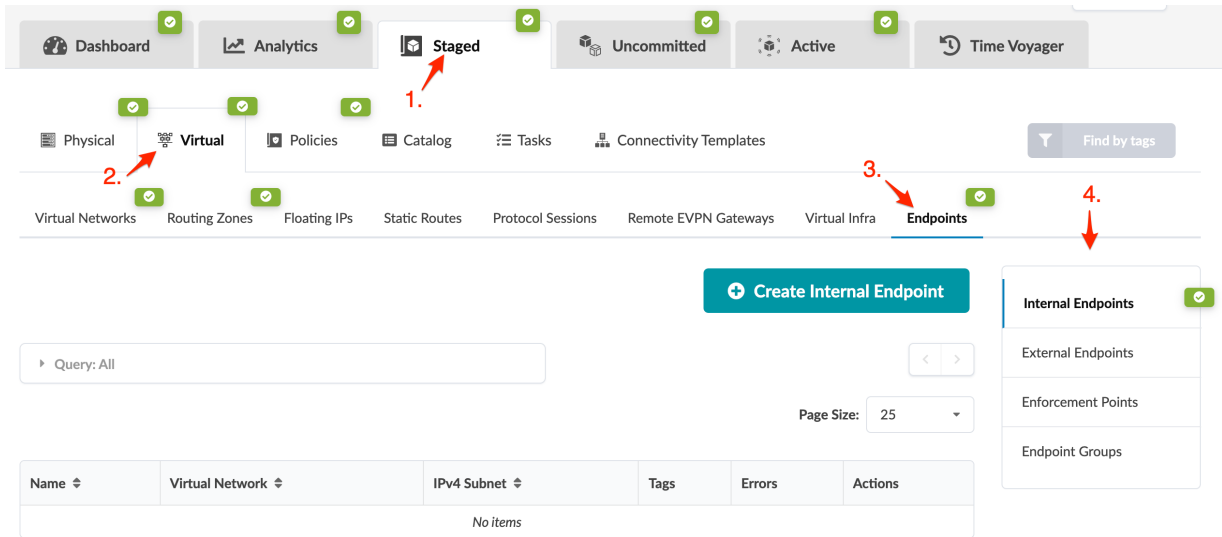
### IN THIS SECTION

- [Internal Endpoints \(Virtual\) | 579](#)
- [External Endpoints \(Virtual\) | 580](#)
- [Enforcement Points \(Virtual\) | 581](#)
- [Endpoint Groups \(Virtual\) | 581](#)

When you want more granularity in your security policies than virtual networks and routing zones can provide, you'll use endpoints. Endpoints can be internal or external to the fabric. You can also combine endpoints into groups.

Endpoints and security policies can be applied to Layer 2 IPv4 blueprints. (Blueprints with IPv6 applications enabled are not supported.) Endpoints are supported on Cisco NX-OS and Arista EOS physical network devices only. For more information about working with security policies, see ["Security Policies" on page 583](#).

From the blueprint, navigate to **Staged > Virtual > Endpoints** to go to endpoints. Click the name of a section to go to its list view. You can create, clone, edit and delete endpoints. Then, when you create a security policy you'll select the endpoints that you've created.



## Internal Endpoints (Virtual)

### IN THIS SECTION

- [Create Internal Endpoint | 579](#)
- [Edit Internal Endpoint | 580](#)
- [Delete Internal Endpoint | 580](#)

### Create Internal Endpoint

1. From the blueprint, navigate to **Staged > Virtual > Endpoints > Internal Endpoints** and click **Create Internal Endpoint**.
2. Configure the endpoint as described below:

Name	32 characters or fewer. Alphanumeric characters, underscores and dashes only.
Virtual Network	Select the virtual network where the endpoint is located.
IPv4 Subnet	Enter the IPv4 Subnet/CIDR.
Tags (optional)	You can add tags for filtering or grouping beyond membership custom groups or virtual networks (for example “web server”, “db” and so on).

3. Click **Create** to stage the endpoint addition and return to the list view. Validation is performed to ensure that the IP address is within the L2 subnet of the virtual network and that no endpoint with the same IP address is within the same routing zone.

#### *Edit Internal Endpoint*

1. From the blueprint, navigate to **Staged > Virtual > Endpoints > Internal Endpoints** and click the **Edit** button for the endpoint to edit.
2. Make your changes.
3. Click **Update** to stage the endpoint change and return to the list view.

#### *Delete Internal Endpoint*

1. From the blueprint, navigate to **Staged > Virtual > Endpoints > Internal Endpoints** and click the **Delete** button for the endpoint to delete.
2. Click **Delete** to stage the endpoint removal and return to the list view.

### External Endpoints (Virtual)

#### IN THIS SECTION

- [Create External Endpoint | 580](#)
- [Edit External Endpoint | 581](#)
- [Delete External Endpoint | 581](#)

#### *Create External Endpoint*

1. From the blueprint, navigate to **Staged > Virtual > Endpoints > External Endpoints** and click **Create External Endpoint**.
2. Configure the endpoint as described below:

Name	32 characters or fewer. Alphanumeric characters, underscores and dashes only.
IPv4 Subnet	Enter the IPv4 Subnet/CIDR.

Tags (optional)	You can add tags for filtering or grouping beyond membership custom groups or virtual networks (for example “web server”, “db” and so on).
Enforcement Points (optional)	Enforcement points are supported on external-facing interfaces on border leaf devices only. They are external-facing points where access lists that involve external endpoints are applied. Any external generic, external connectivity points and enforcement groups can be added.

3. Click **Create** to stage the endpoint addition and return to the list view.

#### *Edit External Endpoint*

1. From the blueprint, navigate to **Staged > Virtual > Endpoints > External Endpoints** and click the **Edit** button for the endpoint to edit.
2. Make your changes.
3. Click **Update** to stage the endpoint change and return to the list view.

#### *Delete External Endpoint*

1. From the blueprint, navigate to **Staged > Virtual > Endpoints > External Endpoints** and click the **Delete** button for the endpoint to delete.
2. Click **Delete** to stage the endpoint removal and return to the list view.

### Enforcement Points (Virtual)

Enforcement points are supported on external-facing interfaces on border leaf devices only. They are automatically created when you add external generics or external connectivity points to a blueprint.

From the blueprint, navigate to **Staged > Virtual > Endpoints > Enforcement Points** to go to enforcement points.

### Endpoint Groups (Virtual)

#### IN THIS SECTION

- [Create Endpoint Group | 582](#)
- [Edit Endpoint Group | 582](#)
- [Delete Endpoint Group | 582](#)

### Create Endpoint Group

1. From the blueprint, navigate to **Staged > Virtual > Endpoints > Endpoint Groups** and click **Create Endpoint Group**.
2. Configure the endpoint group as described below:

Name	32 characters or fewer. Alphanumeric characters, underscores and dashes only
Type	Select the type of endpoint group to create: <b>Internal Endpoint Group</b> , <b>External Endpoint Group</b> , or <b>Enforcement Point Group</b> .
Members	<p>Depending on the type of endpoint group you are creating, options for selecting members are presented.</p> <ul style="list-style-type: none"> <li>• <b>Internal Endpoint Group</b> - Select multiple internal endpoints or other internal endpoint groups.</li> <li>• <b>External Endpoint Group</b> - Select multiple external endpoints or other external endpoint groups, then select enforcement points or enforcement point groups to associate with the external endpoint group.</li> <li>• <b>Enforcement Points Group</b> - Select multiple enforcement points or other enforcement point groups.</li> </ul>

3. Click **Create** to stage the endpoint group addition and return to the list view.

### Edit Endpoint Group

1. From the blueprint, navigate to **Staged > Virtual > Endpoints > Endpoint Groups** and click the **Edit** button for the endpoint group to edit.
2. Make your changes.
3. Click **Update** to stage the endpoint group change and return to the list view.

### Delete Endpoint Group

1. From the blueprint, navigate to **Staged > Virtual > Endpoints > Endpoint Groups** and click the **Delete** button for the endpoint group to delete.
2. Click **Delete** to stage the endpoint group removal and return to the list view.

## Policies

### IN THIS SECTION

- [Security Policies | 583](#)
- [Interface Policies | 591](#)
- [Routing Policies | 599](#)
- [Fabric Addressing Policy | 603](#)
- [Virtual Network Policy | 604](#)
- [Anti-Affinity Policy | 606](#)
- [Modify SVI IP Validation Policy | 608](#)

## Security Policies

### IN THIS SECTION

- [Security Policy Overview | 583](#)
- [Security Policy Parameters | 585](#)
- [Create Security Policy | 587](#)
- [Policy Errors | 588](#)
- [Edit Security Policy | 589](#)
- [Delete Security Policy | 589](#)
- [Security Policy Search | 589](#)
- [Security Policy Conflicts | 590](#)
- [Security Policy Settings | 591](#)

### Security Policy Overview

Endpoint connectivity is determined by reachability (the correct forwarding state in the network) and security (connectivity must be permitted). Policies must be specified between L2 and L3 domains and between more granular L2/L3 IP endpoints. Security policies allow you to permit or deny traffic between the more granular endpoints. They control inter-virtual network traffic (ACLs on SVIs) and

external-to-internal traffic (ACLs in border leafs, external endpoints only). ACLs are rendered in the appropriate device syntax and applied on enforcement points. Adding a new VXLAN Endpoint (for example, adding a rack or adding a leaf to a virtual network) automatically places the ACL on the virtual network interface. Adding a new generic system External Connectivity Point (ECP) (enforcement point) automatically places ACL for external endpoint groups. You can apply security policies to Layer 2 IPv4-enabled blueprints (IPv6 is not supported). For supported devices, refer to the **Connectivity (from Leaf Layer)** table in the Feature Matrix in the Reference section.

Security policies consist of a source point (subnet or IP address), a destination point (subnet or IP address), and rules to allow or deny traffic between those points based on protocol. Rules are stateless, meaning responses to allowed inbound traffic are subject to the rules for outbound traffic (and vice versa).

Rules can include traffic logging. The ACL is configured to log matches using whatever mechanism is supported on the device. Log configuration is local to the network device; It's not on the Apstra server. Parsing these logs is outside the scope of this document.

For a bi-directional security policy, you would create two instances of the policy, one for each direction.

You can apply more than one policy to each subnet/endpoint, which means the ordering of rules has an impact on behavior. An implicit hierarchy exists between routing zones, virtual networks, and IP endpoints, so you must consider how policies are applied at different levels of hierarchy. When one rule's match set contains the other's match set (full containment), the rules can conflict. You can set the rules to execute more specific rules first ("exception" focus/mode) or less specific first ("override" focus/mode).

Rules can also conflict when there is a full containment situation between the rules but the action is the same. In this case, there is potential for compression by using the less specific rule, and the more specific rule becomes a "shadow" rule. When conflicting rules are detected, you are alerted and shown the resolution.

A few cases where conflicting rules are identified are described below:

- Rules in policies between different pairs of IP endpoints (even if one is common to both pairs) are non-overlapping given that the pairs of IP addresses are different. This causes a disjoint match set from a source IP / destination IP perspective (different "IP signature").
- Rules in policies between the same IP endpoints can overlap fields (such as destination port); Apstra software checks for this.
- Rules in policies between different pairs of virtual networks (even if one virtual network is common to both pairs) are non-overlapping given that the pairs of subnets are different. This causes a disjoint match set from the source IP / destination IP perspective (different "IP signature").
- Rules in policies between the same virtual networks can overlap fields (such as destination port); Apstra software checks for this.

- When IP endpoint groups are used, they result in a set of IP endpoint pairs so the above discussion related to IP endpoint pairs applies.
- Rules in policies between a pair of IP endpoints and a pair of parent virtual networks have containment from an IP signature perspective. Apstra software analyzes destination port / protocol overlap and classifies it as full-containment or non-full-containment conflict.
- Rules in policies between a pair of IP endpoints and a pair of virtual networks where at least one virtual network is not parent are non-conflicting (different "IP signature").
- Rules in policies between a pair of IP endpoints and an IP endpoint - virtual network pair where the virtual network is a parent have full containment from an IP signature perspective; Apstra software analyzes the remaining fields.
- Rules in policies that contain external IP endpoints or endpoint groups must be analyzed from an IP signature perspective as external points are not bound by any hierarchical assumptions.
- A routing zone is a set of virtual networks and IP endpoints so the above discussions apply.

To make composition tractable, both from an analysis point of view as well as from comprehending the resulting composition it may be useful to limit the number of security policies that may apply to any given endpoint/group.

### Security Policy Parameters

Security policies include the following details:

Name	Description
Name	32 characters or fewer, underscore, dash and alphanumeric characters only
Description	optional
Enabled	<ul style="list-style-type: none"> <li>• <b>ON</b> to enable security policy (default)</li> <li>• <b>OFF</b> to disable security policy</li> </ul>
Tags	optional

*(Continued)*

Name	Description
Source Point Type	<ul style="list-style-type: none"> <li>• Internal Endpoint (associated with VNs - contain IP /32 address)</li> <li>• External Endpoint (contains /32 or subnet)</li> <li>• External Endpoint Group</li> <li>• Internal Endpoint Group</li> <li>• Virtual Network (contains subnet)</li> <li>• Routing Zone (logical collection of all virtual networks and internal IP endpoints)</li> </ul>
Source Point	Source point (previously created)
Destination Point Type	<ul style="list-style-type: none"> <li>• Internal Endpoint</li> <li>• External Endpoint</li> <li>• External Endpoint Group</li> <li>• Internal Endpoint Group</li> <li>• Virtual Network</li> <li>• Routing Zone</li> </ul>
Destination Point	Destination point (previously created)
Rule Actions	<ul style="list-style-type: none"> <li>• Deny</li> <li>• Deny &amp; Log</li> <li>• Permit</li> <li>• Permit &amp; Log</li> </ul>

(Continued)

Name	Description
Rule Protocols	<ul style="list-style-type: none"> <li>• TCP</li> <li>• UDP</li> <li>• IP</li> <li>• ICMP</li> </ul>
Source Port	For TCP and IP protocols
Destination Port	For TCP and IP protocols

From the blueprint, navigate to **Staged > Policies > Security Policies > Policies** to go to security policies. You can create, clone, edit and delete security policies.

The screenshot shows the network management interface. The top navigation bar includes 'Dashboard', 'Analytics', 'Staged', 'Uncommitted', 'Active', and 'Time Voyager'. The left sidebar has 'Physical', 'Virtual', 'Policies', 'Catalog', 'Tasks', and 'Connectivity Templates'. The 'Policies' section is expanded, showing 'Security Policies', 'Interface Policies', 'Routing Policies', 'Fabric Addressing Policy', 'Virtual Network Policy', 'Anti-Affinity Policy', and 'SVI IP Validation Policy'. The 'Security Policies' section is selected, showing a 'Create Security Policy' button and a table of existing policies.

Name	Source Application Point	Destination Application Point	Rule Count	Rule Conflicts	Tags	Enabled	Errors	Actions
Example	Virtual Network vnet_10_on_rack_2_001_leaf1	Routing Zone Default routing zone	2			ON		Clone, Edit, Delete

### Create Security Policy

Before creating security policies, create "routing zones" on page 496, "virtual networks" on page 488, "endpoints and endpoint groups" on page 578, in that order. They are the basis for creating security policies.

To create security policies:

1. From the blueprint, navigate to **Staged > Policies > Security Policies > Policies** and click **Create Security Policy**.
2. Enter a name, and if you want the policy to be enabled leave the default. Otherwise, click the **Enabled** toggle to disable it.
3. Select a source point type, and enter the source point.
4. Select a destination point type, and enter the destination point.
5. Click **Add Rule**, then enter a name and (optional) description.
6. Select an action from the drop-down list (Deny, Deny & Log, Permit, Permit & Log).
7. Select a protocol from the drop-down list (TCP, UDP, IP ICMP).
8. If you selected TCP or UDP, enter a port (or port range) for source and destination. (If you created ["TCP/UDP port aliases"](#) on page 128, they appear in the drop-down list).
9. To add another rule, click **Add Rule** and configure as above.

**NOTE:** To the right of the **Add Rule** button you can automatically create a blocklist-type policy by clicking **Deny All** or an allowlist-type policy by clicking **Permit All**.

10. You can adjust the rule order by clicking the **Move up** or **Move Down** buttons in each rule.
11. Click **Create** to stage the policy and return to the list view.

### Policy Errors

1. Check the security policy in the list view for errors, which are highlighted in red.

[+ Create Security Policy](#)

Query: All

1-1 of 1    < >    Page Size: 25

Name ^	Source Application Point	Destination Application Point	Rule Count	Rule Conflicts	Tags	Enabled	Actions
External to Compute	External Endpoint Group External	Internal Endpoint Group Databases	2			<div style="width: 20px; height: 10px; background-color: #0070c0; border: 1px solid #0070c0; position: relative;"> <div style="position: absolute; right: 2px; top: 2px; width: 10px; height: 10px; background-color: white; border: 1px solid #0070c0;"></div> </div>	<div style="border: 2px solid red; padding: 2px; display: inline-block; text-align: center;"> </div> <div style="margin-top: 5px; font-size: 0.8em;"> <a href="#" style="background-color: white; border: 1px solid #ccc; padding: 2px 5px;">Show errors</a> </div> <div style="display: flex; gap: 5px;"> <div style="border: 1px solid #ccc; padding: 2px;">📄</div> <div style="border: 1px solid #ccc; padding: 2px;">✎</div> <div style="border: 1px solid #ccc; padding: 2px;">🗑️</div> </div>

2. To see details, click the **Show errors** button.

#### Policy Errors

- Policy 'External to Compute' destination application point is resolved to empty object set
- Policy 'External to Compute' destination application point does not have ip connectivity

3. When you resolve errors, the policy is no longer highlighted red and the **Errors** field is blank.

[+ Create Security Policy](#)

▶ Query: All

1-1 of 1
 

< >

 Page Size: 25

Name ▲	Source Application Point	Destination Application Point	Rule Count	Rule Conflicts	Tags	Enabled	Errors	Actions
External to Compute	External Endpoint Group External	Virtual Network red_125_leaf3_v4	2			<input checked="" type="checkbox"/>		<div style="display: flex; justify-content: space-around; width: 100px;"> <span>📄</span> <span>✎</span> <span>🗑</span> </div>

To activate staged changes, "[commit](#)" on [page 648](#) them to the blueprint.

### Edit Security Policy

1. From the left navigation menu, navigate to **Staged > Policies > Security Policies > Policies** and click the **Edit** button for the policy to edit.
2. Make your changes.
3. Click **Edit** to stage the changes and return to the list view.

### Delete Security Policy

1. From the left navigation menu, navigate to **Staged > Policies > Security Policies > Policies** and click the **Delete** button for the policy to delete.
2. Click **Delete** to stage the deletion and return to the list view.

### Security Policy Search

You can find security policies that are applied to specific subnets or points.

1. From the blueprint, navigate to **Staged > Policies > Security Policies > Policy Search**.
2. Select a source point type and enter a subnet or source point, as applicable.
3. Select a destination point type and enter a subnet or source point, as applicable.

4. Click **Search** to display associated security policies.

### External Endpoint Preview

Name	External Clients
IPv4 Subnet	69.16.128.0/18
Tags	clients
Enforcement Points	<div>Enforcement Point Ethernet1/1.3</div> <div>Enforcement Point Ethernet1/1.2</div> <div>Enforcement Point Ethernet1/1.2</div> <div>Enforcement Point Ethernet1/1</div> <div>Enforcement Point Ethernet1/1</div> <div>Enforcement Point Ethernet1/1.3</div>

### Security Policy Conflicts

From the blueprint, navigate to **Staged > Policies > Security Policies > Conflicts** to see any conflicts that have been detected (**Rule Conflicts** column). Conflicts are resolved automatically whenever possible. By default, more specific policies are applied before less specific ones, but you can change these security policy settings. To see conflict details, click the icon in the **Rule Conflicts** column.

[+ Create Security Policy](#)

Query: All

1-2 of 2    < >    Page Size: 25

Name ▲	Source Application Point	Destination Application Point	Rule Count	Rule Conflicts	Tags	Enabled	Errors	Actions
<a href="#">External to Compute</a>	External Endpoint Group External	Virtual Network red_125_leaf3_v4	2			<input checked="" type="checkbox"/>		
<a href="#">Permit External Clients</a>	External Endpoint External Clients	Virtual Network red_125_leaf3_v4	1			<input checked="" type="checkbox"/>		

If the conflict was resolved automatically, **Resolved by AOS** appears in the **Status** column.

Query: All

1-1 of 1    < >    Page Size: 25

Status ▲	Policy / Rule #1	Policy / Rule #2
Resolved by AOS	<div>External to Compute / Permit HTTPS</div> <div> <div>External Endpoint Group External any</div> <div>Virtual Network red_125_leaf3_v4 443</div> </div>	<div>Permit External Clients / Deny</div> <div> <div>External Endpoint External Clients any</div> <div>Virtual Network red_125_leaf3_v4 any</div> </div>

## Security Policy Settings

You can configure how you want to resolve conflicts and whether to permit or deny traffic.

1. From the blueprint, navigate to **Staged > Policies > Security Policies > Settings**.

2. Select options as appropriate.

- Conflict resolution
  - **More specific first** - more specific IP policy is used (default)
  - **More generic first** - less specific IP policy is used
  - **Disabled** - disables conflict resolution
- Default action
  - **Permit** - permits traffic (default)
  - **Permit & Log** - permits traffic and logs it
  - **Deny** - denies traffic
  - **Deny & Log** - denies traffic and logs it

3. Click **Save Changes** to stage the changes.

To activate staged changes, ["commit" on page 648](#) them to the blueprint.

## Interface Policies

### IN THIS SECTION

- [802.1X Server Port Authentication | 592](#)
- [Common Scenarios | 594](#)
- [802.1X Interface Policy Workflow | 595](#)
- [Create Virtual Networks for Interfaces | 595](#)
- [Create AAA Server for Interface Policy | 595](#)
- [Create 802.1x Interface Policy | 596](#)
- [Assign Ports and Fallback VNs to Interface Policy | 597](#)

## 802.1X Server Port Authentication

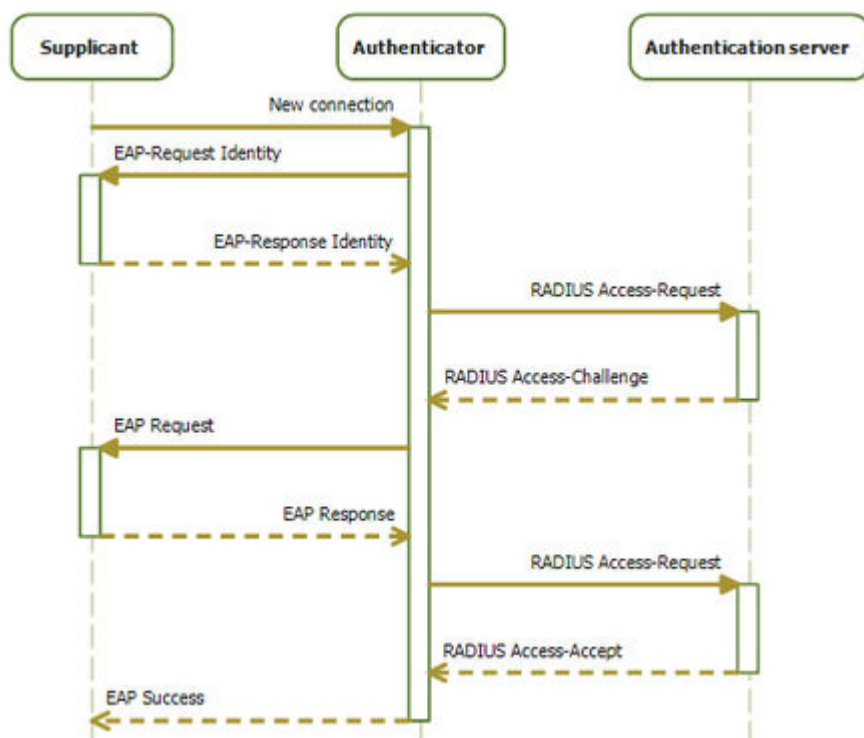
**IEEE 802.1X** is an IEEE Standard for network port-based Network Access Control. It is part of the IEEE 802.1 group of networking protocols. It provides an authentication mechanism to devices wishing to attach to a LAN.

IEEE 802.1X defines the encapsulation of the Extensible Authentication Protocol (EAP) over IEEE 802, which is known as "EAP over LAN" or EAPOL.

802.1X authentication involves three parties: a supplicant, an authenticator, and an authentication server. The **supplicant** is a client device (such as a server) that wishes to attach to the LAN. The term 'supplicant' is also used interchangeably to refer to the software running on the client that provides credentials to the authenticator. The **authenticator** is a network device which provides a data link between the client and the network and can allow or block network traffic between the two, such as an Ethernet switch or wireless access point; and the **authentication server** is typically a trusted server that can receive and respond to requests for network access, and can tell the authenticator if the connection is to be allowed, and various settings that should apply to that client's connection or setting. Authentication servers typically run software supporting the RADIUS and EAP protocols. In some cases, the authentication server software may be running on the authenticator hardware.

The authenticator acts as a security guard to a protected network. The supplicant (i.e., client device) is not allowed access through the authenticator to the protected side of the network until the supplicant's identity has been validated and authorized. With 802.1X port-based authentication, the supplicant must initially provide the required credentials to the authenticator - these will have been specified in advance by the network administrator and could include a user name/password or a permitted digital certificate. The authenticator forwards these credentials to the authentication server to decide whether access is to be granted. If the authentication server determines the credentials are valid, it informs the authenticator, which in turn allows the supplicant (client device) to access resources located on the protected side of the network.

Extensions to 802.1X can also allow the authentication server to pass port-configuration options to the authenticator. An example is using RADIUS value-pair attributes to pass a VLAN ID, allowing the supplicant access to one of several VLANs.



(Source : Wikipedia, revised by Apstra)

You can manage 802.1X configuration on network devices with 802.1X server port authentication, a collection of interface policy settings.

802.1X interface policy is supported on Junos and Arista EOS physical network devices only.

This policy setting enables the network to require L2 servers in a blueprint to authenticate to a RADIUS server before being provided access to the network.

The network operator may require clients to authenticate using EAP-TLS, Certificates, simple username & password, or MAC Authentication bypass.

**NOTE:** Support for encryption protocols, certificates, EAP, is negotiated between RADIUS supplicant and RADIUS server, and is not controlled by the switch.

After authentication occurs, a RADIUS server may optionally set a VLAN ID attribute at authentication time to move the supplicant into a defined VLAN, known by a leaf-specific VLAN ID.

This section describes the necessary tasks to create Interface Policies to be used with 802.1X server port authentication and dynamic VLAN allocation.

## Common Scenarios

The following are some common scenarios for 802.1X port authentication.

### **Device supports 802.1X, credentials and VLAN are configured in Radius**

1. Device (Supplicant) connects to a port
2. Switch (Authenticator) mediates EAP negotiation between supplicant and Radius (Authentication Server)
3. Upon authentication, Radius sends an Access-Accept message to the switch which includes the VLAN number for the device
4. The switch adds the device port to the specified VLAN

### **Device supports 802.1X, but credentials are not configured in Radius**

1. Device (Supplicant) connects to a port
2. Switch (Authenticator) mediates EAP negotiation between supplicant and Radius (Authentication Server)
3. Finding no credential for the supplicant, Radius sends an Access-Reject message to the switch
4. The switch adds the device port to a designated Fallback (aka AuthFail/Parking) VLAN

### **Device does not support 802.1X, but the device MAC address is configured in Radius**

1. Device (Non-Supplicant) connects to a port
2. Switch (Authenticator) does not receive a reply to its EAP-Request Identity message, indicating no 802.1X support
3. Switch authenticates device's MAC address to Radius (Authentication Server)
4. Radius sends an Access-Accept message to the switch which includes the VLAN number for the device
5. The switch adds the device port to the specified VLAN

### **Device does not support 802.1X, and device MAC address is not configured in Radius**

1. Device (Non-Supplicant) connects to a port

2. Switch (Authenticator) does not receive a reply to its EAP-Request Identity message, indicating no 802.1X support
3. Switch authenticates device's MAC address to Radius (Authentication Server)
4. Radius does not find a record for the MAC address
5. Radius sends an Access-Reject or Access-Accept message to the switch without a VLAN
6. The switch adds the device port to a designated Fallback (aka AuthFail/Parking) VLAN

### 802.1X Interface Policy Workflow

1. Create virtual networks (e.g. Data VLAN, Fallback VLAN, Dynamic VLAN)
2. Create AAA servers
3. Create 802.1X interface policy
4. Assign ports and fallback VLANs

### Create Virtual Networks for Interfaces

Create virtual networks for the interface policy per the table below. We suggest creating these virtual networks with a consistent VLAN ID among all leafs (instead of using a resource pool). For more information about creating VLANs, see ["Virtual Networks" on page 488](#).

Data VLAN (assigned to ports)	Interfaces will have 802.1X configuration if at least one VLAN is assigned to the port. If a port does not have any VLANs assigned, 802.1X configuration will not be rendered on the interface. The interface will be configured as a routed port.
Dynamic VLAN (optional, assigned to leafs, not ports)	The RADIUS server itself optionally chooses the VLAN ID dynamically when the user (supplicant) is authenticated and authorized. Apstra software does not have control over Dynamic VLAN assignment. This decision is made by RADIUS configuration, not by the switch configuration.
Fallback VLAN (optional, assigned to leafs, not ports)	<p>Fallback VLAN can be assigned to the user (supplicant) in case of authentication failure. For fallback, the VLAN is controlled by the switch configuration.</p> <p>A RADIUS dynamic VLAN or fallback VLAN must exist on the switch, but there is no requirement to bind any endpoints to that VLAN. It only needs to exist on the switch.</p>

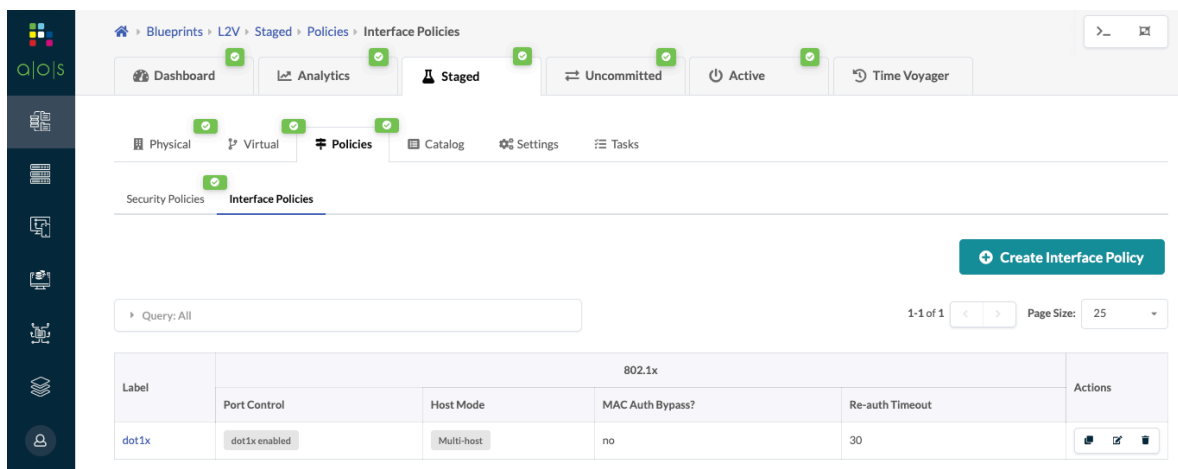
### Create AAA Server for Interface Policy

Create the AAA server. For more information, see ["AAA Servers \(Blueprint\)" on page 618](#).

## Create 802.1x Interface Policy

You must create the policy before you can assign interfaces or fallback VLANs to it.

1. From the blueprint, navigate to **Staged > Policies > Interface Policies** and click **Create Interface Policy**.



2. Enter a name and select **802.1x** from the drop-down list.
3. Select the **Port Control**.
  - **dot1x enabled** - Requires ports to authenticate EAPOL before being given access to the network.
  - **Deny access** - Completely blocks the port; no network access is permitted. No other parameters are needed. Example: as a quarantine configuration to quickly deactivate ports that may be infected.
4. Select the **Host Mode**.
  - **Multi-host\*\*** (default) - Allows all MAC addresses on the port to authenticate after the first successful authorization. After the first host deauthorizes, all MACs on the port are de-authenticated.
  - **Single-host** - Permits a single host to authenticate; all other MACs are not permitted.
5. If you want to enable **MAC Auth Bypass** on Arista EOS, check the **Enabled?** box. Enabling MAC auth bypass allows a switch to send the MAC address to the RADIUS server if the port does not authenticate within the authentication timeout period. MAC Auth bypass (MAB) requests are only sent if the client does not respond to RADIUS requests, or if the client fails authentication.

**NOTE:** MAC Auth bypass must be configured along with 802.1X port control.



**CAUTION:** MAC auth bypass failure behavior may be different between switch vendors and major switch models.

6. Enter **Re-auth Timeout** (optional) to configure a time period (seconds). Re-authentication timeout causes the switch to request any clients to re-authenticate to the network after the timeout expires. This also re-triggers MAC Auth bypass.

If re-authentication timeout is not configured, then no related configuration is rendered on the switch. This means the switchport will be whatever the OS vendor default is. If a value is configured, 802.1X re-authentication will be enabled on the port, and a time value will be configured.

7. Click **Create** to create the interface policy and return to the list view.

### Assign Ports and Fallback VNs to Interface Policy

This steps adds interfaces or dynamic VLANs to the interface policy.

1. From the blueprint, navigate to **Staged > Policies > Interface Policies** and scroll down to the Assign To section.
2. **Assign ports and interfaces:** Click leaf names to expand interfaces, then click ports and interfaces to assign them. Note that you cannot assign ports that are assigned to conflicting policies.
3. **Assign fallback VN:** Assigning the fallback virtual network is leaf-specific. To re-use the fallback on multiple leafs, you have to assign it to each leaf. Any VN that is assigned to the leaf may be used as a fallback virtual network - there are no restrictions.

## Assigned To

Query: All 1-5 of 6 Page Size: 5

Name	Hostname	S/N
l2_virtual_mlag_001_leaf1	l2-virtual-mlag-001-leaf1	52540057E344
<p>Ports: <input checked="" type="checkbox"/> Assigned to the current policy <input type="checkbox"/> Assigned to another policy <input type="checkbox"/> Unavailable for assignment</p> <p><input type="checkbox"/> Not assigned to any policy <input type="checkbox"/> Partial</p> <p>Select port and choose interfaces you want to assign to the current policy <span>Assign All Unassign</span></p> <p>1 2 3 4 <b>5</b> 6 7</p> <p>Port #5 Tr. #1 (10G, default) <span>swp5</span></p> <p>Fallback VN <span>fallback_99</span> <span>blue-2</span> <span>fallback_99</span> <span>red-1</span> <span>Save Changes</span></p>		
l2_virtual_mlag_001_leaf2	l2-virtual-mlag-001-leaf2	5254005B9A65

4. After the policy is configured, the settings are now visible, including interfaces those settings apply to.

**NOTE:** AAA, Dot1x, and Dot1x interface configurations are now pushed to the leafs. The following is a part of sample config rendered for Arista EOS switch.

```
leaf1#sh running-config section dot1x
logging level DOT1X errors
!
aaa group server radius AOS_RADIUS_DOT1X
    server 172.20.191.5 vrf management
!
aaa authentication dot1x default group AOS_RADIUS_DOT1X
aaa accounting dot1x default start-stop group AOS_RADIUS_DOT1X logging
!
interface Ethernet5
    switchport trunk allowed vlan 99
    switchport mode trunk
    switchport
    ipv6 enable
    ipv6 address auto-config
    ipv6 nd ra rx accept default-route
```

```
dot1x pae authenticator
dot1x reauthentication
dot1x port-control auto
dot1x timeout reauth-period 30
!
..snip..
!
dot1x system-auth-control
dot1x dynamic-authorization
```

Routing Policies


IN THIS SECTION


- [Routing Policy Overview | 599](#)
- [Create Routing Policy | 602](#)
- [Edit Routing Policy | 602](#)
- [Delete Routing Policy | 602](#)

Routing Policy Overview

Routing policies include the following details:

Parameter	Description
Name	18 characters or fewer. Alphanumeric, _ and - only.
Import Policy	<ul style="list-style-type: none"><li>● <b>Default</b> - The default BGP route (0.0.0.0/0, ::/0) is permitted. If extra import routes are defined, they are also permitted.</li><li>● <b>All</b> - Any BGP route is permitted.</li><li>● <b>Extra Only</b> - Only user-defined extra import routes are permitted (or denied).</li></ul>
Extra Import Routes (user-defined)	<ul style="list-style-type: none"><li>● <b>Prefix</b> - IPv4 or IPv6 network address (format: network/prefixlen) or IP address (interpreted as /32 network address).</li></ul>

Parameter	Description
	<ul style="list-style-type: none"> <li>• <b>GE Mask and LE Mask</b> - GE Mask matches less-specific prefixes from a parent prefix, up from the GE mask to the prefix length of the route. (IPv4 range: 0-32. IPv6 range: 0-128). If you don't specify GE mask, then the prefix-list entry should be an exact match. You can use this option in combination with LE Mask. GE mask must be longer than the subnet prefix length. If both the LE mask and GE mask are specified, then the LE mask must be greater than the GE mask.</li> <li>• <b>Action</b> - Permit or Deny</li> </ul>
Export Policy	<ul style="list-style-type: none"> <li>• <b>Spine Leaf Links</b> - Exports all spine-leaf (fabric) links within a VRF. EVPN routing zones do not have spine-leaf addressing, so this generated list may be empty. For routing zones of type Virtual L3 Fabric, subinterfaces between spine-leaf are included.</li> <li>• <b>L3 Edge Server Links</b> - Exports all leaf to L3 server links within a routing zone (VRF). On layer 2 blueprints this is an empty list.</li> <li>• <b>L2 Edge Subnets</b> - Exports all virtual networks (VLANs) that have L3 addresses within a routing zone (VRF).</li> <li>• <b>Loopbacks</b> - Exports all loopbacks within a routing zone (VRF) across spine, leaf, and L3 servers.</li> <li>• <b>Static Routes</b> - Exports all subnets in a VRF associated with static routes from all fabric systems to generic systems associated with this routing policy.</li> </ul>
Extra Export Routes (user-defined)	<p>User-defined export routes. These policies are additive. To advertise extra routes only, unselect all export policies.</p> <div>  <p><b>NOTE:</b> To enable default route for EVPN host routes, go to <b>Staged &gt; Settings &gt; Virtual Network Policy</b> and enable the <b>Generate EVPN host routes</b> option.</p> </div> <ul style="list-style-type: none"> <li>• <b>Prefix</b> - IPv4 or IPv6 network address (format: network/prefixlen) or IP address (interpreted as /32 network address).</li> <li>• <b>GE Mask and LE Mask</b> - GE Mask matches less-specific prefixes from a parent prefix, up from the GE mask to the prefix length of the route. (IPv4 range: 0-32. IPv6 range: 0-128). If you don't specify GE mask, then the prefix-list entry should be an exact match. You can use this option in combination with LE Mask. GE mask must be longer than the subnet prefix length. If both the LE mask and GE mask are specified, then the LE mask must be greater than the GE mask.</li> </ul>

Parameter	Description
	<ul style="list-style-type: none"> <li>• <b>Action</b> - Permit or Deny</li> </ul>
Aggregate Prefixes	<p>If you have routing zones associated with your routing policy, and aggregate prefixes are supported on the platform (see the <a href="#">"Feature Matrix" on page 853</a>) you can specify aggregate prefixes. These are the BGP aggregate routes to be imported into the routing zone (VRF) on all border switches. The aggregated routes are sent to all generic system peers in a routing zone (VRF).</p> <div>  <p><b>CAUTION:</b> You apply routing policies with aggregate prefixes to the entire routing zone. You cannot configure them individually for BGP sessions (per connectivity point). If you do attempt to apply them via a connectivity template (CT), you could receive the error "Protocol endpoint routing policy aggregate prefixes should be empty".</p> </div>
Associated Routing Zones	Lists any routing zones that are associated with the routing policy.
Associated Protocol Endpoints	Lists any protocol endpoints that are associated with the routing policy.

From the blueprint, navigate to **Staged > Policies > Routing Policies** to go to routing policies in the blueprint. A default routing policy is associated with the default routing zone. You cannot change the default routing policy, but you can create, clone, edit, and delete other routing policies as described

below.

1. Click **Staged** in the top navigation bar.

2. Click **Policies** in the left sidebar.

3. Click **Routing Policies** in the sub-menu.

**Create Routing Policy**

Query: All 1-1 of 1 Page Size: 25

Name	Type	Description	Import Policy	Spine Leaf Links	L2 Edge Subnets	Loopbacks	Static routes	L3 Edge Server Links	Expect Default IPv4 Route	Expect Default IPv6 Route	Actions
Default Immutable	default Immutable	Associated with routing zones by default, cannot be updated or deleted.	All	no	yes	yes	no	yes	yes	yes	

Click routing policy name for details

### Create Routing Policy

1. From the blueprint, navigate to **Staged > Policies > Routing Policies** and click **Create Routing Policy**.
2. Configure the policy. For parameter details, see the Routing Policy Overview.
3. Click **Create** to stage the policy addition and return to the list view.

### Edit Routing Policy

1. From the blueprint, navigate to **Staged > Policies > Routing Policies** and click the **Edit** button for the policy to edit.
2. Make your changes.
3. Click **Update** (bottom-right) to stage the policy change and return to the list view.

### Delete Routing Policy

1. From the blueprint, navigate to **Staged > Policies > Routing Policies** and click the **Delete** button for the policy to delete.
2. Click **Delete** to stage the policy removal and return to the list view.

## Fabric Addressing Policy

### IN THIS SECTION

- [Enable IPv6 Applications | 603](#)
- [ESI MAC MSB | 603](#)

### Enable IPv6 Applications



**CAUTION:** After IPv6 has been enabled in a blueprint, it cannot be disabled. Although, you could use Time Voyager to rollback to a revision before IPv6 was enabled.

Enabling support for IPv6 virtual networks on EVPN L2 deployments or L3 deployments adds resource requirements and device configurations. This includes IPv6 loopback addresses on leafs and spines, IPv6 addresses for MLAG SVI subnets and IPv6 addresses for leaf L3 peer links. The following caveats apply:

- This feature does not include IPv6 support in the fabric.
  - IPv6 support is not available on non-EVPN L2 networks.
  - IPv6 support is not available on 5-stage Clos networks.
  - When IPv6 is enabled on EVPN L2 deployments, security policy functionality is not available.
1. From the blueprint, navigate to **Staged > Policies > Fabric Addressing Policy** and click **Modify Settings**.
  2. Click the toggle on to enable IPv6 applications.
  3. Click **Save Changes**.

["Assign the required IPv6 IP addresses" on page 420](#). For information about IPv6 configuration for Virtual Networks, see ["Virtual Networks" on page 488](#) documentation.

### ESI MAC MSB



**CAUTION:** Updating the Most Significant Byte (MSB) value regenerates all existing ESI MACs in the blueprint. We recommend that you leave the default value as is.

ESI MAC addresses are internally auto-generated using Most Significant Byte (MSB) values (default: 2). To ensure that multicast MACs are not generated, it must be an even number (up to 254). Config for the

ESI value is rendered as 10 octets. The first octet is zero (0), the second octet is the MSB value. Six (6) octets after the first octet are significant and used as the LACP system-id. The example below is of a rendered ESI value and its respective LACP system id:

```
set interfaces ae1 esi 00:02:00:00:00:00:01:00:00:01
set interfaces ae1 esi all-active
set interfaces ae1 aggregated-ether-options lacp active
set interfaces ae1 aggregated-ether-options lacp system-id 02:00:00:00:00:01
```

1. To change the ESI MAC MSB value from the blueprint, navigate to **Staged > Policies > Fabric Addressing Policy** and click **Modify Settings**.
2. Make your changes, then click **Save Changes**.

### Virtual Network Policy

IN THIS SECTION

●

[Virtual Network Policy Overview | 604](#)

●

[Modify Virtual Network Policy | 606](#)

### Virtual Network Policy Overview

Virtual network policies include the following details:

Policy Details	Description
IP Links to Generic Systems MTU	Specifies the MTU for all L3 IP links facing generic system. A null or empty (default) value implies that any MTU will not be explicitly rendered; the device default MTU is used. Custom larger MTU may be required to provide EVPN DCI functionality or to support fabric wide Jumbo frame functionality. For EVPN-DCI, we recommend an MTU of 9050.
Max External Routes Count	Maximum number of routes to accept from external routers. The default (None) does not render any maximum-route commands on BGP sessions, implying that vendor defaults are used. An integer between range 1 to 2**32-1 sets a maximum limit of routes in BGP config. The value 0 (zero) intends the device to never apply a limit to number of EVPN routes (effectively unlimited). We suggest that this value is effectively unlimited on EVPN blueprints, to permit the high number of /32 and /128 routes to be advertised and

Policy Details	Description
	received between VRFs in the event an external router is providing a form of route leaking functionality.
Max MLAG Routes Count	Maximum number of routes to accept across MLAG peer switches. The default (None) does not render any maximum-route commands on BGP sessions, implying that vendor defaults are used. An integer between range 1 to $2^{32}-1$ sets a maximum limit of routes in BGP config. The value 0 (zero) intends the device to never apply a limit to number down BGP sessions if maximums are exceeded on a session. For EVPN blueprints, this should be combined with max_evpn_routes to permit routes across the L3 peer link which may contain many /32 and /128 from EVPN type-2 routes that convert into BGP route advertisements.
Max EVPN Routes Count	Maximum number of EVPN routes to accept on an EVPN switch. The default (None) does not render any maximum-route commands on BGP sessions, implying that vendor defaults are used. An integer between range 1 to $2^{32}-1$ sets a maximum limit of routes in BGP config. The value 0 (zero) intends the device to never apply a limit to number of EVPN routes (effectively unlimited). Note: Device vendors typically shut down BGP sessions if maximums are exceeded on a session.
Max Fabric Routes Count	Maximum number of routes to accept between spine and leaf in the fabric, and spine-superspine. This includes the default VRF. You may need to set this option in the event of leaking EVPN routes from a routing zone into the default routing zone (VRF) which could generate a large number of /32 and /128 routes. We suggest that this value is effectively unlimited on all blueprints to ensure the network stability of spine-leaf BGP sessions and EVPN underlay. We also suggest unlimited for non-EVPN blueprints considering the impact to traffic if spine-leaf sessions go offline. An integer between $1-2^{32}-1$ will set a maximum limit of routes in BGP config. The value 0 (zero) intends the device to never apply a limit to number of fabric routes (effectively unlimited).
EVPN Type 5 Routes	Default disabled. When enabled all EVPN vteps in the fabric redistribute ARP/IPV6 ND (when possible on NOS type) as EVPN type 5 /32 routes in the routing table. Currently, this option is certified for Juniper Junos only. FRR (SONiC/Cumulus) does this implicitly and cannot be disabled. This setting results in a blueprint warning that it is not supported. This value is disabled by default, as it generates a very large number of routes in the BGP routing table and takes large amounts of TCAM allocation space. When these /32 and /128 routes are generated, it assists in direct unicast routing to host destinations on VNIs that are not stretched to the ingress vtep, and avoids a route lookup to a subnet (such as /24) that may be hosted on many leafs. The directed host route prevents a double lookup to one of many vteps may hosts the /24 and instead routes the destination directly to the correct vtep.

Policy Details	Description
Generate EVPN host routes from ARP/IPV6 ND ARP	Setting " <b>Generate EVPN host routes from ARP/IPV6 ND ARP</b> " adds a policy-statement to the export policy used within the fabric.

## Modify Virtual Network Policy

1. From the blueprint, navigate to **Staged > Policies > Virtual Network Policy** and click **Modify Settings** (right side).

Policy	Value
IP Links to Generic Systems MTU <sup>®</sup>	Default
Max External Routes Count <sup>®</sup>	Unlimited
Max MLAG Routes Count <sup>®</sup>	Unlimited
Max EVPN Routes Count <sup>®</sup>	Unlimited
Max Fabric Routes Count <sup>®</sup>	Unlimited
Generate EVPN host routes from ARP/IPV6 ND ARP <sup>®</sup>	Disabled

2. Make your changes.
3. Click **Save Changes**.

## Anti-Affinity Policy

### IN THIS SECTION

- [Anti-Affinity Policy Overview | 607](#)
- [Enable/Disable Anti-Affinity Policy | 608](#)

## Anti-Affinity Policy Overview

When designing high availability (HA) systems, you want parallel links between two devices to terminate on different physical ports, thus avoiding transceiver failures from impacting both links on a device. Depending on the number of interfaces on a system, manually modifying these links could be time-consuming. With the anti-affinity policy (new in Apstra version 4.0.1) you can apply certain constraints to the cabling map to control automatic port assignments. When you enable the policy, you can specify the maximum number of links as follows:

- **Max Links Count per Slot** - maximum total number of links connected to ports/interfaces of the specified slot regardless of the system they are targeted to. It controls how many links can be connected to one slot of one system. Example: A line card slot in a chassis.
- **Max Links Count per System per Slot** - restricts the number of links to a certain system connected to the ports/interfaces in a specific slot. It controls how many links can be connected to one system to one slot of another system.
- **Max Links Count per Port** - maximum total number of links connected to the interfaces of the specific port regardless of the system they are targeted to. It controls how many links can be connected to one port in one system. Example: Several transformations of one port. In this case, it controls how many transformations can be used in links.
- **Max Link Count per System per Port** - restricts the number of interfaces on a port used to connect to a certain system. It controls how many links can be connected from one system to one port of another system. This is the one that you will most likely use, for port breakouts.

The anti-affinity policy has three modes:

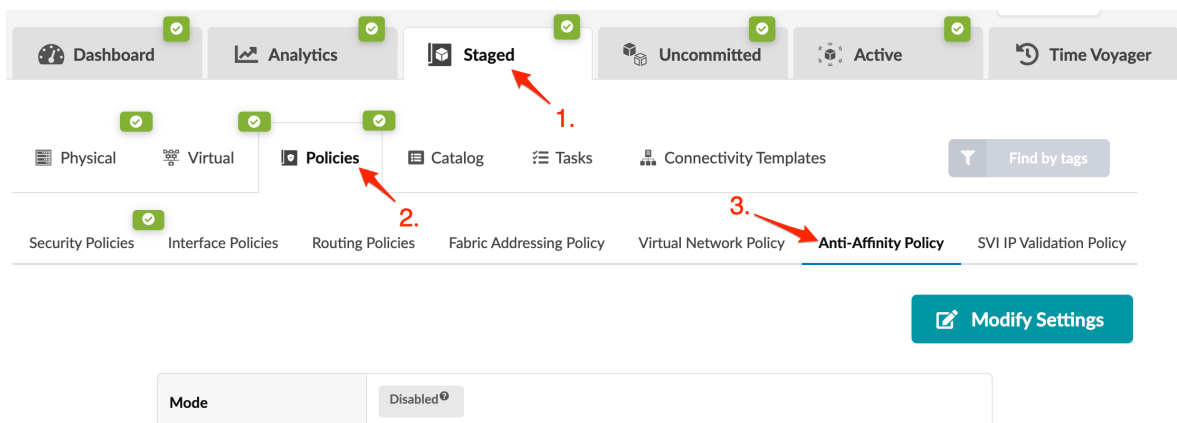
- **Disabled** (default) - ports selection is based on assigned interface maps and interface names (provided or auto-assigned). Port breakouts could terminate on the same physical ports.
- **Enabled (loose)** - controls interface names that were not defined by the user. Does not control or override user-defined cabling. (If you haven't explicitly assigned any interface names, loose and strict are effectively the same policy.)
- **Enabled (strict)** - completely controls port distribution and could override user-defined assignments. When you enable the strict policy, a statement appears at the top of the cabling map (Staged/Active > Physical > Links and Staged/Active > Physical > Topology Selection) stating that the anti-affinity policy is enabled ("forced" for strict).

An example of when you'd want to apply the anti-affinity policy is when you have a QSFP 40G breakout port that you want to break out into 4-10G ports. You can ensure that any links that go to the same device use different QSFP ports instead of 2-10G spine links on the same QSFP port. This gives you an added layer of redundancy if that QSFP port fails.

## Enable/Disable Anti-Affinity Policy

Every time you change the policy, port assignments are recalculated.

1. From the blueprint, navigate to **Staged > Policies > Anti-Affinity Policy** and click **Modify Settings**.



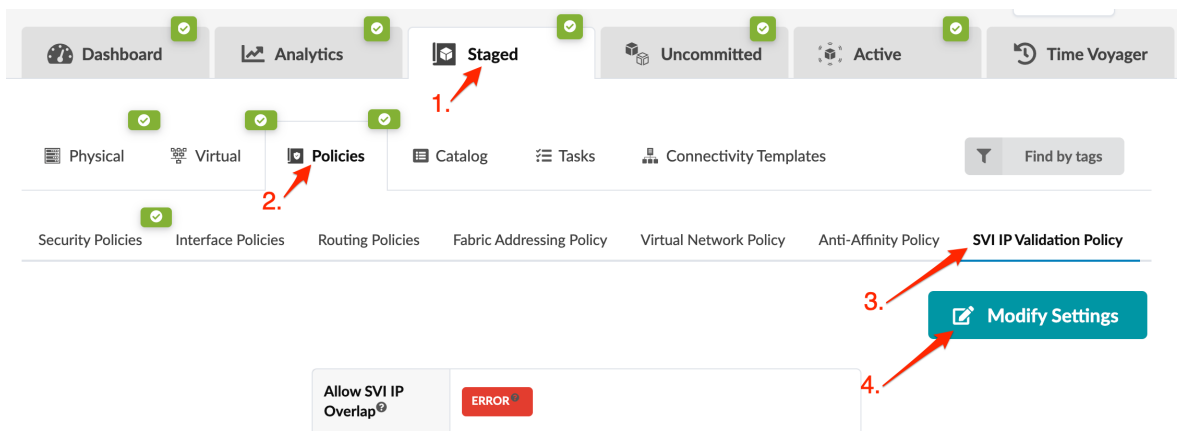
2. Change the policy mode, and if you're enabling the policy, enter a maximum number of links, as applicable.
3. Click **Save Changes** to stage the change and return to the policies view.

To activate staged changes, commit them from the **Uncommitted** tab.

## Modify SVI IP Validation Policy

If you want to use the same IP address for SVIs in the same virtual network, you can change the policy setting in the blueprint to allow duplicates, with or without a warning (as of Apstra version 4.0.1).

1. From the blueprint, navigate to **Staged > Policies > SVI IP Validation Policy** and click **Modify Settings**.



2. Change the setting as applicable:
  - **No Warning** - allows duplicate SVI IP addresses without generating a warning or error.
  - **Warning** - duplicate SVI IP addresses are treated as warnings; you can commit changes.

- **Error** (default) - duplicate SVI IP addresses are treated as errors that must be resolved before you can commit changes.

3. Click **Save Changes** to stage the changes and return to the **SVI IP Validation Policy** page.

To activate staged changes, commit them from the **Uncommitted** tab.

## Catalog

### IN THIS SECTION

- [Logical Devices \(Blueprint Catalog\) | 609](#)
- [Interface Maps \(Blueprint Catalog\) | 610](#)
- [Property Sets \(Blueprint Catalog\) | 612](#)
- [Configlets \(Blueprint Catalog\) | 614](#)
- [AAA Servers \(Blueprint Catalog\) | 618](#)
- [Tags \(Blueprint Catalog\) | 622](#)

## Logical Devices (Blueprint Catalog)

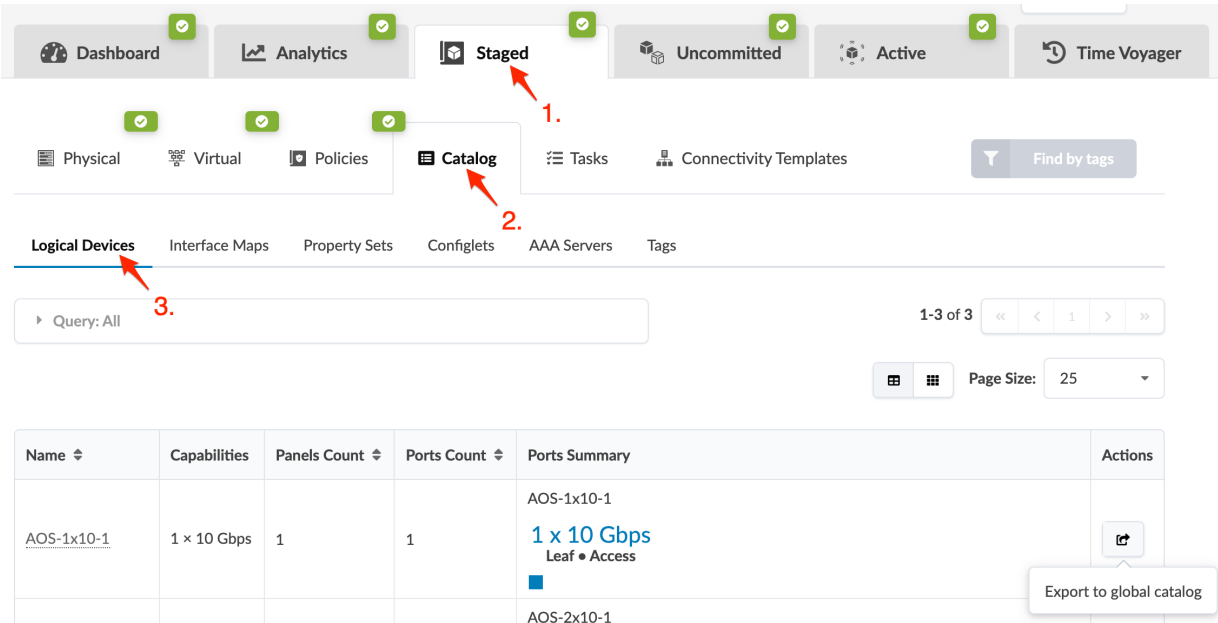
### IN THIS SECTION

- [Logical Devices Overview \(Blueprint Catalog\) | 609](#)
- [Export Logical Device | 610](#)

## Logical Devices Overview (Blueprint Catalog)

The template that was used to create a blueprint determines which logical devices are in that blueprint's catalog. From the blueprint, navigate to **Staged > Catalog > Logical Devices** to go to the logical devices

catalog. You can export logical devices from the blueprint catalog.



## Export Logical Device

1. From the blueprint, navigate to **Staged > Catalog > Logical Devices** and click the **Export to global catalog** button for the logical device to export (in the Actions column on the right side).
2. Select how you want to export the logical device:
  - **Export as new** - to create a new logical device based on the current one in the global catalog. This option doesn't keep references to interface maps. Even if you already have a logical device with the same name in the global catalog you can still export it. Exported logical devices with the same name are identified by the ID instead of by the logical device name.
  - **Export existing** - to create interface maps for this logical device in the global catalog that you can re-import into the blueprint. If you already have a logical device with the same name in the global catalog, you can't use this option. When you export a logical device with this option, the logical device ID and logical device name are the same.
3. Click **Export** to export the logical device and return to the list view.

## Interface Maps (Blueprint Catalog)

### IN THIS SECTION

- [Interface Maps Overview \(Blueprint\) | 611](#)
- [Import Interface Map | 611](#)

● Delete Interface Map (Blueprint) | 611

## Interface Maps Overview (Blueprint)

From the blueprint, navigate to **Staged > Catalog > Interface Maps** to go to the interface maps catalog. You can import and delete interface maps from the blueprint catalog.

1. Catalog

2. Interface Maps

3. Import Interface Map

Name	Device Profile	Logical Device	Actions
Generic_Server_1RU_1x10G_AOS-1x10-1	Generic_Server_1RU_1x10G	AOS-1x10-1	Delete
Generic_Server_1RU_1x10G_Bionic_AOS-1x10-1	Generic_Server_1RU_1x10G_Bionic	AOS-1x10-1	Delete

## Import Interface Map

1. Make sure the "interface map" on page 91 that you want to import is in the global catalog.
2. From the blueprint, navigate to **Staged > Catalog > Interface Maps** and click **Import Interface Map**.
3. Select a logical device and an interface map from the drop-down lists. A preview of your selection appears.
4. Click **Import Selected Interface Map** to stage the import and return to the list view.

## Delete Interface Map (Blueprint)

1. From the blueprint, navigate to **Staged > Catalog > Interface Maps** and click the **Delete** button for the interface map to delete (in the Actions column on the right side).
2. Click **Delete** to stage the deletion and return to the list view.

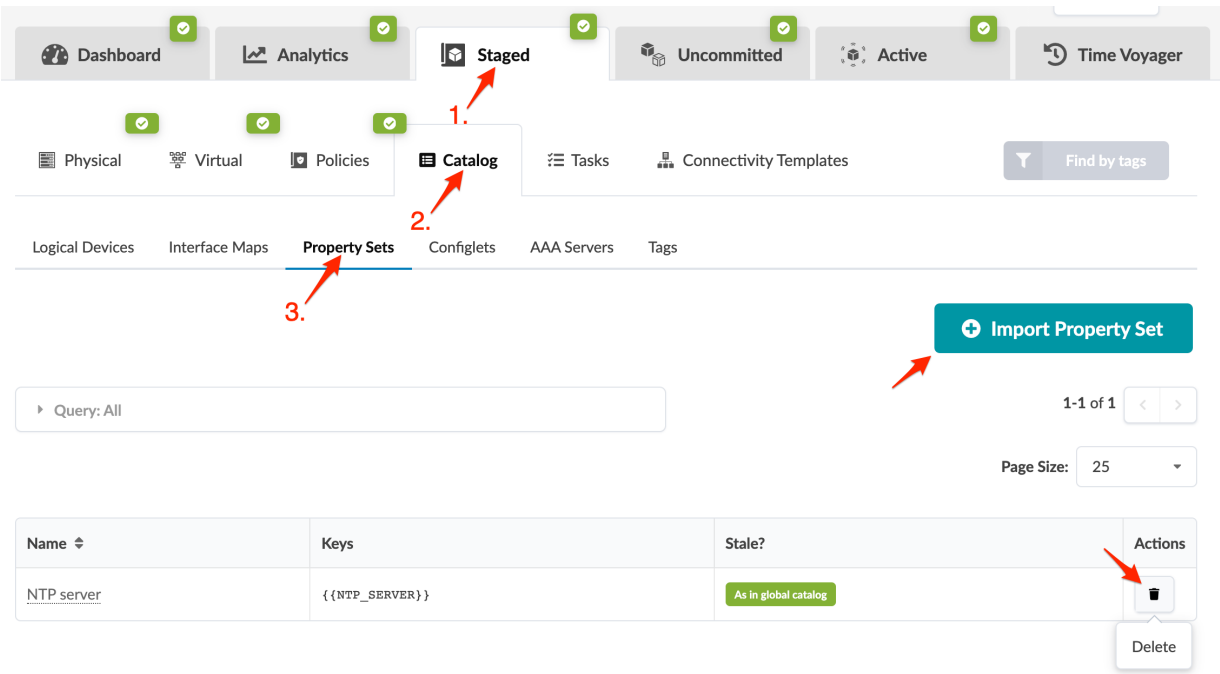
# Property Sets (Blueprint Catalog)

## IN THIS SECTION

- [Property Sets Overview \(Blueprint\) | 612](#)
- [Import Property Set | 612](#)
- [Re-import Property Set | 613](#)
- [Delete Property Set \(Blueprint\) | 613](#)

## Property Sets Overview (Blueprint)

From the blueprint, navigate to **Staged > Catalog > Property Sets** to go to the property sets catalog. You can import, re-import, and delete property sets from the blueprint catalog.



## Import Property Set

1. Make sure the "property set" on page 126 that you want to import is in the design catalog.
2. From the blueprint, navigate to **Staged > Catalog > Property Sets** and click **Import Property Set**.
3. From the drop-down list, select a property set from the design catalog, then click **Import Property Set** to stage the import and return to the list view.

## Re-import Property Set

If a property set that's used in a blueprint is updated in the design (global) catalog, a message appears in the blueprint catalog stating that the property set in the blueprint catalog is **Different from global catalog**. If you want the blueprint to use the updated property set, re-import it.

1. From the blueprint, navigate to **Staged > Catalog > Property Sets**.

The screenshot shows the 'Staged' view of the 'Catalog' with the 'Property Sets' sub-view selected. A table lists property sets, and a red arrow points to the 'Re-import' button in the Actions column for the 'NTP server' property set.

Name	Keys	State?	Actions
NTP server	{{NTP_SERVER}}	Different from global catalog	<div> <div>Re-import</div> </div>

2. Click the **Re-import** button for the "stale" property set, then click **Re-import Property Set** to stage the update and return to the list view.

## Delete Property Set (Blueprint)

As long as a property set is not used in a configlet, you can unassign it from a device at any time. If it is used in a configlet, a build error occurs and you won't be able to commit the change until you remove the property set from the configlet, which resolves that build error.

1. From the blueprint, navigate to **Staged > Catalog > Property Sets** and click the **Delete** button for the property set to delete.
2. Click **Delete** to stage the deletion and return to the list view.



For example, if the NETCONF port (tcp 830) is blocked by a configlet with misconfigured routing engine firewall filter entry, the Junos off-box agent cannot connect to the device to retrieve the running config. The device deployment remains in PENDING state forever and will never time out and fail. Even if you manually change the device config to unblock NETCONF port (tcp 830), Apstra again re-sends the configuration from the last commit which results in a continuing failure. To recover, you have to re-onboard the device. For more details and the workaround, see the [Juniper Support Knowledge Base article KB37291](#).

### Import Configlet

1. Make sure the "configlet" on page 119 that you want to import is in the design (global) catalog.
2. From the blueprint, navigate to **Staged > Catalog > Configlets** and click **Import Configlet**.
3. From the drop-down list, select a configlet from the design (global) catalog.



**CAUTION:** Do not import the same configlet more than once into the same blueprint. If you import the same configlet twice, then remove one of them, the remaining configlet is invalidated.

## Import Configlet from Global Catalog ✕

Configlet <sup>\*</sup>

US-EAST-NTP ✕ Configlets from global catalog appear in drop-down list

**Cumulus: system**

- Template Text
- Negation Template Text Click to see configlet contents

Configlet Scope

role in ["spine","leaf"] and hostname in [] Enter scope directly or...

...select from available criteria

Role ✕ 🗑️

Filter results

✓ Select Search Results

☒ spine

☒ leaf

and ▼

Hostname ✕ 🗑️

Filter results

✓ Select Search Results

☐ spine-1

☐ spine-2

☐ leaf-1-1

☐ leaf-1-2

☐ leaf-2

➕ Add Click to add additional criteria (You may need to scroll down to see it)

**Import Configlet**

4. For interface configlets only - you have the option of selecting specific roles and/or interface names.

**NXOS: INTERFACE**

▸ Template Text

▸ Negation Template Text

**Role**

☐ Interfaces between spine & superspine

☐ Interfaces between spine & leaf

☐ Interfaces facing L2 Generic systems with VLANs assigned

☐ MLAG Peer link (ignored)

☐ L3 Peer link between MLAG Peers

☐ Interfaces facing L3 Generic systems

☐ Unused & server interfaces without vlan assignment yet

**Interface predicate**

☒ or ☐ and

**Interface Names (comma-separated)**

Ethernet1/1,Ethernet1/2,Ethernet1/3,Ethernet1/4,Ethernet1/5,Ethern

5. Enter your scope query (autocomplete assists you) or create queries visually with the interactive cards (new in version 4.0).
6. Click **Import Configlet** to stage the configlet and return to the list view.

### Edit (Blueprint) Configlet

#### IN THIS SECTION

- [Edit Configlet Scope | 617](#)
- [Edit Configlet Generators | 618](#)

### *Edit Configlet Scope*

You can change the configlet scope (roles, IDs, hostnames) directly in the blueprint.

1. From the blueprint, navigate to **Staged > Catalog > Configlets** and click the **Edit** button for the configlet to edit.
2. Make your changes to the configlet scope. The options are the same as when importing a configlet.

**NOTE:** To change configlet generators (template text, negation template text, filename, as applicable) you must change them in the design (global) catalog, then re-import the configlet into the blueprint catalog. See the Edit Configlet Generators section.

3. Click **Update** to stage the update and return to the list view.

### *Edit Configlet Generators*

Configlet generators (template text, negation template text, filename, as applicable) cannot be changed directly in blueprints. If an existing configlet is no longer relevant, you can delete it and import a new or revised one. If you are changing a configlet in a blueprint catalog because of a configuration deviation, see also the Configlets and Config Deviation section.

1. ["Edit" on page 125](#) or create a ["configlet" on page 124](#) in the design (global) catalog.
2. ["Delete" on page 618](#) the configlet from the blueprint catalog.
3. ["Import" on page 615](#) the configlet into the blueprint catalog from the design (global) catalog.
4. ["Commit" on page 648](#) the changes.

### Delete (Blueprint) Configlet

When a configlet is deleted, it is removed from all devices within its scope.

1. From the blueprint, navigate to **Staged > Catalog > Configlets** and click the **Delete** button for the configlet to delete.
2. Click **Delete** to stage the deletion and return to the list view.

## AAA Servers (Blueprint Catalog)

### IN THIS SECTION

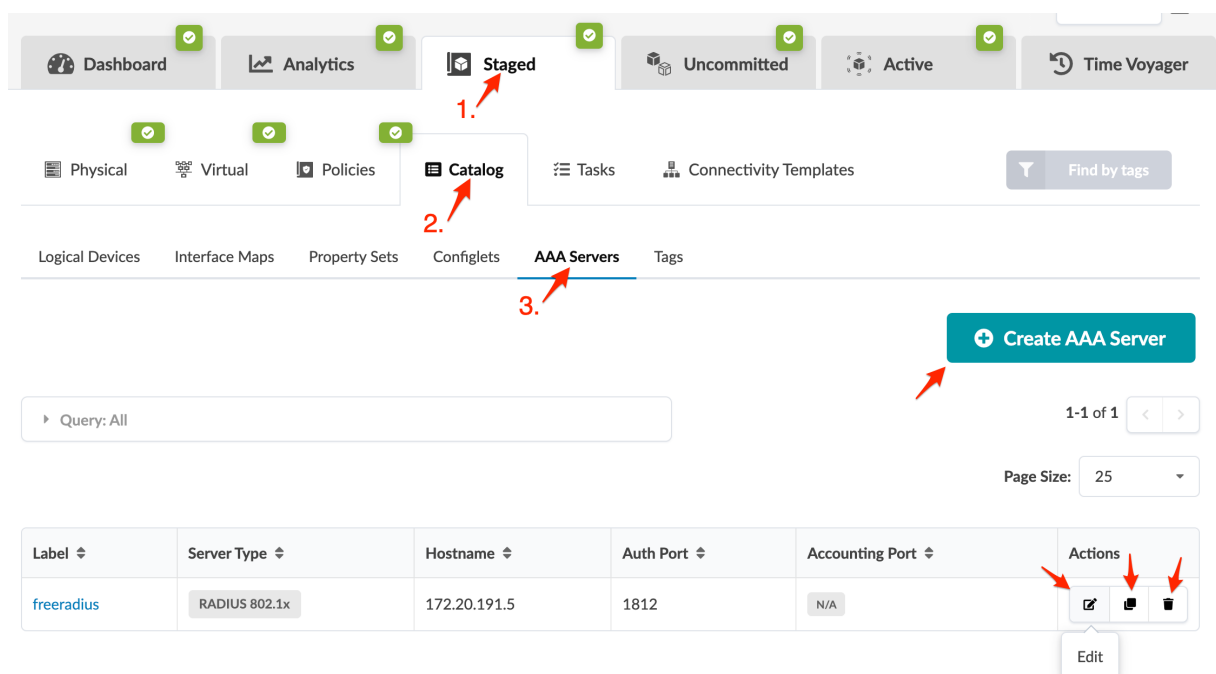
- [AAA Servers Overview | 619](#)
- [Create AAA Server | 620](#)
- [Edit AAA Server | 620](#)
- [Delete AAA Server | 620](#)
- [AAA RADIUS Server Configuration Tasks | 620](#)
- [Client Supplicant Configuration Tasks | 621](#)

AAA Servers Overview

AAA servers are used with ["interface policies" on page 591](#). AAA servers include the following details:

Name	Description
Label	To identify the AAA server
Server Type	<ul style="list-style-type: none"><li>• <b>RADIUS 802.1x</b> - If an 802.1x policy is bound to at least one interface on a switch, all defined AAA RADIUS 802.1x servers will be added to that switch. The server is not rendered unless it is needed.</li><li>• <b>RADIUS COA (Change of Authorization)</b> - Used by switches to enable Dynamic Authorization Server (DAS) requests from RADIUS servers. This enables the switch to 'trust' the given RADIUS server to do dynamic VLAN assignment after authentication instead of during auth. All RADIUS COA implementations are hard-coded to auth port 3799.</li></ul>
Hostname	
Auth Ports	
Accounting Port	optional

From the blueprint, navigate to **Staged > Catalog > AAA Servers** to go to the AAA servers catalog. You can create, clone, edit, and delete AAA servers.



## Create AAA Server

1. From the blueprint, navigate to **Staged > Catalog > AAA Servers** and click **Create AAA Server**.
2. Enter a label, select the server type (RADIUS 802.1x, RADIUS COA), enter a hostname, key, auth port, and (optional) accounting port.
3. Click **Create** to stage the server and return to the list view.

## Edit AAA Server

1. From the blueprint, navigate to **Staged > Catalog > AAA Servers** and click the **Edit** button for the AAA server to edit.
2. Make your changes, then click **Update** to stage the update and return to the list view.

## Delete AAA Server

1. From the blueprint, navigate to **Staged > Catalog > AAA Servers** and click the **Delete** button for the AAA server to delete.
2. Click **Delete** to stage the deletion and return to the list view.

## AAA RADIUS Server Configuration Tasks

AAA RADIUS server configuration tasks are external to Apstra software. The example below shows the files to configure for *FreeRADIUS*.

*/etc/freeradius/clients.conf* -- has credentials for each switch

```
client Arista-7280SR-48C6-1 {
    shortname = Arista-7280SR-48C6-1
    ipaddr    = 172.20.191.10
    secret    = testing123
    nastype   = other
}
```

*/etc/freeradius/users* -- has users and MAC addresses to authenticate. Tunnel-Private-Group-Id shows a dynamic VLAN ID, which is optional.

```
leaf1-server1 ClearText-Password := "password"

"52:54:00:37:d5:e1" Cleartext-Password := "52:54:00:37:d5:e1"
    Tunnel-Type = VLAN,
    Tunnel-Medium-Type = IEEE-802,
    Tunnel-Private-Group-Id = "50"
```

Although this example shows a simple credential, actual implementations may use any EAP method that both the client and RADIUS server support.

## Client Supplicant Configuration Tasks

Client supplicant configuration tasks are external to Apstra software. The following is an example for *wpa\_supplicant*.

*/etc/wpa\_supplicant/aos\_wpa\_supplicant.conf*

```
# Ansible managed
ctrl_interface=/var/run/wpa_supplicant
# Default version is 0 - ensure we're using modern protocols.
eapol_version=2
# Don't scan for wifi.
ap_scan=0
# Hosts will be configured to authenticate with usernames that match their
# Slicer DUT name, configured in radius_server playbook.
network={
```

```

key_mgmt=IEEE8021X
eap=TTLS MD5
identity="leaf1-server1"
anonymous_identity="leaf1-server1"
password="password"
phase1="auth=MD5"
phase2="auth=PAP password=password"
eapol_flags=0
}

```

## Tags (Blueprint Catalog)

### IN THIS SECTION

- [Tags Overview \(Blueprint\) | 622](#)
- [Search Tags \(Blueprint\) | 623](#)
- [Create Tag \(Blueprint\) | 623](#)
- [Import Tag | 624](#)
- [Export Tag | 624](#)
- [Edit Tag \(Blueprint\) | 624](#)
- [Delete Tag \(Blueprint\) | 624](#)

### Tags Overview (Blueprint)

You can apply tags to nodes, links and connectivity templates in your blueprint. When you create a blueprint, if you added tags to the design elements used to create that blueprint (rack types and templates), those tags are added to the blueprint **Tags** catalog. From the blueprint, navigate to **Staged > Catalog > Tags** to go to the tags blueprint catalog. You can add, clone, edit and delete blueprint tags. You can also import global catalog tags to the blueprint catalog and export blueprint tags to the global

catalog.

The screenshot shows the network management interface. The top navigation bar includes 'Dashboard', 'Analytics', 'Staged', 'Uncommitted', 'Active', and 'Time Voyager'. Below this, there are tabs for 'Physical', 'Virtual', 'Policies', 'Catalog', 'Tasks', and 'Connectivity Templates'. The 'Catalog' tab is selected, and the 'Tags' sub-tab is highlighted. A red arrow points to the 'Create Tag' button. Below the navigation bar, there is a search bar with the text 'Query: All'. To the right of the search bar, there is a pagination control showing '1-2 of 2' and a 'Page Size' dropdown set to '25'. Below the search bar, there is a table with the following columns: 'Name', 'Applied To', 'Description', and 'Actions'.

Name	Applied To	Description	Actions
Clients	CONNECTIVITY TEMPLATE : 1		Export, Edit, Delete, Refresh
external router	NODE : 2		Export, Edit, Delete, Refresh

## Search Tags (Blueprint)

You can filter tagged elements based on tag names and/or element types.

1. From the blueprint, navigate to **Staged > Catalog > Tags** and click **Query** to open the dialog.
2. Enter search criteria:
  - To see elements associated with tags, enter tag name(s) in the **Name** field.
  - To see tags that elements are associated with, select element type(s) from the drop-down list in the **Applied To** field.
  - To filter both by tag name and element type, enter details in both fields.
3. Click **Apply** to see filtered results in the table.
4. To go to the list view for a filtered element type, click the element type in the **Applied To** column. From there you can drill down for more details on a specific element.

## Create Tag (Blueprint)

1. From the blueprint, navigate to **Staged > Catalog > Tags** and click **Create Tag**.
2. Select **New** and enter a name and (optional) description. Names are case-insensitive.
3. Click **Create** to stage the new tag.

### Import Tag

1. From the blueprint, navigate to **Staged > Catalog > Tags** and click **Create Tag**.
2. Select **Import from Global Catalog**, select a tag from the drop-down list and enter an (optional) description.
3. Click **Create** to stage the tag import.

### Export Tag

1. From the blueprint, navigate to **Staged > Catalog > Tags** and click the **Export** button for the tag to export. If a tag exists in the global catalog with the same name you won't be able to export it. (The export button will be nonfunctional.)
2. Click **Export** to export the tag to the global catalog and return to the list view.

### Edit Tag (Blueprint)

1. From the blueprint, navigate to **Staged > Catalog > Tags** and click the **Edit** button for the tag to edit.
2. Change the description.
3. Click **Update** to stage the change and return to the list view.

### Delete Tag (Blueprint)

1. From the blueprint, navigate to **Staged > Catalog > Tags** and click the **Delete** button for the tag to delete.
2. Click **Delete** to stage the deletion and return to the list view.

## Tasks

From the blueprint, navigate to **Staged > Tasks** to go to task history. Blueprint task details include type of task, task status (succeeded, failed, in progress), date/time started, date/time last updated, and the duration of the task. For any failed tasks, you can click to see error messages.

## Connectivity Templates

### IN THIS SECTION

- [Primitives | 625](#)
- [View Connectivity Templates | 637](#)
- [Create Connectivity Template for Multiple VNs on Same Interface \(Example\) | 638](#)
- [Create Connectivity Template for Layer 2 Connected External Router \(Example\) | 640](#)
- [Assign Connectivity Template | 643](#)
- [Edit Connectivity Template | 647](#)
- [Delete Connectivity Template | 647](#)

Connectivity templates (introduced in Apstra version 4.0.0) enable you to apply various network configurations to devices connected to generic systems, as a Day 2 operation. Devices could be leafs, spines, or in 5-stage Clos topologies, superspines. Some use cases for connectivity templates include the following:

- Assigning Apstra virtual network endpoints (tagging and untagging VLAN ports) to connect Layer 2 servers.
- Creating Layer 3 interfaces and VLAN-tagged sub-interfaces with BGP routing between Apstra fabric border-leafs and external routers.

As of Apstra version 4.0.0, external router connections to the default routing zone are no longer required. You can use connectivity templates to configure the required external routing connections to routing zones. You can see static routes and protocol sessions by navigating to **Staged > Virtual** in the blueprint.

Connectivity templates consist of combinations of primitives as described in this section.

### Primitives

#### IN THIS SECTION

- [Virtual Network \(Single\) | 627](#)
- [Virtual Network \(Multiple\) | 628](#)

- IP Link | 628
- Static Route | 629
- Custom Static Route | 630
- BGP Peering (IP Endpoint) | 631
- BGP Peering (Generic System) | 632
- Dynamic BGP Peering | 634
- Routing Policy | 635
- User-defined | 636
- Pre-defined | 637

The **Primitives** tab includes the supported configuration functions that can be added to connectivity templates.

Create Connectivity Template

ParametersPrimitivesUser-definedPre-defined

**Virtual Network (Single)**  
Add a single VLAN to interfaces, as tagged or untagged.  
Accepts: interfaceProduce: vn\_endpoint

**Virtual Network (Multiple)**  
Add a list of VLANs to interfaces, as tagged or untagged.  
Accepts: interface

**IP Link**  
Build an IP link between a fabric node and a generic system. This primitive uses AOS resource pool "Link IPs - To Generic" by default to dynamically allocate an IP endpoint (/31) on each side of the link. To allocate different IP endpoints, navigate under Routing Zone>Subinterfaces Table.  
Accepts: interfaceProduce: ip\_link

**Static Route**  
Create a static route to user defined subnet via next hop derived from either IP link or VN endpoint.  
Accepts: ip\_link, vn\_endpoint

**Custom Static Route**  
Create a static route with user defined next hop and destination network.  
Accepts: system

**BGP Peering (IP Endpoint)**  
Create a BGP peering session with a user-specified BGP neighbor addressed peer.  
Accepts: svi, loopback, ip\_linkProduce: protocol\_endpoint

**BGP Peering (Generic System)**  
Create a BGP peering session with Generic Systems inherited from AOS Generic System properties such as loopback and ASN (addressed, or link-local peer).  
Accepts: ip\_link, vn\_endpointProduce: protocol\_endpoint

Application Pointany

You have started blank Connectivity Template creation

Please select one of the possible options to proceed with CT building:

Primitives

Select primitive to use

User-defined

Re-use user-defined Connectivity Template (all primitives it consists of will be added to the current one)

Pre-defined

Re-create Connectivity Template based on a pre-defined template

### Virtual Network (Single)

The virtual network (single) primitive ends with a **vn\_endpoint** point that can optionally connect to another compatible primitive, such as BGP peering (generic system).

#### Create Connectivity Template

Parameters

PrimitivesUser-definedPre-defined

▼ Summary

Title \*

The New CT

Description

Tags

No tags

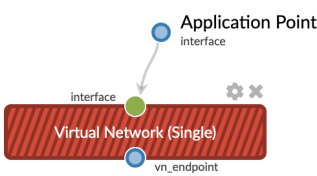
▼ Virtual Network (Single) ⚠

Virtual Network ID \*

Value is required

Virtual Network Tag Type \*

☒ VLAN Tagged☐ Untagged



Virtual Network (Multiple)

Unlike the virtual network (single) primitive, the virtual network (multiple) primitive cannot connect another primitive.

Create Connectivity Template

ParametersPrimitivesUser-definedPre-defined

▼ Summary

Title \*

The New CT

Description

Tags

No tags

▼ Virtual Network (Multiple)

Untagged Virtual Network

Tagged Virtual Networks

Application Point interface

interface

Virtual Network (Multiple)

IP Link

IP link uses Apstra resource pool **Link IPs - To Generics** (by default) to dynamically allocate an IP endpoint (/31) on each side of the link. You can create an IP link for any routing zone including the default routing zone. You can use an untagged link even if it is for a non-default routing zone.

The IP link primitive ends with an **ip\_link** point that can optionally connect to another compatible primitive, such as BGP peering (generic system).

Create Connectivity Template

Parameters

Primitives

User-defined

Pre-defined

Tags

No tags

IP Link

Routing Zone

Value is required

OFF

 Untagged

VLAN ID

2

IPv4 Addressing Type

None

Numbered

IPv6 Addressing Type

None

Link local

Application Point interface

interface

IP Link

ip\_link

Static Route

Next-hop is derived from either the IP link or virtual network endpoint. If the remote peer IP is shared across the generic system, then share the IP endpoint.

The **Static Route** primitive uses the next available IP address as the next-hop. To use a specific next-hop IP address, use the **Custom Static Route** instead.

Create Connectivity Template

Parameters

Primitives

User-defined

Pre-defined

▼ Summary

Title \*  

The New CT

Description

Tags  

No tags

▼ Static Route ⚠

Network \*  

203.0.113.0/24 or 2001:db8::/32

Value is required

OFF

 Share IP Endpoint ⓘ

Application Point  
ip\_link, vn\_endpoint

ip\_link, vn\_endpoint

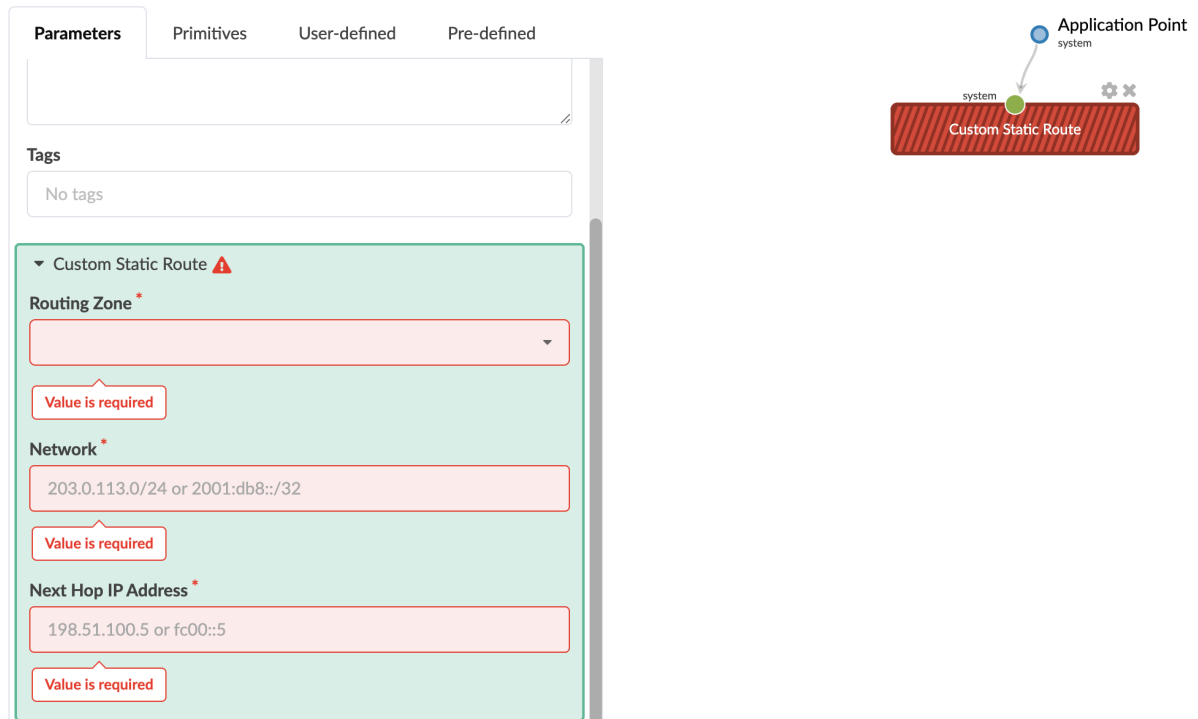
Static Route

Custom Static Route

If the next-hop IP address is not accessible, the static route will not be installed. Apstra software cannot monitor the next-hop IP and will not alert you if it is not accessible. It is your responsibility to configure the custom static route primitive correctly.

Connectivity templates using this primitive can only be assigned to leaf systems and cannot be combined with interface primitives.

### Create Connectivity Template



The screenshot displays the 'Create Connectivity Template' interface. The 'Parameters' tab is selected, showing a 'Custom Static Route' primitive. The form includes the following fields and errors:

- Routing Zone \***: A dropdown menu with a red border and a 'Value is required' error message.
- Network \***: A text input field containing '203.0.113.0/24 or 2001:db8::/32' with a red border and a 'Value is required' error message.
- Next Hop IP Address \***: A text input field containing '198.51.100.5 or fc00::5' with a red border and a 'Value is required' error message.

On the right, a diagram illustrates the connectivity template. It shows an 'Application Point system' (blue circle) connected to a 'system' node (green circle), which is then connected to a 'Custom Static Route' block (red rectangle with diagonal lines). A gear icon and a close icon are visible next to the 'Custom Static Route' block.

### BGP Peering (IP Endpoint)

The BGP peering (IP endpoint) primitive creates a BGP peering session with a user-specified BGP neighbor addressed peer. You can use this to create a BGP peering session to a Layer 3 server running BGP connected to an Apstra virtual network.

The following parameters must be configured:

- Neighbor ASN type (static, dynamic)
- If the neighbor ASN type is static, the ASN
- IPv4 AFI
- IPv6 AFI
- BGP Time to Live (TTL)
  - When you set TTL to 0, nothing is configured and the device defaults are used.

- When you set TTL to 1, Cisco NX-OS and FRR-based BGP (Cumulus Linux and SONiC) render disable-connected-check. Otherwise, TTL values render ebgp-multihop on specific BGP neighbors.
- Single-hop BFD
  - This enables BFD for the BGP peering. Multihop BFD is only supported for Junos, which is activated by default.
- BGP Password
- BGP Keep Alive Timer (seconds)
- BGP Hold Time Timer (seconds)
- IPv4 address of peer (if IPv4 AFI is enabled)
- IPv6 address of peer (if Ipv6 AFI is enabled)

You can connect a routing policy primitive to a BGP peering (IP endpoint)

### Create Connectivity Template

Parameters Primitives User-defined Pre-defined

▼ BGP Peering (IP Endpoint)

Neighbor ASN Type \*

☒ Static
 ☐ Dynamic

ASN

64496

ON ☐ IPv4 AFI \*

OFF ☐ IPv6 AFI \*

TTL ⓘ

2

OFF ☐ Single-hop BFD ⓘ

Password

Keep Alive Timer (sec)

Hold Time Timer (sec)

Application Point  
svi, loopback, ip\_link

svi, loopback, ip\_link

BGP Peering (IP Endpoint)

protocol\_endpoint

### BGP Peering (Generic System)

The BGP peering (generic system) primitive creates a BGP peering session with a generic system. The generic system is inherited from Apstra generic system properties, such as loopback and ASN

(addressed, link-local peer). This primitive connects to a virtual network (single) or IP link connectivity point primitive.

The following parameters must be configured:

- IPv4 AFI
- IPv6 AFI
- BGP Time to Live (TTL)
  - When you set TTL to 0, nothing is configured and the device defaults are used.
  - When you set TTL to 1, Cisco NX-OS and FRR-based BGP (Cumulus Linux and SONiC) renders disable-connected-check. Otherwise, TTL values render ebgp-multihop on specific BGP neighbors.
- Single-hop BFD
  - This enables BFD for the BGP peering. Multihop BFD is only supported for Junos, which is activated by default.
- BGP Password
- BGP Keep Alive Timer (seconds)
- BGP Hold Time Timer (seconds)
- IPv4 Addressing Type (none, addressed)
- IPv6 Addressing Type (none, (addressed if IPv6 applications are enabled) link local)
- Neighbor ASN Type (static, dynamic)
- Peer From (loopback, interface)
- Peer To (loopback, interface/IP endpoint, interface/shared IP endpoint)
  - Loopback: use this option to peer with the loopback address of a single remote system.
  - Interface/IP endpoint: use this option to peer with the IP address of a single remote system link or routed vlan interface.
  - Interface/Shared IP endpoint: use this option for any scenario where the remote peer IP address is shared across multiple remote systems.

You can connect a routing policy primitive to a BGP peering (generic system).

### Create Connectivity Template

The screenshot shows the 'Create Connectivity Template' interface with the 'Parameters' tab selected. The 'BGP Peering (Generic System)' section is expanded, showing the following configuration options:

- IPv4 AFI**: ☒ ON
- IPv6 AFI**: ☐ OFF
- TTL**: 2
- Single-hop BFD**: ☐ OFF
- Password**: [Empty text field]
- Keep Alive Timer (sec)**: [Empty text field]
- Hold Time Timer (sec)**: [Empty text field]
- IPv4 Addressing Type**: ☒ Addressed
- IPv6 Addressing Type**: [Empty text field]

To the right of the form is a diagram illustrating the connection. It shows a box labeled 'BGP Peering (Generic System)' with two endpoints: 'ip\_link, vn\_endpoint' (top) and 'protocol\_endpoint' (bottom). An 'Application Point' is shown above the 'ip\_link, vn\_endpoint' endpoint, with a green dot indicating the connection point.

### Dynamic BGP Peering

The dynamic BGP peering primitive enables dynamic peering on selected devices and virtual networks.

The following parameters must be configured:

- IPv4 AFI
- IPv6 AFI
- BGP Time to Live (TTL)
  - When you set TTL to 0, nothing is configured and the device defaults are used.
  - When you set TTL to 1, Cisco NX-OS and FRR-based BGP (Cumulus Linux and SONiC) renders disable-connected-check. Otherwise, TTL values render ebgp-multihop on specific BGP neighbors.
- Single-hop BFD
  - This enables BFD for the BGP peering. Multihop BFD is only supported for Junos, which is activated by default.

- BGP Password
- BGP Keep Alive Timer (seconds)
- BGP Hold Time Timer (seconds)
- IPv4
- IPv6
- IPv4 subnet for BGP prefix dynamic neighbors. If you leave this field blank, Apstra derives the subnet from the application point.
- IPv6 subnet for BGP prefix dynamic neighbors. If you leave this field blank, Apstra derives the subnet from the application point.

### Create Connectivity Template

The screenshot displays the 'Create Connectivity Template' interface. The 'Parameters' tab is active, showing configuration options for 'Dynamic BGP Peering'. The 'IPV4 AFI' and 'IPV6 AFI' are both set to 'OFF'. The 'TTL' is set to '2'. 'Single-hop BFD' is set to 'OFF'. The 'Password' field is empty. The 'Keep Alive Timer (sec)' and 'Hold Time Timer (sec)' fields are also empty. At the bottom, 'IPv4' and 'IPv6' are set to 'OFF', and the 'IPv4 Subnet for BGP Prefix Dynamic Neighbors' field is empty. To the right, a diagram shows an 'Application Point' (ip\_link, svi) connected to a 'Dynamic BGP Peering' block, which is then connected to a 'protocol\_endpoint'.

### Routing Policy

The routing policy primitive applies a routing policy to an application endpoint. This overrides the routing policy configured for the routing zone. You must select the routing policy that was defined in the

blueprint (Staged > Policies > Routing Policies).

Create Connectivity Template

ParametersPrimitivesUser-definedPre-defined

▼ Summary

Title \*

The New CT

Description

Tags

No tags

▼ Routing Policy ⚠

Routing Policy \*

Value is required

Application Point  
protocol\_endpoint

protocol\_endpoint

Routing Policy

User-defined

From the **User-defined** tab, you can add grouped primitives that you previously created as connectivity templates.

Create Connectivity Template

ParametersPrimitivesUser-definedPre-defined

Quick Search

rtr\_leaf1\_leaf2:l3:ct\_bgp\_subintf\_to\_subintf:ipv4

vn\_endpoints\_blue\_300\_leaf3\_v4\_vlan\_tagged

vn\_endpoints\_blue\_301\_leaf4\_v4\_vlan\_tagged

vn\_endpoints\_blue\_302\_leaf5\_v4\_vlan\_tagged

vn\_endpoints\_blue\_303\_leaf\_pair001\_00\_v4\_vlan\_tagged

vn\_endpoints\_blue\_vxlan\_34\_v4\_1\_vlan\_tagged

You have started blank Connectivity Template creation

Please select one of the possible options to proceed with CT building:

Primitives

Select primitive to use

User-defined

Re-use user-defined Connectivity Template (all primitives it consists of will be added to the current one)

Pre-defined

Re-create Connectivity Template based on a pre-defined template

## Pre-defined

From the **Pre-defined** tab, you can add grouped primitive provided by Apstra software.

### Create Connectivity Template

Parameters Primitives User-defined **Pre-defined**

Quick Search

[BGP Dynamic over L3 connectivity](#)

[BGP over L2 connectivity](#)

[BGP over L3 connectivity](#)

**You have started blank Connectivity Template creation**  
Please select one of the possible options to proceed with CT building:

- Primitives** Select primitive to use
- User-defined** Re-use user-defined Connectivity Template (all primitives it consists of will be added to the current one)
- Pre-defined** Re-create Connectivity Template based on a pre-defined template

## View Connectivity Templates

From the blueprint, navigate to **Staged > Connectivity Templates** to go to the connectivity template list view. You can create, assign, edit, and delete connectivity templates.

Dashboard Analytics **Staged** Uncommitted Active Time Voyager

Physical Virtual Policies Catalog Tasks **Connectivity Templates** Find by tags

Application Endpoints **Add Template**

Query: All 1-25 of 29 Page Size: 25

Filter selected by ☒ all ☐ selected only ☐ unselected only

**Click connectivity template for details**

Name	Description	Tags	Primitives	Status	Actions
<a href="#">rtr_leaf1_leaf2:l3:ct_bgp_subintf_to_subintf:ipv4</a>			<ul style="list-style-type: none"> <li>BGP Peering (Generic System)</li> <li>IP Link</li> </ul>	Assigned on 2 endpoint(s)	<a href="#">Link</a> <a href="#">Edit</a> <a href="#">Delete</a>
<a href="#">vn_endpoints_blue_300_leaf3_v4_vlan_tagged</a>			<ul style="list-style-type: none"> <li>Virtual Network (Single)</li> </ul>	Assigned on 1 endpoint(s)	<a href="#">Link</a> <a href="#">Edit</a> <a href="#">Delete</a>
<a href="#">vn_endpoints_blue_301_leaf4_v4_vlan_tagged</a>			<ul style="list-style-type: none"> <li>Virtual Network (Single)</li> </ul>	Assigned on 1 endpoint(s)	<a href="#">Link</a> <a href="#">Edit</a> <a href="#">Delete</a>
<a href="#">vn_endpoints_blue_302_leaf5_v4_vlan_tagged</a>			<ul style="list-style-type: none"> <li>Virtual Network (Single)</li> </ul>	Assigned on 1 endpoint(s)	<a href="#">Link</a> <a href="#">Edit</a> <a href="#">Delete</a>
<a href="#">vn_endpoints_blue_303_leaf_pair001_00_v4_vlan_tagged</a>			<ul style="list-style-type: none"> <li>Virtual Network (Single)</li> </ul>	Assigned on 3 endpoint(s)	<a href="#">Link</a> <a href="#">Edit</a> <a href="#">Delete</a>
<a href="#">vn_endpoints_blue_vxlan_34_v4_1_vlan_tagged</a>			<ul style="list-style-type: none"> <li>Virtual Network (Single)</li> </ul>	Assigned on 6 endpoint(s)	<a href="#">Link</a> <a href="#">Edit</a> <a href="#">Delete</a>
<a href="#">vn_endpoints_blue_vxlan_36_v4_no_eps_vlan_tagged</a>			<ul style="list-style-type: none"> <li>Virtual Network (Single)</li> </ul>	Ready	<a href="#">Link</a> <a href="#">Edit</a> <a href="#">Delete</a>

## Create Connectivity Template for Multiple VNs on Same Interface (Example)

To create connectivity templates you add primitives (either singly or in groups) to a staging area, then you configure the parameters of those primitives. You can include 64 primitives in each connectivity template (increased from 18 as of Apstra version 4.0.1). We'll use examples to illustrate the process. We'll start by showing you how to create multiple virtual networks for the same interface.

1. From the blueprint, navigate to **Staged > Connectivity Templates** and click **Add Template**. The staging area on the right contains the application point.

### Create Connectivity Template

**Parameters** Primitives User-defined Pre-defined

▼ Summary

**Title \***

The New CT

**Description**

**Tags**

No tags

**You have started blank Connectivity Template creation**  
Please select one of the possible options to proceed with CT building:

**Primitives** Select primitive to use

**User-defined** Re-use user-defined Connectivity Template (all primitives it consists of will be added to the current one)

**Pre-defined** Re-create Connectivity Template based on a pre-defined template

2. In the **Parameters** tab, enter a connectivity name in the **Title** field. You can optionally enter a description, and tags that you'll be able to use during subsequent searches.
3. The tabs **Primitives**, **User-defined**, and **Pre-defined** all contain primitives either singly or in groups. They are described in more detail in the overview. For this example, we'll add primitives one at a time from the **Primitives** tab. Click the **Primitives** tab, then click **Virtual Network (Single)**. It's added to the staging area, and it's connected to the application point.

### Create Connectivity Template

**Parameters** **Primitives** User-defined Pre-defined

Quick Search

**Virtual Network (Single)**  
Add a single VLAN to interfaces, as tagged or untagged.  
Accepts: interface Produces: vn\_endpoint

**Virtual Network (Multiple)**  
Add a list of VLANs to interfaces, as tagged or untagged.  
Accepts: interface

**IP Link**  
Build an IP link between a fabric node and a generic system. This primitive uses AOS resource pool "Link IPs - To Generic" by default to dynamically allocate an IP endpoint (/31) on each side of the link. To allocate different IP endpoints, navigate under Routing Zone>Subinterfaces Table.  
Accepts: interface Produces: ip\_link

**3. The selected primitive appears in the staging area**

Application Point interface

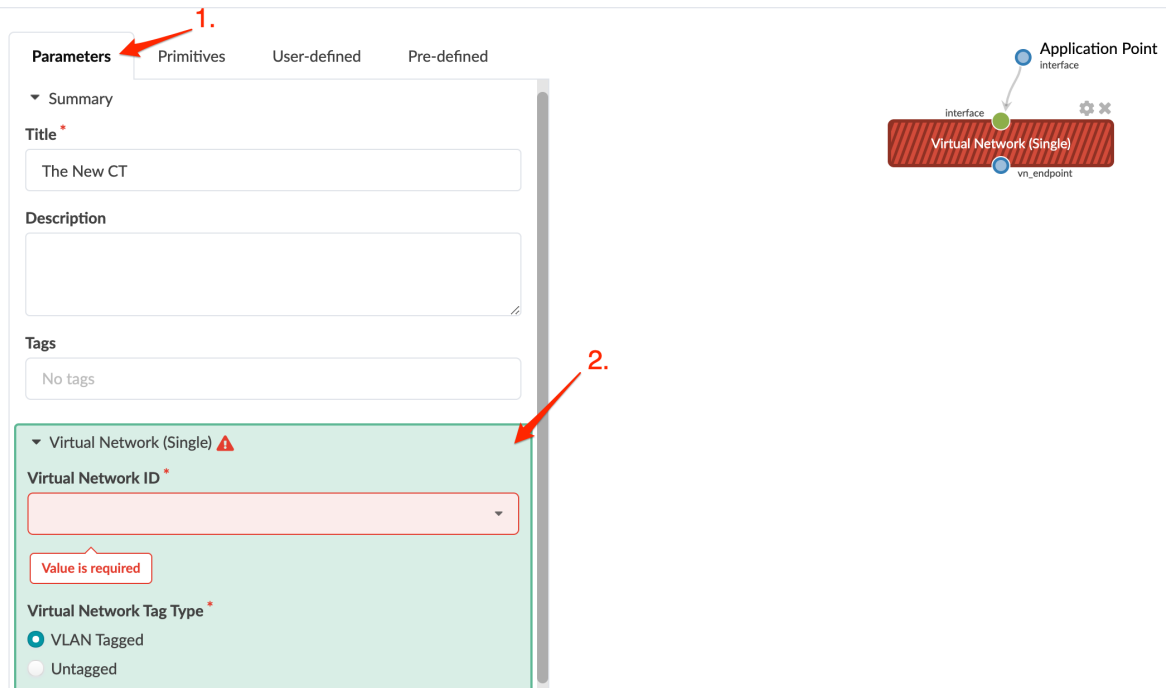
interface

Virtual Network (Single)

vn\_endpoint

- Click the **Parameters** tab to see what you need to configure for that primitive. In this example, you need to select a virtual network and specify whether it is VLAN tagged or untagged.

#### Create Connectivity Template



**Parameters** Primitives User-defined Pre-defined

▼ Summary

**Title \***

The New CT

**Description**

**Tags**

No tags

▼ Virtual Network (Single) ⚠

**Virtual Network ID \***

Value is required

**Virtual Network Tag Type \***

☒ VLAN Tagged

☐ Untagged

Application Point interface

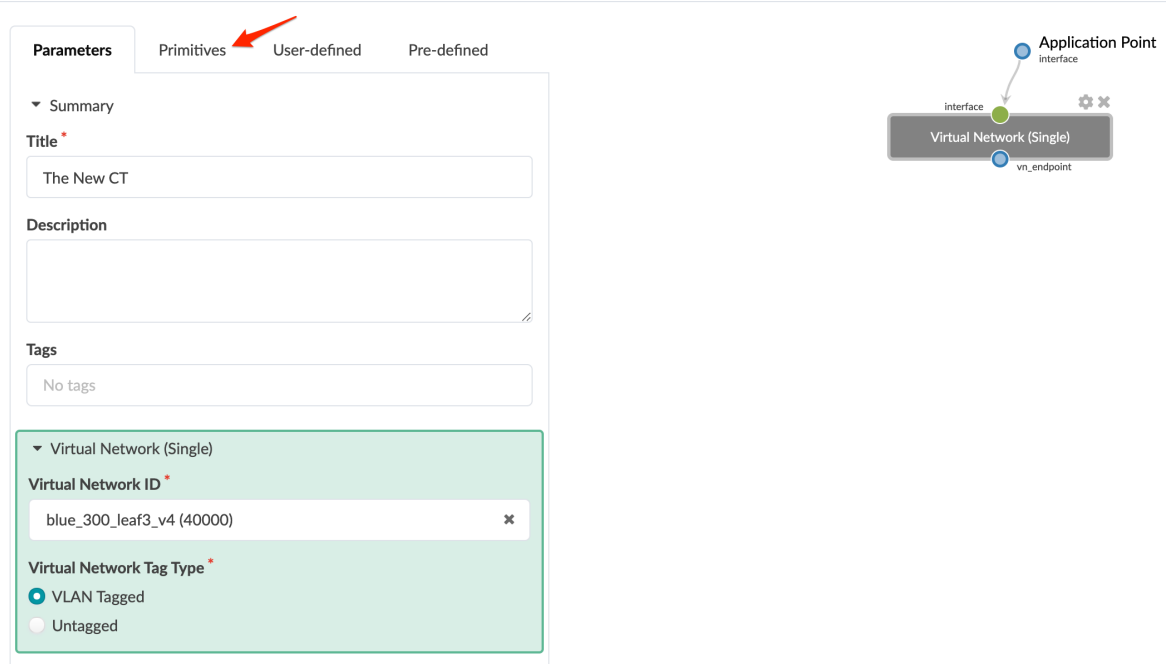
interface

Virtual Network (Single)

vn\_endpoint

- When it's successfully configured, the color of the selected primitive changes from red to gray. Click the **Primitives** tab.

#### Create Connectivity Template



**Parameters** Primitives User-defined Pre-defined

▼ Summary

**Title \***

The New CT

**Description**

**Tags**

No tags

▼ Virtual Network (Single)

**Virtual Network ID \***

blue\_300\_leaf3\_v4 (40000) ✕

**Virtual Network Tag Type \***

☒ VLAN Tagged

☐ Untagged

Application Point interface

interface

Virtual Network (Single)

vn\_endpoint

6. From the **Primitives** tab, click **Virtual Network (Multiple)**.

### Create Connectivity Template

1. Primitives tab

2. Virtual Network (Multiple)

3. Virtual Network (Multiple) in diagram

7. In the staging area, click **Virtual Network (Multiple)** (to make sure it's selected), click the **Parameters** tab and configure the primitive.

8. Click **Create** to create the connectivity template and return to the list view where you'll see your newly created connectivity template.

Physical Virtual Policies Catalog Tasks Connectivity Templates

Find by tags

Application Endpoints Add Template

Query: All 1-25 of 30

Page Size: 25

Filter selected by ☒ all ☐ selected only ☐ unselected only

Name	Description	Tags	Primitives	Status	Actions
rtr_leaf1_leaf2:l3:ct_bgp_subintf_to_subintf:ipv4			<ul style="list-style-type: none"> <li>BGP Peering (Generic System)</li> <li>IP Link</li> </ul>	Assigned on 2 endpoint(s)	<a href="#">Link</a> <a href="#">Edit</a> <a href="#">Delete</a>
The New CT			<ul style="list-style-type: none"> <li>Virtual Network (Multiple)</li> <li>Virtual Network (Single)</li> </ul>	Ready	<a href="#">Link</a> <a href="#">Edit</a> <a href="#">Delete</a>

### Create Connectivity Template for Layer 2 Connected External Router (Example)

In addition to applying multiple primitives to the application point interface, you can connect compatible primitives to each other. For example, let's configure a Layer 2 connected external router.

1. From the **Create Connectivity Template** dialog, click **Primitives**, click **Virtual Network (Single)**, and configure it on the **Parameters** tab (similar to the first example).

### Create Connectivity Template

Parameters

Primitives

User-defined

Pre-defined

▼ Summary

Title \*  
The New CT

Description

Tags  
No tags

▼ Virtual Network (Single)

Virtual Network ID \*  
blue\_300\_leaf3\_v4 (40000) ✕

Virtual Network Tag Type \*  
☒ VLAN Tagged  
☐ Untagged

Application Point  
interface

interface

Virtual Network (Single)

vn\_endpoint

- Click **Primitives**. When a primitive is selected, the other primitives that you can add to it are highlighted (new in Apstra version 4.0).

## Create Connectivity Template

1. With the primitive selected...

The screenshot displays the 'Create Connectivity Template' interface with the 'Primitives' tab selected. The interface lists several primitives, with 'Static Route' highlighted in green. A diagram on the right shows a 'Virtual Network (Single)' connected to an 'Application Point interface' and a 'vn\_endpoint'.

**Parameters** **Primitives** User-defined Pre-defined

**Virtual Network (Multiple)**  
Add a list of VLANs to interfaces, as tagged or untagged.  
**Accepts:** interface

**IP Link**  
Build an IP link between a fabric node and a generic system. This primitive uses AOS resource pool "Link IPs - To Generic" by default to dynamically allocate an IP endpoint (/31) on each side of the link. To allocate different IP endpoints, navigate under Routing Zone>Subinterfaces Table.  
**Accepts:** interface **Produces:** ip\_link

**Static Route**  
Create a static route to user defined subnet via next hop derived from either IP link or VN endpoint.  
**Accepts:** ip\_link, vn\_endpoint

**Custom Static Route**  
Create a static route with user defined next hop and destination network.  
**Accepts:** system

**BGP Peering (IP Endpoint)**  
Create a BGP peering session with a user-specified BGP neighbor addressed peer.  
**Accepts:** svi, loopback, ip\_link **Produces:** protocol\_endpoint

**BGP Peering (Generic System)**  
Create a BGP peering session with Generic Systems inherited from AOS Generic System properties such as loopback and ASN (addressed, or link-local peer).  
**Accepts:** ip\_link, vn\_endpoint **Produces:** protocol\_endpoint

**Virtual Network (Single)**  
interface Application Point interface  
vn\_endpoint

2. ...you can see what you can attach to it.

3. With **Virtual Network (Single)** selected in the staging area, click **BGP Peering (Generic System)** to add it to the staging area and connect it to the virtual network.

## Create Connectivity Template

The screenshot shows the 'Create Connectivity Template' interface. The 'Primitives' tab is selected, displaying a list of primitives. The 'BGP Peering (Generic System)' primitive is highlighted with a red box and an arrow pointing to it with the text '1. Select a primitive...'. To the right, a diagram shows the 'Virtual Network (Single)' component connected to an 'Application Point interface' and a 'BGP Peering (Generic System)' component. The 'BGP Peering (Generic System)' component is connected to the 'Virtual Network (Single)' component via a 'vn\_endpoint' and a 'protocol\_endpoint'. A red arrow points to the 'BGP Peering (Generic System)' component with the text '2. ...to add it to the staging area'.

4. Proceed with configuring the parameters and click **Create** to create the template.

## Assign Connectivity Template

### IN THIS SECTION

- [Assign Connectivity Template Overview | 643](#)
- [Method 1 | 644](#)
- [Method 2 | 645](#)
- [Force Assign VN Templates | 646](#)

## Assign Connectivity Template Overview

You can assign connectivity templates that have an active **Assign** button. These include connectivity templates in the **Ready** or **Assigned** status. (**Incomplete** status means that more configuration is required.) You can use one of two methods to assign connectivity points:

- **Method 1**- Select connectivity templates from the list view, and add application endpoints.
- **Method 2** - Click **Application Endpoints**, and assign connectivity templates to them.

## Method 1

1. From the blueprint, navigate to **Staged > Connectivity Templates** and click the **Assign** button (in the **Actions** section on the right) for the connectivity template to assign. (You can select multiple connectivity templates, to the left of the CT name, then click the **Assign** button that appears above the list.)

Dashboard Analytics Staged Uncommitted Active Time Voyager

Physical Virtual Policies Catalog Tasks Connectivity Templates Find by tags

Application Endpoints Add Template

Query: All 1-25 of 28 Page Size: 25

Filter selected by ☒ all ☐ selected only ☐ unselected only

<input type="checkbox"/>	Name	Description	Tags	Primitives	Status	Actions
<input type="checkbox"/>	rtr_leaf1_leaf2:l3:ct_bgp_subintf_to_subintf:ipv4			<ul style="list-style-type: none"> <li>BGP Peering (Generic System)</li> <li>IP Link</li> </ul>	Assigned on 2 endpoints	
<input type="checkbox"/>	vn_endpoints_blue_300_leaf3_v4_vlan_tagged			<ul style="list-style-type: none"> <li>Virtual Network (Single)</li> </ul>	Assigned on 1 endpoint(s)	
<input type="checkbox"/>	vn_endpoints_blue_301_leaf4_v4_vlan_tagged			<ul style="list-style-type: none"> <li>Virtual Network (Single)</li> </ul>	Assigned on 1 endpoint(s)	

The available fabric application endpoints appears in a dialog.

2. Click boxes on the connectivity template column to assign the connectivity template to the application endpoint. The **Tags** column shows the tags that are applied to each available application point. You can click the **Query** dialog to search by tags or labels.

Assign vn\_endpoints\_blue\_300\_leaf3\_v4\_vlan\_tagged

Table view

Query: All All bulk actions (⚙️) will be applied only to the loaded connectivity templates.

Fabric	Tags	vn_endpoints_blue_300_leaf3_v4_vlan_tagged
pod1 (Pod)		<input type="checkbox"/>
evpn_single_001_001 (Rack)		<input type="checkbox"/>
leaf3 (Leaf)		<input type="checkbox"/>
xe-0/0/2 -> switch3-server1 (Interface)		<input type="checkbox"/>

Assign

You can use "bulk actions" to select multiple "children" application endpoints.

3. Click **Assign** to complete the connectivity template assignments.

4. You can view application endpoints in **Table view (tech preview feature)**. From the table view, you can filter application endpoints by pod, rack, node, applied connectivity templates, or tags. You can also copy/paste connectivity template assignments from the table view.

Assign vn\_endpoints\_blue\_300\_leaf3\_v4\_vlan\_tagged

☒ On ☐ Table view

Pod: All, Rack: All, Node: All, Applied templates: All

Application point search: Search..., Tags: All

Bulk assign templates

1-1 of 1, Page Size: 25

Filter selected by: ☒ all ☐ selected only ☐ unselected only

Pod	Rack	Node	Application point	Tags	Applied templates	Actions
pod1	evpn_single_001_001	leaf3 (Leaf)	xe-0/0/2 -> switch3-server1 (Interface)		vn_endpoints_vlan_30_leaf3_v4_untagged vn_endpoints_blue_vxlan_33_v4_1_vlan_tagged vn_endpoints_red_304_leaf3_v4_vlan_tagged ... show 3 more	<input type="button" value="Copy"/> <input type="button" value="Paste"/> <input type="button" value="More"/>

Assign

**NOTE:** This feature has been classified as a Juniper Apstra Technology Preview feature. These features are "as is" and voluntary use. Juniper Support will attempt to resolve any issues that customers experience when using these features and create bug reports on behalf of support cases. However, Juniper may not provide comprehensive support services to Tech Preview features.

For additional information, refer to the ["Juniper Apstra Technology Previews"](#) on page 1082 page or contact ["Juniper Support"](#) on page 777

## Method 2

1. From the blueprint, navigate to **Staged > Connectivity Templates** and click **Application Endpoints**.
2. You can click the + button to add a column for multiple connectivity templates.

Application Endpoints

☒ On ☐ Table view

Query: All

All bulk actions (⚙️) will be applied only to the loaded connectivity templates.

Fabric	Tags	Templates Applied
pod1 (Pod)		N/A
evpn_esl_001_001 (Rack)		N/A
leaf1 (Leaf)		N/A
xe-0/0/0 -> rtr_leaf1_leaf2 (Interface)		rtr_leaf1_leaf2:3:ct_bgp_subintf_to_subintf:ipv4

3. You can then query and select the desired assignment combination of connectivity templates and application endpoints.
4. After a connectivity template is applied, its configuration may require additional resources in the blueprint. For example, if you're adding Layer 3 links to connect a generic system (such as an external router), you must assign **Generic Link IPs**.
5. You can view as a **Table view (tech preview feature)**. From the table view, you can filter application endpoints by pod, rack, node, applied connectivity templates, or tags. You can also copy/paste connectivity template assignments from the table view.

Application Endpoints ✕

☒ Table view

Pod:  Rack:  Node:  Applied templates:

Application point search:  Tags:

Applicable templates:   Page Size:

Filter selected by: ☒ all ☐ selected only ☐ unselected only

<input type="checkbox"/>	Pod	Rack	Node	Application point	Tags	Applied templates	Actions
<input type="checkbox"/>	pod1	evpn_es1_001_001	leaf1 (Leaf)	xe-0/0/0-> rtr_leaf1_leaf2 (Interface)		<input type="text" value="rtr_leaf1_leaf2:13:ct_bgp_subintf_to_subintfipv4 ✕"/>	<input type="button" value="Copy"/> <input type="button" value="Paste"/>

**NOTE:** This feature has been classified as a Juniper Apstra Technology Preview feature. These features are "as is" and voluntary use. Juniper Support will attempt to resolve any issues that customers experience when using these features and create bug reports on behalf of support cases. However, Juniper may not provide comprehensive support services to Tech Preview features.

For additional information, refer to the ["Juniper Apstra Technology Previews" on page 1082](#) page or contact ["Juniper Support" on page 777](#)

### Force Assign VN Templates

When a virtual network (single) or virtual network (multiple) template is already assigned to a port and you want to assign a new VN template, you'll receive a validation error indicating that the port already has a VN template assigned to it. As of Apstra version 4.0.1 you can force assign the new VN template, which automatically unassigns the existing VN template(s) and assigns the new one(s) on the selected port(s). You don't need to manually unassign the existing VN template.

To force assign VN templates, from the CT assignment screen, click **Remove all conflicts**, then click **Assign**.

Assign BLUE\_TAGGED\_VN

OFF

Table view

Query: All

All bulk actions (⚙️) will be applied only to the loaded connectivity templates.

Remove all conflicts

OFF

Show only conflicting rows?

Fabric	Tags	Conflicts ⚙️	Row Actions	BLUE_TAGGED_VN
▼ pod1 (Pod)			⚙️	<input type="checkbox"/> ⚙️
▼ I2_virtual_001 (Rack)			⚙️	<input type="checkbox"/> ⚙️
▼ I2_virtual_001_leaf1 (Leaf)			⚙️	<input type="checkbox"/> ⚙️
xe-0/0/4 -> I2_virtual_001_sys001 (Interface)		BLUE_UNTAGGED_MULTIPLE_VN	⚙️	<input checked="" type="checkbox"/>
xe-0/0/6 -> I2_virtual_001_sys002 (Interface)		BLUE_UNTAGGED_MULTIPLE_VN	⚙️	<input checked="" type="checkbox"/>
▼ I2_virtual_002 (Rack)			⚙️	<input type="checkbox"/> ⚙️
▼ I2_virtual_002_leaf1 (Leaf)			⚙️	<input type="checkbox"/> ⚙️
xe-0/0/4 -> I2_virtual_002_sys001 (Interface)			⚙️	<input type="checkbox"/>
xe-0/0/6 -> I2_virtual_002_sys002 (Interface)			⚙️	<input type="checkbox"/>
▼ I2_virtual_003 (Rack)			⚙️	<input type="checkbox"/> ⚙️
▼ I2_virtual_003_leaf1 (Leaf)			⚙️	<input type="checkbox"/> ⚙️

Assign

## Edit Connectivity Template

1. Either from the list view (Staged > Connectivity Templates) or the details view, click the **Edit** button for the connectivity template to edit.
2. Make your changes.
3. Click **Update** to update the connectivity template and return to the list view. (If you decide not to change the connectivity template, click **Revert Changes** to discard your changes.)

## Delete Connectivity Template

You cannot delete connectivity templates that have been assigned.

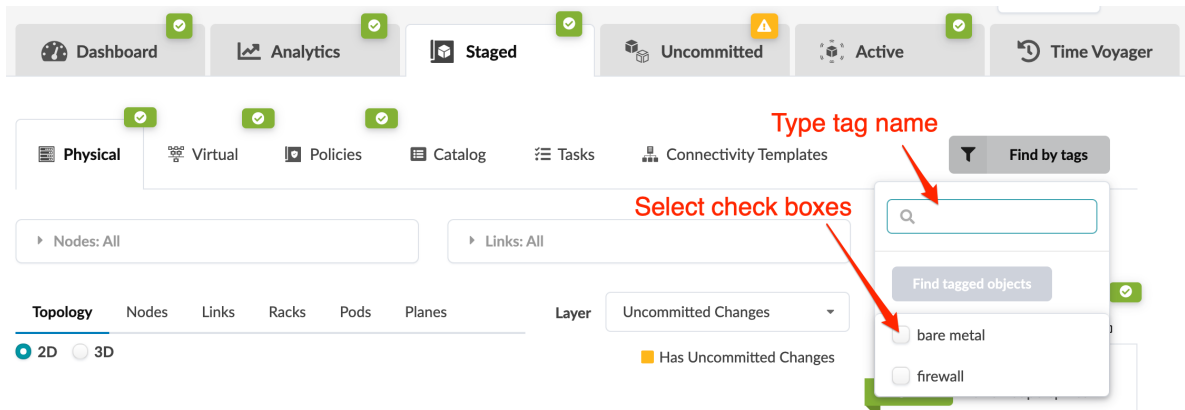
1. Either from the list view (Staged > Connectivity Templates) or the details view, click the **Delete** button for the connectivity template to delete.
2. Click **Delete** to delete the connectivity template and return to the list view.

## Find by Tags

With **Find by Tags**, you can search the entire blueprint for nodes, links, and connectivity templates that have associated tags.

1. From any page in the staged (or active) blueprint click **Find by Tags** (right side).

2. Either start typing to filter tags for selection, or select one or more check boxes.



3. Click **Find tagged objects** to display all objects with those tags.

# Uncommitted (Blueprints)

## IN THIS SECTION

- [Uncommitted Overview | 648](#)
- [Review Staged Changes | 650](#)
- [Commit Staged Changes | 653](#)
- [Revert Staged Changes | 653](#)

## Uncommitted Overview

When you start staging your new blueprint (under the **Staged** tab), the status indicator on the **Uncommitted** tab is red. When you've finished staging the blueprint and resolved any build errors, the indicator turns yellow (or orange if you have warnings, as of Apstra version 4.0.1) and the **Commit** button turns from gray to black indicating that the blueprint is ready to be committed. When you commit your pending changes you are pushing configuration to the **Active** blueprint. The meaning of the status indicator colors are shown in the table below:

Table 28: Uncommitted Status Indicators

Status Indicator Color	Description
Red	The blueprint needs staging or has <b>Build Errors</b> that must be resolved before you can commit.
Orange	The blueprint has <b>Warnings</b> to notify you of potential issues. The blueprint may or may not have staged changes. You can commit to a blueprint that has warnings and pending changes.
Yellow	The blueprint has pending changes that you can commit to the blueprint.
Green	The blueprint does not have any pending changes, warning, or errors. The blueprint is active and there is nothing to commit.

The blueprint below has warnings and pending changes. You can commit these changes.

Dashboard

Analytics

Staged

Uncommitted

Active

Logical Diff

Full Nodes Diff

Build Errors

Warnings

Warning Message

Node leaf001\_002\_1\_loopback: Loopback ipv4 subnet '10.0.0.8/32' overlaps with Loopback 'rack1-server1\_loopback' ipv4 subnet '10.0.0.8/32' error in the next releases which will prevent configuration from being deployed. Please make sure the warning is fixed.

Node rack1-server1\_loopback: Loopback ipv4 subnet '10.0.0.8/32' overlaps with Loopback 'leaf001\_002\_1\_loopback' ipv4 subnet '10.0.0.8/32' error in the next releases which will prevent configuration from being deployed. Please make sure the warning is fixed.

The blueprint below has warnings and no pending changes. There is nothing to commit.

Dashboard

Analytics

Staged

Uncommitted

Active

Time Voyager

Logical Diff

Full Nodes Diff

Build Errors

Warnings

1-2 of 2

Page Size: 25

Warning Message

Node leaf001\_002\_1\_loopback: Loopback ipv4 subnet '10.0.0.8/32' overlaps with Loopback 'rack1-server1\_loopback' ipv4 subnet '10.0.0.8/32'. This warning may be turned to error in the next releases which will prevent configuration from being deployed. Please make sure the warning is fixed.

Node rack1-server1\_loopback: Loopback ipv4 subnet '10.0.0.8/32' overlaps with Loopback 'leaf001\_002\_1\_loopback' ipv4 subnet '10.0.0.8/32'. This warning may be turned to error in the next releases which will prevent configuration from being deployed. Please make sure the warning is fixed.

You can review pending changes, and then decide to commit those changes or discard them. For more information, see the sections below.

## Review Staged Changes

1. From the blueprint top menu, click **Uncommitted** to go to pending changes. You can review **Logical Diff**, **Full Nodes Diff**, **Build Errors**, and **Warnings**. Full nodes diff shows all uncommitted changes in one place, organized by node type, change type and raw data. You can sort and search the diffs, then preview the changed element. Full nodes diff requires a fair amount of resources and time to generate.

Dashboard

Analytics

Staged

Uncommitted

Active

Time Voyager

Logical Diff

Full Nodes Diff

Build Errors

Warnings

Query: All

1-14 of 14

Page Size: 25

Type

Action

Name

Connectivity Template

CHANGED

rtr\_leaf1\_leaf2:l3:ct\_bgp\_subintf\_to\_subintf:ipv4

Protocol Sessions

ADDED

92a88b01-a710-4970-a9d8-e3ad6ee3e34f

Protocol Sessions

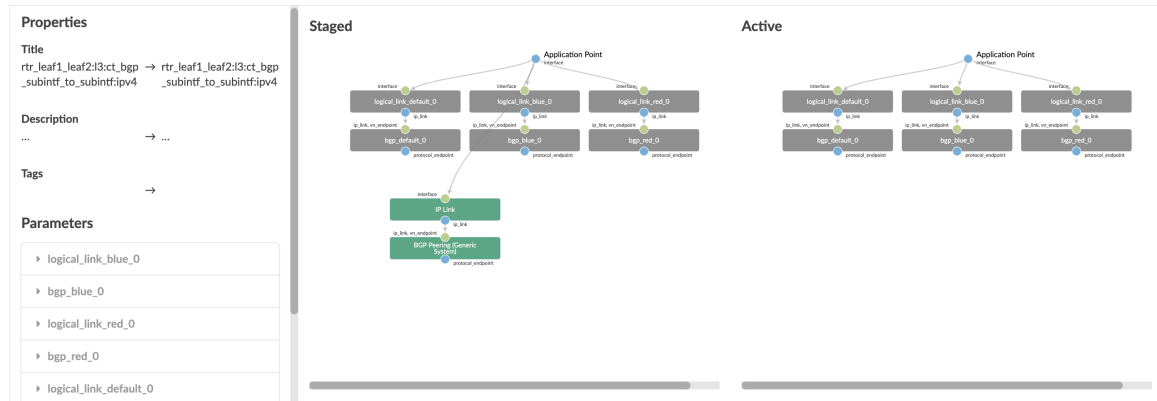
ADDED

baa208f3-e9a6-4ce8-8f93-766e2c13d7b9

Click to see changes

- From **Logical Diff**, click a name from the **Name** column to see detailed changes, additions or deletions for that element.

## Connectivity Template Preview



In some cases, you have the option of viewing only the differences, as shown below.

### Virtual Network Preview



☐ Show Diff Only?

#### Parameters

	Active	Staged
Name	red_vxlan_43_v4_no_eps	red_vxlan_43_v4_no_eps
Type	VXLAN	VXLAN
VNI	30009	30009
DHCP Service	Enabled	Enabled
IPv4 Connectivity	Enabled	Enabled
IPv4 Subnet	10.1.0.240/28	10.1.0.240/28
Virtual Gateway IPv4	10.1.0.241	10.1.0.241

#### Assigned To

### Virtual Network Preview



☒ Show Diff Only?

#### Endpoints

Query: All

1-4 of 4

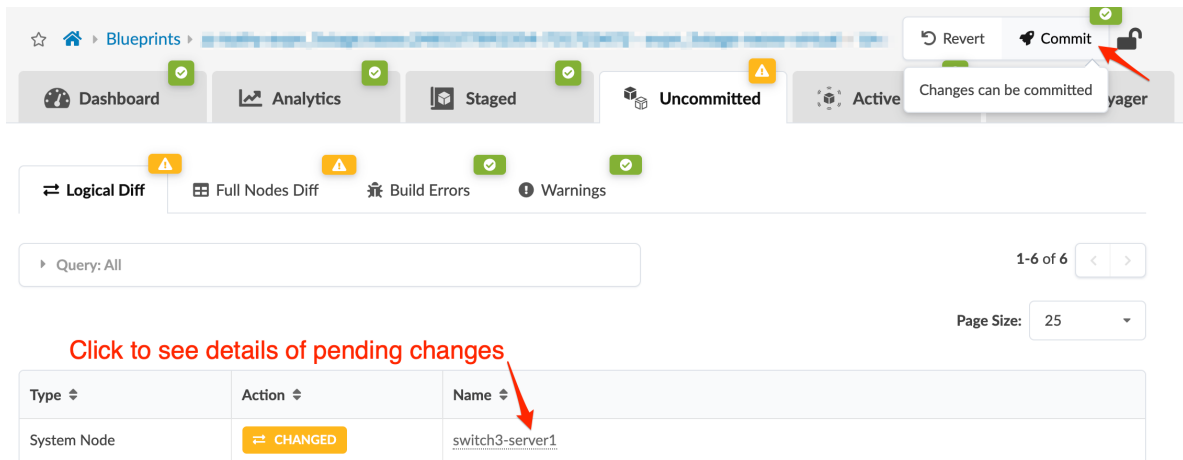
Page Size: 25

Leaf	Interface Name(s)	Generic System	Link Label	Generic System Group	Endpoint
leaf1	Ethernet1/4	switch1-server1	single-link	single-server-1	<div>Unassigned</div> <div>↓</div> <div>VLAN Tagged</div>
leaf2	Ethernet1/4	switch2-server1	single-link	single-server-2	<div>Unassigned</div> <div>↓</div> <div>VLAN Tagged</div>

3. When you are finished reviewing your changes and you've resolved any build errors, proceed to commit your changes to the blueprint or discard them, as applicable.

## Commit Staged Changes

1. From the blueprint top menu, click **Uncommitted** and review changes as needed.



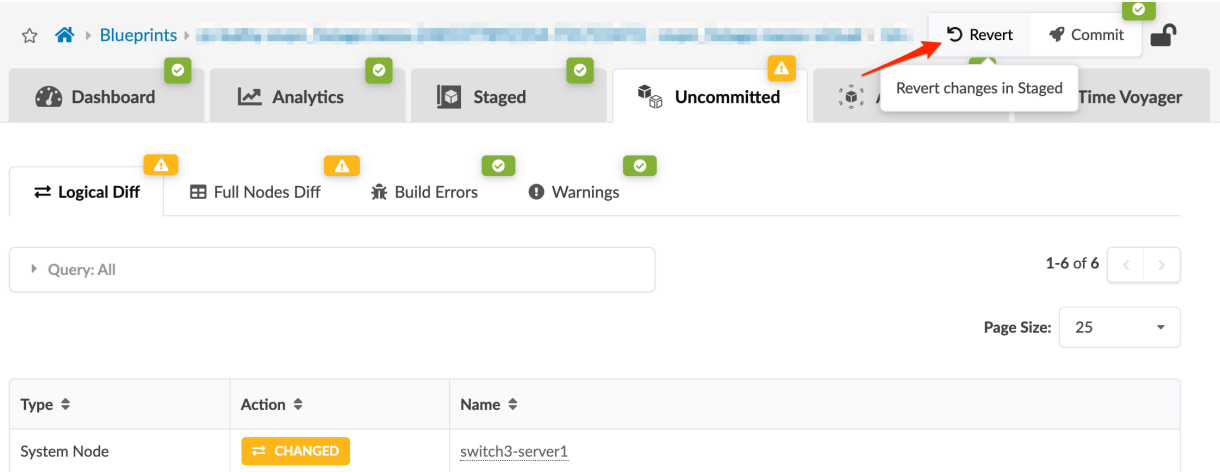
2. Click **Commit** and add a description of the changes. We recommend that you enter the optional revision description to identify changes. These descriptions are displayed in the **Revisions** section of "Time Voyager" on page 682. If you don't add a description now you can always add one later. If you need to roll back to a previous revision, this description helps to determine the appropriate revision. Specific diffs between revisions are not displayed, so the description is the only change information available for that revision.
3. Click **Commit** to push the staged changes to the active blueprint and create a revision.
4. While the task is active, you can click **Active Tasks** at the bottom of the screen for information about task progress. (Additional task history is available in the blueprint at **Staged > Tasks**.)

When a blueprint has been committed and devices have been deployed, the network is up and running. However, networks are not static and can require modifications as they evolve. Due to Juniper Apstra's approach of the *network as a single entity* this is extremely easy; all required device configurations are generated and pushed to the devices when you commit the change. We call this *Flexible Fabric Expansion (FFE)*.

## Revert Staged Changes

If you decide not to commit staged changes to a blueprint, you can discard them.

From the blueprint top menu, click **Uncommitted**, then click **Revert**. In some cases, you might also need to ["reset resource group overrides" on page 420](#).



# Active (Blueprints)

IN THIS SECTION

Status (Active) | 655

Selection (Active) | 655

Topology (Active) | 656

Nodes (Active) | 665

Links (Active) | 667

Racks (Active) | 668

Pods (Active) | 669

Query | 670

Anomalies (Service) | 671

Root Causes | 679

The details of a deployed network are shown in the active view. From here, you can view telemetry status in a topology-centric view. Alerts and anomalies can be filtered by different layers to conduct root cause analysis of any problems.

## Status (Active)

From the blueprint, navigate to **Active > Physical** to go to the statuses for services and deploy modes, deployment statuses for discovery, drain and service, as well as traffic heat.

The screenshot displays the 'Active > Physical' status page. The top navigation bar includes 'Dashboard', 'Analytics', 'Staged', 'Uncommitted', 'Active', and 'Time Voyager'. The left sidebar shows 'Physical' selected. The main area displays a topology view with nodes, links, racks, pods, and planes. A right-hand panel shows a list of anomalies and deployment statuses. Red arrows indicate navigation steps: 1. Click 'Active' in the top navigation bar. 2. Click 'Physical' in the left sidebar. 3. Click 'Status' in the right-hand panel.

**Top Navigation Bar:** Dashboard, Analytics, Staged, Uncommitted, Active, Time Voyager.

**Left Sidebar:** Physical, Virtual, Policies, Catalog, Query, Anomalies, Root Causes, Connectivity Templates, Find by tags.

**Main Area:** Nodes: All, Links: All, Selection, Status. Topology view showing 2D/3D options, Layer: Anomalies: All Services, and various filters for Selected Plane, Pod, Rack, Node, and Topology Label.

**Right Panel (Anomalies and Deployment Statuses):**

- Anomalies: All Services (0)
- Anomalies: BGP (0)
- Anomalies: Cabling (0)
- Anomalies: Config (0)
- Anomalies: Hostname (0)
- Anomalies: Interface (0)
- Anomalies: LAG (0)
- Anomalies: Liveness (0)
- Anomalies: MLAG (0)
- Anomalies: Probes (0)
- Anomalies: Route (0)
- 11/0/0/0 Deploy Mode
- 0/0/0 Deployment Status: Discovery
- 0/0/0 Deployment Status: Drain

## Selection (Active)

When you select a node in the active topology, information about telemetry, device, properties, tags, and VMs is available in the right panel.

Selection

Status

✓

rack\_2\_001\_sys001

Role: Generic System

Group label: generic

✓

Telemetry

Device

Properties

Tags

VMs

N/A

i

Not Deployed Yet

Telemetry for this node will become available when node will be deployed

## Topology (Active)

### IN THIS SECTION

- 2D Topology View (Active) | 656
- 3D Topology View (Active) | 658
- Neighbors View (Active) | 659
- Links View (Active Topology) | 662
- Virtual Networks Endpoints (Active) | 663
- Headroom (Topology) | 663

You can look at topologies as 2D views or 3D views. When you select a node from a topology view (by clicking its element in the topology, or by selecting it from the **Selected Nodes** drop-down list), details for the selection are displayed. You can view the selection to show neighbors, links, virtual network endpoints (as of Apstra version 4.0.1), or headroom. Telemetry and other device properties are displayed in the selection panel on the right side of the window.

### 2D Topology View (Active)

From the blueprint, navigate to **Active > Physical > Topology**. The default view is **2D**.

- To make topology elements larger, click the **Expand Nodes** check box.
- To show the links between elements, click the **Show Links** check box.
- To show node name, hostname (and role and tags as of Apstra version 4.0.1) as applicable, hover over an element.
- To display a different label (name, hostname, S/N), select a different label from the **Topology Label** drop-down list.
- To show rack details, select a rack by either clicking its element or by selecting it from the **Selected Rack** drop-down list.
- To show node details, select the node by either clicking its element in the topology or by selecting it from the **Selected Node** drop-down list.

The screenshot displays the Apstra interface with the following components and annotations:

- Top Navigation Bar:** Includes tabs for Dashboard, Analytics, Staged, Uncommitted, and Active. A red arrow labeled "1." points to the **Active** tab.
- Left Sidebar:** Contains icons for Physical, Virtual, Policies, Catalog, Query, Anomalies, Root Causes, and Cc. A red arrow labeled "2." points to the **Physical** icon.
- Filters:** Includes "Nodes: All" and "Links: All" dropdowns. A red arrow labeled "3." points to the **Nodes: All** dropdown.
- Topology View:** Shows a 2D/3D toggle (2D is selected), a "Layer" dropdown set to "Anomalies: All Services", and a legend for "No Anomalies" (green) and "Anomalies Present" (red).
- Selection Controls:** Includes "Selected Rack" and "Selected Node" dropdowns, both currently set to "All". A red arrow points to the "Selected Rack" dropdown with the text "Make selection from drop-down lists, or...".
- Display Options:** Includes checkboxes for "Expand Nodes?" and "Show Links?".
- Topology Diagram:** Shows a network topology with elements:
  - sys001** (grey box)
  - spine1** and **spine2** (green boxes)
  - rack\_1\_001** (dashed box containing **rack\_1\_001\_leaf1**, **rack\_1\_001\_leaf2**, and **rack\_1\_001\_sys001**)
  - rack\_2\_001** (dashed box containing **rack\_2\_001\_leaf1** and **rack\_2\_001\_sys001**)
- Annotations:**
  - A red arrow points to **spine2** with the text "...click selection directly for more info".
  - A red arrow points to **rack\_1\_001\_leaf1** with the text "Rollover an element to see info".
- Topology Label:** A dropdown menu currently set to "Name".

## 3D Topology View (Active)

**NOTE:** This feature has been classified as a Juniper Apstra Technology Preview feature. These features are "as is" and voluntary use. Juniper Support will attempt to resolve any issues that customers experience when using these features and create bug reports on behalf of support cases. However, Juniper may not provide comprehensive support services to Tech Preview features.

For additional information, refer to the ["Juniper Apstra Technology Previews" on page 1082](#) page or contact ["JuniperSupport" on page 777](#).

From the blueprint, navigate to **Active > Physical > Topology** and click **3D**.

- You can zoom in and out, move left and right, and reset to the default size and orientation.
- To show node name (and hostname as applicable) hover over an element.
- To show rack details, select a rack by either clicking its element or by selecting it from the **Selected Rack** drop-down list.
- To show node details, select a node by either clicking its element or by selecting it from the **Selected Node** drop-down list.

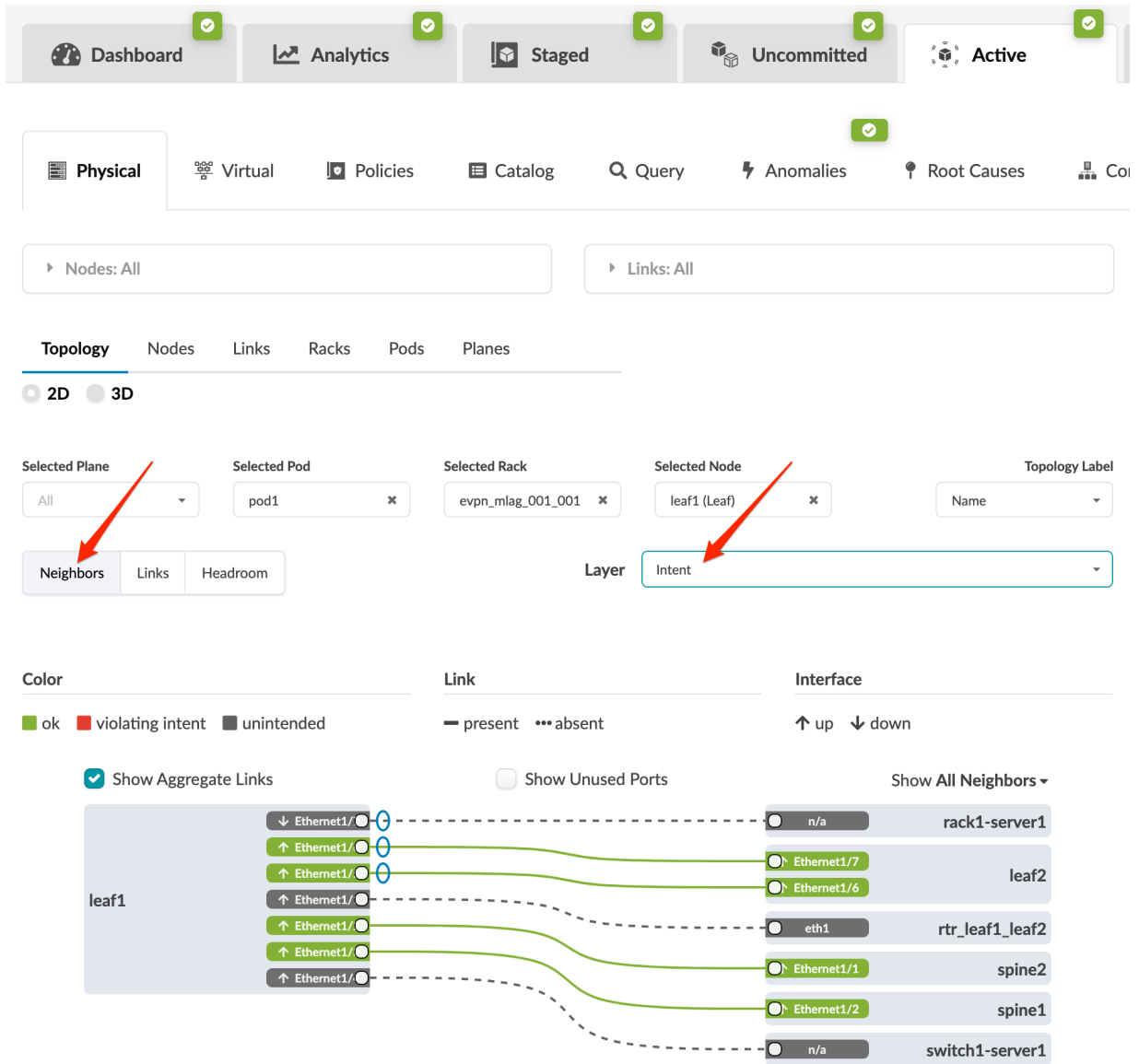
The screenshot displays the 'Active' view of a system monitoring interface. At the top, a navigation bar includes tabs for Dashboard, Analytics, Staged, Uncommitted, and Active. Below this, a secondary bar shows Physical, Virtual, Policies, Catalog, Query, Anomalies, Root Causes, and Cc. A dropdown menu for 'Nodes: All' and 'Links: All' is visible. The 'Topology' view is selected, showing a 3D diagram of nodes and links. A 'Layer' dropdown is set to 'Anomalies: All Services'. A legend indicates 'No Anomalies' (green) and 'Anomalies Present' (red). Below the diagram, 'Selected Rack' and 'Selected Node' dropdowns are shown. A 'Show Links?' checkbox is checked. A red arrow points to a node in the diagram, with a tooltip indicating 'Rollover an element to see info'. Another red arrow points to a node in the diagram, with a tooltip indicating '...click selection directly for more info'. A third red arrow points to the 'Selected Rack' dropdown, with a tooltip indicating 'Make selection from drop-down lists, or...'. A fourth red arrow points to the 'Selected Node' dropdown, with a tooltip indicating '...click selection directly for more info'.

### Neighbors View (Active)

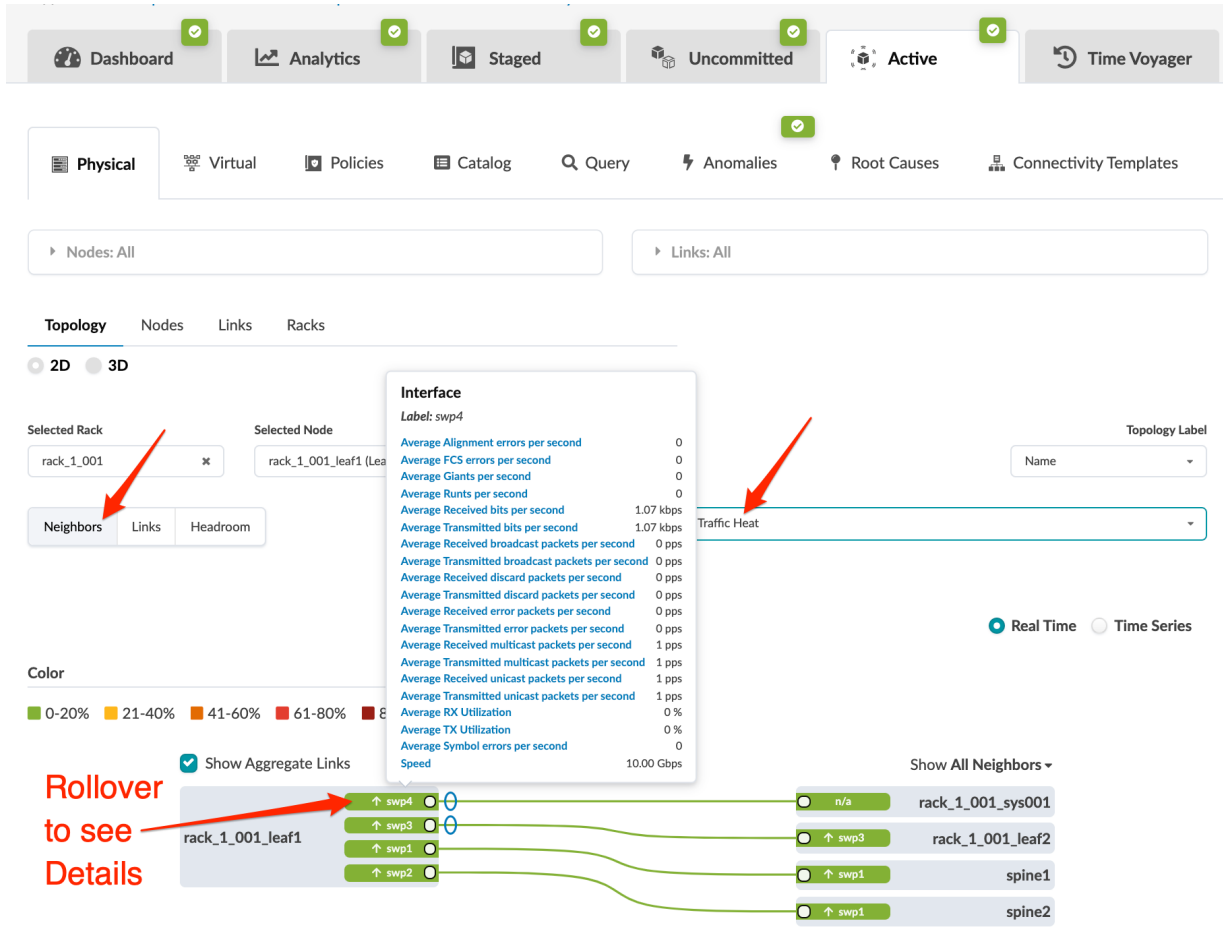
- To show aggregate links, click the **Show Aggregate Links** check box.
- To show unused ports, click the **Show Unused Ports** check box.
- To show a different label (name, hostname, S/N), select a different label from the **Topology Label** drop-down list (right side).
- To show a different layer, select a different layer from the **Layer** drop-down list.

- Choose to show all neighbors or only specific ones (generic, leaf, spine, and so on).

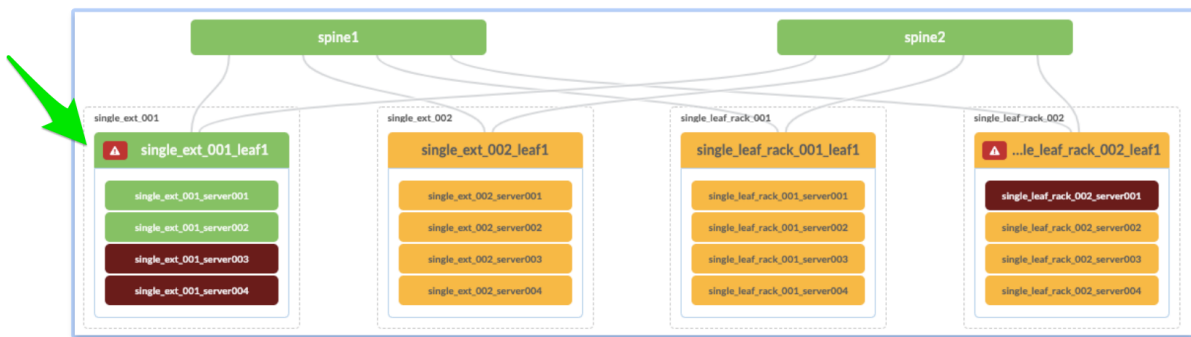
The intent layer is shown below.



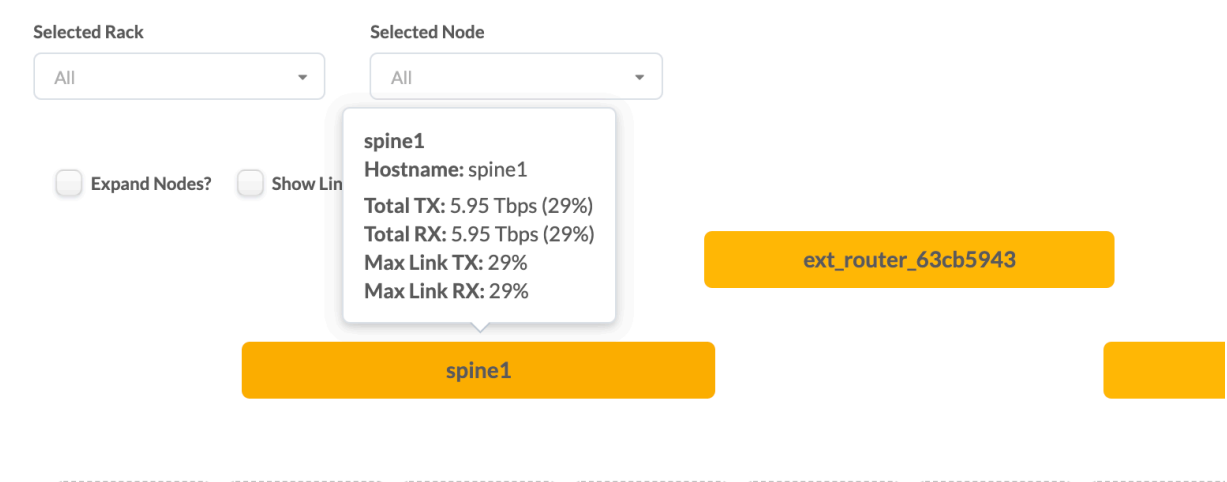
The traffic heat layer is shown below. The colors represent different available/used capacity based on the current system level TX/RX, averaged to 2 minutes, by default. If the aggregated TX or RX across all the device interfaces is < 20% it's **green**. If it's between 21-40%, it's **yellow** and so on. For each 20% difference, capacity is shown with a different color. (Server color is calculated based on the interface counters of the leaf ports facing that server. To see RX/TX per interface for a single node, click the node.



If any of a device's deployed ports are > 81% of its capacity in either RX or TX, a new "Alert" icon is shown on the device.



Mousing over a node shows exact aggregated values.



### Links View (Active Topology)

Dashboard

Analytics

Staged

Uncommitted

Active

Time View

Physical

Virtual

Policies

Catalog

Query

Anomalies

Root Causes

Connectivity Templates

Nodes: All

Links: All

Topology

Nodes

Links

Racks

Pods

Planes

2D

3D

Selected Plane: All

Selected Pod: pod1

Selected Rack: evpn\_mlag\_001\_001

Selected Node: leaf1 (Leaf)

Topology Label: Name

Neighbors

Links

Headroom

1-7 of 7

Page Size: 25

Filter selected by ☒ all ☐ selected only ☐ unselected only

Name	Role	Tags	Speed	Link label	Port Channel ID	Endpoint 1				Endpoint 2			
						Name	Role	Interface	Lag Mode	Name	Role	Interface	Lag Mode
leaf001_001_1<->leaf001_001_2(l3_peer_link)[1]	Leaf L3 Peer Link		10G	l3_peer_link	3	leaf1	Leaf	Ethernet1/5	N/A	leaf2	Leaf	Ethernet1/6	N/A
leaf001_001_1<->leaf001_001_2[1]	Leaf Peer Link		10G	N/A	2	leaf1	Leaf	Ethernet1/6	N/A	leaf2	Leaf	Ethernet1/7	N/A
spine001_001_1<->leaf001_001_1[1]	Spine to Leaf		10G	N/A	N/A	leaf1	Leaf	Ethernet1/3	N/A	spine1	Spine	Ethernet1/2	N/A

# Virtual Networks Endpoints (Active)

New in Apstra version 4.0.1.

Dashboard

Analytics

Staged

Uncommitted

Active

Physical

Virtual

Policies

Catalog

Query

Anomalies

Root Causes

Nodes: All

Links: All

Topology

Nodes

Links

Racks

Pods

Planes

2D

3D

Selected Plane

Selected Pod

Selected Rack

Selected Node

Topology Label

All

pod1

evpn\_single\_001\_001

switch3-server1 (Gen eric System)

Name

Neighbors

Links

Virtual Networks Endpoints

Headroom

Query: All

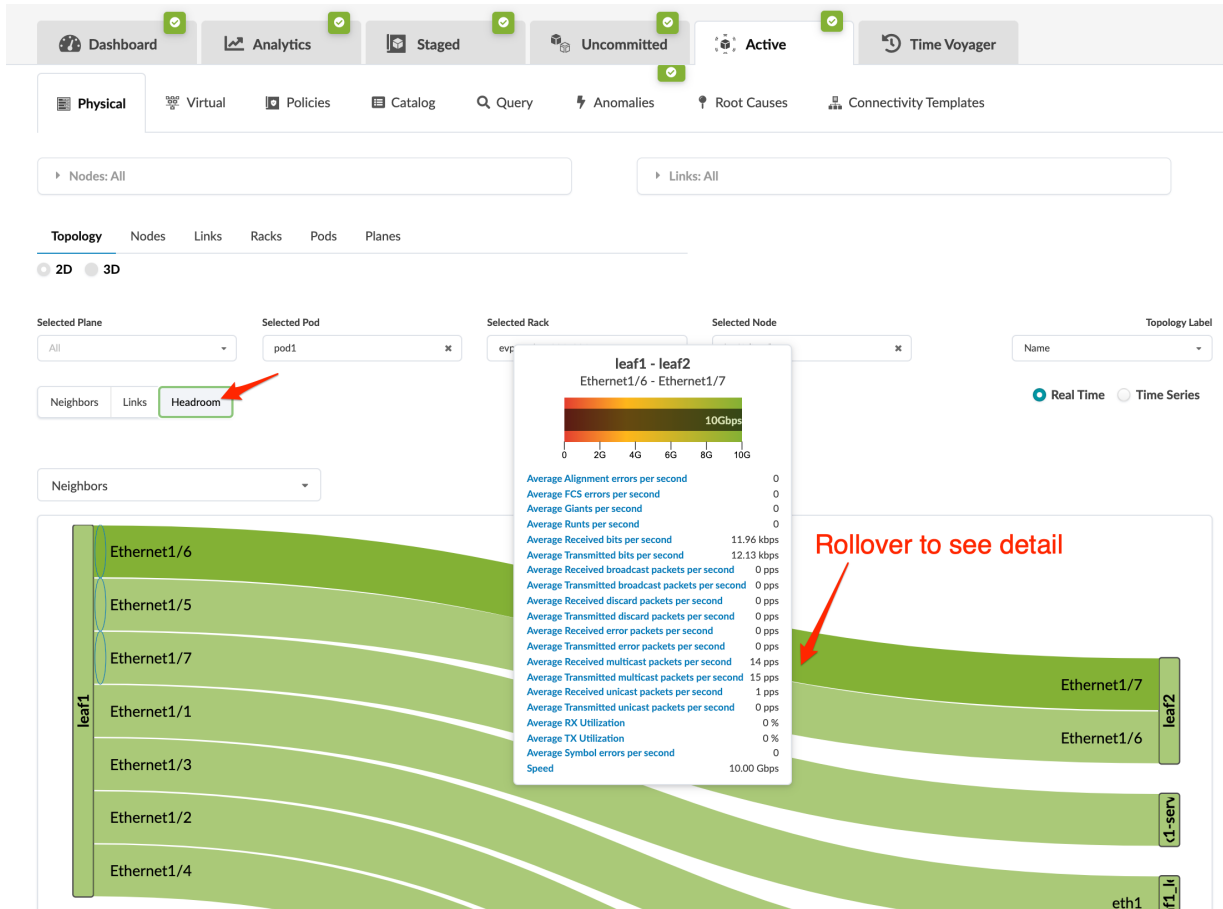
1-7 of 7

Page Size: 25

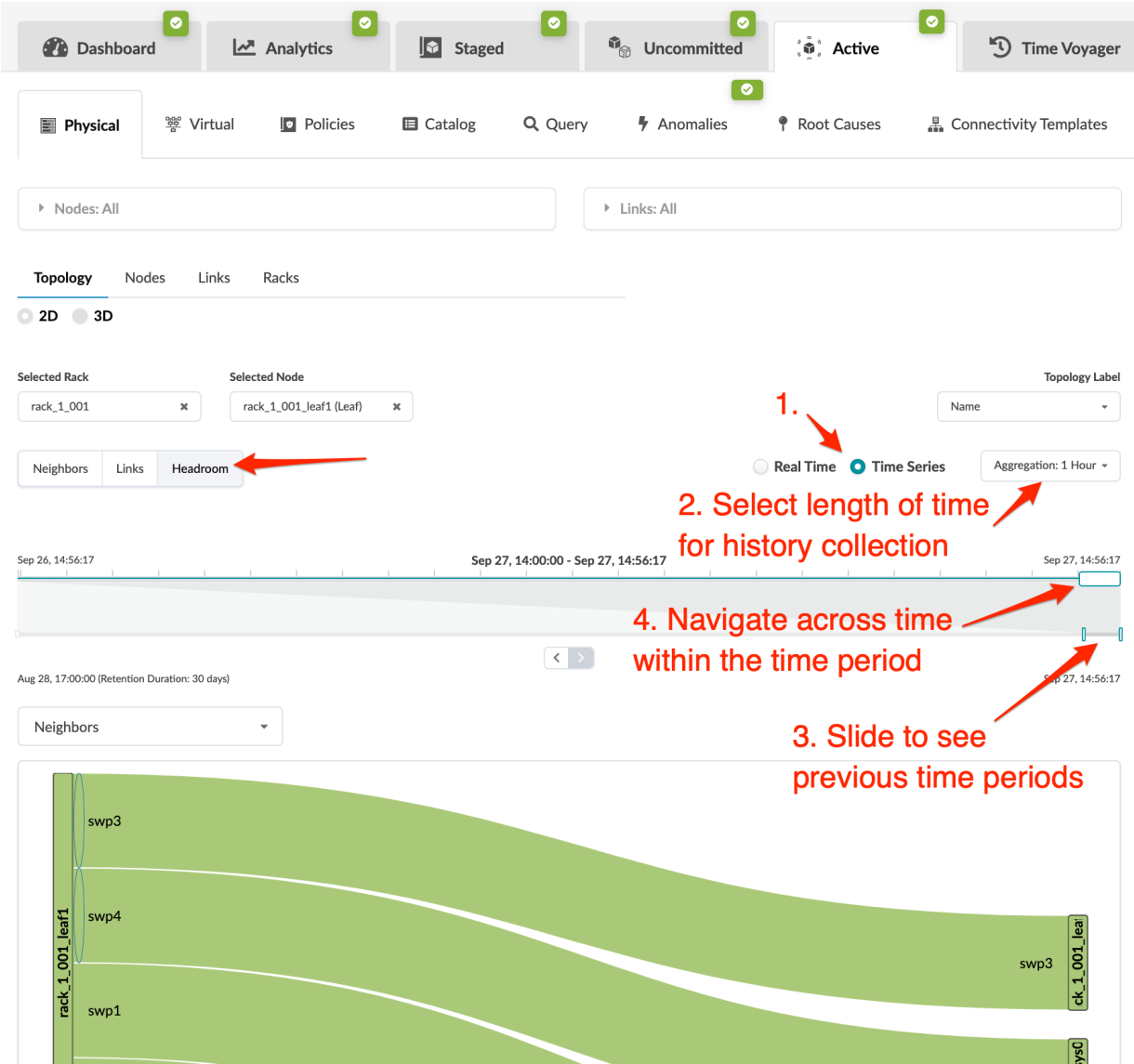
Virtual Network	Tag Type	Leaf(s)	Port Channel ID	Interface Name(s)
vlan_30_leaf3_v4	Untagged	leaf3	N/A	Ethernet1/3
blue_300_leaf3_v4	VLAN Tagged	leaf3	N/A	Ethernet1/3

## Headroom (Topology)

**NOTE:** To see the headroom view, the **Device Traffic probe** must be enabled. If you disable or delete the probe, the traffic heat layer in the active topology is not available. For more information, see ["Device Traffic probe" on page 928](#).



To view traffic history on top of the physical topology from the headroom view, select **Time Series**.



## Nodes (Active)

### IN THIS SECTION

- Active Nodes Overview | 666
- Apply Full Config | 666

## Active Nodes Overview

From the blueprint, navigate to **Active > Physical > Nodes** to go to nodes in the active topology. You can view nodes in table view or card view. In table view, you can select which details to display (from the drop-down list). To see additional details (such as telemetry, properties, and tags) for a specific node, select it, then the right panel displays tabs with more information.

The screenshot shows the 'Active' tab selected in the top navigation bar. Below it, the 'Physical' tab is selected in the sub-navigation bar. The main area displays a table of nodes. Annotations indicate the following steps:

1. Click on the 'Active' tab in the top navigation bar.
2. Click on the 'Physical' tab in the sub-navigation bar.
3. Click on the 'Nodes' tab in the sub-navigation bar.

Below the 'Nodes' tab, there are filters for 'Selected Plane' (superspine\_plane1), 'Selected Pod' (All), and 'Selected Rack' (All). A table of nodes is displayed with columns: Name, Tags, Role, External?, Deploy Mode, Device Profile, Hostname, and Deploy Status. A dropdown menu is open for 'Columns (10/20)', showing options: Name, Tags, Role, External?, and Pod. A red arrow points from the text 'Select what to display' to this dropdown. Another red arrow points from the text 'Select a name to see more details in right panel' to the 'sspine1' node in the table. On the right side, a panel for 'sspine1' is shown with tabs for Telemetry, Device, Properties, and Tags. The 'Telemetry' tab is active, showing a list of services: Probes, All Services, Liveness, Config, Interface, and Cabling. A red arrow points from the text 'Table Card' to the table view.

## Apply Full Config



**CAUTION:** Applying a full config is a disruptive operation and results in a temporary loss of service to the device. For information about when to apply a full config, see ["Anomalies - Configuration Deviation" on page 671](#).

1. From the blueprint, navigate to **Active > Physical > Nodes** and select the device.
2. From the selection panel (right-side) click **Device**, then click **Rendered**, **Incremental**, or **Pristine** to review the different configurations.
3. Click **Apply Full Config**.

## Links (Active)

### IN THIS SECTION

- [Active Links Overview | 667](#)
- [Export Cabling Map | 667](#)

### Active Links Overview

From the blueprint, navigate to **Active > Physical > Links** to go to nodes in the active topology. To search for specific nodes or links, click its query box, enter your criteria and click **Apply** to go to results. To go to properties of a particular link (in the right panel), click its name.

The screenshot shows the 'Active' tab selected in the top navigation bar. Below it, the 'Physical' tab is selected in the sub-navigation bar. The 'Links' tab is selected in the 'Topology' section. The 'Export cabling map' button is highlighted with a red arrow and labeled '2.'. The 'Links' tab is highlighted with a red arrow and labeled '3.'. The 'Properties' panel on the right shows the details for a selected link, with a red arrow pointing to the link name and labeled '1.'. The table below shows a list of links with columns for Name, Role, Speed, Tags, Endpoint 1, and Endpoint 2. A red arrow points to the link name 'leaf001\_001\_1<->leaf001\_001\_2[1]' and is labeled 'Click a name...'. Another red arrow points to the 'Properties' panel and is labeled '... to see properties'.

1. Click a name... ... to see properties

2. Export cabling map

3. Links

Name	Role	Speed	Tags	Endpoint 1	Endpoint 2						
Name	Role	Speed	Tags	Name	Role	Interface	IPv4	Name	Role	Interface	IPv4
leaf001_001_1<->leaf001_001_2[1]	Leaf Peer Link	10G		leaf1	Leaf	Ethernet1/6	N/A	leaf2	Leaf	Ethernet1/7	N/A
leaf001_001_1<->leaf001_001_2[1]	Leaf Peer Link	10G		leaf2	Leaf	Ethernet1/6	N/A	leaf1	Leaf	Ethernet1/5	N/A

### Export Cabling Map

1. From the blueprint, navigate to **Active > Physical > Links**, click the **Export cabling map** button and select **JSON** or **CSV**.
2. Click **Copy** to copy the contents or click **Save As File** to download the file.
3. When you've copied or downloaded the cabling map, close the dialog to return to the **Links** view.

**NOTE:** Cabling maps can also be exported from the **Staged >Physical >Links** view.

## Racks (Active)

### IN THIS SECTION

- [Change Rack Name](#) | 668

From the blueprint, navigate to **Active > Physical > Racks** to go to rack details in the active blueprint. You can change the default view from a table to a list. You can search for specific nodes or links and select a layer to see anomalies, deploy mode/status, traffic heat and more.

### Change Rack Name

You may want to use your own rack naming schema (for example, your rack names could be based on their physical locations). In these cases you can modify the existing rack names.

1. From the blueprint, navigate to **Active > Physical > Racks** and select the rack that you want to change.

The screenshot shows the network management interface with the following components:

- Top Navigation Bar:** Dashboard, Analytics, Staged, Uncommitted, **Active** (selected), Time Voyager.
- Left Sidebar:** Physical (selected), Virtual, Policies, Catalog, Query, Anomalies, Root Causes, Connectivity Templates.
- Main Content Area:**
  - Nodes:** All
  - Links:** All
  - Selection:** Status
  - Topology:** Nodes, Links, **Racks** (selected), Pods, Planes
  - Layer:** Anomalies: All Services
  - Legend:** No Anomalies (green square), Anomalies Present (red square)
  - Query:** All
  - Page Size:** 25
  - Rack Properties:**
    - Name:** evpn\_mlag\_001\_001 (with an edit button)
- Rack Details:**
  - evpn\_mlag\_001\_001 pod1:**
    - Generic Systems Capacity: 1-3 of 3
    - Query: All
    - Table with 3 columns: Name, Used, Available. Rows: dual-server (1/1), single-server-1 (1/1), single-server-2 (1/1).
  - evpn\_single\_001\_001 pod1:**
    - Generic Systems Capacity: 1-1 of 1
    - Query: All
    - Table with 3 columns: Name, Used, Available. Rows: single-server (1/4).

2. In **Rack Properties** (right panel selection) click the **Edit** button for the rack name.
3. Change the name and click the **Save** button to stage the change.

**NOTE:** You can also change rack names from the staged blueprint.

## Pods (Active)

From the blueprint, navigate to **Active > Physical > Pods** to see details about the deployed pods in the 5-stage Clos network. (You'll only see the pod view on "5-stage" on page 802 blueprints.) You can search for specific nodes or links and select a layer to see anomalies, deploy modes, deployment status and

more. Click a rack name in a pod to see its rack type preview.

The screenshot shows the 'Active' blueprint in the network management interface. The top navigation bar includes 'Dashboard', 'Analytics', 'Staged', 'Uncommitted', 'Active', and 'Time Voyager'. The left sidebar has 'Physical', 'Virtual', 'Policies', 'Catalog', 'Query', 'Anomalies', 'Root Causes', and 'Connectivity Templates'. The main area shows 'Nodes: All' and 'Links: All' filters. The 'Pods' tab is selected, showing a table of pods. The 'Layer' dropdown is set to 'Anomalies: All Services'. The 'pod1' and 'pod2' details are shown below.

Annotations:

- 1. Arrow pointing to the 'Active' blueprint tab.
- 2. Arrow pointing to the 'Physical' tab in the left sidebar.
- 3. Arrow pointing to the 'Pods' tab in the main area.
- Text: 'Select layer to see anomalies, deployment status and more' with an arrow pointing to the 'Layer' dropdown.
- Text: 'Click rack type name to see preview' with an arrow pointing to the 'Type' column in the pod1 table.

pod1 Capacity: Query: All 1-5 of 12

Name	Type	Used	Available
evpn-mlag	global	0	1
evpn-mlag	embedded	1	1
evpn-single	global	0	2
evpn-single	embedded	1	2
L2 MLAG 1x access	global	0	1

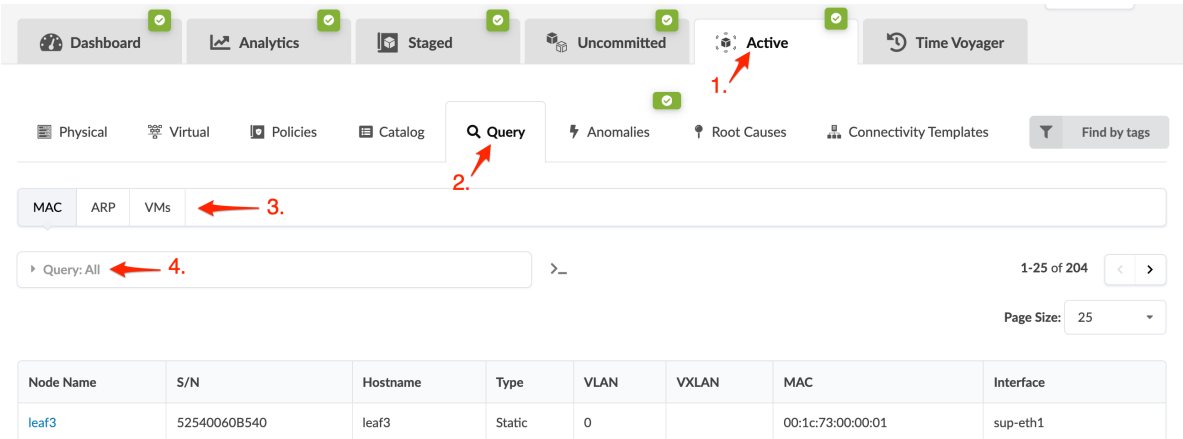
pod2 Capacity: Query: All 1-5 of 14

Name	Type	Used	Available
evpn-mlag	global	0	1
evpn-mlag	embedded	0	1
evpn-single	global	0	3
evpn-single	embedded	2	3
L2 ESI 2x Links	global	0	1

## Query

You can search for MAC addresses, IP addresses and VMs by using the query feature in the active blueprint.

1. From the blueprint, navigate to **Active > Query**.



2. Click **MAC**, **ARP**, or **VMs** depending on your query.

3. Click **Query:All**, enter search criteria, and click **Apply** to see results.

## Anomalies (Service)

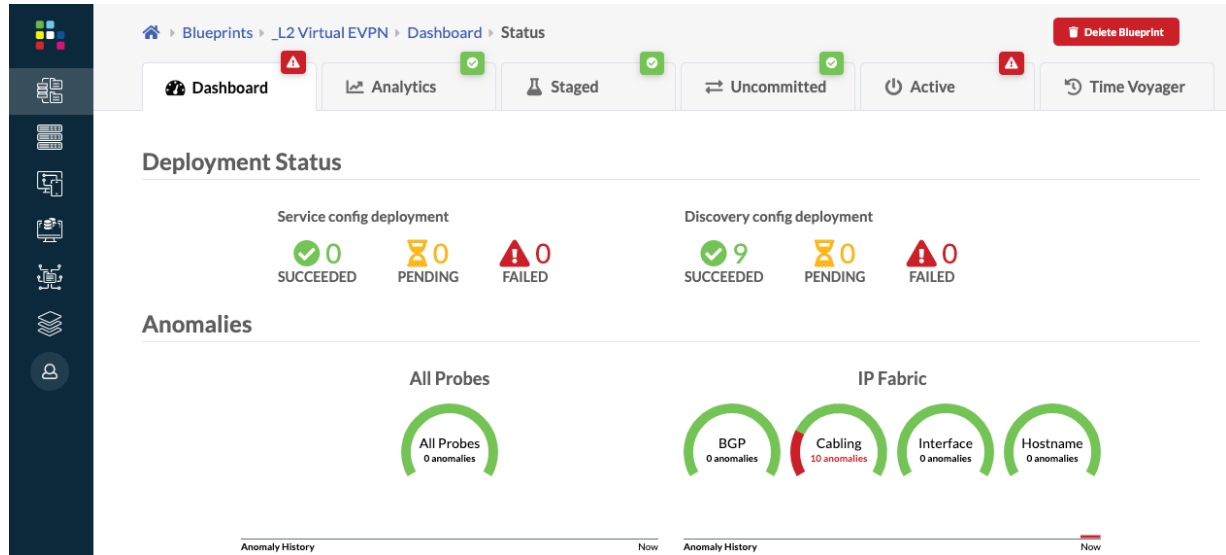
### IN THIS SECTION

- [Discovery Anomalies | 672](#)
- [Configuration Deviation | 676](#)

This section covers service anomalies. For analytics anomalies see ["IBA Anomalies." on page 407](#)

## Discovery Anomalies

To demonstrate anomalies during the discovery phase, cabling errors have been deliberately configured in the example below to trigger alarms.



To see the list of the cabling anomalies, click the **Cabling** gauge on the dashboard.

The screenshot shows the OLS dashboard for the blueprint "\_L2 Virtual EVPN" with the "Anomalies" tab selected. The main content area displays a list of anomalies under the "Cabling" category. The table below shows the details of these anomalies.

Node	Hostname	Service	Anomaly Type	Role	Anomaly Extra Details	Expected	Actual	Time Updated
spine1	spine1	IP Fabric	cabling	Spine to Leaf	Property: Value Interface: "evp3"	Property: Value neighbor interface: "evp1" neighbor name: "12-virtual-evpn-001-leaf2"	Property: Value neighbor interface: "Ethernet1/1" neighbor name: "12-virtual-evpn-002-leaf1"	15 minutes ago
_l2_virtual_evpn_001_leaf2	l2-virtual-evpn-001-leaf2	IP Fabric	cabling	Spine to Leaf	Property: Value Interface: "evp1"	Property: Value neighbor interface: "evp3" neighbor name: "spine1"	Property: Value neighbor interface: "evp2" neighbor name: "spine1"	15 minutes ago
_l2_virtual_evpn_002_leaf1	l2-virtual-evpn-002-leaf1	IP Fabric	cabling	Spine to Leaf	Property: Value Interface: "Ethernet1/1"	Property: Value neighbor interface: "evp1" neighbor name: "spine1"	Property: Value neighbor interface: "evp3" neighbor name: "spine1"	15 minutes ago
spine1	spine1	IP Fabric	cabling	Spine to Leaf	Property: Value Interface: "evp5"	Property: Value neighbor interface: "Ethernet1" neighbor name: "12-virtual-evpn-003-leaf1"	Property: Value neighbor interface: "Ethernet1" neighbor name: "12-virtual-evpn-003-leaf2"	15 minutes ago
spine1	spine1	IP Fabric	cabling	Spine to Leaf	Property: Value Interface: "evp5"	Property: Value neighbor interface: "Ethernet1"	Property: Value neighbor interface: "Ethernet1"	15 minutes ago

To see the anomalies in the topology view, click **Active**.

The screenshot displays the OVS network management interface. The top navigation bar includes tabs for Dashboard, Analytics, Staged, Uncommitted, **Active** (highlighted), and Time Voyager. The breadcrumb trail shows: Blueprints > \_I2 Virtual EVPN > Active > Physical > Status.

Below the navigation bar, the **Physical** tab is selected. The interface shows filters for Nodes (All) and Links (All). The **Topology** section includes tabs for Nodes, Links, and Racks. The **Layer** dropdown is set to "Anomalies: All Services". A legend indicates "No Anomalies" (green) and "Anomalies Present" (red).

The topology diagram shows a central **router1** node connected to three spine nodes: **spine1** (red, anomalies present), **spine2** (green, no anomalies), and **spine3** (green, no anomalies). Each spine node is connected to a corresponding virtual EVPN node (e.g., \_I2\_virtual\_evpn\_001, \_I2\_virtual\_evpn\_002, \_I2\_virtual\_evpn\_003). These virtual nodes contain leaf nodes (e.g., \_I2\_virtual\_evpn\_001\_leaf1, \_I2\_virtual\_evpn\_001\_leaf2) and server nodes (e.g., \_I2\_virtual\_evpn\_001\_server001).

On the right side, a panel titled "Anomalies: All Services" lists various anomaly categories with their counts:

- Anomalies: All Services: 10
- Anomalies: BGP: 0
- Anomalies: Cabling: 10
- Anomalies: Config: 0
- Anomalies: Hostname: 0
- Anomalies: Interface: 0
- Anomalies: LAG: 0
- Anomalies: Liveness: 0
- Anomalies: MLAG: 0
- Anomalies: Probes: 0
- Anomalies: Route: 0
- Deploy Mode: 0/9/0
- Deployment Status: Discovery: 9/0/0
- Deployment Status: Service: 0/0/0

To see the topology view of the anomalies affecting spine1, click **Spine1** in the topology.

The screenshot displays the network management interface for spine1. The top navigation bar includes tabs for Dashboard, Analytics, Staged, Uncommitted, Active, and Time Voyager. The main area shows a physical topology view with nodes and links. A legend indicates 'No Anomalies' (green) and 'Anomalies Present' (red). A right panel shows a list of services with status indicators: Probes (green), All Services (red with 5), Liveness (green), Config (green), Interface (green), Cabling (red with 5), BGP (green), Route (green), and Hostname (green).

You can see the cabling violations on spine1. In the right panel, click the red status indicator for **All Services** to see a comparison of expectations vs. actual. If other anomalies existed in addition to the

cabling anomalies, they would be shown in this list as well.

Blueprints > \_L2 Virtual EVPN > System Nodes > spine1 > Active > Telemetry > Anomalies

Staged

Active

Physical

Virtual

Telemetry

Anomalies

Config

Interface

MAC

LLDP

BGP

Route

Hostname

Counters

ARP

Transceivers

Utilization

Query: All

1-5 of 5

Page Size: 25

Service	Anomaly Type	Role	Anomaly Extra Details	Expected	Actual	Time Updated																
IP Fabric	cabling	Spine to Leaf	<table><tr><td>Property</td><td>Value</td></tr><tr><td>Interface</td><td>"svp3"</td></tr></table>	Property	Value	Interface	"svp3"	<table><tr><td>Property</td><td>Value</td></tr><tr><td>neighbor interface</td><td>"svp1"</td></tr><tr><td>neighbor name</td><td>"12-virtual-evpn-001-leaf2"</td></tr></table>	Property	Value	neighbor interface	"svp1"	neighbor name	"12-virtual-evpn-001-leaf2"	<table><tr><td>Property</td><td>Value</td></tr><tr><td>neighbor interface</td><td>"Ethernet1/1"</td></tr><tr><td>neighbor name</td><td>"12-virtual-evpn-002-leaf1"</td></tr></table>	Property	Value	neighbor interface	"Ethernet1/1"	neighbor name	"12-virtual-evpn-002-leaf1"	4 hours ago
Property	Value																					
Interface	"svp3"																					
Property	Value																					
neighbor interface	"svp1"																					
neighbor name	"12-virtual-evpn-001-leaf2"																					
Property	Value																					
neighbor interface	"Ethernet1/1"																					
neighbor name	"12-virtual-evpn-002-leaf1"																					
IP Fabric	cabling	Spine to Leaf	<table><tr><td>Property</td><td>Value</td></tr><tr><td>Interface</td><td>"svp6"</td></tr></table>	Property	Value	Interface	"svp6"	<table><tr><td>Property</td><td>Value</td></tr><tr><td>neighbor interface</td><td>"Ethernet1"</td></tr><tr><td>neighbor name</td><td>"12-virtual-evpn-003-leaf1"</td></tr></table>	Property	Value	neighbor interface	"Ethernet1"	neighbor name	"12-virtual-evpn-003-leaf1"	<table><tr><td>Property</td><td>Value</td></tr><tr><td>neighbor interface</td><td>"Ethernet1"</td></tr><tr><td>neighbor name</td><td>"12-virtual-evpn-003-leaf2"</td></tr></table>	Property	Value	neighbor interface	"Ethernet1"	neighbor name	"12-virtual-evpn-003-leaf2"	4 hours ago
Property	Value																					
Interface	"svp6"																					
Property	Value																					
neighbor interface	"Ethernet1"																					
neighbor name	"12-virtual-evpn-003-leaf1"																					
Property	Value																					
neighbor interface	"Ethernet1"																					
neighbor name	"12-virtual-evpn-003-leaf2"																					
IP Fabric	cabling	Spine to Leaf	<table><tr><td>Property</td><td>Value</td></tr><tr><td>Interface</td><td>"svp5"</td></tr></table>	Property	Value	Interface	"svp5"	<table><tr><td>Property</td><td>Value</td></tr><tr><td>neighbor interface</td><td>"Ethernet1"</td></tr><tr><td>neighbor name</td><td>"12-virtual-evpn-003-leaf2"</td></tr></table>	Property	Value	neighbor interface	"Ethernet1"	neighbor name	"12-virtual-evpn-003-leaf2"	<table><tr><td>Property</td><td>Value</td></tr><tr><td>neighbor interface</td><td>"Ethernet1"</td></tr><tr><td>neighbor name</td><td>"12-virtual-evpn-003-leaf1"</td></tr></table>	Property	Value	neighbor interface	"Ethernet1"	neighbor name	"12-virtual-evpn-003-leaf1"	4 hours ago
Property	Value																					
Interface	"svp5"																					
Property	Value																					
neighbor interface	"Ethernet1"																					
neighbor name	"12-virtual-evpn-003-leaf2"																					
Property	Value																					
neighbor interface	"Ethernet1"																					
neighbor name	"12-virtual-evpn-003-leaf1"																					
IP Fabric	cabling	Spine to Leaf	<table><tr><td>Property</td><td>Value</td></tr><tr><td>Interface</td><td>"svp1"</td></tr></table>	Property	Value	Interface	"svp1"	<table><tr><td>Property</td><td>Value</td></tr><tr><td>neighbor interface</td><td>"Ethernet1/1"</td></tr><tr><td>neighbor name</td><td>"12-virtual-evpn-002-leaf1"</td></tr></table>	Property	Value	neighbor interface	"Ethernet1/1"	neighbor name	"12-virtual-evpn-002-leaf1"	<table><tr><td>Property</td><td>Value</td></tr><tr><td>neighbor interface</td><td>"svp1"</td></tr><tr><td>neighbor name</td><td>"12-virtual-evpn-001-leaf1"</td></tr></table>	Property	Value	neighbor interface	"svp1"	neighbor name	"12-virtual-evpn-001-leaf1"	4 hours ago
Property	Value																					
Interface	"svp1"																					
Property	Value																					
neighbor interface	"Ethernet1/1"																					
neighbor name	"12-virtual-evpn-002-leaf1"																					
Property	Value																					
neighbor interface	"svp1"																					
neighbor name	"12-virtual-evpn-001-leaf1"																					
IP Fabric	cabling	Spine to Leaf	<table><tr><td>Property</td><td>Value</td></tr><tr><td>Interface</td><td>"svp2"</td></tr></table>	Property	Value	Interface	"svp2"	<table><tr><td>Property</td><td>Value</td></tr><tr><td>neighbor interface</td><td>"svp1"</td></tr><tr><td>neighbor name</td><td>"12-virtual-evpn-001-leaf1"</td></tr></table>	Property	Value	neighbor interface	"svp1"	neighbor name	"12-virtual-evpn-001-leaf1"	<table><tr><td>Property</td><td>Value</td></tr><tr><td>neighbor interface</td><td>"svp1"</td></tr><tr><td>neighbor name</td><td>"12-virtual-evpn-001-leaf2"</td></tr></table>	Property	Value	neighbor interface	"svp1"	neighbor name	"12-virtual-evpn-001-leaf2"	4 hours ago
Property	Value																					
Interface	"svp2"																					
Property	Value																					
neighbor interface	"svp1"																					
neighbor name	"12-virtual-evpn-001-leaf1"																					
Property	Value																					
neighbor interface	"svp1"																					
neighbor name	"12-virtual-evpn-001-leaf2"																					

To see additional details specific to LLDP only, click LLDP.

**Staged** **Active**

Physical Virtual **Telemetry**

Anomalies Config Interface MAC **LLDP** BGP Route Hostname Counters ARP Transceivers Utilization

1-6 of 6 Page Size: 25

	Expected		Actual					
Interface ↕	Neighbor node ↕	Neighbor interface ↕	Neighbor node ↕	Neighbor interface ↕	Intent status ↕	Neighbor system	Last fetched ↕	Last modified ↕
swp1	I2-virtual-evpn-002-leaf1	Ethernet1/1	I2-virtual-evpn-001-leaf1	swp1	mismatch	Cumulus Linux version 3.7.11 running on QEMU Standard PC (i440FX + PIIX, 1996)	a few seconds ago	5 hours ago
swp2	I2-virtual-evpn-001-leaf1	swp1	I2-virtual-evpn-001-leaf2	swp1	mismatch	Cumulus Linux version 3.7.11 running on QEMU Standard PC (i440FX + PIIX, 1996)	a few seconds ago	5 hours ago
swp3	I2-virtual-evpn-001-leaf2	swp1	I2-virtual-evpn-002-leaf1	Ethernet1/1	mismatch	Cisco Nexus Operating System (NX-OS) Software 9.2(2) TAC support: http://www.cisco.com/tac Copyright (c) 2002-2018, Cisco Systems, Inc. All rights reserved.	a few seconds ago	4 hours ago
swp4	I2-virtual-evpn-002-leaf2	Ethernet1/1	I2-virtual-evpn-002-leaf2	Ethernet1/1	ok	Cisco Nexus Operating System (NX-OS) Software 9.2(2) TAC support: http://www.cisco.com/tac Copyright (c) 2002-2018, Cisco Systems, Inc. All rights reserved.	a few seconds ago	5 hours ago
swp5	I2-virtual-evpn-003-leaf2	Ethernet1	I2-virtual-evpn-003-leaf1	Ethernet1	mismatch		a few seconds ago	5 hours ago
swp6	I2-virtual-evpn-003-leaf1	Ethernet1	I2-virtual-evpn-003-leaf2	Ethernet1	mismatch		a few seconds ago	5 hours ago

To see how to resolve these cabling issues, see ["Fetching Discovered LLDP Data"](#) on page 473.

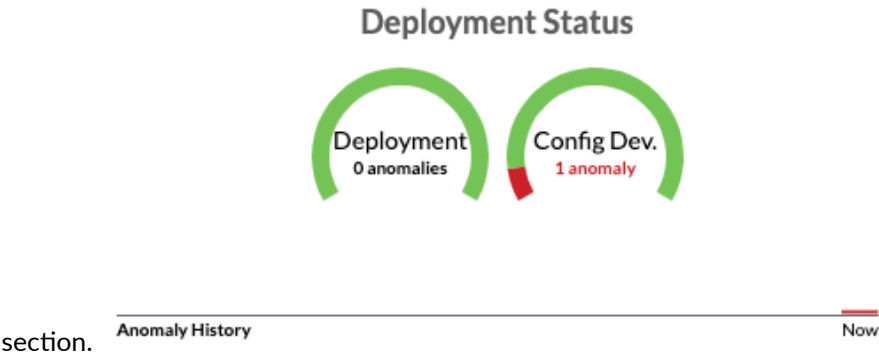
## Configuration Deviation

## IN THIS SECTION

- Config Deviation and Configlets | 679

Running configurations on devices are continuously compared with the ["Golden Config" on page 140](#). If a config deviation is found, a configuration anomaly is raised. Typically such deviations are seen when changes were made outside of Apstra (by using the device CLI), or attempting to deploy configuration on a switch that is not able to take the change. These anomalies remain active until either the anomalous configuration is removed from the device or the anomaly is suppressed.

1. From the blueprint dashboard, any configuration deviations are displayed in the **Deployment Status**



2. Click **Config Dev.** to see the list of node(s) with anomalies.

The screenshot shows the 'Anomalies' page in the dashboard. The breadcrumb trail is 'Blueprints > L2V > Active > Anomalies'. There are tabs for 'Dashboard', 'Analytics', 'Staged', 'Uncommitted', and 'Active'. The 'Active' tab is selected. Below the tabs, there are filters for 'Physical', 'Virtual', 'Policies', 'Settings', 'Query', 'Anomalies', and 'Root Causes'. A query is entered: 'Service = Deployment Status and Anomaly Type = config'. The results are shown in a table with columns: Node, Hostname, Service, Anomaly Type, Role, Anomaly Extra Details, Expected, Actual, and Time Updated. The first row is highlighted with a green box, showing the node 'l2\_virtual\_003\_leaf1'.

Node	Hostname	Service	Anomaly Type	Role	Anomaly Extra Details	Expected	Actual	Time Updated
l2_virtual_003_leaf1	l2-virtual-003-leaf1	Deployment Status	config		-	Property: config, Value: Show in modal	Property: config, Value: Show in modal	20 minutes ago

3. Click a node name to see the device telemetry page, then click **Config** to see a side-by-side comparison of the actual config to the golden config. (The difference is not shown in the image below.)

The screenshot shows the 'Config' page for the node 'l2\_virtual\_003\_leaf1'. The breadcrumb trail is 'Blueprints > L2V > System Nodes > l2\_virtual\_003\_leaf1 > Active > Telemetry > Config'. There are tabs for 'Staged' and 'Active'. The 'Active' tab is selected. Below the tabs, there are filters for 'Physical', 'Virtual', and 'Telemetry'. A 'Config' tab is highlighted with a green box. Below the filters, there are buttons for 'Apply Full Config' and 'Accept Changes'. A red warning message states 'Actual config deviated from golden config'. Below the message, there is a table with two columns: 'Intended running configuration' and 'Actual running configuration'. Both columns show the same configuration commands.

Intended running configuration	Actual running configuration
1 ! Command: show running-config	1 ! Command: show running-config
2 ! device: l2-virtual-003-leaf1 (vEOS, EOS-4.22.3M)	2 ! device: l2-virtual-003-leaf1 (vEOS, EOS-4.22.3M)
3 !	3 !
4 ! boot system flash:./boot-image.swi	4 ! boot system flash:./boot-image.swi
5 !	5 !
6 daemon AosEosProxySdkAgent	6 daemon AosEosProxySdkAgent

4. To keep the configuration difference, click **Accept Changes**. This **suppresses** the configuration anomaly, and does not affect "Intended" or Apstra-rendered config. the primary purpose of "Accept Changes" is to mitigate *cosmetic* configuration anomalies.

**NOTE:** Out-of-band (OOB) changes to the fabric are not supported. Do not **Accept Changes** to attempt to add OOB changes. For custom changes, use ["configlets" on page 119](#).



**CAUTION:**

- Depending on the change, Apstra may overwrite out-of-band changes. There is no way to avoid this. As such, always avoid OOB changes in the Apstra environment.
- Using *Accept Changes* does **not** make the OOB change persistent. In the event of a full config push or Apstra writing to the same config, all OOB changes are discarded.



**CAUTION:** Do not use **Accept Changes** on Cumulus Linux, unless suppressing a cosmetic anomaly. On Cumulus, Apstra maintains device configuration using updates to various files, contents of which are overwritten. Any blueprint commit requiring a change to a file that has out-of-band changes results in those changes being discarded.

5. To make the actual configuration conform to the intended configuration, click **Apply Full Config**, then click **Confirm**. Applying the full config erases the device's current (unintended) configuration before re-applying the complete intended configuration. A full configuration push does not include any OOB changes, and therefore erases them, regardless of their "Accepted" state.

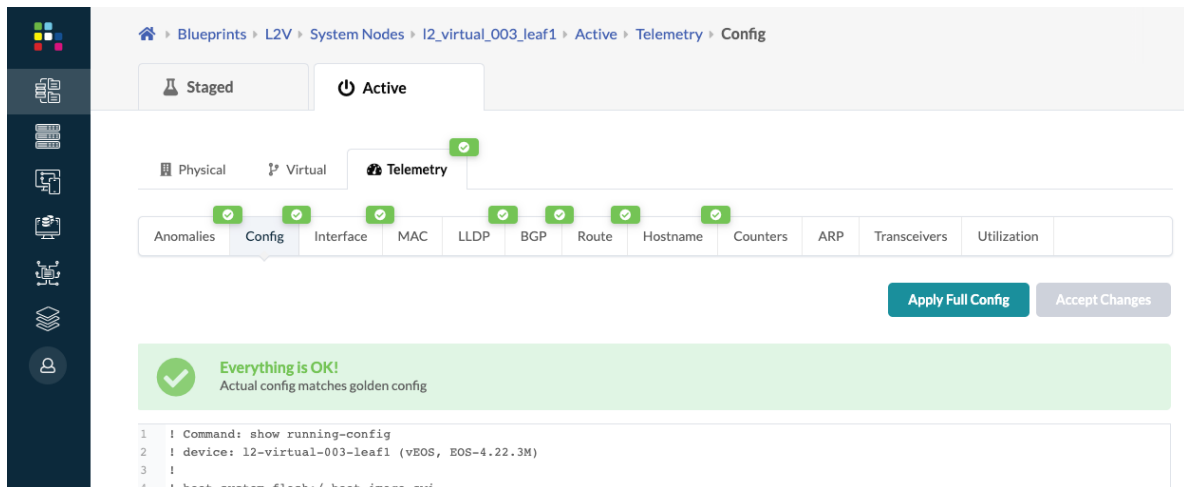


**CAUTION:** Applying a full config is a disruptive operation and results in a temporary loss of service to the device.



**CAUTION:** Never directly modify any Apstra-rendered config that affects routing and connectivity. Doing so can potentially impact the network's operation. When in doubt, contact ["Juniper Support" on page 777](#).

- After resolving the config deviation anomaly (either by accepting changes or applying a full config) the actual config matches the golden config and the anomaly is cleared.



## Config Deviation and Configlets

If an improperly-configured configlet causes Apstra deployment errors (when the command is rejected by the device), a **service config deployment** failure occurs. In this case, follow the steps below to resolve the anomaly.

- From the blueprint, navigate to **Staged > Catalog > Configlets** and delete the configlet.
- Click **Uncommitted** and commit the change. The configuration deviation remains because the golden config is empty. The golden config is the running config of the device after *successful* deployment of Apstra-rendered config. If deployment fails there is no golden config, thus causing the config deviation.
- Click **Dashboard**, then click **Config Dev.** (in the **Deployment Status** section).
- Click the node name, then select **Accept Changes** to notify Apstra that the failure can be ignored.

## Root Causes

### IN THIS SECTION

- [Root Cause Overview | 680](#)
- [Enable Root Cause Analysis | 680](#)
- [View Root Cause Analysis | 681](#)

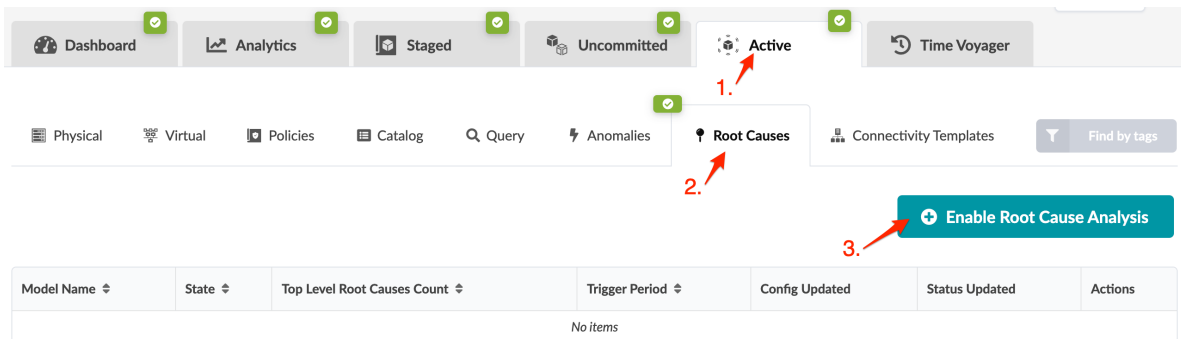
## Root Cause Overview

Root Cause Identification (RCI) is a technology integrated into Apstra software that automatically determines root causes of complex network issues. RCI leverages the Apstra datastore for realtime network status, and automatically correlates telemetry with each active blueprint intent. Root cause use cases include the following:

Root Cause	Description
Link broken	Symptoms: Both interfaces are operationally down, LLDP is missing on both sides, BGP peered across that link is operationally down.
Link miscabled	Symptoms: LLDP indicates wrong neighbors, BGP peered across that link is operationally down.
Operator shut interface	Symptoms: Both interfaces on the link are operationally down; the interface in question is administratively down; LLDP missing on both sides, BGP peered across that link is operationally down.
Disconnection between 2 devices	<p>Symptoms: Union of symptoms for link broken / link miscabled / operator shut interface for all constituent links between a spine and a leaf</p> <p>For instance, if there are 3 links between a spine and a leaf, then 2 could be miscabled and 1 is broken - this results in a disconnection between that spine and that leaf.</p>

## Enable Root Cause Analysis

1. From the blueprint, navigate to **Active > Root Causes** and click **Enable Root Cause Analysis**.



2. Enter a **Trigger Period** or leave the default, and click **Create** to enable root cause analysis and return to the list view.

## View Root Cause Analysis

From the blueprint, navigate to **Active > Root Causes** and click the model name **connectivity** in the list.

DashboardAnalyticsStagedUncommittedActiveTime Voyager

PhysicalVirtualPoliciesCatalogQueryAnomaliesRoot CausesConnectivity TemplatesFind by tags

Enable Root Cause Analysis

Model Name	State	Top Level Root Causes Count	Trigger Period	Config Updated	Status Updated	Actions
connectivity	OPERATIONAL	0	30s	a minute ago	a few seconds ago	

Root cause analysis runs periodically and produces zero or more root causes. Any root causes that are found include a description, a timestamp of when it was detected and a list of symptoms.

DashboardAnalyticsStagedUncommittedActive

PhysicalVirtualPoliciesCatalogQueryAnomaliesRoot CausesCc

Back to list

Configuration

Model Name	connectivity
State	OPERATIONAL
Trigger Period	30s
Config Updated	an hour ago
States Updated	a few seconds ago

Root Causes

✓

No Root Causes Found

# Time Voyager (Blueprints)

## IN THIS SECTION

- [Time Voyager Overview | 682](#)
- [Jump to Previous Blueprint Revision | 684](#)
- [Keep Saved Blueprint Revision | 685](#)
- [Update Blueprint Revision Description | 685](#)
- [Delete Kept Blueprint Revision | 685](#)

## Time Voyager Overview

When you commit a staged blueprint, thereby deploying updates to the network, you may find that the result is not what you expected. Or maybe you've committed changes to a blueprint by mistake and you want to undo those changes. Another scenario may be that you've decided to return the network to the state it was in several revisions ago. Depending on the level of complexity, manually staging and committing changes to undo what you've done can be difficult and error-prone. In these cases you'll want to use Time Voyager to automatically restore previous revisions of a blueprint.

A blueprint can be jumped back to any retained revision. The five (5) most recent blueprint commits are retained. When you commit a sixth time, the first revision is discarded, and the sixth revision becomes the fifth, the second revision becomes the first, and so on as additional blueprint changes are committed. You can retain a particular revision indefinitely by *keeping* it. When you keep a revision it is not included in the five revisions that cycle out. You can keep up to twenty-five (25) revisions, effectively having thirty (30) blueprint revisions to choose from. Keep in mind that each revision requires storage space. If you decide that you no longer want to keep a revision you can simply delete it.

When committing a blueprint you can add a revision description to help identify the changes made in that revision. These descriptions are displayed in the revision history section of the blueprint as long as that revision is retained. If you don't add a description when you commit you can always add one later. When jumping to a revision, this description helps you choose the correct one. Specific differences between revisions are not displayed, so the description is the only change information available for that revision.

When jumping to a revision, any previously staged changes that have not been committed are discarded. If this is an issue, do not jump until you've addressed the uncommitted changes.

Time Voyager is not just an UNDO function. When using Time Voyager you roll back to a previous commit. This means that anything deleted on the last commit is re-applied when rolling back. There can be many changes in-between revisions, both additions and removals, all of which would be included in the rollback. It is important to do a detailed review of changes before committing a rollback. Therefore Time Voyager is better compared with a Revision Control System (for the whole network!) than an UNDO function.

#### Unsupported Time Voyager Scenarios

- After you've upgraded Apstra server, you cannot jump to a blueprint with an older version because the blueprint revision history is discarded on upgrade. If you need to return to a previous Apstra version that was taken prior to upgrading Apstra, refer to ["Restore Database" on page 41](#). This method could cause issues from a device config standpoint.
- It's not supported when the Pristine config has changed between revisions.
- It's not supported when the NOS versions are different between revisions. You could downgrade the NOS version to the same version using the device manager, then jump to a previous revision.
- Devices that were allocated in a previous revision that are no longer available result in the build error *system ID does not exist*. (Conversely, *adding* a device and jumping to a previous revision without that device *will* be successful. The added device will be removed.)
- Resources that were assigned in a previous revision that have been reassigned cause the build error *resource already in use*. To resolve the build error, you must manually assign resources to each member in that group or reset the resource group overrides. (Jumping to a previous revision after a previously assigned global resource pool is modified *may* be successful, but it could cause an intent violation.)
- It's not supported if manual device config changes have been accepted.
- It's not supported in any other cases where the resulting device config state is different.

**NOTE:** Why not use Apstra server backup/restore to jump to a previous revision? Time Voyager maintains synchronized configuration between the Apstra server and devices (as much as possible); Apstra backup/restore does not. Effectively, the Apstra backup/restore is an out-of-band change from a device configuration standpoint. If a backup is restored, you would need to push a full config to make sure the device configuration reflects what you restored from the database backup. This would most likely be disruptive.

From the blueprint, click **Time Voyager** to go to the retained blueprint revisions. The first revision in the list is the active one. Successive revisions are ordered by date from most recent to oldest.

The screenshot shows the 'Time Voyager' interface with a navigation bar at the top containing tabs: Dashboard, Analytics, Staged, Uncommitted, Active, and Time Voyager. Below the navigation bar is a 'Revisions' section with a search bar (Query: All) and a pagination control (1-8 of 8, Page Size: 25). The main content is a table of revisions.

Description	Created At	User	Actions
	2021-10-13, 16:09:30 <span>current</span>	admin	<a href="#">Jump to this revision</a> <a href="#">Update description</a> <a href="#">Delete</a>
	2021-10-13, 13:51:56	admin	<a href="#">Jump to this revision</a> <a href="#">Update description</a> <a href="#">Delete</a>
snmp trap option configlet	2021-10-13, 11:52:38	admin	<a href="#">Jump to this revision</a> <a href="#">Update description</a> <a href="#">Delete</a>
	2021-10-12, 13:17:49	admin	<a href="#">Jump to this revision</a> <a href="#">Update description</a> <a href="#">Delete</a>
	2021-10-12, 13:16:58	admin	<a href="#">Jump to this revision</a> <a href="#">Update description</a> <a href="#">Delete</a>
with NTP error	2021-10-11, 18:37:38	admin	<a href="#">Jump to this revision</a> <a href="#">Update description</a> <a href="#">Delete</a>
with NTP	2021-10-11, 17:58:39	admin	<a href="#">Jump to this revision</a> <a href="#">Update description</a> <a href="#">Delete</a>
base	2021-10-11, 17:45:59	admin	<a href="#">Jump to this revision</a> <a href="#">Update description</a> <a href="#">Delete</a>

## Jump to Previous Blueprint Revision

**NOTE:** When you rollback to a previous revision, any previously staged changes that have not been committed are discarded. If this is an issue, do not jump to a different revision until you've committed the uncommitted changes.

1. From the blueprint, click **Time Voyager**, then click the **Jump to this revision** button for the revision to jump to (first of four buttons in **Actions** section).
2. Any uncommitted changes in the staged area are discarded. If this is an issue, close the dialog and address the uncommitted changes before proceeding. To proceed, click **Rollback**.
3. You can make additional changes to the blueprint before committing. For example, if you've replaced a device, the device ID (serial number) will change, but the IP won't. You can create the device agent and update the serial number in your blueprint before committing the revision change.
4. Click **Uncommitted**, then click the diff tabs to review the changes.

5. If you decide that you don't want to jump to this revision, click the **Revert** button to discard the changes.
6. To proceed, click the **Commit** button (top-right) to see the dialog for committing changes and creating a revision.
7. We recommend that you enter the optional revision description to identify the changes. Specific differences between revisions are not displayed, so the description is the only change information available for the revision.
8. Click **Commit** to commit your changes to the active blueprint and create a revision. In some cases, you might also need to ["reset resource group overrides" on page 420](#).
9. If you click **Time Voyager** you'll see the revision as the current one.

## Keep Saved Blueprint Revision

1. From the blueprint, click **Time Voyager**, then click the **Keep this revision** button for the revision to keep (second of four buttons in **Actions** section).
2. Click **Save** to confirm and proceed. The button turns gray indicating that the revision has been saved indefinitely. It won't be deleted until you manually delete it.

## Update Blueprint Revision Description

1. From the blueprint, click **Time Voyager**, then click the **Update description** button for the revision to keep (third of four buttons in **Actions** section.)
2. Enter or change the description.
3. Click **Update** to change the description and return to the list view.

## Delete Kept Blueprint Revision

1. From the blueprint, click **Time Voyager**, then click the **Delete** button for the revision to delete (fourth of four buttons in **Actions** section). You can't delete a revision if there are five (5) or fewer of them in the list.
2. Click **Delete** to delete the revision and return to the list view.

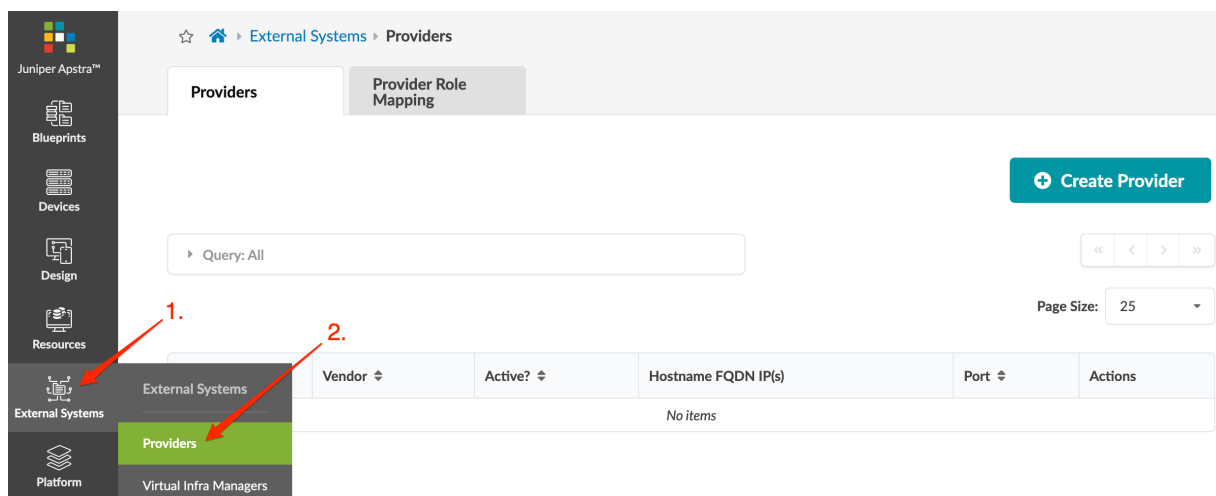
## Providers (External Systems)

### IN THIS SECTION

- [LDAP Provider | 687](#)
- [Active Directory Provider | 690](#)
- [TACACS+ Provider | 691](#)
- [RADIUS Provider | 693](#)
- [Edit / Delete Provider | 695](#)
- [Provider Role Map Overview | 696](#)
- [Create Provider Role Map | 697](#)
- [Edit / Delete Role Map | 697](#)

You can use Role-Based Access Control (RBAC) for specifying access permissions. RBAC servers are remote network servers that authenticate and authorize network access based on roles assigned to individual users within an enterprise (The accounting part of AAA is not included). If a user's group in the RBAC server is not specified, or if the provider group is not mapped to any user roles, that user cannot log in. This restriction avoids security issues by ignoring users without mapped groups. You can use the following protocols to authenticate and authorize users: LDAP, Active Directory, TACACS+, and RADIUS. Only Active Directory is supported as an external authentication server. No other versions are supported as external authentication servers, including RedHat IdM and Open LDAP. See the individual protocol sections for mor information.

From the left navigation menu, navigate to **External Systems > Providers** to go to providers. You can create, clone, edit and delete providers.



## LDAP Provider

### IN THIS SECTION

- [Create LDAP Provider | 687](#)
- [Configure LDAP Provider | 689](#)

## Create LDAP Provider

Lightweight Directory Access Protocol (LDAP)

1. From the left navigation menu, navigate to **External Systems > Providers** and click **Create Provider**.
2. Enter a **Name** (64 characters or fewer), select **LDAP**, and if you want LDAP to be the active provider, toggle on **Active?**.
3. For **Connection Settings**, enter/select the following:
  - **Port** - The TCP port - LDAP: **389**, LDAPS: **636**
  - **Hostname FQDN IP(s)** - The fully qualified domain name (FQDN) or IP address of the LDAP server. For high availability (HA) environments, specify multiple LDAP servers using the same settings. If the first server cannot be reached, connections to succeeding ones are attempted in order.
4. For **Provider-specific Parameters** enter/select the following, as appropriate:

- **Groups Search DN** - The LDAP Distinguished Name (DN) path for the RBAC Groups Organizational Unit (OU)
  - **Users Search DN** - The LDAP Distinguished Name (DN) path for the RBAC Users Organization Unit (OU)
  - **Bind DN** - The LDAP Distinguished Name (DN) path for the active server user that the Apstra server will connect as
  - **Password** - The LDAP server user password for the Apstra server to connect as
  - **Encryption** - None, SSL/TLS or STARTTLS
  - **Advanced Config**
    - **Timeout** (seconds)
    - **Username Attribute Name** - The LDAP attribute from the user entry that Apstra Server uses for authentication. (usually cn or uid)
    - **User Search Attribute Name**
    - **User First Name Attribute Name**
    - **User Last Name Attribute Name**
    - **User Email Attribute Name**
    - **User Object Class Attribute Name**
    - **User Member Attribute Name**
    - **Group Name Attribute Name**
    - **Group DN Attribute Name**
    - **Group Search Attribute Name**
    - **Group Member Attribute Name**
    - **Group Member Mapping Attribute Name**
    - **Group Object Class Attribute Name**
5. You can **Check provider parameters** and **Check login** (to verify authentication with the remote user credentials) before creating the provider.
  6. Click **Create** to create the provider and return to the list view.

## Configure LDAP Provider

To authorize Apstra users via a LDAP provider, the LDAP server must be configured to properly return a provider group attribute. This attribute must be mapped to a defined Apstra Role. The example configuration below is for the open-source OpenLDAP server.

```
dn: ou=People,dc=example,dc=com
objectClass: organizationalUnit
ou: People

dn: ou=Groups,dc=example,dc=com
objectClass: organizationalUnit
ou: Groups

dn: cn=user,ou=Groups,dc=example,dc=com
gidNumber: 5000
cn: user
objectClass: posixGroup
memberUid: USER1

dn: cn=USER1,ou=People,dc=example,dc=com
cn: USER1
givenName: USER1
loginShell: /bin/sh
objectClass: inetOrgPerson
objectClass: posixAccount
uid: USER1
userPassword: USER1
uidNumber: 10000
gidNumber: 5000
sn: USER1
homeDirectory: /home/users/USER1
mail: USER1@example.com
```

The **user** group must be mapped to a defined Apstra Role.

After configuring and activating a provider, you must ["map" on page 696](#) that provider to one or more user roles to give access permissions to users with those roles.

## Active Directory Provider

### IN THIS SECTION

- [Create Active Directory Provider | 690](#)

Active Directory (AD) is a database-based system that provides authentication, directory, policy, and other services in a Windows environment.

### Create Active Directory Provider

1. From the left navigation menu, navigate to **External Systems > Providers** and click **Create Provider**.
2. Enter a **Name** (64 characters or fewer), select **Active Directory**, and if you want Active Directory to be the active provider, toggle on **Active?**.
3. For **Connection Settings**, enter/select the following:
  - **Port** - The TCP port used by the server
  - **Hostname FQDN IP(s)** - The fully qualified domain name (FQDN) or IP address of the AD server. For high availability (HA) environments, specify multiple AD servers using the same settings. If the first server cannot be reached, connections to succeeding ones are attempted in order.
4. For **Provider-specific Parameters** enter/select the following, as appropriate:
  - **Groups Search DN** - The AD Distinguished Name (DN) path for the RBAC Groups Organizational Unit (OU)
  - **Users Search DN** - The AD Distinguished Name (DN) path for the RBAC Users Organization Unit (OU)
  - **Bind DN** - The AD Distinguished Name (DN) path for the active server user that the Apstra server will connect as
  - **Password** - The AD server user password for Apstra server to connect as
  - **Encryption** - None, SSL/TLS or STARTTLS
  - **Advanced Config**
    - **Timeout** (seconds)
    - **Username Attribute Name** - The AD attribute from the user entry that the Apstra server uses for authentication. (usually **cn** or **uid**)

- User Search Attribute Name
- User First Name Attribute Name
- User Last Name Attribute Name
- User Email Attribute Name
- User Object Class Attribute Name
- User Member Attribute Name
- Group Name Attribute Name
- Group DN Attribute Name
- Group Search Attribute Name
- Group Member Attribute Name
- Group Member Mapping Attribute Name
- Group Object Class Attribute Name

5. You can **Check provider parameters** and **Check login** (to verify authentication with the remote user credentials) before creating the provider.

6. Click **Create** to create the provider and return to the list view.

After configuring and activating a provider, you must ["map" on page 696](#) that provider to one or more user roles to give access permissions to users with those roles.

## TACACS+ Provider

### IN THIS SECTION

- [Create TACACS+ Provider | 692](#)
- [Configure TACACS+ Provider | 692](#)

Terminal Access Controller Access-Control Systems (TACACS+)

## Create TACACS+ Provider

1. From the left navigation menu, navigate to **External Systems > Providers** and click **Create Provider**.
2. Enter a **Name** (64 characters or fewer), select **TACACS+**, and if you want TACACS+ to be the active provider, toggle on **Active?**.
3. For **Connection Settings**, enter/select the following:
  - **Port** - The TCP port used by the server, usually **49**
  - **Hostname FQDN IP(s)** - The fully qualified domain name (FQDN) or IP address of the TACACS+ server. For high availability (HA) environments, specify multiple TACACS+ servers using the same settings. If the first server cannot be reached, connections to succeeding ones are attempted in order.
4. For **Provider-specific Parameters** enter/select the following, as appropriate:
  - **Shared Key** - shared key configured on the server

Caution

Shared key is not displayed when editing a configured TACACS+ provider. If you do not change it, the previously configured shared key is retained. If you test the provider and you have not re-entered the shared key, a null shared key is used for the test and may not work.
- **Auth Mode** - Authentication mode - ASCII (clear-text), PAP (Password Authentication Protocol), or CHAP (Challenge-Handshake Authentication Protocol)

  5. You can **Check provider parameters** and **Check login** (to verify authentication with the remote user credentials) before creating the provider.
  6. Click **Create** to create the provider and return to the list view.

## Configure TACACS+ Provider

To authorize Apstra users via a TACACS+ provider, the TACACS+ server must be configured to properly return an **aos-group** attribute. This attribute must be mapped to a defined Apstra Role. The example configuration below is for the open-source tac\_plus TACACS+ server.

```
user = jdoe {
    default service = permit
    name = "John Doe"
    member = admin
    login = des LQqpIWvpXDXDw
}

group = admin {
    service = exec {
        priv-lvl = 15
    }
}
```

```

    }
    cmd=show {
        permit .*
    }
    service = aos-exec {
        default attribute = permit
        priv-lvl = 15
        aos-group = apstra-admins
    }
}

```

The **apstra-admins** group must be mapped to a defined Apstra Role.

After configuring and activating a provider, you must ["map" on page 696](#) that provider to one or more user roles to give access permissions to users with those roles.

## RADIUS Provider

### IN THIS SECTION

- [RADIUS Limitations | 693](#)
- [Create RADIUS Provider | 694](#)

Remote Authentication Dial-In User Service (RADIUS). See below for limitations.

### RADIUS Limitations

- No support for changing the RADIUS user's password on a remote RADIUS server.
- RADIUS authentication does not control Linux user login via SSH.
- No support for group role-mapping changes.
- Nested groups are not allowed. You must explicitly assign each group to a role.
- When a user logs in, only username and password are required for authenticating against the remote RADIUS server. Log in credentials are not cached. Therefore, when a user logs in, a connection between Apstra and the remote RADIUS server is required.

## Create RADIUS Provider

1. From the left navigation menu, navigate to **External Systems > Providers** and click **Create Provider**.
2. Enter a **Name** (64 characters or fewer), select **RADIUS**, and if you want RADIUS to be the active provider, toggle on **Active?**.
3. For **Connection Settings**, enter/select the following:
  - **Port** - The TCP port used by the server, default is **1812** as specified in RFC 2865.
  - **Hostname FQDN IP(s)** - The fully qualified domain name (FQDN) or IP address of the RADIUS server. For high availability (HA) environments, specify multiple RADIUS servers using the same settings. If the first server cannot be reached, connections to succeeding ones are attempted in order.
4. For **Provider-specific Parameters** enter/select the following, as appropriate:
  - **Shared Key** (64 characters or fewer) - shared key configured on the server



**CAUTION:** Shared key is not displayed when editing a configured RADIUS provider. If you do not change it, the previously configured shared key is retained. If you test the provider and you have not re-entered the shared key, a null shared key is used for the test and may not work.

An example of a pre-shared key configuration that tests successfully with Apstra software is from Ubuntu FreeRADIUS (an open source RADIUS server). The Shared Key as given in the RADIUS server configuration must be provided in Apstra.

```
home_server localhost {
  ipaddr = 127.0.0.1
  port = 1812
  type = "auth"
  secret = "testing123"
  response_window = 20
  max_outstanding = 65536
```

- **Advanced Config**
  - **Group Name Attribute Name** - To specify a role that a user belongs to, the RADIUS server must specify the users' group. The user group information must be specified with **Framed-Filter-ID** as the attribute. It is used to assign users to different RADIUS groups.

For example, the FreeRADIUS config below specifies the **Framed-Filter-ID** attribute to be **freerad**. In this case, when mapping later, you would enter **freerad** for the Provider Group.

```
/etc/freeradius/users
freerad Cleartext-Password := "testing123"
Framed-Filter-Id = "freerad"
```

So that the user can be mapped to an existing group in the Apstra environment, the RADIUS server must return the Apstra group name as part of the authentication response.



**CAUTION:** If the group is unmapped, users cannot log in.

- **Timeout** (seconds) - Defaults to 30 seconds

After configuring and activating a provider, you must ["map" on page 696](#) that provider to one or more user roles to give permissions to users with those roles.

## Edit / Delete Provider

### IN THIS SECTION

- [Edit Provider | 695](#)
- [Delete Provider | 696](#)

### Edit Provider



**CAUTION:** Any users who are logged into Apstra software when a setting is changed in an active RBAC provider, are immediately logged out without notification. To continue, the user must log back into the Apstra server. This does not affect users who are defined locally on the Apstra server (for example, **admin**).

1. Either from the list view (External Systems > Providers) or the details view, click the **Edit** button for the provider to edit.
2. Make your changes.

3. Click **Update** (bottom-right) to edit the provider and return to the list view.

## Delete Provider

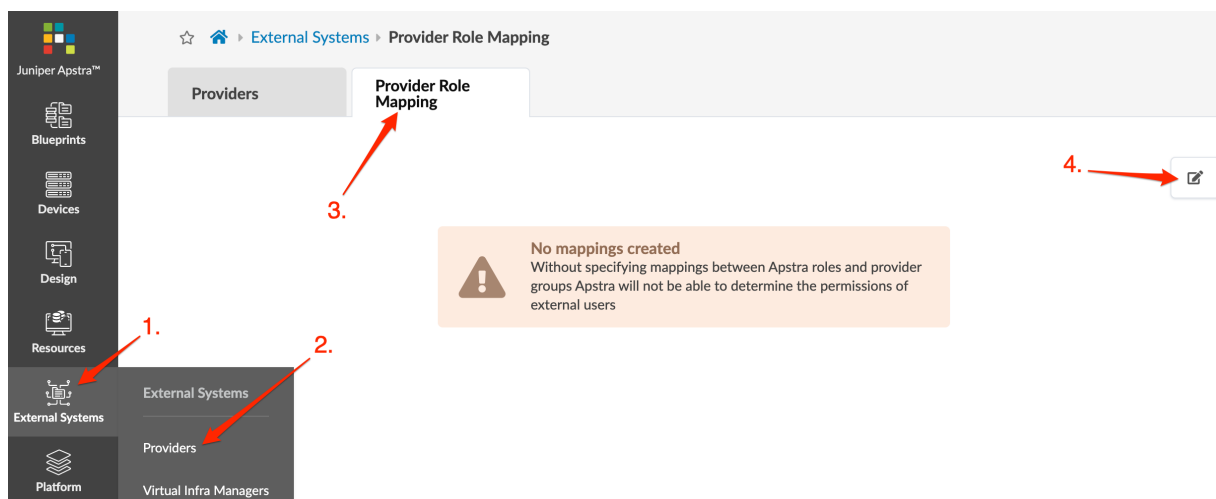
1. Either from the list view (External Systems > Providers) or the details view, click the **Delete** button for the provider to delete.
2. Click **Delete** to delete the provider and return to the list view.

## Provider Role Map Overview

After configuring an RBAC provider, you must map the provider to one or more user roles to give access permissions to users with those roles. You can create, edit and delete provider role mappings, as needed. Other details to be aware of include the following:

- Only one provider can be active at a time.
- You can map more than one Apstra role to the same provider group (new in version 4.0).
- When the same username exists both locally and in the RBAC provider, the local user is used to authenticate login attempts.
- Changing users with the web-based RBAC feature does not modify accounts on the Apstra server VM. To change these credentials, use standard Linux CLI commands: "useradd", "usermod", "userdel", "passwd".

From the left navigation menu, navigate to **External Systems > Providers > Provider Role Mapping** to go to provider role mapping.



# Create Provider Role Map

1. From the left navigation menu, navigate to **External Systems > Providers > Provider Role Mapping** and click the **Edit** button (top-right).
2. Click **Add mapping**, select a role from the drop-down list, then enter a provider group. The following is an example for mapping the **apstra-admins** group that was configured in TACACS+ configuration.

Edit Role Mappings

Apstra Role	Provider Group
administrator	apstra-admins

+ Add mapping

Update

**TIP:** To see user role details, navigate to **Platform > User Management > Roles**. From there, you can also create new roles, as needed.

3. To add another role mapping, click **Add mapping** and select an **Apstra Role** and **Provider Group**. You can have more than one role associated with the same provider group.
4. Click **Update** to create the role map. If the provider that you mapped is the active provider, then users with the mapped roles can log in with their usernames and passwords defined in the RBAC server.

# Edit / Delete Role Map

## IN THIS SECTION

- [Edit Role Map | 698](#)
- [Delete Role Map | 698](#)

## Edit Role Map



**CAUTION:** Changing role mappings for an active provider causes all remotely logged in users to be logged out (because the session tokens are cleared when changes are made). Users will need to log back into the system. This includes user **admin**, if **admin** is not logged in locally.

1. From the left navigation menu, navigate to **External Systems > Providers > Provider Role Mapping** and click the **Edit** button (top-right).
2. Edit role mapping as needed.
3. Click **Update** to update the role map.

## Delete Role Map

1. From the left navigation menu, navigate to **External Systems > Providers > Provider Role Mapping**, click the **Edit** button (top-right), then click the **X** next to the mapping to delete.
2. Click **Update** to update the role map.

# Platform

### IN THIS SECTION

- [User/Role Management \(Platform\) | 699](#)
- [Security \(Platform\) | 713](#)
- [Syslog Configuration \(Platform\) | 718](#)
- [Receivers \(Platform\) | 721](#)
- [Global Statistics \(Platform\) | 724](#)
- [Event Log \(Platform\) | 725](#)
- [Apstra Cluster \(Platform\) | 728](#)
- [Developers \(Platform\) | 734](#)
- [Technical Support \(Platform\) | 777](#)

## User/Role Management (Platform)

### IN THIS SECTION

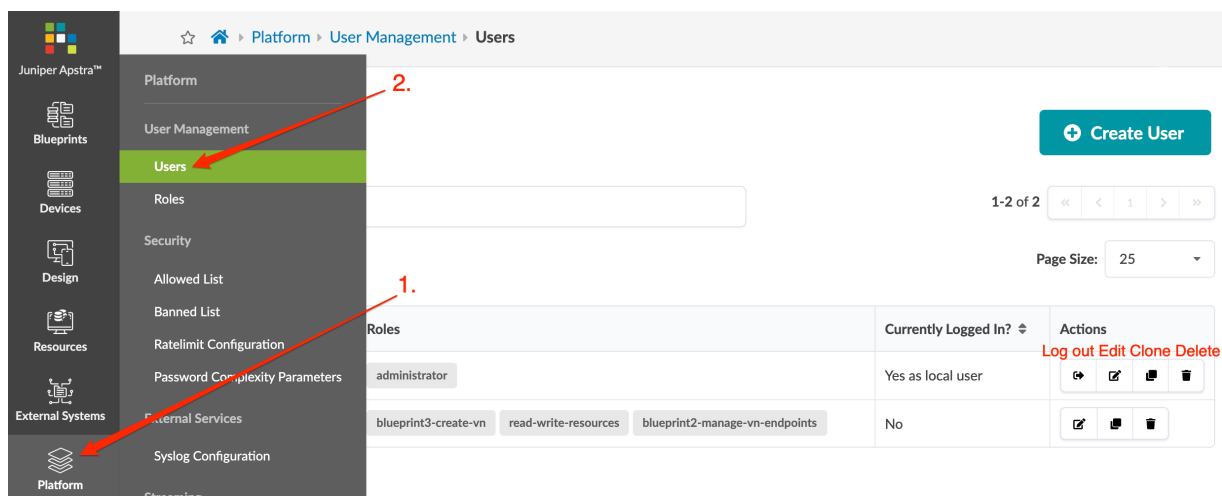
- [User Profile Management | 699](#)
- [User Role Management | 700](#)
- [User Profile Use Cases | 702](#)
- [Create User Profile | 706](#)
- [Change User Password | 706](#)
- [Log Out User | 706](#)
- [Edit / Delete User Profile | 706](#)
- [User Role Use Cases | 707](#)
- [Create User Role | 712](#)
- [Edit / Delete User Role | 712](#)

### User Profile Management

User profiles include the following details and options:

- Username
- First Name (optional)
- Last Name (optional)
- Email (optional)
- Password
- Roles

From the left navigation menu in the Apstra GUI, navigate to **Platform > User Management > Users** to go to user profiles.



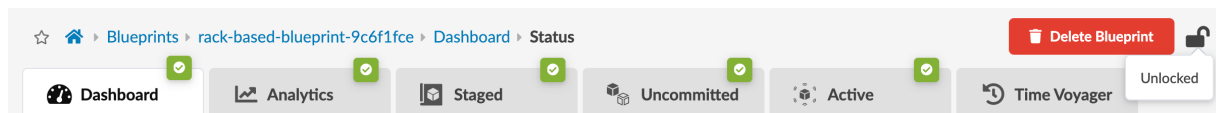
## User Role Management

Users with the **administrator** role can create, clone, edit and delete user roles (which are assigned to user profiles). These roles can also be "mapped" on page 696 to external groups used by authentication providers such as LDAP, Active Directory, TACACS+, and RADIUS.

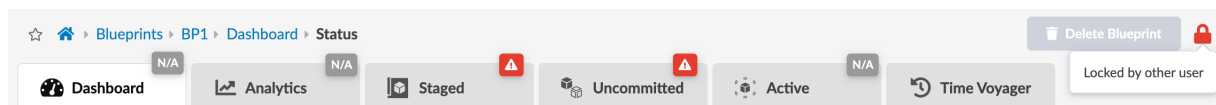
With Enhanced Role Based Access Control, you can create blueprint-specific roles with very specific privileges allowing limited control to associated users. This allows you to create more hierarchical roles and protect against accidental changes to the network.

The blueprint locking feature prevents restricted users (based on their roles) from making changes that effectively are not permitted. In particular, a restricted user should not be able to commit changes made by another user.

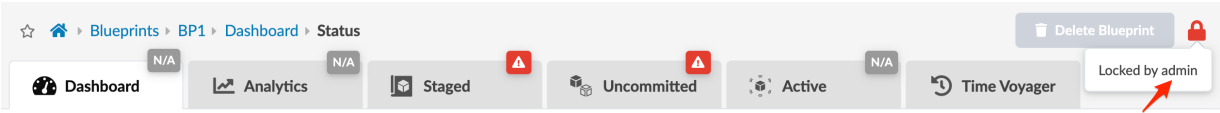
A blueprint with no uncommitted changes is considered "unlocked".



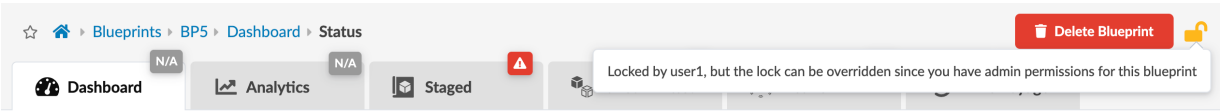
If you have permission (based on your assigned roles) to create/update/delete virtual networks, and another user has made uncommitted changes to the blueprint. The blueprint is considered "locked", and you will not be able to create/update/delete virtual networks until the changes are committed or reverted by the "locking user" who made the uncommitted changes, unless you are the locking user.



If you have permission (based on your assigned roles) the name of the user who created the pending changes is displayed.



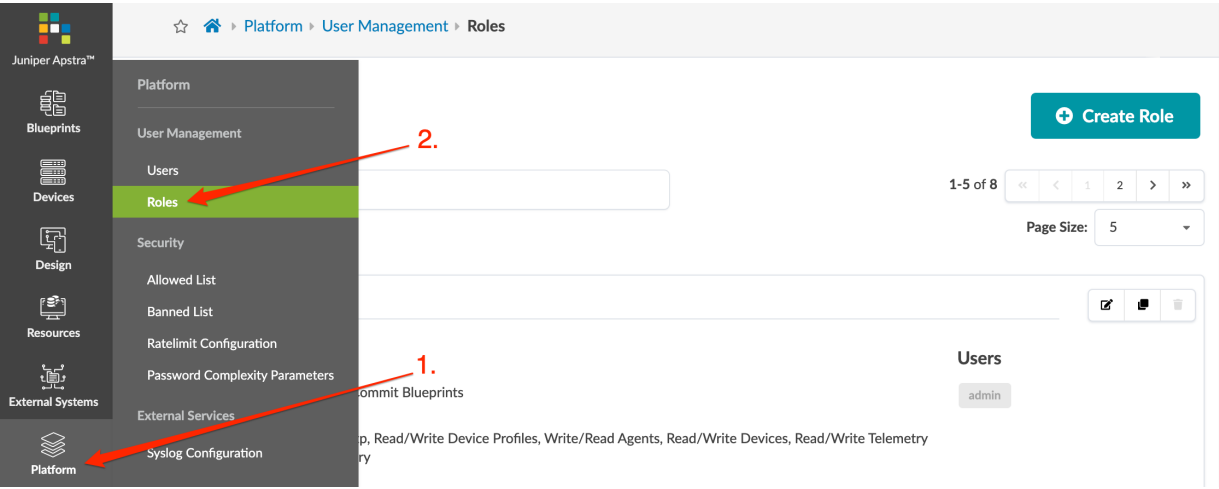
An admin user who has "Write/Commit Blueprints" permissions can make any changes to, apply changes for, revert changes for any blueprint.



User roles include the following details and options:

Parameter	Description
Name	role name
Type	global permission or per-blueprint permissions
Global Permissions (read, write, commit, as applicable)	<ul style="list-style-type: none"> <li>Blueprints - blueprints</li> <li>Devices - device profiles, agents, devices</li> <li>Design - configlets, templates, rack types, logical devices, property sets, interface maps</li> <li>Resources - IP pools, IPv6 pools, ASN pools, VNI pools</li> <li>AAA - sysdb data, AAA providers, roles, audit config, audit events, users</li> <li>Other - streaming, Apstra metric logs, ztp, port setting schema, Apstra cluster management, virtual infra manager, telemetry service registry, port aliases</li> </ul>
Per-Blueprint Permissions	<ul style="list-style-type: none"> <li>Which Blueprints? All or by ID</li> <li>Permissions                             <ul style="list-style-type: none"> <li>Read blueprint</li> <li>Make any changes to staging blueprint (includes managing VNs and their endpoints)</li> <li>Commit changes</li> <li>Manage virtual networks (includes managing VN endpoints)</li> <li>Manage virtual network endpoints</li> </ul> </li> </ul>

From the left navigation menu, navigate to **Platform > User Management > Roles** to go to user roles. You can create, clone, edit, and delete user roles, except for the four predefined user roles (administrator, device\_ztp, user, viewer) which cannot be modified.



## User Profile Use Cases

IN THIS SECTION

- [Use Case Overview | 702](#)

## Use Case Overview

IN THIS SECTION

- [Use Case 1: Create Virtual Networks Only \(not Including Allocating Resources\) | 704](#)
- [Use Case 2: Create Virtual Networks and Allocate Resources | 705](#)

The following use cases are described below.

☆ [Home](#) > [Platform](#) > [User Management](#) > [Users](#)

Create User

Query: All

1-3 of 3

Page Size: 25

Username	E-Mail	Roles	Currently Logged In?	Actions
admin	not_set	administrator	Yes as local user	<div></div>
blueprint3-vn		blueprint3-create-vn read-write-resources blueprint2-manage-vn-endpoints	No	<div></div>
create-vn-only		blueprint3-create-vn	No	<div></div>

Use case 2

Use case 1

*Use Case 1: Create Virtual Networks Only (not Including Allocating Resources)*

To limit a user's role to only create virtual networks and look at blueprint details, assign them the role as described in ["User Role Use Case 3" on page 707](#).

**Username \***

**First Name**

**Last Name**

**Email**

**Password \***

- ✓ Length should be at least 8
- ✓ Must contain uppercase letter
- ✓ Must contain lowercase letter
- ✓ Must contain digit
- ✓ Must contain special character

**Repeat Password \***

**Global Roles**

- ☐ administrator
- ☐ device\_ztp
- ☐ read-write-resources
- ☐ user
- ☐ viewer

**Per-Blueprint Roles**

- ☐ blueprint1-read-write-commit
- ☐ blueprint2-manage-vn-endpoints
- ☒ blueprint3-create-vn

*Use Case 2: Create Virtual Networks and Allocate Resources*

To allow a user to create virtual networks and allocate resources to them, you must assign them multiple roles. For more information, see ["User Role Use Cases 3A and 4" on page 707](#).

**Username \***

blueprint3-vn

**First Name**

virtual network managment

**Last Name**


**Email**

**Password \***

..... 

- ✓ Length should be at least 8
- ✓ Must contain uppercase letter
- ✓ Must contain lowercase letter
- ✓ Must contain digit
- ✓ Must contain special character

**Repeat Password \***

..... 

**Global Roles**

- ☐ administrator
- ☐ device\_ztp
- ☒ read-write-resources
- ☐ user
- ☐ viewer

**Per-Blueprint Roles**

- ☐ blueprint1-read-write-commit
- ☒ blueprint2-manage-vn-endpoints
- ☒ blueprint3-create-vn

## Create User Profile

1. From the left navigation menu, navigate to **Platform > User Management > Users** and click **Create User**.
2. Enter a username.
3. Enter a password that meets password complexity requirements. (For more information, see ["Password Complexity Parameters" on page 717.](#))
4. Re-enter the password.
5. Select one or more roles. If custom roles have been created, they appear as options along with the predefined roles that ship with the software. (You can see the permissions specified for each of the roles at **Platform > User Management > Roles**.)
6. Click **Create** to create the user profile and return to the list view.

## Change User Password

1. From the left navigation menu, navigate to **Platform > User Management > Users**, click the username to change, then click the **Change Password** button (top-right).
2. Enter a new password that meets password complexity requirements. (For more information, see ["Password Complexity Parameters" on page 717.](#))
3. Re-enter the new password.
4. Click **Change Password** to update the password.

## Log Out User

From the left navigation menu, navigate to **Platform > User Management > Users** and click the **Log Out** button for the user.

## Edit / Delete User Profile

### IN THIS SECTION

- [Edit User Profile | 706](#)
- [Delete User Profile | 707](#)

## Edit User Profile

1. Either from the list view (Platform > User Management > Users) or the details view, click the **Edit** button for the user profile.
2. Change roles and/or other details.

3. Click **Update** to update the user profile and return to the list view.

### Delete User Profile

1. From the left navigation menu, navigate to **Platform > User Management > Users** and click the **Delete** button for the user profile.
2. Click **Delete** to delete the user profile and return to the list view. (User **admin** cannot be deleted.)

**NOTE:** You can also use REST API to manage user profiles. Navigate to **Platform > Developers** for **REST API Documentation** and tools. See the **aaa** section for user-related APIs.

## User Role Use Cases

### IN THIS SECTION

- [Use Cases Overview | 707](#)

## Use Cases Overview

### IN THIS SECTION

- [Use Case 1: Read, Write and Commit Specific Blueprints | 708](#)
- [Use Case 2: Manage VN Endpoints on Specific Blueprints | 709](#)
- [Use Case 3: Create Virtual Networks \(not Including Allocating Resources\) | 710](#)
- [Use Case 3A: Create Virtual Networks and Allocate Resources | 711](#)
- [Use Case 4: Read and Write Resources on All Blueprints | 712](#)

The following use cases are described below.

☆
🏠
Platform
>
User Management
>
Roles

blueprint1-read-write-commit
← Use case 1

BP1

Commit changes  
Read blueprint  
Make any change to staging blueprint

Users

blueprint2-manage-vn-endpoints
← Use case 2

BP2

Read blueprint  
Manage virtual network endpoints

Users

blueprint3-create-vn
← Use case 3

BP3

Commit changes  
Read blueprint  
Manage virtual network endpoints  
Manage virtual networks

Users

device\_ztp

Permissions

Devices: Write ztp

Users

read-write-resources
← Use case 3A and 4

Permissions

Resources: Read/Write IP Pools, Write/Read IPv6 Pools, Write/Read ASN Pools, Write/Read VNI Pools

Users

### *Use Case 1: Read, Write and Commit Specific Blueprints*

To create a role that gives a user permission to read, write, and commit to specific blueprints, select **Per-Blueprint Permissions**, select one or more blueprint IDs (or **All** for all blueprints), then toggle on **Read blueprint**, **Make any change to staging blueprint**, and **Commit changes**. The changes that can be made include **Manage virtual networks** and **Manage virtual network endpoints** even though those permissions

may or may not be toggled on.

Name \*

blueprint1-read-write-commit

Description

A user with this role can read, write and commit the blueprint named BP1.

Type

☐ Global Permissions

☒ Per-Blueprint Permissions

Which Blueprints?

All

☒ OFF

By ID

☒ BP1

☐ BP2

☐ BP3

Permissions \*

Read blueprint

ON

Make any change to staging blueprint

ON

Commit changes

ON

Read information about user who locked blueprint

OFF

Manage virtual networks

OFF

Manage virtual network endpoints

OFF

*Use Case 2: Manage VN Endpoints on Specific Blueprints*

To create a role that gives a user permission to only manage virtual network endpoints on specific blueprints, select **Per-Blueprint Permissions**, select one or more blueprint IDs (or **All** for all blueprints),

then toggle on **Manage virtual network endpoints**.

Name \*

blueprint2-manage-vn-endpoints

Description

A user with this role can manage VN endpoints on the blueprint named BP2.

Type

☐ Global Permissions ☒ Per-Blueprint Permissions

Which Blueprints?

All

☐ OFF

By ID

☐ BP1

☒ BP2

☐ BP3

Permissions \*

Read blueprint

☒ ON

Make any change to staging blueprint

☐ OFF

Commit changes

☐ OFF

Read information about user who locked blueprint

☐ OFF

Manage virtual networks

☐ OFF

Manage virtual network endpoints

☒ ON

### *Use Case 3: Create Virtual Networks (not Including Allocating Resources)*

To create a role that gives a user permission to only create virtual networks, select **Per-Blueprint Permissions**, select one or more blueprint IDs (or toggle on **All** for all blueprints), then toggle on **Read Blueprint**, **Commit changes**, **Manage virtual networks**, and **Manage virtual network endpoints**. By not selecting **Make any change to staging blueprint** you are limiting the changes that can be made to virtual

networks only.

Name \*

blueprint3-create-vn

Description

A user with this role can create virtual networks on the blueprint named BP3. (For the user to be able to allocate resources to the virtual network, they need two additional roles: one with global permissions to read and write resources and one with per-blueprint permissions to make any change to staging blueprint.)

Type

☐ Global Permissions ☒ Per-Blueprint Permissions

Which Blueprints?

All

☐ OFF

By ID

☐ BP1

☐ BP2

☒ BP3

Permissions \*

Read blueprint

☒ ON

Make any change to staging blueprint

☐ OFF

Commit changes

☒ ON

Read information about user who locked blueprint

☐ OFF

Manage virtual networks

☒ ON

Manage virtual network endpoints

☒ ON

### Use Case 3A: Create Virtual Networks and Allocate Resources

For a user with the role in use case 3 above to be able to allocate resources to the virtual networks that they create, they must also be assigned two additional roles: one with global permissions to read and write resources (see use case 4 below) and another one with per-blueprint permissions to **Make any change to staging blueprint**, effectively giving them access to make other changes in addition to making changes to virtual networks. Of course, this second one would not be needed if the role for creating virtual networks also enabled **Make any change to staging blueprints**.

Use Case 4: Read and Write Resources on All Blueprints

To create a role that gives a user permission to read and write resources on any blueprint, select **Global Permissions**, then toggle on **Resources** for **Read** and **Write**, which toggles on all resource types.

Name \*

read-write-resources

Description

A user with this role can read and write resources in any blueprint.

Type

☒ Global Permissions

☐ Per-Blueprint Permissions

Permissions \*

Permission	Read	Write	Commit
templates	<input type="checkbox"/>	<input type="checkbox"/>	
Resources	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
ASN Pools	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
IP Pools	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
IPv6 Pools	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
VNI Pools	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
AAA	<input type="checkbox"/>	<input type="checkbox"/>	
AAA	<input type="checkbox"/>	<input type="checkbox"/>	

Create User Role

1. From the left navigation menu of the Apstra GUI, navigate to **Platform > User Management > Roles** and click **Create Role**.
2. Enter a name and description, then select permission type and one or more permissions.
3. Click **Create** to create the role and return to the list view.

Edit / Delete User Role

IN THIS SECTION

Edit User Role | 713

Delete User Role | 713

## Edit User Role

The four predefined user roles (administrator, device\_ztp, user, viewer) cannot be modified.

1. Either from the list view (Platform > User Management > Roles) or the details view, click the **Edit** button for the user role.
2. Change permissions, as applicable.
3. Click **Update** to update the role and return to the list view.

## Delete User Role

The four predefined user roles (administrator, device\_ztp, user, viewer) cannot be deleted.

1. Either from the list view (Platform > User Management > Roles) or the details view, click the **Delete** button for the user role to delete.
2. Click **Delete** to delete the role and return to the list view.

**NOTE:** You can also use REST API to manage user roles. Navigate to **Platform > Developers** for REST API documentation and tools. See the **aaa** section for role-related APIs.

## Security (Platform)

### IN THIS SECTION

- [Allowed List | 714](#)
- [Banned List | 715](#)
- [Rate Limit Configuration | 716](#)
- [Edit Password Complexity Requirements | 717](#)

## Allowed List

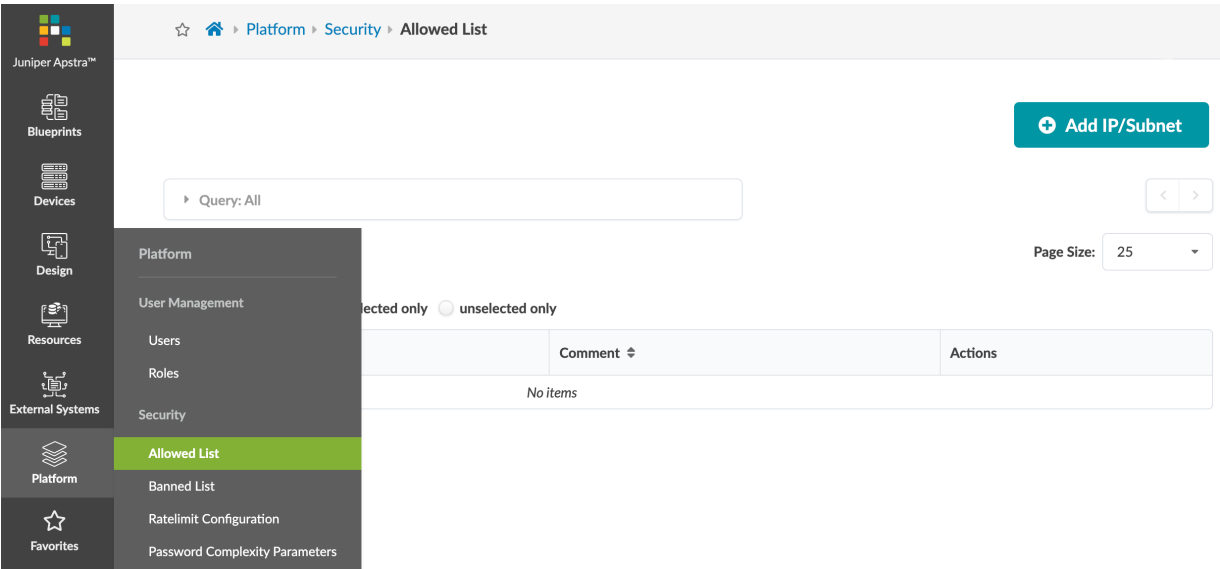
### IN THIS SECTION

- [Allowed List Overview | 714](#)
- [Add IP/Subnet to Allowed List | 714](#)
- [Edit IP/Subnet to Allowed List | 715](#)
- [Delete IP/Subnet from Allowed List | 715](#)

### Allowed List Overview

You can add trusted IP/subnets to the allowed list so they are never locked out, even if they violate rate limit rules. You can add and change comments about those IP/subnets.

From the left navigation menu, navigate to **Platform > Security > Allowed List**. You can search and sort the list. You can add, edit, and delete IP/subnets.



### Add IP/Subnet to Allowed List

1. From the left navigation menu, navigate to **Platform > Security > Allowed List** and click **Add IP/Subnet**.
2. Enter an IP address or subnet, and a comment.
3. To keep the dialog open to add another IP/subnet, check the **Create Another** check box.

4. Click **Create** to add the IP/subnet and return to the list view (or, if you checked **Create Another**, return to the dialog to enter another IP/subnet).

### Edit IP/Subnet to Allowed List

1. From the left navigation menu, navigate to **Platform > Security > Allowed List** and click the **Edit** button for the IP/subnet to edit.
2. Change the comment.
3. Click **Update** to complete the change and return to the list view.

### Delete IP/Subnet from Allowed List

1. From the left navigation menu, navigate to **Platform > Security > Allowed List**.
2. Select the IP/subnet(s) to delete.
  - To delete a single IP/subnet, click the **Delete** button for the IP/subnet (right-side).
  - To delete one or more IP/subnets, click the checkbox (left-side) for one or more IP/subnets and click the **Delete** button above the list.
3. Click **Update** to complete the deletion and return to the list view.

## Banned List

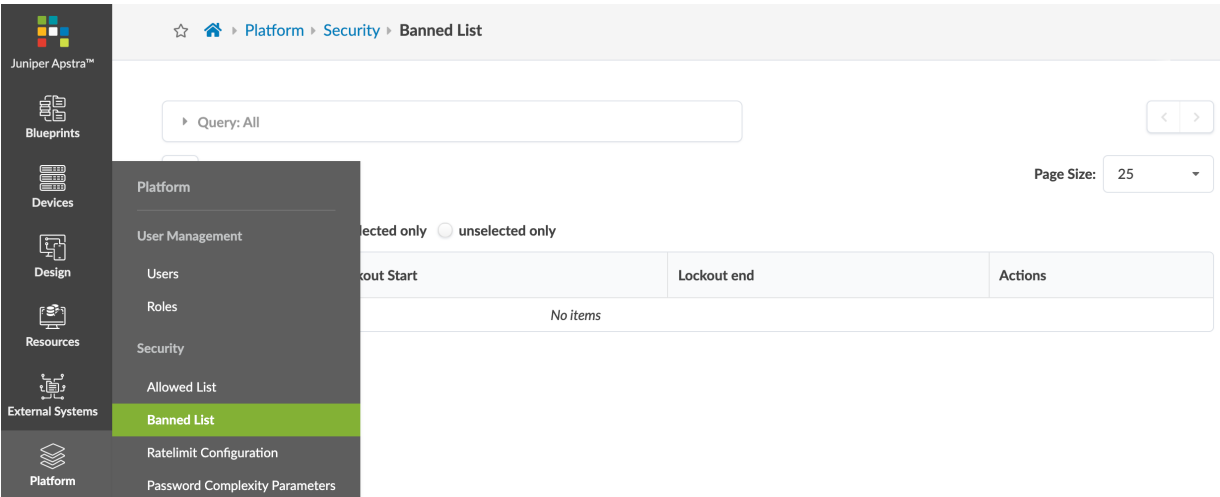
### IN THIS SECTION

- [Banned List Overview | 715](#)
- [Delete IP/Subnet from Banned List | 716](#)

### Banned List Overview

IP/subnets that violate rate limit rules are automatically added to the banned list and are locked out for the configured lockout period, or until an admin removes them from the banned list. The banned list has a lower precedence than the allowed list, so an IP/subnet on the banned list may actually not be banned. From the left navigation menu, navigate to **Platform > Security > Banned List** to go to IP/subnets on the

banned list. You can search and sort the list. You can remove IP/subnets from the banned list.



### Delete IP/Subnet from Banned List

1. From the left navigation menu, navigate to **Platform > Security > Banned List** and click the **Delete** button to the right of the IP/subnet(s) to delete.
2. Click **Delete** to remove the IP/subnet from the banned list and immediately allow logins from that IP/subnet.

## Rate Limit Configuration

### IN THIS SECTION

- [Rate Limit Configuration Overview | 716](#)
- [Edit Rate Limit Configuration | 717](#)

### Rate Limit Configuration Overview

Default settings allow 5 login attempts within 60 seconds. After the fifth failed attempt, the IP/subnet is blocked and added to the banned list for 3 minutes (found at **Platform > Security > Banned List**), or until an admin removes it from the list. When you change rate limit configuration, any banned IP/subnets are immediately affected. For example, if you change the lockout period from 3 minutes to 5 minutes, an IP/subnet that's already on the banned list would remain on the banned list for an additional 2 minutes.

## Edit Rate Limit Configuration

1. From the left navigation menu, navigate to **Platform > Security > Ratelimit Configuration** and click the **Edit** button (top-right).

Juniper Apstra™

Platform > Security > Ratelimit Configuration

Platform

User Management

Users

Roles

Security

Allowed List

Banned List

**Ratelimit Configuration**

Password Complexity Parameters

Lockout period (sec)	180
Observation window (sec)	60
Number of observations	5

Edit

2. Change parameter values (lockout period, time period, number of attempts).
3. Click **Update** to complete the change and return to the Rate Limit Configuration page.

## Edit Password Complexity Requirements

You can change password complexity requirements. Changes are applied to newly created passwords. Existing passwords are not affected. Rules are entered as regular expressions.

1. From the left navigation menu, navigate to **Platform > Security > Password Complexity Parameters** and click the **Edit** button (top-right).

Juniper Apstra™

Platform > Security > Password Complexity Parameters

Platform

User Management

Users

Roles

Security

Allowed List

Banned List

Ratelimit Configuration

**Password Complexity Parameters**

Regular Expression	Error Message
^(?=.{8,})	Length should be at least 8
^(?=.*[A-Z])	Must contain uppercase letter
^(?=.*[a-z])	Must contain lowercase letter
^(?=.*[0-9])	Must contain digit
^(?=.*[!@A-Za-z0-9])	Must contain special character

Edit

2. Change parameter values:
  - To add a rule, click **Add** and enter a regular expression and error message.
  - To change a rule, change values as appropriate and update the error message.
  - To delete a rule, click the red **X** to the right of the rule to delete.
3. Click **Update** to complete the change and return to the Password Complexity Parameters page.

## Syslog Configuration (Platform)

### IN THIS SECTION

- [Syslog Configuration Overview | 718](#)
- [Create Syslog Config | 720](#)
- [Edit Syslog Config | 721](#)
- [Delete Syslog Config | 721](#)

### Syslog Configuration Overview

You can use the Syslog protocol for auditing and for forwarding anomaly messages to one or more external Syslog servers. Alarms can be generated for use in an NMS alarming environment.

Syslog messages follow Common Event Format (CEF) conventions as shown below:

AOS Log Format:

```
'{timestamp} {host} '
  'CEF:{version}|{device_vendor}|{device_product}|{device_version}|'
  '{device_event_class_id}|{name}|{severity}|{extension}'
```

Where:

```
{version}      : always "0"
{device_vendor} : always "Apstra"
{device_product} : always "AOS"
{device_version} : current AOS version
{device_event_class_id} : "100" for audit logs, "101" for anomaly logs
{name}         : "Audit event" for audit logs, "Alert" for anomaly logs
{severity}     : "medium" for audit logs, "Very-High" for anomaly logs
```

And where {extension} is either :

```
For anomaly logs : msg=<json payload>
For audit logs   : cat=<activity> src=<src_IP> suser=<username> act=<activity result>
cs1Label=<field1_type> cs1=<field1_value> cs2Label=<field2_type> cs2=<field2_value>
cs3Label=<field3_type> cs3=<field3_value>
```

### Anomaly Log JSON Format

u'blueprint\_label' : Name of the blueprint the anomaly was raised in.  
 u'timestamp' : Unix timestamp when the Anomaly was raised.  
 u'origin\_name' : Serial Number of the device the anomaly affects.  
 u>alert' : The value is a JSON Payload with the actual anomaly (see next table)  
 u'origin\_hostname' : Hostname of the device the anomaly affects.  
 u'device\_hostname' : Hostname of the device the anomaly affects.  
 u'origin\_role' : Role of the device the anomaly affects.  
 u'first\_seen' : Unix timestamp when the Anomaly was raised for the first time.  
 u'raised' : Always True.  
 u'severity' : The severity level of the anomaly. Always 3.

### Audit Log Extension Format:

cat : Activity performed. Valid values: "Login", "Logout", "BlueprintCommit",  
 "DeviceConfigChange", "BlueprintDelete".  
 src : Source IP of the client making HTTP requests.  
 suser : Who performed the activity.  
 act : Outcome of the activity - free-form string. In case of error, include error  
 string. Ex: Unauthorized  
 cs1Label : The string "Blueprint Name"  
 cs1 : Name of the blueprint on which action was taken.  
 cs2Label : The string "Blueprint ID"  
 cs2 : Id of the blueprint on which action was taken.  
 cs3Label : The string "Commit Message". Only exists if user has added a commit message  
 (optional)  
 cs3 : Commit Message. Only exists if user has added a commit message (optional)  
 deviceExternalId : Id (typically serial number) of the managed device on which action was  
 taken.  
 deviceConfig : Config that is pushed and applied on the device where "#012" is used to  
 indicate a line break to log collectors and parsers.

An example of Syslog messages is shown below:

```

Jul 31 03:10:36 aos-server - 2019-07-31T03:10:36.797839+0000 aos-server
CEF:0|Apstra|AOS|3.0.0-151|101|Alert|Very-High|msg={'device_hostname':
'<device hostname unknown>', u'timestamp': 1564542636797839, u>alert':
{'u'probe_alert': {'u'stage_name': u'Anomaly', u'actual_int': 9711, u'probe_id':
u'46b827c2-be4d-47a5-95f9-1ed0a57224f6', u'key_value_pairs': [{u'value':
u'"Ethernet5"', u'key': u'interface'}, {u'value': u'"2CC260994101"', u'key':

```

```
u'system_id']], u'item_id': u'anomaly,probe\=46b827c2-be4d-47a5-95f9-1ed0a57224f6
,proc\=Anomaly,stage\=out,interface\=Ethernet5,system_id\=2CC260994101',
u'expected_int': 8000}, u'first_seen': 1564542636797795, u'raised': True,
u'severity': 3, u'id': u'4981e646-e665-481b-bada-391689e7ebf3'}, u'origin_name':
u'anomaly,probe\=46b827c2-be4d-47a5-95f9-1ed0a57224f6,proc\=Anomaly,stage\=out,
interface\=Ethernet5,system_id\=2CC260994101'}
```

```
Jul 31 03:11:01 aos-server - 2019-07-31T03:11:01.699190+0000 aos-server
CEF:0|Apstra|AOS|3.0.0-151|101|Alert|Very-High|msg={'device_hostname':
'<device hostname unknown>', u'timestamp': 1564542661699190, u'alert':
{'u'probe_alert': {'u'stage_name': u'Anomaly', u'actual_int': 12890, u'probe_id':
u'46b827c2-be4d-47a5-95f9-1ed0a57224f6', u'key_value_pairs': [{u'value':
u'"Ethernet5"', u'key': u'interface'}, {u'value': u'"2CC260994101"', u'key':
u'system_id'}], u'item_id': u'anomaly,probe\=46b827c2-be4d-47a5-95f9-1ed0a57224f6,
proc\=Anomaly,stage\=out,interface\=Ethernet5,system_id\=2CC260994101',
u'expected_int': 8000}, u'first_seen': 1564542636797795, u'raised': False,
u'severity': 3, u'id': u'4981e646-e665-481b-bada-391689e7ebf3'}, u'origin_name':
u'anomaly,probe\=46b827c2-be4d-47a5-95f9-1ed0a57224f6,proc\=Anomaly,stage\=
out,interface\=Ethernet5,system_id\=2CC260994101'}
```

From the left navigation menu, navigate to **Platform > External Services > Syslog Configuration** to see configurations. You can create, clone, edit and delete syslog configuration.

1. Platform

2. Syslog Configuration

Toggle on to use syslog for audit

Toggle on to use syslog for anomaly forwarding

Protocol	Facility	Time Zone	Use for Audit	Forward Anomalies	Actions
UDP	user	America/Los_Angeles	OFF	OFF	Edit Clone Delete

## Create Syslog Config

- From the left navigation menu, navigate to **Platform > External Services > Syslog Configuration** and click **Create Syslog Config** (top-right).
- Configure the Syslog server.
  - IP Address (or hostname)
  - Port

- Protocol - UDP, TCP
  - Facility - kern, user, mail, daemon, auth, syslog, lpr, news, uucp, authpriv, ftp, cron, local0, local1, local2, local3, local4, local5, local6, local7
  - Time Zone (optional) (new in version 4.0)
3. Click **Create** to save the configuration and return to the list view.
  4. To configure another Syslog server, repeat the steps above.
  5. To enable messages to be sent to a configured server, toggle on **Use for Audit** and/or **Forward Anomalies**, as appropriate. (Before version 4.0, messages would include the default aos-server hostname even if the hostname had been changed.)

## Edit Syslog Config

1. From the left navigation menu, navigate to **Platform > External Services > Syslog Configuration** and click the **Edit** button for the Syslog configuration to edit.
2. Make your changes.
3. Click **Update** to update the Syslog configuration and return to the list view.

## Delete Syslog Config

1. From the left navigation menu, navigate to **Platform > External Services > Syslog Configuration** and click the **Delete** button for the Syslog configuration to delete.
2. Click **Delete Syslog Config** to delete the Syslog configuration and return to the list view.

## Receivers (Platform)

### IN THIS SECTION

- [Streaming Receivers Overview | 722](#)
- [Create Receiver | 722](#)
- [Delete Receiver | 723](#)
- [Configure Receivers Using Telegraf Plugin | 723](#)

## Streaming Receivers Overview

You can configure the Apstra server to stream alerts, events and perfmon, or any combination thereof. Each data type is sent to a streaming receiver over its own TCP socket. Even if all three data types are configured for the same streaming receiver, three (3) connections are created between the Apstra server and the streaming receiver. This also allows for all three types to be sent to three different streaming receivers.

Receivers include the following details:

- **Hostname** - Hostname
- **Port** - default: 4444
- **Message Type** - alerts, events, perfmon
- **Sequencing Mode** - unsequenced, sequenced

From the left navigation menu, navigate to **Platform > Streaming > Receivers** to go to receivers. You can create and delete receivers.

Juniper Apstra™

Platform > Streaming > Receivers

Create Receiver

1-5 of 5

Page Size: 25

Message Type	Sequencing Mode	Alive	Connection Reset Count	Last Transmitted Message	Last Disconnected	Actions
perfmon	Sequenced	✓	0			[Delete]
events	Sequenced	✓	0			[Delete]
alerts	Sequenced	✓	0			[Delete]
perfmon	Sequenced	✓	0			[Delete]
events	Sequenced	✓	0			[Delete]

## Create Receiver

1. From the left navigation menu of the Apstra GUI, navigate to **Platform > Streaming > Receivers** and click **Create Receiver**.
2. Enter/select required values.
3. Click **Create** to create the receiver and return to the list view.

## Delete Receiver

1. From the left navigation menu of the Apstra GUI, navigate to **Platform > Streaming > Receivers** and click the delete button for the receiver to delete.
2. Click **Delete** to delete the receiver from the system and return to the list view.

## Configure Receivers Using Telegraf Plugin

You can use the Apstra Telegraf input plugin to receive streaming telemetry from Apstra. [Telegraf](#) is an agent for collecting, processing, aggregating, and writing metrics. This is the component of AOSOM-Streaming that handles the reception of the protobuf messages from the Apstra environment. For more information, see the ["AOSOM Streaming Guide" on page 828](#). The Telegraf platform consists of input and output plugins that you can choose from for aggregating and storing metrics to different backend databases. The Apstra input plugin for Telegraf deserializes the protobuf stream and creates metrics that can then be sent to a particular backend database, such as Prometheus, InfluxDB, or Elasticsearch.

The configuration described here assumes you are using the Apstra Telegraf input plugin. You can configure streaming receivers in Apstra with the Telegraf plugin by providing it Apstra credentials. We recommend that you use a separate Apstra account with only the streaming credentials. If you configure through the GUI, then there is no need to supply credentials in the Telegraf config file.

The easiest way to run the Telegraf receiver is in a docker container. The `docker-compose.yml` snippet below shows the configuration for the Telegraf container. This pulls the latest Apstra supported Telegraf container from Docker Hub.

```
# Telegraf container config
telegraf-prom:
  image: apstra/telegraf:latest
  command: telegraf
  volumes:
    - ./config/telegraf-prom.toml:/etc/telegraf/telegraf.conf
  ports:
    - '9999:9999'
```

The Telegraf configuration file - `./config/telegraf-prom.toml` - is mapped to `/etc/telegraf/telegraf.conf` on the container. It includes the following parameters:

- **address** - specifies the IP address of the streaming receiver
- **port** - specifies the port that the streaming receiver will be listening on
- **streaming\_type** - specifies the type of data to be streamed from Apstra to this receiver

The remaining parameters are only necessary if you want the Apstra Telegraf plugin to configure the streaming receivers in Apstra via the API.

- **aos\_server** - specifies the IP address of the Apstra server
- **aos\_port** - should always be 443
- **aos\_login** - Apstra's username
- **aos\_password** - Apstra password

The input and output plugin configurations are shown in the snippet below. The output plugin is configured for the Prometheus client and listens on port 9126. The input plugin is configured for Apstra.

```
# Configuration for Prometheus server to expose metrics
[[outputs.prometheus_client]]
  listen = ":9126"
  expiration_interval = "0"

[[inputs.aos]]
  address = "10.1.1.200"
  port = 9999
  streaming_type = [ "perfmon", "alerts", "events" ]
  aos_server = "$AOS_SERVER"
  aos_port = $AOS_PORT
  aos_login = "$AOS_LOGIN"
  aos_password = "$AOS_PASSWORD"
```

## Global Statistics (Platform)

Global statistics include information that is unrelated to any specific receiver. These statistics provide crucial information required for better planning of receivers. Whenever you reset the Apstra server, these global statistics are reset.

From the left navigation menu, navigate to **Platform > Streaming > Global Statistics** to see global statistics.

	alerts	events	perfmon
Users	420	888	1,796,213
Roles	75.86 KB	129.14 KB	229.03 MB
	0 messages/sec	0 messages/sec	20 messages/sec
	0 Bytes/sec	1.00 Bytes/sec	2.65 KB/sec

Last Fetched: 20

## Event Log (Platform)

### IN THIS SECTION

- [Event Log Overview | 725](#)
- [Export Event Log to CSV File | 727](#)
- [Send Event Log to External System | 727](#)

## Event Log Overview

Activity within the Apstra environment is recorded in an event log which you can use for auditing purposes. Events for the following event types are logged):

- User login (success and failure)
- User logout
- Blueprint commit (applies changes from staged to active blueprint)

- Blueprint commit (applies changes from staged to active blueprint)
- Blueprint rollback (rolls back the staged blueprint to a previous version)
- Blueprint deletion
- Device Config change
- Operation Mode changes (maintenance, normal, read-only)
- Changes to login banned/allowed list (new in release 4.1.1)

Each event includes the following information which is searchable and sortable:

- Time - when the event occurred (mouse over time field to see date and time)
- User - username of person who performed the activity
- Source IP - The source IP address of the client making the HTTP request
- Type - type of event (listed above)
- Device ID (as applicable) - typically the serial number of the managed device on which the action was taken
- Device Config (as applicable) - The config that is pushed and applied on the device
- Blueprint ID (as applicable) - The ID of the blueprint on which action was taken
- Blueprint name (as applicable) - The blueprint label on which action was taken
- Result - The outcome of the activity. Success means operation is accepted by the system. In the case of an error, the error string is included (unauthorized, for example)

From the left navigation menu, navigate to **Platform > Event Log** to go to the table of events that have been logged.

The screenshot shows the Juniper Apstra web interface. The left navigation menu has 'Platform' selected, and 'Event Log' is highlighted. The main area displays a table of events. Red arrows and text annotations guide the user: '1.' points to the 'Event Log' menu item, '2.' points to the 'Event Log' table, and 'Click device ID for details' points to a device ID in the table. Another arrow points to a 'View Config' button with the text 'Click to see device config'.

Device IP Address	Type	Result	Details
	DeviceConfigChange	Success	Device: 525400B7F346 View Config
	DeviceConfigChange	Success	Device: 52540036D734 View Config
	DeviceConfigChange	Success	Device: 52540036D734 View Config

You can search recent history for audit events.

- To filter the table, click **Query:All** and enter your query.
- To view device details (info, pristine config, telemetry), click a device ID.
- To view device configuration, click **View Config**.

Audit events are written to log-rotated files as a second repository. You can configure logrotate parameters in the Apstra server configuration file (/etc/aos/aos.conf). You can export and ship audit events to syslog.

## Export Event Log to CSV File

1. From the left navigation menu, navigate to **Platform > Event Log** and click **Export to CSV** (top-right).
2. To filter the data to export, enter your query.
3. Click **Save as CSV File** to download the CSV file.

## Send Event Log to External System

For details about sending the event log to an external system with the Syslog protocol, see "[Syslog Configuration](#)" on page 718.

## Apstra Cluster (Platform)

### IN THIS SECTION

- [Cluster Nodes | 728](#)
- [Cluster Management | 732](#)

You can monitor and manage different aspects of the Apstra environment, such as its configuration, usage, and containers. If your network includes many devices with off-box agents, or if you're taking advantage of Apstra's Intent Based Analytics feature, you might need more resources than can be provided from just one virtual machine (VM). To increase resource capacity, you can add worker node VMs to create a cluster with the Apstra controller node VM.

### Cluster Nodes

### IN THIS SECTION

- [Nodes Overview | 728](#)
- [Create Apstra Node | 731](#)
- [Edit Apstra Node | 731](#)
- [Delete Apstra Node | 731](#)

### Nodes Overview

When a worker VM is added to the main Apstra controller VM, it registers with the Apstra server VM through sysdb, collects facts about the VM (such as core/memory/disk configuration and usage), and launches a container of the local VM. The Apstra controller VM reacts to REST API requests, configures the worker VM for joining or leaving the cluster, and keeps track of cluster-wide runtime information. It also reacts to container configuration entities and schedules them to the worker VM.

Apstra VM nodes include the following details:

Table 29: Apstra VM Nodes Parameters

Name	Description
Address	IP address or Fully-Qualified Domain Name (FQDN) of the VM
Name	Apstra VM name, such as <b>controller</b> (the main Apstra controller node) or <b>worker - iba</b> (a worker node)
State	ACTIVE, MISSING, or FAILED
Roles	Controller or worker
Tags	The controller node and any worker nodes that you add are tagged with iba and offbox, by default. If you delete one or both of these tags or delete a worker node with one or both of these tags, any IBA and/or off-box containers in that node automatically move to a VM with those tags. Make sure there is another node with the tag(s) you're deleting or the containers will be deleted when you delete the tag or node.
Capacity Score	Calculated by Apstra software
CPU	Number of CPUs
Errors	As applicable. An example of an error is when an agent process has restarted because an agent has crashed.
Usage	<ul style="list-style-type: none"> <li>• Container Service Usage - Current VM container server usage (percentage)</li> <li>• Containers Count</li> <li>• Memory Usage (percentage)</li> <li>• CPU Usage (percentage)</li> <li>• Disk Usage - Current VM disk usage per logical volume (GB and percentage)</li> </ul>
Containers	The containers running on the node and the resources used by each container
Username/ Password	Apstra Server VM SSH username/password login credentials

From the left navigation menu, navigate to **Platform > Apstra Cluster** to go to Apstra nodes. Click a node address to see its details. You can create, clone, edit and delete Apstra nodes.

1. Platform

2. Apstra Cluster

State	Roles	Tags	Capacity Score	Containers Count	CPU	Memory Usage, Gb	CPU Usage	Disk Usage	Container Service Usage	Actions
ACTIVE	controller		160	4	4	6.24 (39%)	1%	7%	0%	Edit Clone Delete
ACTIVE	worker	lba	320	3	4	1.16 (7%)	0%	9%	12%	Edit Clone Delete
ACTIVE	worker	offbox	320	4	4	1.57 (10%)	10%	9%	4%	Edit Clone Delete

You have continuous visibility of platform health at the bottom left of every screen (GA in Apstra version 4.0.2 and Tech Preview in Apstra version 4.0.1). Green indicates the active state. Red indicates otherwise, such as a missing agent, the disk being in read only mode, or when an agent is rebooting (when agent has rebooted, the status returns to active). You can go directly to node details from any page by clicking one of the status indicators, then clicking the controller node or a worker node name.

1. Platform

2. Click a node to see details

Address	Name	Roles	Tags	Capacity Score	CPU
10.28.29.3	controller	controller		160	4

The feature above is classified as GA in Apstra version 4.0.2 and Technology Preview in Apstra version 4.0.1.

**NOTE:** This feature has been classified as a Juniper Apstra Technology Preview feature. These features are "as is" and for voluntary use. Juniper Support will attempt to resolve any issues that customers experience when using these features and create bug reports on behalf of support cases. However, Juniper may not provide comprehensive support services to Tech Preview features. For additional information, refer to ["Juniper Apstra Technology Previews" on page 1082](#) or contact ["Juniper Support" on page 777](#).

### Create Apstra Node

1. Install Apstra software on the VMs to be clustered, making sure they are all the same Apstra version as the main Apstra controller (which acts as the cluster manager). If they are not the same version, the controller will not accept them as part of the cluster.
2. From the left navigation menu, navigate to **Platform > Apstra Cluster** and click **Add Node**.
3. Enter a name, tags (optional), address (IP or FQDN), and Apstra Server VM SSH username/password login credentials. (iba and offbox tags are added by default.)
4. Click **Create**. As the main Apstra controller connects to the new Apstra VM worker node, the state of the new Apstra VM changes from **INIT** to **ACTIVE**.

### Edit Apstra Node

1. Either from the list view (Platform > Apstra Cluster) or the details view, click the **Edit** button for the VM to edit.
2. Make your changes. If you delete iba and/or offbox tags from the node, the IBA and/or off-box containers (as applicable) are moved to another node with those tags. Make sure the cluster has another node with those tags, or the containers will be deleted instead of moved.



**CAUTION:** To prevent containers from being deleted, don't delete tags unless another node in the cluster has the same tags.

3. Click **Update** to update the Apstra VM worker node.

### Delete Apstra Node

When you delete a node that includes iba and/or offbox tags, the IBA and/or off-box containers (as applicable) are moved to another node with those tags. Make sure the cluster has another node with those tags, or the containers will be deleted instead of moved.



**CAUTION:** To prevent containers from being deleted, don't delete nodes with `iba` and/or `offbox` tags unless another node in the cluster has the same tags.

1. Either from the list view (Platform > Apstra Cluster) or the details view, click the **Delete** button for the Apstra VM to delete.
2. Click **Delete** to delete the Apstra VM.

## Cluster Management

From the left navigation menu, navigate to **Platform > Apstra Cluster > Cluster Management** to go to Apstra cluster configuration and status (new in Apstra version 4.0.1).

Juniper Apstra™

Platform > Apstra Cluster > Cluster Management

Nodes Cluster Management

Expanded View Compact View

Configuration

Operation Mode<sup>®</sup> Normal

Status

Anomaly	Normal
Cluster	Normal
Device	Normal

User: admin

You have continuous visibility of platform health at the bottom left of every screen (GA in Apstra version 4.0.2 and Tech Preview in Apstra version 4.0.1). Green indicates the normal state. Red indicates otherwise, such as when it's in maintenance mode. You can go directly to cluster management details

from any page by clicking one of the status indicators, then clicking **Operation Mode**.

The screenshot displays the Juniper Apstra web interface for Cluster Management. The left sidebar contains navigation icons for Blueprints, Devices, Design, Resources, External Systems, Platform (highlighted), Favorites, and User: admin. The main area shows the 'Cluster Management' tab with 'Nodes' and 'Cluster Management' sub-tabs. Under the 'Configuration' section, the 'Operation Mode' is set to 'Normal'. Under the 'Status' section, a table shows 'Anomaly', 'Cluster', and 'Device' all with 'Normal' status. A red arrow labeled '1.' points to the 'Operation Mode' dropdown, and another red arrow labeled '2.' points to the 'Platform' icon in the sidebar.

The feature above is classified as GA in Apstra version 4.0.2 and Technology Preview in Apstra version 4.0.1.

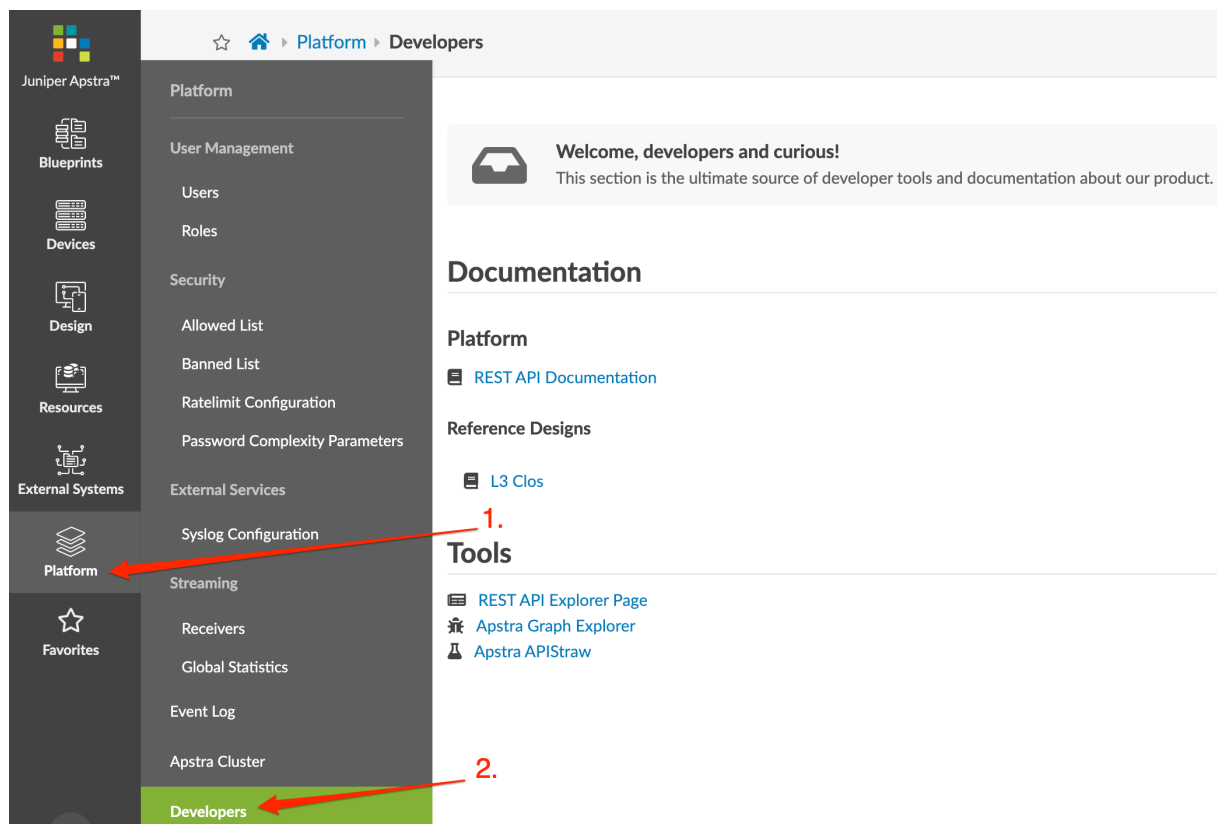
**NOTE:** This feature has been classified as a Juniper Apstra Technology Preview feature. These features are "as is" and for voluntary use. Juniper Support will attempt to resolve any issues that customers experience when using these features and create bug reports on behalf of support cases. However, Juniper may not provide comprehensive support services to Tech Preview features. For additional information, refer to ["Juniper Apstra Technology Previews" on page 1082](#) or contact ["Juniper Support" on page 777](#).

## Developers (Platform)

### IN THIS SECTION

- [Resource Pools \(API\) | 735](#)
- [Configlets \(API\) | 746](#)
- [Property Sets \(API\) | 749](#)
- [Interface Descriptions \(API\) | 751](#)
- [Probes \(API\) | 755](#)
- [RCI Fault Model \(API\) | 769](#)
- [Apstra Cluster \(API\) | 773](#)
- [API From Python | 774](#)
- [REST API Explorer | 776](#)

From the left navigation menu, navigate to **Platform > Developers** to go to developer documentation and tools.



The **Documentation** section includes links to Apstra in-product API documentation.

- **Platform REST API Documentation** includes API documentation for APIs used outside of Apstra blueprints (such as Apstra global catalog logical devices).
- **Reference Designs L3 Clos** includes API documentation for APIs used in standard Apstra L3 Clos blueprints (such as Apstra blueprint virtual networks).

## Resource Pools (API)

### IN THIS SECTION

- [API - ASN Pools | 735](#)
- [API - IP Pools | 740](#)

This reference demonstrates the resource group API usage with parity to the UI. For full API documentation, view the REST Platform API reference under the Apstra web interface.

To list resource group slots in a blueprint, perform an authenticated HTTP GET to [https://aos-server/api/blueprints/<blueprint\\_id>/resource\\_groups](https://aos-server/api/blueprints/<blueprint_id>/resource_groups)

Both **ASN pools** and **IP pools** must be assigned in order for a blueprint to complete the build phase.

### API - ASN Pools

#### Create ASN Pool

An example payload for creating an ASN Pool:

If an ID is not specified, one will be created and returned in the HTTP response.

```
{
  "id": "RFC6996-Private",
  "display_name": "RFC6996-Private",
  "tags": [ "default" ],
  "ranges": [
    {
      "last": 65534,
      "first": 64512
    }
  ]
}
```

```
]
}
```

To create an ASN pool perform an HTTP POST to <https://aos-server/api/resources/asn-pools> with a JSON payload.

```
curl 'https://192.168.25.250/api/resources/asn-pools?comment=create'
-H 'AuthToken: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6
ImFkbWluIiwiaXNlZGF9hdCI6IjIwMTctMDUzMzFUMDA6MjI6MDcuNTIwMTgzWiIsIn
Nlc3Npb24iOiJjOTliOGVlOS05Y2NjLTRjZTAtYTU5NS0wODI3N2ZkYjA0ZDYifQ.FnJMR3
crPoD0-lQRXnpPOJ8TCsRG9Wr-DaddnAIj6ko' - --data-binary '{"display_name"
:"Example","ranges":[{"first":100,"last":200}], "tags":[]}' --compressed
--insecure
```

## List ASN Pools

```
curl 'https://192.168.25.250/api/resources/asn-pools' -H 'AuthToken:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFkbWluIiwiaXNl
ZGF9hdCI6IjIwMTctMDUzMzFUMDA6MjI6MDcuNTIwMTgzWiIsInNlc3Npb24iOiJjOTliO
GVlOS05Y2NjLTRjZTAtYTU5NS0wODI3N2ZkYjA0ZDYifQ.FnJMR3crPoD0-lQRXnpPOJ8TCsR
G9Wr-DaddnAIj6ko' --compressed --insecure
```

```
{
  "items": [
    {
      "created_at": "2017-05-30T12:56:07.293082Z",
      "display_name": "Private ASN",
      "id": "c23ea447-8f37-419a-9b1c-c48cc55d5b9c",
      "last_modified_at": "2017-05-30T12:56:07.293082Z",
      "ranges": [
        {
          "first": 65412,
          "last": 65534,
          "status": "pool_element_in_use"
        }
      ],
      "status": "in_use",
      "tags": []
    }
  ]
}
```

```
]
}
```

## Delete ASN Pool

To delete an ASN Pool perform an HTTP DELETE to [https://aos-server/resources/asn-pools/{pool\\_id}](https://aos-server/resources/asn-pools/{pool_id})

A successful DELETE returns HTTP 200 OK.

```
curl
'https://192.168.25.250/api/resources/asn-pools/d0312b4a-017e-4478-8b8d-df0417ce8d3b'
-X DELETE -H 'AuthToken: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2Vybm
FtZSI6ImFkbWluIiwiaXNlc3Npb24iOiJjOTliOGVlOS05Y2NjLTRjZTAyYTY5NS0wODI3N2ZkYjA0ZDYifQ.FnJ
iIsInNlc3Npb24iOiJjOTliOGVlOS05Y2NjLTRjZTAyYTY5NS0wODI3N2ZkYjA0ZDYifQ.FnJ
MR3crPoD0-lQRXnpPJ8TCsRG9Wr-DaddnAIj6ko' --compressed --insecure
```

## Assign ASN to Blueprint

To assign an IP pool to the blueprint perform an HTTP PUT to [https://aos-server/blueprints/<blueprint\\_id>/resource\\_groups/ip/<pool\\_name>](https://aos-server/blueprints/<blueprint_id>/resource_groups/ip/<pool_name>)

For instance, to post a resource pool to **spine\_loopback\_ips**, first obtain the ID of the resource pool, and append it to a list for slot assignation. When updating the IP Pool resource group, specify all pools in the payload at the same time. We cannot add single pools, so PUT them all at once.

Payload:

```
{"pool_ids": ["pool_id1", "pool_id2", "pool_id3"] }
```

```
curl
'https://192.168.25.250/api/blueprints/4c1e69c6-97bd-4c99-9504-7818f138b17f/resource_groups/asn/
spine_asns'
-X PUT -H 'AuthToken: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2Vybm
mFtZSI6ImFkbWluIiwiaXNlc3Npb24iOiJjOTliOGVlOS05Y2NjLTRjZTAyYTY5NS0wODI3N2ZkYj
wMTgzWiIsInNlc3Npb24iOiJjOTliOGVlOS05Y2NjLTRjZTAyYTY5NS0wODI3N2ZkYj
A0ZDYifQ.FnJMR3crPoD0-lQRXnpPJ8TCsRG9Wr-DaddnAIj6ko' --data-binary
'{"pool_ids":["c23ea447-8f37-419a-9b1c-c48cc55d5b9c"]}' --compressed --insecure
```

A successful ASSIGNMENT returns HTTP 200 OK.



```

        "pool_ids": [
            "c23ea447-8f37-419a-9b1c-c48cc55d5b9c"
        ],
        "type": "asn"
    },
    {
        "name": "spine_asns",
        "pool_ids": [
            "c23ea447-8f37-419a-9b1c-c48cc55d5b9c"
        ],
        "type": "asn"
    },
    {
        "name": "leaf_loopback_ips",
        "pool_ids": [
            "56e8e0dc-babd-4652-92a5-fc37294a7b26"
        ],
        "type": "ip"
    },
    {
        "name": "mlag_domain_svi_subnets",
        "pool_ids": [
            "ed7d8830-c703-4ac0-8252-77e0f272a677"
        ],
        "type": "ip"
    },
    {
        "name": "spine_leaf_link_ips",
        "pool_ids": [
            "ed7d8830-c703-4ac0-8252-77e0f272a677"
        ],
        "type": "ip"
    },
    {
        "name": "spine_loopback_ips",
        "pool_ids": [
            "56e8e0dc-babd-4652-92a5-fc37294a7b26"
        ],
        "type": "ip"
    }
]
}

```

## API - IP Pools

### Create IP Pool

JSON Payload for creating an IP Pool:

```
{
  "id": "example_ip_pool",
  "display_name": "example_ip_pool",
  "tags": ["default"],
  "subnets": [
    {"network": "10.0.0.0/8"}
  ]
}
```

The **subnets** section requires a list of dictionaries with keyword **network** and value matching a CIDR mask. The subnets cannot overlap with each other in the same pool. That is to say, 192.168.10.0/24 and 192.168.0.0/16 cannot be configured in the same pool.

Tags are optional and are not currently used in Apstra. If ID is specified, it will be saved, otherwise an ID will be returned in the HTTP Response after creating the pool.

An HTTP POST to <https://aos-server/api/resources/ip-pools> with JSON payload will reply with the ID of the new IP pool.

```
curl 'https://192.168.25.250/api/resources/ip-pools' -X
POST -H 'AuthToken: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybWludGZlYXRlZF9hdCI6IjIwMTctMDUzMzFUMDA6MjI6MDcuNTIwMTgzWiIsInNlc3Npb24iOiJjOTliOGVlOS05Y2NjLTRjZTA0ZDYifQ.FnJMR3crPoD0-lQRXnpPOJ8TCsRG9Wr-DaddnAIj6ko' --data-binary '{"display_name":
"example_ip_pool","subnets":[{"network":"10.0.0.0/8"}, {"network":
"192.168.0.0/16"}],"tags":[]}' --compressed --insecure
```

```
{"id": "d0312b4a-017e-4478-8b8d-df0417ce8d3b"}
```

## List IP Pools

Perform an HTTP GET to <https://aos-server/api/resources/ip-pools> -

```
jp@ApstraVM ~ $ curl 'https://192.168.25.250/api/resources/ip-pools' -H
'AuthToken: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFkbW
luIiwiaWY3JlYXRlZF9hdCI6IjIwMTctMDUzMzFUMDA6MjI6MDcuNTIwMTgzWiIsInNlc3Npb24
iOiJjOTliOGVlOS05Y2NjLTRjZTAtYTU5NS0wODI3N2ZkYjA0ZDYifQ.FnJMR3crPoD0-lQRXnpP
OJ8TCsRG9Wr-DaddnAIj6ko' --compressed --insecure | python -m json.tool
```

```
{
  "items": [
    {
      "created_at": "2017-05-31T03:48:38.562331Z",
      "display_name": "example_ip_pool",
      "id": "d5046aa6-eab2-4990-9816-0a519ce1a8db",
      "last_modified_at": "2017-05-31T03:48:38.562331Z",
      "status": "not_in_use",
      "subnets": [
        {
          "network": "10.0.0.0/8",
          "status": "pool_element_available"
        },
        {
          "network": "192.168.0.0/16",
          "status": "pool_element_available"
        }
      ],
      "tags": []
    },
    {
      "created_at": "2017-05-30T12:56:50.576598Z",
      "display_name": "L3-CLOS",
      "id": "ed7d8830-c703-4ac0-8252-77e0f272a677",
      "last_modified_at": "2017-05-30T12:56:50.576598Z",
      "status": "in_use",
      "subnets": [
        {
          "network": "10.16.0.0/16",
          "status": "pool_element_in_use"
        }
      ]
    }
  ]
}
```

```

    ],
    "tags": []
  },
  {
    "created_at": "2017-05-30T12:56:24.222906Z",
    "display_name": "Loopbacks",
    "id": "56e8e0dc-babd-4652-92a5-fc37294a7b26",
    "last_modified_at": "2017-05-30T12:56:24.222906Z",
    "status": "in_use",
    "subnets": [
      {
        "network": "10.254.0.0/16",
        "status": "pool_element_in_use"
      }
    ],
    "tags": []
  },
  {
    "created_at": "2017-05-31T03:49:15.485164Z",
    "display_name": "example_ip_pool",
    "id": "d0312b4a-017e-4478-8b8d-df0417ce8d3b",
    "last_modified_at": "2017-05-31T03:49:15.485164Z",
    "status": "not_in_use",
    "subnets": [
      {
        "network": "10.0.0.0/8",
        "status": "pool_element_available"
      },
      {
        "network": "192.168.0.0/16",
        "status": "pool_element_available"
      }
    ],
    "tags": []
  }
]
}

```

## Delete IP pool

To delete an IP Pool perform an HTTP DELETE to [https://aos-server/resources/ip-pools/{pool\\_id}](https://aos-server/resources/ip-pools/{pool_id})

A successful DELETE returns HTTP 200 OK and an empty JSON response {}

```
curl
'https://192.168.25.250/api/resources/ip-pools/d0312b4a-017e-4478-8b8d-df0417ce8d3b'
-X DELETE -H 'AuthToken: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFkbWluIiwia3JlYXRlZF9hdCI6IjIwMTctMDU0MzFUMDA6MjI6MDcuNTIwMTgzWiIsInNl
c3Npb24iOiJjOTliOGVlOS05Y2NjLTRjZTA0YTY5NS0wODI3N2ZkYjA0ZDYifQ.FnJMR3crPoD0
-lQRXnp0J8TCsRG9Wr-DaddnAIj6ko' --compressed --insecure
```

## Assign IP to Blueprint

To assign an IP pool to the blueprint perform an HTTP PUT to [https://aos-server/blueprints/<blueprint\\_id>/resource\\_groups/ip/<group\\_name>](https://aos-server/blueprints/<blueprint_id>/resource_groups/ip/<group_name>)

For instance, to associate a resource pool **spine\_loopback\_ips** with a blueprint first obtain the ID of the resource pool, and append it to a list for slot assignation. When updating the IP Pool resource group, specify all pools in the payload at the same time. We cannot add single pools, so PUT them all at once. Instruct Apstra to associate IP pool with ID 'ed7d8830-c703-4ac0-8252-77e0f272a677' to the blueprint. You may have to GET existing pool IDs prior to adding a new one to avoid deleting existing pools.

Payload:

```
{"pool_ids": ["pool_id1", "pool_id2", "pool_id3"] }
```

```
curl
'https://192.168.25.250/api/blueprints/4c1e69c6-97bd-4c99-9504-7818f138b17f/resource_groups/ip/
spine_loopback_ips'
-X PUT -H 'AuthToken: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImF
kbWluIiwia3JlYXRlZF9hdCI6IjIwMTctMDU0MzFUMDA6MjI6MDcuNTIwMTgzWiIsInNl
c3Npb24iOiJjOTliOGVlOS05Y2NjLTRjZTA0YTY5NS0wODI3N2ZkYjA0ZDYifQ.FnJMR3crPoD0-lQRXnp
0J8TCsRG9Wr-DaddnAIj6ko' --data-binary '{"pool_ids":["ed7d8830-c703-4ac0-825
2-77e0f272a677"]}' --compressed --insecure
```

A successful ASSIGNMENT returns an HTTP 200 OK.

## Remove IP from Blueprint

To remove IP pools from the blueprint PUT an empty pool\_id list to the blueprint with the payload []:

PUT to the HTTP endpoint [https://aos-server/api/blueprints/<blueprint\\_id>/resource\\_groups/ip/<allocation\\_group\\_name>](https://aos-server/api/blueprints/<blueprint_id>/resource_groups/ip/<allocation_group_name>)

With the payload:

```
{ "pool_ids": [] }
```

## CURL Example

```
curl
'https://192.168.25.250/api/blueprints/4c1e69c6-97bd-4c99-9504-7818f138b17f/resource_groups/ip/
spine_loopback_ips'
-X PUT -H 'AuthToken: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZ
SI6ImFkbWluIiwia3JlYXRlZF9hdCI6IjIwMTctMDUzMzFUMDA6MjI6MDcuNTIwMTgzWiIsI
nNlc3Npb24iOiJjOTI0GVlOS0S0Y2NjLTRjZTA0YTY5NS0wODI3N2ZkYjA0ZDYifQ.FnJMR3cr
PoD0-LQRXnpPOJ8TCsRG9Wr-DaddnAIj6ko' --data-binary '{"pool_ids":[]}'
--compressed --insecure
```

A successful REMOVAL returns an empty response: {}

### List IPs Assigned to Blueprint

```
curl
'https://192.168.25.250/api/blueprints/4c1e69c6-97bd-4c99-9504-7818f138b17f/resource_groups'
-H 'AuthToken: eyJhbGciOiJIUzI1NiMTctMDUtMzFUMDA6MjI6MDcuNTIwMTgzWiIsInNlc3
Npb24iOiJjOTliOGVlOS05Y2NjLTRjZTAtYTty5NS0wODI3N2ZkYjA0ZDYifQ.FnJMR3crPoD
0-lQRXnp0J8TCsRG9Wr-DaddnAIj6ko' --compressed --insecure | python -m json.tool
```

```
{
  "items": [
    {
      "name": "leaf_asns",
      "pool_ids": [
        "c23ea447-8f37-419a-9b1c-c48cc55d5b9c"
      ],
      "type": "asn"
    },
    {
      "name": "spine_asns",
```

```

        "pool_ids": [
            "c23ea447-8f37-419a-9b1c-c48cc55d5b9c"
        ],
        "type": "asn"
    },
    {
        "name": "leaf_loopback_ips",
        "pool_ids": [
            "56e8e0dc-babd-4652-92a5-fc37294a7b26"
        ],
        "type": "ip"
    },
    {
        "name": "mlag_domain_svi_subnets",
        "pool_ids": [
            "ed7d8830-c703-4ac0-8252-77e0f272a677"
        ],
        "type": "ip"
    },
    {
        "name": "spine_leaf_link_ips",
        "pool_ids": [
            "ed7d8830-c703-4ac0-8252-77e0f272a677"
        ],
        "type": "ip"
    },
    {
        "name": "spine_loopback_ips",
        "pool_ids": [
            "56e8e0dc-babd-4652-92a5-fc37294a7b26"
        ],
        "type": "ip"
    }
]
}

```

## Configlets (API)

### IN THIS SECTION

- [API - Create Configlet | 747](#)
- [API - Delete Configlet | 747](#)
- [API - Assign Configlet | 747](#)
- [CURL Example - HTTP PUT | 748](#)
- [API - Unassign Configlet | 749](#)

For full API documentation, view the Platform API reference from the web interface. This is a targeted section to demonstrate configlet API similarly to the UI. The main difference between the Web UI and REST API is that the Apstra API does not make any use of the configlets stored under `api/design/` configlets when working with a blueprint. Design-configlets are meant for consumption under the UI. When working with configlets on the API, work directly with the blueprint.

Configlets live in <http://aos-server/api/design/configlets> and are referenced by ID.

```
{
  "ref_archs": [
    "two_stage_l3clos"
  ],
  "created_at": "string",
  "last_modified_at": "string",
  "id": "string",
  "generators": [
    {
      "config_style": "string",
      "template_text": "string",
      "negation_template_text": "string"
    }
  ],
  "display_name": "string",
  "section": "string"
}
```

## API - Create Configlet

To create a configlet, POST to <https://aos-server/api/design/configlets> with a valid JSON structure representing the configlet. Creating a configlet this way only allows it to be available for assignment in the web interface - it is not required in this method for the REST API to assign to a blueprint. See the assigning a configlet section for more details.

A POST will create a new configlet. A PUT will overwrite an existing configlet. PUT requires the URL of the configlet. <https://aos-server/api/design/configlets/{id}>

```
curl -H "AuthToken: EXAMPLE" -d '{"display_name":"DNS","ref_archs":
["two_stage_l3clos"],"section":"system","generators":[{"config_style":"eos","template_text":"ip
name-server 192.168.1.1","negation_template_text":"no ip name-server 192.168.1.1"}]}' -X POST
"http://aos-server/api/design/configlets"
```

The response will contain the ID of the newly created configlet {"id": "995446c7-de7d-46bb-a88a-786839556064"}

## API - Delete Configlet

Deleting a configlet requires an HTTP DELETE to the configlet by URL <http://aos-server/api/design/configlets/{id}>

```
curl -H "AuthToken: EXAMPLE" -X DELETE "http://aos-server/api/design/configlets/995446c7-
de7d-46bb-a88a-786839556064"
```

A successful DELETE has an empty response {}

## API - Assign Configlet

Assigning a configlet to a blueprint requires assignment of device conditions as well as embedding the configlet details. When assigning a configlet to a blueprint, the configlets available as design resources aren't necessary. These are only used for UI purposes.

The assigned configlet lives in [https://aos-server/api/blueprints/blueprint\\_id/configlets](https://aos-server/api/blueprints/blueprint_id/configlets)

JSON Syntax for putting a configlet to a blueprint. Basically, this is just an 'items' dictionary element containing a list of configlet schemas.

```
{
  "items": [
    {
```

```

    "template_params": [
      "string"
    ],
    "configlet": {
      "generators": [
        {
          "config_style": "string",
          "template_text": "string",
          "negation_template_text": "string"
        }
      ],
      "section": "string",
      "display_name": "string"
    },
    "condition": "string"
  }
]
}

```

### CURL Example - HTTP PUT

```

curl "http://aos-server/api/blueprints/e4068e99-813c-4290-b7cc-e145d85a98a8/configlets" -X PUT -
H "AuthToken: EXAMPLE" -H "Content-Type: application/json; charset=utf-8" --data
"[{"configlet":{"generators":[{"config_style":"eos","template_text":"ip name-server
192.168.1.1","negation_template_text":"no ip name-server
192.168.1.1"}],"section":"system","display_name":"DNS"},"condition":"role==spine"},
{"configlet":{"generators":[{"config_style":"eos","template_text":"ip name-server
192.168.1.1","negation_template_text":"no ip name-server
192.168.1.1"}],"section":"system","display_name":"DNS"},"condition":"role==leaf"}]"

```

### Response

```

{"items": [{"configlet": {"generators": [{"config_style": "eos", "template_text": "ip name-
server 192.168.1.1", "negation_template_text": "no ip name-server 192.168.1.1"}], "section":
"system", "display_name": "DNS"}, {"condition": "role==spine"}, {"configlet": {"generators":
[{"config_style": "eos", "template_text": "ip name-server 192.168.1.1",
"negation_template_text": "no ip name-server 192.168.1.1"}], "section": "system",
"display_name": "DNS"}, {"condition": "role==leaf"}]}

```

## API - Unassign Configlet

To unassign a configlet, remove it from the items list by PUT with an empty json post.

```
curl "http://aos-server/api/blueprints/e4068e99-813c-4290-b7cc-e145d85a98a8/configlets" -X PUT -H "AuthToken: EXAMPLE" -H "Content-Type: application/json; charset=utf-8" --data ""
```

The response will be an empty json set once the configlet is deleted: {"items": []}

## Property Sets (API)

### IN THIS SECTION

- [API - Create Property Set | 750](#)
- [API - Delete Property Set | 750](#)
- [API - Assign Property Set | 750](#)
- [CURL Example - API HTTP PUT | 751](#)
- [API - Unassign Property Set | 751](#)

For full API documentation, view the Platform API reference from the web interface. This is a targeted section to demonstrate property sets API similarly to the web interface.

Property sets live in <http://aos-server:8888/api/property-sets> and are referenced by ID.

```
{
  "items": [
    {
      "label": "string",
      "values": {
        "additionalProp1": "string",
        "additionalProp2": "string",
        "additionalProp3": "string"
      },
      "id": "string"
    }
  ]
}
```

## API - Create Property Set

To create a property set, POST to <https://aos-server/api/property-sets> with a valid JSON structure representing the property set. Creating a property set this way only allows it to be available for assignment in the web interface - it is not required in this method for the REST API to assign to a blueprint. See the assigning a property set section for more details.

A POST will create a new property set. A PUT will overwrite an existing property set. PUT requires the URL of the property set. <https://aos-server:8888/api/design/property-sets/{id}>

```
curl -H "AuthToken: EXAMPLE" -d '{"values": {"NTP_SRV1": "192.168.1.1", "NTP_SRV1": "192.168.1.1"}, "label": "NTP-servers"}' -X POST "http://aos-server:8888/api/design/property-sets"
```

The response will contain the ID of the newly created property-set {"id": "73223e81-a451-4e7f-91fb-fb476f4b9fc8"}

## API - Delete Property Set

Deleting a property set requires an HTTP DELETE to the property set by URL <http://aos-server:8888/api/design/property-sets/{id}>

```
curl -H "AuthToken: EXAMPLE" -X DELETE "http://aos-server:8888/api/design/property-sets/73223e81-a451-4e7f-91fb-fb476f4b9fc8"
```

A successful DELETE has an empty response {}

## API - Assign Property Set

Assigning a property set to a blueprint requires an HTTP POST to the blueprint by URL [http://aos-server:8888/api/blueprints/{blueprint\\_ID}/property-sets](http://aos-server:8888/api/blueprints/{blueprint_ID}/property-sets)

```
{
  "id": "73223e81-a451-4e7f-91fb-fb476f4b9fc8"
}
```

The response will contain the ID of the assigned property-sets {"id": "73223e81-a451-4e7f-91fb-fb476f4b9fc8"}

## CURL Example - API HTTP PUT

```
curl "http://aos-server:8888/api/blueprints/e4068e99-813c-4290-b7cc-e145d85a98a8/property-sets/73223e81-a451-4e7f-91fb-fb476f4b9fc8" -X DELETE -H "AuthToken: EXAMPLE"
```

### Response

```
{"id": "73223e81-a451-4e7f-91fb-fb476f4b9fc8"}
```

## API - Unassign Property Set

Deleting a property set requires an HTTP DELETE to the blueprint property set by URL [http://aos-server:8888/api/blueprints/{blueprint\\_ID}/property-sets{id}](http://aos-server:8888/api/blueprints/{blueprint_ID}/property-sets{id})

```
curl "http://aos-server:8888/api/blueprints/e4068e99-813c-4290-b7cc-e145d85a98a8/property-sets/73223e81-a451-4e7f-91fb-fb476f4b9fc8" -X DELETE -H "AuthToken: EXAMPLE"
```

A successful DELETE has an empty response {}

## Interface Descriptions (API)

### IN THIS SECTION

- [Apstra REST API - Interface descriptions | 752](#)

Besides main parameters of network interfaces like name, speed and port mode, Apstra also configures a description for physical interfaces and aggregated logical interfaces (so called port channels). Interface description is automatically generated if the following conditions are met:

1. The interface is connected to a peer.
2. The interface belongs to leaf, spine or generic system.
3. The peer interface belongs to leaf, spine, or generic system with virtual network endpoint on this server.

The generated description has the form <facing\_|to.>peer-device-label>[:peer-interface-name]. Examples:

- facing\_spine2:Ethernet1/2

- to.server1:eth0
- to.server2

The prefix of the name is `facing_` if the peer is leaf, spine or external router. The prefix is `to.` in case peer device is an L2 or L3 server. The peer interface name part is present only when the peer device is controlled by Apstra.

### Apstra REST API - Interface descriptions

The Apstra API is able to change the auto-generated interface description. However, there is no such functionality in the Apstra UI.

The interface description may contain ASCII characters with codes 33-126 and spaces, except "?", which is interpreted as a command-completion. The description length is limited to 240 characters, which is the longest possible length across supported switch models.

Interfaces are stored internally as graph nodes with certain set of properties. Description is one of these properties. To modify the description, use the generic API to interact with graph nodes.

### API - Obtain interface configuration

To obtain interface configuration, send GET request to <https://aos-server/api/blueprints/{blueprint-id}/nodes/{interface-node-id}>.

Request:

```
{
  "description": "facing_dkl-2-leaf:Ethernet1/2",
  "mlag_id": null,
  "tags": null,
  "if_name": "swp2",
  "label": null,
  "port_channel_id": null,
  "ipv4_addr": "203.0.113.10/31",
  "mode": null,
  "if_type": "ip",
  "type": "interface",
  "id": "interface-id-1",
  "protocols": "ebgp"
}
```

## API - Create or modify interface description

To create or modify interface description, send PATCH request to <https://aos-server/api/blueprints/{blueprint-id}/nodes/{interface-node-id}> with a valid JSON. The JSON should contain the "description" field with a valid data.

```
curl -X PATCH -H "AuthToken: EXAMPLE" \
  -d '{"description": "New description I want!"}'
http://aos-server:8888/api/blueprints/id-1/nodes/interface-id-1
```

Response:

```
{
  "description": "New description I want!",
  "mlag_id": null,
  "tags": null,
  "if_name": null,
  "label": null,
  "port_channel_id": null,
  "ipv4_addr": null,
  "mode": null,
  "if_type": "ip",
  "type": "interface",
  "id": "interface-id-1",
  "protocols": "ebgp"
}
```

## API - Delete interface description

To delete custom interface description and get back to automatic description generation, set the description to empty value.

Request:

```
curl -X PATCH -H "AuthToken: EXAMPLE" \
  -d '{"description": ""}'
http://aos-server:8888/api/blueprints/id-1/nodes/interface-id-1
```

Response:

```
{
  "description": "",
  "mlag_id": null,
  "tags": null,
  "if_name": null,
  "label": null,
  "port_channel_id": null,
  "ipv4_addr": null,
  "mode": null,
  "if_type": "ip",
  "type": "interface",
  "id": "interface-id-1",
  "protocols": "ebgp"
}
```

Subsequent GET request will show that the description was automatically generated.

Request:

```
curl -H "AuthToken: EXAMPLE" \
  http://aos-server:8888/api/blueprints/id-1/nodes/interface-id-1
```

Response:

```
{
  "description": "facing_dkl-2-leaf:Ethernet1/2",
  "mlag_id": null,
  "tags": null,
  "if_name": "swp2",
  "label": null,
  "port_channel_id": null,
  "ipv4_addr": "203.0.113.10/31",
  "mode": null,
  "if_type": "ip",
  "type": "interface",
  "id": "interface-id-1",
  "protocols": "ebgp"
}
```

## Probes (API)

### IN THIS SECTION

- [Generic Probe REST API | 755](#)
- [Create Probe | 755](#)
- [Inspect Probe | 761](#)
- [Query Probe Anomalies | 764](#)
- [Introspect Processors | 765](#)
- [Stream Data | 768](#)

### Generic Probe REST API

The information below describes as much of the API as necessary to understand how to use IBA for someone already familiar with Apstra API conventions. Formal API documentation is reserved for the API documentation itself.

We will walk through the API as it's used for the example workflow described in the introduction, demonstrating its general capability by specific example.

### Create Probe

To create a probe, the operator POSTs to `/api/blueprints/<blueprint_id>/probes` with the following form:

```
{
  "label": "server_tx_bytes",
  "description": "Server traffic imbalance",
  "tags": ["server", "imbalance"],
  "disabled": false,
  "processors": [
    {
      "name": "server_tx_bytes",
      "outputs": {
        "out": "server_tx_bytes_output"
      },
      "properties": {
        "counter_type": "tx_bytes",
        "graph_query": "node('system',
```

```

name='sys').out('hosted_interfaces').node('interface', name='intf').out('link').node('link',
link_type='ethernet', speed=not_none()).in_('link').node('interface',
name='dst_intf').in_('hosted_interfaces').node('system', name='dst_node',
role='server').ensure_different('intf', 'dst_intf')",
        "interface": "intf.if_name",
        "system_id": "sys.system_id"
    },
    "type": "if_counter"
},
{
    "inputs": {
        "in": "server_tx_bytes_output"
    },
    "name": "std",
    "outputs": {
        "out": "std_dev_output"
    },
    "properties": {
        "ddof": 0,
        "group_by": []
    },
    "type": "std_dev"
},
{
    "inputs": {
        "in": "std_dev_output"
    },
    "name": "server_imbalance",
    "outputs": {
        "out": "std_dev_output_in_range"
    },
    "properties": {
        "range": {
            "max": 100
        }
    },
    "type": "range_check"
},
{
    "inputs": {
        "in": "std_dev_output_in_range"
    },
    "name": "server_imbalance_anomaly",

```

```

        "outputs": {
            "out": "server_traffic_imbalanced"
        },
        "type": "anomaly"
    }
],
"stages": [
    {
        "name": "server_tx_bytes_output",
        "description": "Collect server tx_bytes",
        "tags": ["traffic counter"],
        "units": "Bps"
    }
]
}

```

As seen above, the endpoint is given an input of probe metadata, a processor instance list, and output stage list.

Probe metadata is composed of the following fields:

<b>label</b>	human-readable probe label; required,
<b>description</b>	optional description of the probe,
<b>tags</b>	list of strings with the probe tags; optional,
<b>disabled</b>	optional boolean that tells whether probe should be disabled. Disabled probes don't provide any data and don't consume any resources. The probe is not disabled by default.

Each processor instance contains an instance name (defined by user), processor type (a selection from a catalog defined by the platform and the reference design), and inputs and/or outputs. All additional fields in each processor are specific to that type of processor, are specified in the properties sub-field, and can be learned by introspection via our introspection API at `/api/blueprints/<blueprint_id>/telemetry/processors`; we will go over this API later.

Matching our working example, we will go through each entry we have in the processor list in the above example.

In the first entry, we have a processor instance of type `if_counter` that we name `server_tx_bytes`. It takes as input a query called `graph_query` which is a graph query. It then has two other fields named `interface` and `system_id`. These three fields together indicate that we want to collect a (first time-derivative of) counter for every server-facing port in the system. For every match of the query specified by `graph_query`, we extract a `system_id` by taking the `system_id` field of the `sys` node in the resulting path (as specified in the `system_id` processor field) and an interface name by taking the `if_name` field of the `intf` node in the resulting

path (as specified in the `interface` processor field). The combination of system ID and interface is used to identify an interface in the network, and its `tx_bytes` counter (as specified by `counter_type`) is put into the output of this processor. The output of this processor is of type "Number Set" (NS); stage types are discussed exhaustively later. This processor has no inputs, so we do not supply an `input` field. It has one output, labeled `out` (as defined by the `if_counter` processor type); we map that output to a stage labeled `server_tx_bytes_output`.

The second processor is of type `std_dev` and takes as input the stage we created before called `server_tx_bytes_output`; see the processor-specific documentation for the meaning of the `ddof` field. Also, see the processor-specific documentation for the full meaning of the `group_by` field. It will suffice to say for now that in this case `group_by` tells us to construct a single output "Number" (N) from the input NS; that is, this processor outputs a single number-the standard deviation taken across each of the many input numbers. This output is named "std\_dev\_output".

The third processor is of type `range_check` and takes as input `std_dev_output`. It checks that the input is out of the expected range specified by `range` - in this case if the input is ever greater-than 100 (we have chosen this arbitrary value to indicate when the server-directed traffic is unbalanced). This processor has a single output we choose to label `std_dev_output_in_range`. This output (as defined by the `range_check` processor type) is of type DS (Discrete State) and can take values either `true` or `false`, indicating whether or not a value is out of the range.

Our final processor is of type `anomaly` and takes as input `std_dev_output_in_range`. It raises an Apstra anomaly when the input is in the `true` state. This processor has a single output we choose to label `server_traffic_imbalanced`. This output (as defined by the `anomaly` processor type) is of type DS (Discrete State) and can take values either `true` or `false`, indicating whether or not an anomaly is raised. We do not do any further processing with this anomalous state data in this example, but that does not preclude its general possibility.

Finally, we have a `stages` field. This is a list of a subset of output stages, with each stage indicated by the `name` field which refers to the stage label. This list is meant to add metadata to each output stage that cannot be inferred from the DAG itself. Currently, supported fields are:

<b>description</b>	string with a stage description,
<b>tags</b>	list of strings that make a set of tags for stage,
<b>units</b>	string that is meant to describe the units of the stage data.

All these fields are optional.

This stage metadata is returned when fetching data from that stage via the REST API and used by the GUI in visualization.

HTTP POST can be sent to `/api/blueprints/<blueprint_id>/probes`. Here, we POST probe configuration, as exemplified in the "POST for Probe Creation" figure to create a new probe. POSTing to this endpoint will

return a UUID, as most of the other creation endpoints in Apstra, which can be used for further operations.

Changed in version 2.3: To get a predictable probe id instead of a UUID described above, one could specify it by adding an "id" property to the request body.

```
{
  "id": "my_tx_bytes_probe",
  "label": "server_tx_bytes",
  "processors": [],
  "rest_of_the": "request_body"
}
```

Changed in version 2.3: Previously, stage definitions were inlined into processor definitions like this:

```
{
  "label": "test probe",
  "processors": [
    {
      "name": "testproc",
      "outputs": {"out": "test_stage"},
      "stages": [{"name": "out", "units": "pps"}]
    }
  ]
}
```

This no longer works, and stage name should refer to the stage label instead of the internal stage name. So the example above should look this way:

```
{
  "stages": [{"name": "test_stage", "units": "pps"}]
}
```

Additional note: it's recommended not to inline stage definitions into processor definitions, and place that as a stand-alone element like in POST example above.

HTTP DELETE can be sent to `/api/blueprints/<blueprint_id>/probes/<probe_id>` where to delete the probe specified by its `probe_id`.

HTTP GET can be sent to `/api/blueprints/<blueprint_id>/probes/<probe_id>` to retrieve the configuration of the probe as it was POSTed. It will contain more fields than it was specified at probe creation:

- id** with id of the probe (or UUID if it was not specified at creation time),
- state** with actual state of the probe; possible values are "created" for a probe being configured, "operational" for a successfully configured probe, and "error" if probe configuration has failed.
- last\_error** contains detailed error description for the most-recent error for probes in the "error" state. It has the following sub-fields:
- level: a message level, such as "error" or "info".
  - message: text with error details.
  - timestamp: when the message was registered.

The complete list of probe messages could be obtained by issuing HTTP GET request to `/api/blueprints/<blueprint_id>/probes/<probe_id>/messages`.

Messages are sorted by the 'timestamp' field, oldest come first.

Additionally, HTTP GET can be sent to `/api/blueprints/<blueprint_id>/probes` to retrieve all the probes for blueprint `<blueprint_id>`.

## 2.3

HTTP PATCH and PUT methods for probes are available since Apstra version 2.3.

HTTP PATCH can be sent to `/api/blueprints/<blueprint_id>/probes/<probe_id>` to update the probe metadata or disable or enable the probe.

```
{
  "label": "new server_tx_bytes",
  "description": "some better probe description",
  "tags": ["production"],
  "stages": [
    {
      "name": "server_tx_bytes",
      "description": "updated stage description",
      "tags": ["server traffic"],
      "units": "bps"
    }
  ]
}
```

This example updates probe metadata for the probe that was created with the POST request listed above. All fields here are optional, values that were not specified remain unchanged.

Every stage instance is also optional, that is, only specified stages will be updated, and not specified stages remain unchanged.

Tags collection is updated entirely, i.e. if it was tags: ["a", "b"] and the PATCH payload specified tags: ["c"], then the resulting collection will look like tags: ["c"] (NOT tags: ["a", "b", "c"]).

With PATCH it's not possible to change probe's set of processor and stages. Please read further for PUT description which allows to do that.

HTTP PUT can be sent to `/api/blueprints/<blueprint_id>/probes/<probe_id>` to replace a probe.

This is very similar to POST, with the difference being that it replaces the old configuration for probe `<probe_id>` with the new one specified in the payload. Payload format for this request is the same as for POST, but `id` is not allowed.

## Inspect Probe

Stages are implicitly created by being named in the input and output of various processors. You can inspect the various stages of a probe. The API for reading a particular stage is `/api/blueprints/<blueprint_id>/probes/<probe_id>/stages/<stage_name>`

**NOTE:** Each stage has a type. This is a function of the generating processor and the input stage(s) to that processor. The types are: Number (N); Number Time Series (NTS); Number Set (NS); Number Set Time Series (NSTS); Text (T); Text Time Series (TTS); Text Set (TS); Text Set Time Series (TSTS); Discrete State (DS); Discrete State Time Series (DSTS); Discrete State Set (DSS); Discrete Set Time Series (DSSTS)

A NS is exactly that: a set of numbers.

Similarly, a DSS is a set of discrete-state variables. Part of the specification of a DSS (and DSSTS) stage is the possible values the discrete-state variable can take.

A text set is a set of strings.

A NSTS is a set of time-series with numbers as values. For example, a member of this set would be: (time=0 seconds, value=3), (time=3 seconds, value=5), (time=6 seconds, value=23), and so-on.

An DSTS is the same as an NSTS except values are discrete-state.

An TSTS is the same as an NSTS except values are strings.

Number (N), Discrete-State (DS), and Text (T) are simply Number Sets, Discrete State Sets, and Text Sets guaranteed to be of length one.

NTS, DSTS, and TS are the same as above, but are time-series instead of single values.

Let's consider the first stage - "server\_tx\_bytes". This stage contains the tx\_bytes counter for every server-facing port in the system. We can get it from the url `/api/blueprints/<blueprint_id>/probes/<probe_id>/stages/server_tx_bytes_output`

The response we get would be of the same form as the following:

```
{
  "properties": [
    "interface",
    "system_id"
  ],
  "type": "ns",
  "units": "bytes_per_second",
  "values": [
    {
      "properties": {
        "interface": "intf1",
        "system_id": "spine1"
      },
      "value": 22
    },
    {
      "properties": {
        "interface": "intf2",
        "system_id": "spine1"
      },
      "value": 23
    },
    {
      "properties": {
        "interface": "intf1",
        "system_id": "spine3"
      },
      "value": 24
    }
  ]
}
```

```
]
}
```

As we know from our running example, the "server\_tx\_bytes" stage contains the tx\_bytes value for every server-facing interface in the network. Looking at the above example, we can see that this stage is of type "ns", indicating NS or Number-Set. As mentioned before, data in stages is associated with context. This means that every element in the set of a stage is associated with a group of key-value pairs. Per every stage, the keys are the same for every piece of data (or, equivalently, item in the set). These keys are listed in the "properties" field of a given stage, and are generally a function of the generating processor. Each of the items in "values" assigns a value to each of the properties of the stage and provides a value (the "Number" in the "Number Set"). The meaning of this data in this stage is that tx\_bytes on intf1 of spine1 is 22, on intf2 of spine1 is 23, and on intf1 of spine3 is 24 bytes per second.

Notice that "units" is set for this stage as specified in the running example.

To query the second stage in our probe, send an HTTP GET to the std endpoint /api/blueprints/<blueprint\_id>/probes/<probe\_id>/stages/std\_dev\_output.

```
{
  "type": "n",
  "units": "",
  "value": 1
}
```

This stage is a number. It has no context, only a single value. In our example, this is the standard deviation across all spines.

The penultimate stage in our probe can be queried at the endpoint /api/blueprints/<blueprint\_id>/probes/<probe\_id>/stages/server\_traffic\_imbalanced.

```
{
  "possible_values": [
    "true",
    "false"
  ],
  "type": "ds",
  "units": "",
  "value": false
}
```

As shown, this stage indicates whether server traffic is imbalanced ("true") or not ("false") by indicating if the standard deviation across of tx\_bytes across all server-facing ports is greater-than 100. Note the "possible\_values" field describes all values that the discrete-state "value" can take.

All processors of a probe can also be queried via `/api/blueprints/<blueprint_id>/probes/<probe_id>/processors/<processor_name>`. By doing such a query, you can discover the configuration used for creation of said processor.

### Query Probe Anomalies

The final stage of our example processor raises an Apstra Anomaly (and sets its output to "true"), when the standard deviation of tx\_bytes across server-facing interfaces is greater-than 100.

You can query probe anomalies via the standard anomaly API at `/api/blueprints/<blueprint_id>/anomalies?type=probe`.

Following is the JSON form of an anomaly that would be raised by our example probe (with ellipses for data we don't care about for this example):

```
{
  "actual": {
    "value_int": 101
  },
  "anomaly_type": "probe",
  "expected": {
    "value_int": 100
  },
  "id": "...",
  "identity": {
    "anomaly_type": "probe",
    "probe_id": "efb2bf7f-d8cc-4a55-8e9b-9381e4dba61f",
    "properties": {},
    "stage_id": "server_traffic_imbalanced"
  },
  "last_modified_at": "...",
  "severity": "critical"
}
```

As seen in the above example, the identity contains the probe\_id and the name of the stage on which the anomaly was raised and which requires further inspection by the operator. Within a given stage, if the type of the stage were a set-based type, the "properties" field of the anomaly would be filled with the properties of the specific item in the set that caused the anomaly. This brings up the important point

that multiple anomalies can be raised on a single stage, as long as each is on a different item in the set. In our example, since the stage in question is of type NS, the "properties" field is not set.

## Introspect Processors

The set of processors available to the operator is a function of the platform and the reference design. Apstra provides an API for the operator to list all available processors, learn what parameters they take, and learn what inputs they require and outputs they yield.

The API in question is found at `/api/blueprints/<blueprint_id>/telemetry/processors`.

It yields a list of processor descriptions. In the following example, we show the description for the `std_dev` processor.

```
{
  "description": "Standard Deviation Processor.\n\n Groups as described by group_by, then
calculates std deviation and\n outputs one standard deviation for each group. Output is NS.\n
Input is an NS or NSTS.\n ",
  "inputs": {
    "in": {
      "required": true,
      "types": [
        {
          "keys": [],
          "possible_values": null,
          "type": "ns"
        },
        {
          "keys": [],
          "possible_values": null,
          "type": "nsts"
        }
      ]
    }
  },
  "outputs": {
    "out": {
      "required": true,
      "types": [
        {
          "keys": [],
          "possible_values": null,
          "type": "ns"
        }
      ]
    }
  }
}
```

```

        }
    ]
}
},
"label": "Standard Deviation",
"name": "std_dev",
"schema": {
    "additionalProperties": false,
    "properties": {
        "ddof": {
            "default": 0,
            "description": "Standard deviation correction value, is used to correct divisor (N - ddof) in calculations, e.g. ddof=0 - uncorrected sample standard deviation, ddof=1 - corrected sample standard deviation.",
            "title": "ddof",
            "type": "integer"
        },
        "enable_streaming": {
            "default": false,
            "type": "boolean"
        },
        "group_by": {
            "default": [
                "system_id"
            ],
            "items": {
                "type": "string"
            },
            "type": "array"
        }
    },
    "type": "object"
}
}

```

As seen above, there is a string-based description, the name of type processor type (as supplied to the REST API in probe configuration). The set of parameters specific to a given probe is described in the "schema".

Special notice must be paid to "inputs" and "outputs". Even though these are in the "schema" section, they are present on every type of processor. Each processor can take zero-or-more more input stages and must output one-or-more stages. Optional stages have "required" set to false. The names of the stages (relative to a particular instance of a processor) they take are described in these variables. We can

see that the "std\_dev" processor takes a single input named "in" and a single output named "out". This is reflected in our usage of it in the previous example.

There's one special input name: \*. For example:

```
"inputs": {
  "*": {
    "required": true,
    "types": [
      {
        "keys": [],
        "possible_values": null,
        "type": "ns"
      },
      {
        "keys": [],
        "possible_values": [],
        "type": "dss"
      },
      {
        "keys": [],
        "possible_values": null,
        "type": "ts"
      }
    ]
  }
}
```

It means the processor accepts one or more inputs of the specified types with arbitrary names.

Changed in 3.0: Previously, inputs and outputs section didn't specify whether specific inputs or outputs were required, so the format was changed from the following:

This syntax is deprecated and invalid.

```
"inputs": {
  "in": [
    {
      "data_type": "ns",
      "keys": [
        "system_id"
      ],
    },
  ],
}
```

```

        "value_map": null,
        "value_type": "int64"
    }
    ...
]
}

```

## Stream Data

Any processor instance in any probe can be configured to have its output stages streamed in the "perfmon" channel of Apstra streaming output. If the property "enable\_streaming" is set to "true" in the configuration for any processor, its output stages will have all their data streamed.

For Non-Time-Series-based stages, each will generate a message whenever their value changes. For Time-Series based stages, each will generate a message whenever a new entry is made into the time-series. For Set-based stages, each item in the set will generate a message according to the two prior rules.

Each message that is generated has a value, a timestamp, and a set of key-value pairs. The value is self-explanatory. The timestamp is the time at which the value changed for Non Time-series-based stages and the timestamp of the new entry for Time-series based stages. The key-value pairs correspond to the "properties" field we observed earlier in the "values" section of stages, thus providing context.

Below we have the format for messages from IBA which is encapsulated in a PerfMon message (and that in-turn in an AosMessage). The key-value pairs of context are put into the "property" repeated field (with "name" as the key and "value" as the value) while the value is put into the "value" field. "probe\_id" and "stage\_name" are as they appear. The blueprint\_id is put into the "origin\_name" of the encapsulated AosMessage. Similarly the timestamp is put into the generic "timestamp" field.

```

message ProbeProperty {
    required string name = 5;
    required string value = 6;
}
message ProbeMessage {
    repeated ProbeProperty property = 1;
    oneof value {
        int64 int64_value = 2;
        float float_value = 3;
        string string_value = 4;
    }
    required string probe_id = 5;
}

```

```
required string stage_name = 6;
}
```

## RCI Fault Model (API)

### IN THIS SECTION

- [Create Root Cause Identification Instance | 769](#)
- [Update Root Cause Identification Instance | 770](#)
- [Delete Root Cause Identification Instance | 771](#)
- [List Root Cause Identification Instances | 772](#)

You can access complete Apstra API documentation from the web interface in the **Platform > Developers** section.

- A blueprint is associated with zero or more Root Cause Identification instances.
- Root Cause Identification instances are enabled (created) / disabled (deleted) via CRUD API for Root Cause Identification sub-resource under the blueprint.
- The instances that can be created depends on the reference design of the blueprint. In this first phase of Root Cause Identification, only two\_stage\_l3clos has Root Cause Identification support, and right now it only allows one Root Cause Identification instance per blueprint.

### Create Root Cause Identification Instance

```
POST /api/blueprints/<blueprint_id>/arca
Request Payload schema
{
  "model_name": s.String() # Name of ARCA instance's system fault model (ref
design specific)
  "trigger_period": s.Float(min=10.0) # ARCA instance runs every <trigger_period>
seconds.
}
```

Example for blueprints for ref design two\_stage\_l3clos:

```
{
  "model_name": "default",
  "trigger_period": 10.0
}
```

Return values:

201 - Successfully created the RCI instance. Response payload:

```
{"id": <RCI instance ID>}
```

The ID is used in GET, PUT, DELETE

404 - Blueprint does not exist or is not deployed

422 - Validation error. Response payload:

```
{"error": <message>}
```

Possible error messages:

Model name is not found for the reference design

An ARCA instance already exists for given model name

trigger\_period is too small

## Update Root Cause Identification Instance

Using the PUT API, you can tweak the execution frequency of the Root Cause Identification instance.

PUT /api/blueprints/<blueprint\_id>/arca/<arca\_id>

Request Payload schema

```
{
  "trigger_period": s.Float(min=10.0)
}
```

Return values:

200 - Update succeeded.

404 - ARCA instance not found.

422 - Validation error. Response payload:

```
{"error": <message>}
```

Possible error messages:  
trigger\_period is too small

## Delete Root Cause Identification Instance

Using the GET API, you can obtain the current status (set of root causes) of the Root Cause Identification instance.

GET /api/blueprints/<blueprint\_id>/arca/<arca\_id>

Return values:

200 - see response schema below

404 - ARCA instance not found

Response payload schema

```
{
  "id": String,      # ARCA instance ID
  "model_name": String, # see POST payload
  "trigger_period": Float, # see POST payload
  "state": Enum("created", "operational"),
  "config_updated_at": Timestamp # of last update to instance via POST/PUT
  "status_updated_at": Timestamp # of last update to ARCA results
  "root_cause_count": Integer(min=0) # Number of root causes identified
  "root_causes": List(ROOT_CAUSE_OBJ) # Actual root causes
}
```

Timestamps are in ISO8601 format in UTC timezone, e.g. "2018-10-16T22:12:34+0000" If state == "created", then Status\_updated\_at == UNIX epoch root\_cause\_count == 0 "root\_causes" key is not returned

Each ROOT\_CAUSE\_OBJ has the following schema:

```
{
  "id": String, # Unique ID for the root cause in the ARCA instance
  "context": String, # Encoded context such as references to graph nodes
  "description": String, # Human-readable text, e.g. "link <blah> broken"
  "timestamp": Timestamp, # of when RC is detected (ISO8601 format)
  "symptoms": List(SYMPOM_OBJ), # List of symptoms; always non-empty
}
```

Notes on root cause detection and IDs: A root cause may be detected multiple times over the blueprint's lifetime. For instance, a root cause is defined for broken cable between spine1 and leaf1. This root cause can appear at any time, and it may disappear once the problem is fixed. A root cause has a unique ID scoped in the ARCA instance. This means that the ID may appear and disappear corresponding to whether the problem occurs or gets fixed, e.g. cable gets broken or reconnected. What to expect as root cause ID: In two\_stage\_l3clos the root cause ID is a composition of graph node and relationship IDs, and some immutable but readable name of the root cause. Example: <graph link node id>/broken.

Each SYMPTOM\_OBJ has the following schema:

```
{
  "id": String, # Unique ID for the symptom in the ARCA instance
  "context": String, # Encoded context such as system ID, service name
  "description": String, # Readable, e.g. "interface swp1 on leaf1 is down"
}
```

Given the same ARCA system fault model, the set of symptom IDs are always the same for given root cause. However, the context may be different. For instance, the symptom "interface swp1 on leaf1 is down" is the same, while context of different instances of this symptom may have different system IDs depending on which system ID is assigned to leaf1 when the root cause for this symptom is detected. Example symptom ID: <graph interface node id>/down

### List Root Cause Identification Instances

```
GET /api/blueprints/<blueprint_id>/arca
```

Return values

200 - see response schema below

404 - blueprint not found or blueprint not deployed

Response schema:

```
{
  "items": List(ARCA_INSTANCE_DIGEST), # list may be empty
}
```

ARCA\_INSTANCE\_DIGEST has the same schema as the response payload of GET individual ARCA instance, except that it does not contain the "root\_causes" key.

In this phase, for two\_stage\_l3clos blueprints, there is at most 1 element in the list, because only 1 ARCA instance is allowed per blueprint.

## Apstra Cluster (API)

### IN THIS SECTION

- [Health Check Apstra VMs via Apstra REST API | 773](#)

In addition to using the Apstra web interface to check the health of Apstra VMs, you can also use Apstra REST API.

### Health Check Apstra VMs via Apstra REST API

From the left navigation menu of the Apstra GUI, navigate to **Platform > Developers** to access REST API documentation. From there you can access cluster APIs.

```
/api/cluster/nodes/{node_id} .. Get AOS slave node status.
/api/cluster/nodes/{node_id}/errors .. Retrieve error for an AOS cluster node.
```

Here is an example of REST API with curl command:

```
curl -X GET "https://172.20.159.3/api/cluster/nodes/AosController/errors" -H "accept:
application/json"
```

If no error occurs, the output is as follows:

```
{
  "state": "active",
  "errors": []
}
```

If the agent process has rebooted, the error is shown as follows:

```
{
  "state": "active",
  "errors": [
    "agentReboot"
  ]
}
```

```
]
}
```

## API From Python

### IN THIS SECTION

- [API User Login | 774](#)
- [API - Blueprints | 775](#)
- [API - Blueprint Racks | 775](#)
- [API - Blueprint Routing Zones \(Security Zones\) | 775](#)
- [API - Blueprint Virtual Networks | 776](#)
- [Run Python | 776](#)

Following are examples of Python3 code using the Apstra API.

### API User Login

```
import requests, sys

# IP of Cloudlabs AOS Server
aos_server = '172.16.90.3'
username = 'admin'
password = 'aos aos'

# authenticate and get a auth token
url = 'https://' + aos_server + '/api/user/login'
headers = { 'Content-Type': "application/json", 'Cache-Control': "no-cache" }
data = '{ \"username\": \"' + username + '\", \"password\": \"' + password + '\" }'
response = requests.request("POST", url, data=data, headers=headers, verify=False)
print('POST',url,response.status_code)
if response.status_code != 201:
    sys.exit('error: authentication failed')
auth_token = response.json()['token']
print(auth_token)
```

```
headers = { 'AuthToken':auth_token, 'Content-Type':"application/json", 'Cache-Control':"no-cache" }
```

## API - Blueprints

```
# get blueprint ID ... assuming there is only one
url = 'https://' + aos_server + '/api/blueprints'
response = requests.request('GET', url, headers=headers, verify=False)
print('GET', url, response.status_code)
blueprint_id = response.json()['items'][0]['id']
blueprint_name = response.json()['items'][0]['label']
print(blueprint_name, blueprint_id)
```

## API - Blueprint Racks

```
# get a list of racks
bound_to = ''
url = 'https://' + aos_server + '/api/blueprints/' + blueprint_id + '/racks'
response = requests.request('GET', url, headers=headers, verify=False)
print('GET', url, response.status_code)
for item in response.json()['items']:
    bound_to += '{"system_id":"' + item['leafs'][0]['id'] + '",'
bound_to = bound_to[:-1]
print(bound_to)
```

## API - Blueprint Routing Zones (Security Zones)

```
# get routing zone ID ... assuming there is only one
url = 'https://' + aos_server + '/api/blueprints/' + blueprint_id + '/security-zones'
response = requests.request('GET', url, headers=headers, verify=False)
print('GET', url, response.status_code)
for item in response.json()['items']:
    if(response.json()['items'][item]['vrf_name'] != 'default'):
        security_zone_name = response.json()['items'][item]['vrf_name']
        security_zone_id = item
        break
print(security_zone_name, security_zone_id)
```

## API - Blueprint Virtual Networks

```
# create a virtual network
vn_name = "My-VN"
url = 'https://' + aos_server + '/api/blueprints/' + blueprint_id + '/virtual-networks'
data = '{"label":"' + vn_name + '","vn_type":"' + vxlan + '","bound_to":[' + bound_to +
'],'security_zone_id":"' + security_zone_id + '"]}'
print(data)
response = requests.request('POST', url, data=data, headers=headers, verify=False)
print('POST', url, response.status_code)
```

## Run Python

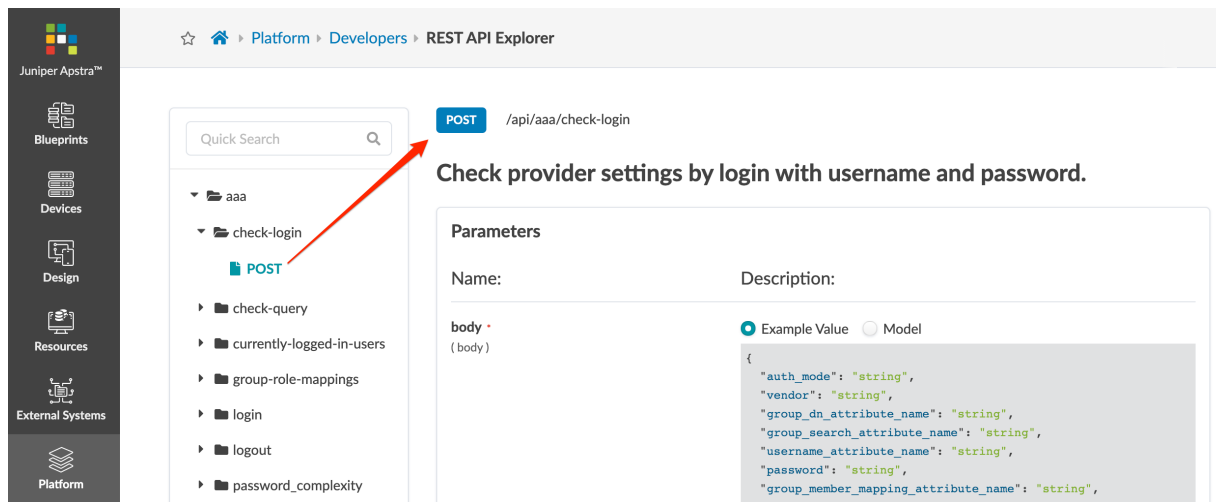
The proceeding Python3 code can be run on the Cloudlabs AOS Server. Use the python3.6 command to run the Python script.

```
admin@aos-server:~$ python3.6 test.py
POST https://192.168.3.3/api/user/login 201
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFkbWluIiwiaWF0IjY3JlYXRlZF9hdCI6IjIwMjAtMDItMjFUMTc6NDM6NTkuNTU1MDQzIiwidXNlcl9zZXNzaW9uIjoiaWUyY2YwODktNzZmYS00NDg4LTlhNmItYWVhYjc3MmQyNDE2IiwiaXhwIjojNTgyMzkzNDM5fQ.GWsy292pfpPVpisbQNKc3EHRDlxh10UmpQaQ-dF-mwY
GET https://192.168.3.3/api/blueprints 200
neil-blueprint cbfe7a43-4da7-4b2c-90a2-ea0bae4ed79a
GET https://192.168.3.3/api/blueprints/cbfe7a43-4da7-4b2c-90a2-ea0bae4ed79a/racks 200
{"system_id":"2cbb0fc0-5f87-4671-8d8b-e909cbf84fdd"}, {"system_id":"98002bb9-d0a9-484c-86e7-2aac2b926bf7"}, {"system_id":"73bd231c-f78e-499f-bf98-fa80c1102a4a"}, {"system_id":"19fb6155-e9eb-4ae7-b5b3-933416f0e3cd"}
GET https://192.168.3.3/api/blueprints/cbfe7a43-4da7-4b2c-90a2-ea0bae4ed79a/security-zones 200
Finance 4aaa4499-3194-4904-a1ae-daabbe3ed329
{"label":"My-VN","vn_type":"vxlan","bound_to":[{"system_id":"2cbb0fc0-5f87-4671-8d8b-e909cbf84fdd"}, {"system_id":"98002bb9-d0a9-484c-86e7-2aac2b926bf7"}, {"system_id":"73bd231c-f78e-499f-bf98-fa80c1102a4a"}, {"system_id":"19fb6155-e9eb-4ae7-b5b3-933416f0e3cd"}], "security_zone_id":"4aaa4499-3194-4904-a1ae-daabbe3ed329"}
POST https://192.168.3.3/api/blueprints/cbfe7a43-4da7-4b2c-90a2-ea0bae4ed79a/virtual-networks 201
admin@aos-server:~$
```

## REST API Explorer

With Apstra's REST API explorer, you can browse and search for specific REST API endpoints relevant to both the platform and reference designs.

From the left navigation menu, navigate to **Platform > Developers > REST API Explorer** to see the screen as shown below. The left column contains a list of API categories from which you can browse. You can also search for a specific endpoint by entering a query in the **Quick Search** field. The details view of an endpoint includes information about the URL, method, summary, parameters and responses. The example below shows the model for checking provider settings by login with username and password.



## Technical Support (Platform)

### IN THIS SECTION

- [Show Tech: Apstra Controller and On-box Agents \(GUI\) | 778](#)
- [Show Tech: Off-box Agents \(CLI\) | 781](#)
- [Show Tech: Infra Off-box Agents \(CLI\) | 781](#)
- [Show Tech: Apstra Server \(CLI\) | 782](#)
- [Show Tech: On-box Agents \(CLI\) | 783](#)

Technical Support is available to all customers with a valid and current license for Juniper Apstra software. This includes customers who have purchased a license directly or via a partner or reseller. This also includes customers who have obtained an evaluation license. If your purchased or evaluation license is expired, Juniper Support may not be able to offer support and will refer you to the appropriate sales team to purchase a current license. For more information about working with Juniper Support, refer to [Guidelines & Policies](#)

If you require assistance with registration or with opening a technical support case via phone, call Juniper Customer Care at +1-888-314-5822 (toll free, US & Canada). If you are outside the US or Canada, call +1-408-745-9500 or a country number listed on the [Contact Support](#) page.

To aid the support process, we ask that you provide Juniper Support with diagnostic information from the Apstra environment. Separate *show tech* files are needed from the Apstra controller and from each of the affected device agents. You can obtain the show tech files for the Apstra controller and connected on-box agents from the Apstra GUI or via the Apstra controller Linux CLI. You must obtain the show tech files for off-box agents from the Apstra controller Linux CLI.

## Show Tech: Apstra Controller and On-box Agents (GUI)

You can collect show tech files for the Apstra controller and connected device agents (on-box and offbox) from the Apstra web GUI.

if you haven't configured local credentials for the Apstra controller, from the left navigation menu, navigate to **Platform > Apstra Cluster** and edit the controller to configure credentials. These are the credentials you use for the VM console or SSH.

1. From the left navigation menu, navigate to **Platform > Technical Support** and click **Collect Show Tech** to see the dialog for selecting and collecting show tech files.
2. To collect show tech from the controller, leave **Apstra Controller** selected.

**NOTE:** For Apstra server controllers with large databases, the operation may timeout. If this happens, you must collect show tech using the CLI. For more information, see "[Show Tech: Apstra Server \(CLI\)](#)" on page 782.

3. Check the box for **Managed Devices** to see the list of connected managed devices (devices with installed agents that have been acknowledged).
4. Select up to twenty (20) devices for show tech collection.

**NOTE:** The configured device system agent username and password authentication is used to collect device show tech. If you have configured the device to use another authentication (AAA) method with a different username and password (such as RADIUS and TACACS) you cannot collect show tech from the GUI. You must collect show tech with CLI. For more information, see "[Show Tech: On-box Agents \(CLI\)](#)" on page 783.

5. Click **Collect** to start the collection process.

**Collect Show Tech**

Choose log source: \*

☒ AOS Controller

☒ Managed Devices <sup>?</sup>

Choose from 1 to 20 devices to collect show tech for.

▸ Query: All

1-11 of 11 < >

Page Size: 25 ▾

Filter selected by ☒ all ☐ selected only ☐ unselected only

<input checked="" type="checkbox"/>	Address ▾	Platform ▾	Platform Version ▾	Hostname ▾
11 selected				
<input checked="" type="checkbox"/>	10.29.29.15	NXOS	9.3(7)	sspine1
<input checked="" type="checkbox"/>	10.29.29.21	NXOS	9.3(7)	spine4
<input checked="" type="checkbox"/>	10.29.29.22	NXOS	9.3(7)	leaf3
<input checked="" type="checkbox"/>	10.29.29.23	NXOS	9.3(7)	leaf2
<input checked="" type="checkbox"/>	10.29.29.19	NXOS	9.3(7)	spine2
<input checked="" type="checkbox"/>	10.29.29.13	NXOS	9.3(7)	leaf5
<input checked="" type="checkbox"/>	10.29.29.17	NXOS	9.3(7)	leaf4
<input checked="" type="checkbox"/>	10.29.29.14	NXOS	9.3(7)	sspine2
<input checked="" type="checkbox"/>	10.29.29.20	NXOS	9.3(7)	leaf1

Collect

**TIP:** If the image below appears, you still need to configure local credentials on the node. Click the link to go to the controller node screen, click the **Edit** button (right side), then enter

the username and password you use for the VM console or SSH.

☆ 🏠 > Platform > Technical Support

Collect Show Tech

Query: All 1-1 of 1 Page Size: 25

Job ID: 0eb4d194-4e41-4f3b-9cf0-4d92782baaa1

FAILED

1-1 of 1

Device Address	State	Started	Finished	Logs
AOS Controller	FAILED	2021-11-08, 14:52:00	2021-11-08, 14:52:00	Need to specify username and password for <a href="#">AOS Controller</a>

6. After the jobs are complete and marked **SUCCESS**, click the download button for *each* of the files (under **Logs**).

☆ 🏠 > Platform > Technical Support


Collect Show Tech

Query: All 1-13 of 13 Page Size: 25

Job ID: 2011a972-e77e-4720-9018-b2120a66b68f

SUCCESS

1-1 of 1

Device Address	State	Started	Finished	Logs
AOS Controller	SUCCESS	2021-11-12, 15:55:26	2021-11-12, 15:58:35	 (118 MB)

**TIP:** When the show tech files have been downloaded, you can free up disk space by deleting jobs.

7. From a computer with the ability to upload, [upload the show tech files to your customer case](#).

## Show Tech: Off-box Agents (CLI)

Show tech files for installed off-box agents (for all supported devices and all Apstra versions) must be obtained from the CLI. You cannot collect them from the GUI. You'll need the device management IP address(es) and a valid device SSH username and password.

**NOTE:** If your off-box agents are for infra, you'll collect show tech with a different method. Refer to ["Show-Tech: Infra Off-box Agents \(CLI\)" on page 781](#) for details.

1. SSH into the Apstra server that the off-box agent is running on. (`ssh admin@<apstra-server-ip>` where `<apstra-server-ip>` is the IP address of the Apstra server.)
2. Run the `aos_offbox_show_tech_collector` command with the IP address(es) of the device(s) and a valid admin username and password. The show tech file is copied to your user directory. For example:

```
admin@aos-server:~$ sudo aos_offbox_show_tech_collector --ips 172.20.202.6 --user admin --
password admin-password
2020-08-10 12:03:46,116 aos-offbox-172_20_202_6-f
2020-08-10 12:03:46,384 collecting offbox container aos-offbox-172_20_202_6-f showtech
2020-08-10 12:03:50,873 invoking DI container to collect device 172.20.202.6 show_tech
2020-08-10 12:05:42,155 Done collecting device show_tech
2020-08-10 12:05:42,204 AOS offbox show tech generated at /home/admin/
aos_show_tech_20200810_120542_172_20_202_6.tar.gz
admin@aos-server:~$
```

3. Copy the file from the Apstra server to a local computer with the ability to upload. The file is owned by root. You may need to change permissions to copy the file. For example:

```
admin@aos-server:~$ ls -l aos_show_tech_20200810_120542_172_20_202_6.tar.gz
-rw----- 1 root root 238156 Aug 10 12:05 aos_show_tech_20200810_120542_172_20_202_6.tar.gz
admin@aos-server:~$ sudo chmod a+r aos_show_tech_20200810_120542_172_20_202_6.tar.gz
admin@aos-server:~$ ls -l aos_show_tech_20200810_120542_172_20_202_6.tar.gz
-rw-r--r-- 1 root root 238156 Aug 10 12:05 aos_show_tech_20200810_120542_172_20_202_6.tar.gz
admin@aos-server:~$
```

4. [Upload the show tech file to your customer case.](#)

## Show Tech: Infra Off-box Agents (CLI)

The instructions below are for collecting show tech files for infra off-box agents. If your off-box agents are not for infra, refer to ["Show Tech: Apstra Off-box Agents \(CLI\)" on page 781](#).

1. SSH into the Apstra server that the off-box agent is running on. (ssh admin@<apstra-server-ip> where <apstra-server-ip> is the IP address of the Apstra server.)
2. Run `docker ps` to get the name of the container (in the NAMES column).
3. Run the `docker exec -ti <offbox_container_name> aos_show_tech` command where <offbox\_container\_name> is the name you retrieved when you ran `docker ps`. For example:

```
admin@aos-server:~$ docker exec -ti aos-offbox-172_20_47_6-f aos_show_tech
AOS show tech generated at /tmp/aos_show_tech_20200401_181128.tar.gz
admin@aos-server:~$
```

4. Using SCP, run the `docker cp` command to copy the show tech file from the off-box agent Docker container to the /tmp directory of the Apstra server. For example:

```
admin@aos-server:~$ docker cp aos-offbox-172_20_47_6-f:/tmp/
aos_show_tech_20200401_181128.tar.gz .
admin@aos-server:~$ ls
aos_show_tech_20200401_181128.tar.gz  docker.service.log
admin@aos-server:~$
```

5. Locate the file archive in the /tmp directory and copy it to a local computer with the ability to upload. Then [upload the show tech file to your customer case](#).

## Show Tech: Apstra Server (CLI)

We recommend using the GUI to obtain Apstra server show tech files, but you have the option of using the Apstra server Linux CLI instead, as described below.

1. SSH into the Apstra server. (ssh admin@<apstra-server-ip> where <apstra-server-ip> is the IP address of the Apstra server.)
2. Run the `sudo aos_show_tech` command to generate and copy the show tech file to the current working directory of the Apstra server. For example:

```
admin@aos-server:~$ sudo aos_show_tech
[sudo] password for admin:
Generating technical support data under directory /tmp/tmp.YmjJDhatJ
--- collecting sysinfo/cpuinfo from /proc/cpuinfo ---
--- collecting network/etc_hosts from /etc/hosts ---
--- collecting aos/aos.conf from /etc/aos/aos.conf ---
--- collecting sysinfo/meminfo from /proc/meminfo ---
--- collecting sysinfo/vmstat from /proc/vmstat ---
--- collecting network/etc_hostname from /etc/hostname ---
--- collecting network/interfaces_config from /etc/network/interfaces ---
```

```

--- collecting network/resolv.conf from /etc/resolv.conf ---
--- collecting logs/kern_log from /var/log/kern.log* ---
--- collecting logs/syslog from /var/log/syslog* ---
--- collecting filesystem/aos_cachaca_db_usage with command: du -a /var/lib/aos/cachaca ---
--- collecting sysinfo/uptime with command: uptime ---
--- collecting filesystem/aos_db_usage with command: du -a /var/lib/aos/db ---
--- collecting filesystem/disk_free with command: df -h ---
[snip]
Remaining dump took 8.477 ms
2020-04-01 03:35:39,010 131:INFO:aos.infra.core.entity_util:Create partition mount factory
for partition Anomaly
Dumping entity (anomaly_sysdb_dump/Tac) took 0.389 ms
Dumping entity (anomaly_sysdb_dump/alert_aggregation) took 3.986 ms
Dumping entity (anomaly_sysdb_dump/streaming) took 0.173 ms
Dumping entity (anomaly_sysdb_dump/alerts) took 4.174 ms
Dumping entity (anomaly_sysdb_dump/counters) took 0.160 ms
Dumping entity (anomaly_sysdb_dump/telemetry_adaptor) took 0.156 ms
Dumping entity (anomaly_sysdb_dump/deployment) took 0.214 ms
Dumping entity (anomaly_sysdb_dump/device) took 0.675 ms
Dumping entity (anomaly_sysdb_dump/cachaca) took 0.144 ms
Dumping entity (anomaly_sysdb_dump/var) took 0.201 ms
Skipping SysDB dump
Archiving show tech data into aos_show_tech_20200401_033431.tar.gz
Removing working directory /tmp/tmp.YmjuJDhatJ
All done.
admin@aos-server:~$

```

3. Locate the file archive in the /tmp directory (for example, aos\_show\_tech\_20200401\_033431.tar.gz), and via SCP, copy the file to a local computer with the ability to upload.
4. [Upload the show tech file to your customer case.](#)

## Show Tech: On-box Agents (CLI)

We recommend using the ["Apstra GUI" on page 778](#) to obtain on-box agent show tech files, but you have the option of using the Apstra server Linux CLI instead, as described below.

1. SSH to the device.
2. For Arista only, run bash to go the Arista Networks EOS shell.
3. For Cisco only, run guestshell.

4. From the device, run the `sudo aos_show_tech --platform <platform>` command where <platform> is the platform you're using: eos, nxos, cumulus, or sonic. The file is generated and copied to the /tmp directory. See below for SONiC example:

```
admin@l2-virtual-ext-001-leaf1:mgmt-vrf:~$ sudo aos_show_tech --platform sonic
AOS show tech generated at /tmp/aos_show_tech_20200401_034527.tar.gz
admin@l2-virtual-ext-001-leaf1:mgmt-vrf:~$
```

5. Locate the file archive in the /tmp directory (for example, aos\_show\_tech\_20200401\_034527.tar.gz) and copy it, via SCP, to a local computer with the ability to upload.
6. [Upload the show tech file to your customer case.](#)

## Favorites & User

### IN THIS SECTION

- [Manage Favorites | 784](#)
- [Change Your User Password | 786](#)
- [Change Your User Name/Email | 786](#)
- [Log Out | 786](#)

You can return quickly to frequently visited pages by saving them as favorites. From your user profile page, you can manage favorites, change your password, username and email; and log out of the Apstra software.

### Manage Favorites

- To add a favorite - click the star in the upper-left corner of the page to save. Leave the default name or rename it, then click **Add**. The outlined star becomes a shaded star to indicate that it is saved as a favorite.
- To remove a favorite - click the shaded star on the saved page. The star becomes an outline.

- To go to your list of favorites from anywhere in the GUI, click **Favorites** in the left navigation menu.
- To go to a favorite page from the **Favorites** menu - click its name. Up to five saved pages appear in the drop-down list.
- To go to your list of favorites from the **Favorites** menu - click **Show more** to go to your profile page where you can link to all favorite pages and change their names.
- To go to your profile page to see all your favorites, click your user name in the left navigation menu (bottom), then click **Profile**.
- To go to a favorite page from your profile page - click its link.
- To change the name of a link from your profile page - click the **Edit label** button, change the name, then click **Update**.
- To remove a favorite page from your profile page - click the **Remove** button (trash can) and click **Delete**.

The screenshot shows the Juniper Apstra GUI. On the left is a dark navigation sidebar with icons for Blueprints, Devices, Design, Resources, External Systems, Platform, Favorites, and User: admin. The top navigation bar shows a star icon, a home icon, and the text 'Profile'. Below this, the 'User profile' section contains a table with user details. The 'Favorites' section below it shows a list of saved pages with columns for Label, URL, and Actions. Red arrows and text annotations highlight key features: 'Click to add any page to favorites' points to the star icon; 'Click to see saved pages' points to the Favorites menu item; 'Click a favorite to go to the page' points to a favorite entry in the dropdown; 'Click to see all saved pages on your user profile' points to the 'Show more...' link in the dropdown; and another arrow points to the URL column in the Favorites table.

**User profile**

Username	admin
First Name	admin
Last Name	admin
Email	not_set
Roles	<input type="checkbox"/>

**Favorites**

Query: All 1-3 of 3 Page Size: 25

Label	URL	Actions
Juniper Apstra / Devices / Managed Devices	/devices/systems	
Juniper Apstra / Design / Configlets	/design/configlets	
Juniper Apstra / Platform / Event Log	/platform/events	

## Change Your User Password

1. From any page, click your username in the left navigation menu (bottom) and click **Profile** to see your profile page.
2. Click the **Change Password** button (top-right), enter your current password, then enter your new password, twice.
3. Click **Change Password** to update your password and return to your profile.

## Change Your User Name/Email

1. From any page, click your username in the left navigation menu (bottom) and click **Profile** to go to your profile page.
2. Click the **Edit** button (top-right), then change your name and/or email, as applicable.
3. Click **Save** to update your details and return to your profile.

## Log Out

From any page, click your username in the left navigation menu (bottom) and click **Log Out**.

The screenshot shows the Juniper Apstra user interface. On the left is a dark navigation menu with icons for Blueprints, Devices, Design, Resources, External Systems, Platform, Favorites, and a user profile section at the bottom labeled 'User: admin'. A red arrow labeled '1.' points to the 'User: admin' section. A sub-menu is open, showing 'User: admin', 'Profile', and 'Log Out'. A red arrow labeled '2.' points to the 'Profile' option. The main content area is titled 'Profile' and contains a 'User profile' table with fields: Username (admin), First Name (admin), Last Name (admin), Email (not\_set), and Roles (a toggle switch). Below this is a 'Favorites' section with a search bar and a 'Page Size' dropdown set to 25. At the bottom is a table with columns 'Label', 'URL', and 'Actions', currently showing 'No items'. In the top right corner of the profile page, there are two buttons: 'Change password' (indicated by a red arrow) and 'Edit' (indicated by a red arrow).

Label	URL	Actions
No items		

# Guides

## IN THIS SECTION

- [Extensible Telemetry Guide | 787](#)
- [5-Stage Clos Architecture | 802](#)
- [Access Layer Switches | 805](#)
- [Juniper EVPN Support | 806](#)
- [Intent-Based Analytics with AOS-CLI Utility | 814](#)
- [AOSOM-Streaming Guide | 828](#)
- [Mixed Uplink Speeds between Leafs and Spines | 842](#)
- [Enable VXLAN Routing on Cumulus Tomahawk | 845](#)

## Extensible Telemetry Guide

## IN THIS SECTION

- [Extensible Telemetry Overview | 787](#)
- [Set Up Development Environment | 788](#)
- [Develop Collector | 789](#)
- [Write Collector | 792](#)
- [Unit Test Collector | 798](#)
- [Package Collector | 800](#)
- [Upload Packages | 800](#)
- [Use Telemetry Collector | 801](#)

## Extensible Telemetry Overview

You can collect additional telemetry by installing Apstra device drivers and telemetry collectors, which can then be used in ["IBA probes" on page 410](#). The device drivers enable Apstra to connect to a NOS

and collect telemetry. Apstra ships with drivers for EOS, Cumulus, NX-OS, Ubuntu, and CentOS. To add a driver for an operating system not listed here, contact ["Juniper Support" on page 777](#).

Telemetry collectors are Python modules that help collect extended telemetry. The following sections describe the pipeline for creating telemetry collectors and extending Apstra with new collectors. You need familiarity with Python to be able to develop collectors.

## Set Up Development Environment

To get access to telemetry collectors (which are housed in the *aos\_developer\_sdk* repository) contact ["Juniper Support" on page 777](#). Contribute any new collectors that you develop to the repository.

To keep your system environment intact, we recommend that you use a virtual environment to isolate the required Python packages (for development and testing). You can download the base development environment, *aos\_developer\_sdk.run*, from <https://support.juniper.net/support/downloads/?p=apstra/>. To load the environment, execute:

```
aos_developer_sdk$ bash aos_development_sdk.run
4d8bbfb90ba8: Loading layer [=====>] 217.6kB/
217.6kB
7d54ea05a373: Loading layer [=====>] 4.096kB/
4.096kB
e2e40f457231: Loading layer [=====>] 1.771MB/
1.771MB
Loaded image: aos-developer-sdk:2.3.1-129

=====

Loaded AOS Developer SDK Environment Container Image
aos-developer-sdk:2.3.1-129.

Container can be run by
  docker run -it \
    -v <path to aos_developer_sdk cloned repo>:/aos_developer_sdk \
    --name <container name> \
    aos-developer-sdk:2.3.1-129

=====
```

This command loads the `aos_developer_sdk` Docker image. After the image load is complete, the command to start the environment is printed. Start the container environment as specified by the command. To install the dependencies, execute:

```
root@f2ece48bb2f1:/# cd /aos_developer_sdk/
root@f2ece48bb2f1:/aos_developer_sdk# make setup_env
...
```

The environment is now set up for developing and testing the collectors. Apstra SDK packages, such as device drivers and REST client, are also installed in the environment.

## Develop Collector

To develop a telemetry collector, specify the following *in order*.

1. **Service for which the collector is developed** - Identify what the service is. For example, the service could be to collect received and transmitted bytes from the switch interfaces. Identify a name for the service. Using service names that are reserved for built-in services (ARP, BGP, interface, hostname, route, MAC, XCVR, LAG, MLAG) is prohibited.
2. **The schema of the data provided to Apstra** - Identify how the collector output is to be structured. A collection of key-value pairs should be posted to Apstra. Identify what each item is, that is, what is the key/value syntactically and semantically. For the above mentioned example, key is a string that identifies the interface name. The value is a JSON string, with the JSON having two keys 'rx' and 'tx' both having an integer value.
3. **Network Operating System (NOS) for which the collector is developed** - The collector plugins are NOS-specific. Before writing a collector, identify the NOS(s) for which collector(s) are required.
4. **How the required data can be obtained from the device** - Identify the commands that can be used in the device to retrieve the required information. For example, 'show interfaces' command gives received and transmitted bytes from an Arista EOS device.
5. **Storage Schema Path** - The storage schema path is determined by the type of key and value in each item. The type of collector selected determines the storage schema for the application. The storage schema defines the high level structure of the data returned by the service. The storage schema path for your collector can be determined using the following table:

**Table 30: Determining Storage Schema Path**

Key Type	Value Type	Storage Schema Path
String	String	aos.sdk.telemetry.schemas.generic
String	Dict	aos.sdk.telemetry.schemas.generic

Table 30: Determining Storage Schema Path *(Continued)*

Key Type	Value Type	Storage Schema Path
Dict	String	aos.sdk.telemetry.schemas.iba_string_data
Dict	Integer	aos.sdk.telemetry.schemas.iba_integer_data

6. **Application Schema** - Application schema defines the schema for each item posted to the framework. Application schema is expressed using draft 4 version of [json schema](#). Each item is comprised of a key and value. The following table specifies two sample items.

Table 31: Sample item with its storage schema path

Storage Schema Path	Sample Item
aos.sdk.telemetry.schemas.generic	<pre>{   "identity": "eth0",   "value": "up", }</pre>
aos.sdk.telemetry.schemas.iba_string_data	<pre>{   "key": {     "source_ip": "10.1.1.1",     "dest_ip": "10.1.1.2",   },   "value": "up", }</pre>

**NOTE:** \* An item returned by collectors with generic storage schema should specify the key value using the key 'identity' and the value using the key 'value'.

\* An item returned by collectors with IBA-based schemas should specify the key value using the key 'key' and the value using the key 'value'.

Using this information, you can write the JSON schema. The following table maps the sample item specified above to its corresponding JSON schema.

Table 32: Sample Application Schema

Sample Item	Application Schema
<pre>{   "identity": "eth0",   "value": "up", }</pre>	<pre>{   "type": "object",   "properties": {     "identity": {       "type": "string",     },     "value": {       "type": "string",     }   } }</pre>
<pre>{   "key": {     "source_ip": "10.1.1.1",     "dest_ip": "10.1.1.2",   },   "value": "up", }</pre>	<pre>{   "type": "object",   "properties": {     "key": {       "type": "object",       "properties": {         "source_ip": {           "type": "string",           "format": "ipv4"         },         "dest_ip": {           "type": "string",           "format": "ipv4"         }       },       "required": ["source_ip", "dest_ip"],     },     "value": {       "type": "string",     }   } }</pre>

You can specify more complex schema using the constructs available in JSON schema. Update the schema in the file `aos_developer_sdk/aosstdcollectors/aosstdcollectors/json_schemas/<service_name>.json`

**NOTE:** As of Apstra version 4.0.1, you can ["import the service schema" on page 174](#) via the GUI.

## Write Collector

### IN THIS SECTION

- [Collect Data from Device | 792](#)
- [Parse Data | 793](#)
- [Post Data to Framework | 794](#)

Collector is a class that must derive from *aos.sdk.system\_agent.base\_telemetry\_collector.BaseTelemetryCollector*. Override the *collect* method of the collector with the logic to:

### Collect Data from Device

The device driver instance inside the collector provides methods to execute commands against the devices. For example, most Apstra device drivers provide methods *get\_json* and *get\_text* to execute commands and return the output.

**NOTE:** The device drivers for *aos\_developer\_sdk* environment are preinstalled. You can explore the methods available to collect data. For example:

```
>>> from aos.sdk.driver.eos import Device
>>> device = Device('172.20.180.10', 'admin', 'admin')
>>> device.open()
>>> pprint.pprint(device.get_json('show version'))
{'architecture': u'i386',
 'bootupTimestamp': 1548302664.0,
 'hardwareRevision': u'',
 'internalBuildId': u'68f3ae78-65cb-4ed3-8675-0ff2219bf118',
 'internalVersion': u'4.20.10M-10040268.42010M',
 'isIntlVersion': False,
 'memFree': 3003648,
```

```

u'memTotal': 4011060,
u'modelName': u'vEOS',
u'serialNumber': u'',
u'systemMacAddress': u'52:54:00:ce:87:37',
u'uptime': 62620.55,
u'version': u'4.20.10M'}
>>> dir(device)
['AOS_VERSION_FILE', '__class__', '__delattr__', '__dict__', '__doc__',
 '__format__', '__getattr__', '__hash__', '__init__', '__module__',
 '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__',
 '__sizeof__', '__str__', '__subclasshook__', '__weakref__', 'close',
 'device_info', 'driver', 'execute', 'get_aos_server_ip',
 'get_aos_version_related_info', 'get_device_aos_version',
 'get_device_aos_version_number', 'get_device_info', 'get_json',
 'get_text', 'ip_address', 'onbox', 'open', 'open_options', 'password',
 'probe', 'set_device_info', 'upload_file', 'username']

```

## Parse Data

The collected data needs to be parsed and re-formatted per the Apstra framework and the service schema identified above. Collectors with generic storage schema follow the following structure:

```

{
  "items": [
    {
      "identity": <key goes here>,
      "value": <value goes here>,
    },
    {
      "identity": <key goes here>,
      "value": <value goes here>,
    },
    ...
  ]
}

```

Collectors with IBA-based schema follow the following structure:

```
[
  {
    "key": <key goes here>,
    "value": <value goes here>,
  },
  {
    "key": <key goes here>,
    "value": <value goes here>,
  },
  ...
]
```

In the structures above, the data posted has multiple items. Each item has a key and a value. For example, to post interface specific information, there would be an identity/key-value pair for each interface you want to post to the framework.

**NOTE:** In the case when you want to use a third party package to parse data obtained from a device, list the Python package and version in the path.

<aos\_developer\_sdk>/aosstdcollectors/requirements\_<NOS>.txt. The packages installed by the dependency do not conflict with packages used by Apstra. The Apstra-installed packages are available at /etc/aos/python\_dependency.txt in the development environment.

## Post Data to Framework

When data is collected and parsed as per the required schema, post the data to the framework. You can use the `post_data` method available in the collector. It accepts one argument, and that is the data that should be posted to the framework.

The folder `aos_developer_sdk/aosstdcollectors/aosstdcollectors` in the repository contains folders for each NOS. Add your collector to the folder that matches the NOS. For example, to write a collector for Cumulus, add the collector to `aos_developer_sdk/aosstdcollectors/aosstdcollectors/cumulus`, and name the file after the service name. For example, if the service name is `interface_in_out_bytes`, then name the file `interface_in_out_bytes.py`.

In addition to defining the collector class, define the function `collector_plugin` in the collector file. The function takes one argument and returns the collector class that is implemented.

For example, a generic storage schema based collector looks like:

```
"""
Service Name: interface_in_out_bytes
Schema:
    Key: String, represents interface name.
    Value: Json String with two possible keys:
        rx: integer value, represents received bytes.
        tx: integer value, represents transmitted bytes.
DOS: eos
Data collected using command: 'show interfaces'
Type of Collector: BaseTelemetryCollector
Storage Schema Path: aos.sdk.telemetry.schemas.generic
Application Schema: {
    'type': 'object',
    'properties': {
        'identity': {
            'type': 'string',
        },
        'value': {
            'type': 'object',
            'properties': {
                'rx': {
                    'type': 'number',
                },
                'tx': {
                    'type': 'number',
                }
            },
            'required': ['rx', 'tx'],
        }
    }
}

"""

import json
from aos.sdk.system_agent.base_telemetry_collector import BaseTelemetryCollector

# Inheriting from BaseTelemetryCollector
class InterfaceRxTxCollector(BaseTelemetryCollector):
```

```

# Overriding collect method
def collect(self):

    # Obtaining the command output using the device instance.
    collected_data = self.device.get_json('show interfaces')

    # Data is in the format
    # "interfaces": {
    #     "<interface_name>": {
    #         ....
    #         "interfaceCounters": {
    #             ....
    #             "inOctets": int
    #             "outOctets": int
    #             ....
    #         }
    #     }
    #     ...
    # }

    # Parse the data as per the schema and structure required.
    parsed_data = json.dumps({
        'items': [
            {
                'identity': intf_name,
                'value': json.dumps({
                    'rx': intf_stats['interfaceCounters'].get('inOctets'),
                    'tx': intf_stats['interfaceCounters'].get('outOctets'),
                })
            } for intf_name, intf_stats in collected_data['interfaces'].iteritems()
            if 'interfaceCounters' in intf_stats
        ]
    })

    # Post the data to the framework
    self.post_data(parsed_data)

# Define collector_plugin class to return the Collector
def collector_plugin(_device):
    return InterfaceRxTxCollector

```

An IBA storage schema based collector looks like:

```

"""
Service Name: iba_bgp
Schema:
    Key: JSON String, specifies local IP and peer IP.
    Value: String. '1' if state is established '2' otherwise
DOS: eos
Data collected using command: 'show ip bgp summary vrf all'
Storage Schema Path: aos.sdk.telemetry.schemas.iba_string_data
Application Schema: {
    'type': 'object',
    'properties': {
        key: {
            'type': 'object',
            'properties': {
                'local_ip': {
                    'type': 'string',
                },
                'peer_ip': {
                    'type': 'string',
                }
            },
            'required': ['local_ip', 'peer_ip'],
        },
        'value': {
            'type': 'string',
        }
    }
}
"""

from aos.sdk.system_agent.base_telemetry_collector import IBATelemetryCollector

def parse_text_output(collected):
    result = [
        {'key': {'local_ip': str(vrf_info['routerId']), 'peer_ip': str(peer_ip)},
         'value': str(
             1 if session_info['peerState'] == 'Established' else 2)}
        for vrf_info in collected['vrfs'].itervalues()
        for peer_ip, session_info in vrf_info['peers'].iteritems()]
    return result

```

```
# Inheriting from BaseTelemetryCollector
class IbaBgpCollector(BaseTelemetryCollector):
    # Overriding collect method
    def collect(self):
        # Obtaining the command output using the device instance.
        collected_data = self.device.get_json('show ip bgp summary vrf all')
        # Parse the data as per the schema and structure required and
        # post to framework.
        self.post_data(parse_text_output(collected_data))

# Define collector_plugin class to return the Collector
def collector_plugin(device):
    return IbaBgpCollector
```

## Unit Test Collector

The folder `aos_developer_sdk/aosstdcollectors/test` in the repository contains folders based on the NOS. Add your test to the folder that matches the NOS. For example, a test to a collector for Cumulus is added to `aos_developer_sdk/aosstdcollectors/test/cumulus`. We recommend that you name the unit test with the prefix `test_`.

The existing infrastructure implements a Pytest fixture `collector_factory` that is used to mock the device driver command response. The general flow for test development is as follows.

1. Use the collector factory to get a collector instance and mocked Apstra framework. The collector factory takes the collector class that you have written as input.
2. Mock the device response.
3. Invoke collect method.
4. Validate the data posted to the mocked Apstra framework.

For example, a test looks like:

```
import json
from aosstdcollectors.eos.interface_in_out_bytes import InterfaceRxTxCollector

# Test method with prefix 'test_'
def test_sanity(collector_factory):

    # Using collector factory to retrieve the collector instance and mocked
```

```

# Apstra framework.
collector, mock_framework = collector_factory(InterfaceRxTxCollector)

command_response = {
    'interfaces': {
        'Ethernet1': {
            'interfaceCounters': {
                'inOctets': 10,
                'outOctets': 20,
            }
        },
        'Ethernet2': {
            'interfaceCounters': {
                'inOctets': 30,
                'outOctets': 40,
            }
        }
    }
}

# Set the device get_json method to retrieve the command response.
collector.device.get_json.side_effect = lambda _: command_response

# Invoke the collect method
collector.collect()

expected_data = [
    {
        'identity': 'Ethernet1',
        'value': json.dumps({
            'rx': 10,
            'tx': 20,
        })
    },
    {
        'identity': 'Ethernet2',
        'value': json.dumps({
            'rx': 30,
            'tx': 40,
        })
    }
]

# validate the data posted by the collector

```

```
data_posted_by_collector = json.loads(mock_framework.post_data.call_args[0][0])
assert sorted(expected_data) == sorted(data_posted_by_collector["items"])
```

To run the test, execute:

```
root@1df9bf89aeaf:/aos_developer_sdk# make test
root@1df9bf89aeaf:/aos_developer_sdk# make test
```

This command executes all the tests in the repository.

### Package Collector

All the collectors are packaged based on the NOS. To generate all packages, execute make at aos\_develop\_sdk. You can find the build packages at aos\_developer\_sdk/dist. The packages build can be broadly classified as:

Package	Description
Built-In Collector Packages	These packages have the prefix <i>aosstdcollectors_builtin_</i> . To collect telemetry from a device per the reference design, Apstra requires services as listed in the <deviceblah> section. Built-In collector packages contain collectors for these services. The packages are generated on a per NOS basis.
Custom Collector Packages	These package have the prefix <i>aosstdcollectors_custom_</i> in their names. The packages are generated on a per NOS basis. The package named <i>aosstdcollectors_custom_&lt;NOS&gt;-0.1.0-py2-none-any.whl</i> contains the developed collector.
Apstra SDK Device Driver Packages	These packages have a prefix <i>apstra_devicedriver_</i> . These packages are generated on a per NOS basis. Packages are generated for NOS that are not available by default in Apstra.

### Upload Packages

If the built-in collector packages and the Apstra SDK Device Driver for your Device Operating System (NOS) were not provided with the Apstra software, you must upload them to the Apstra server.

If you are using an off-box solution and your NOS is not EOS or Cumulus, you must upload the built-in collector package.

Upload the package containing your collector(s) and assign them to a Device System Agent or System Agent Profile.

## Use Telemetry Collector

IN THIS SECTION

- Set up Telemetry Service Registry | 801
- Start Collector | 801
- Delete Collector | 802
- Get Collected Data | 802
- List Running Collector Services | 802

### Set up Telemetry Service Registry

The registry maps the service to its application schema and the storage schema path. You can manage the telemetry service registry with the REST endpoint `/api/telemetry-service-registry`. You can't enable the collector for a service without adding a registry entry for the particular service. The registry entry for a service cannot be modified while the service is in use.

**NOTE:** When executing `make`, all application schemas are packaged together to a tar file (`json_schemas.tgz`) in the `dist` folder. With `aos-cli`, you have the option of importing all the schemas in the `.tgz` file. For more information about using the `aos-cli` utility, see [Juniper Support Knowledge Base article KB36747](#)

### Start Collector

To start a service, use the POST API `/api/systems/<system_id>/services` with the following three arguments:

Arguments	Description
Input_data	The data provided as input to the collector. Defaults to None.
Interval	Interval at which to run the service. Defaults to 120 seconds.
Name	Name of the service.

**NOTE:** You can also manage collectors via the aos-cli utility.

### Delete Collector

To delete a service, use the DELETE API `/api/systems/<system_id>/services/<service_name>`.

### Get Collected Data

To retrieve collected data, use the GET API `/api/systems/<system_id>/services/<service_name>/data`. Only the data collected in the last iteration is saved. Data does not persist over Apstra restart.

### List Running Collector Services

To retrieve the list of services enabled on a device, use the GET API `/api/systems/<system_id>/services`.

## 5-Stage Clos Architecture

### IN THIS SECTION

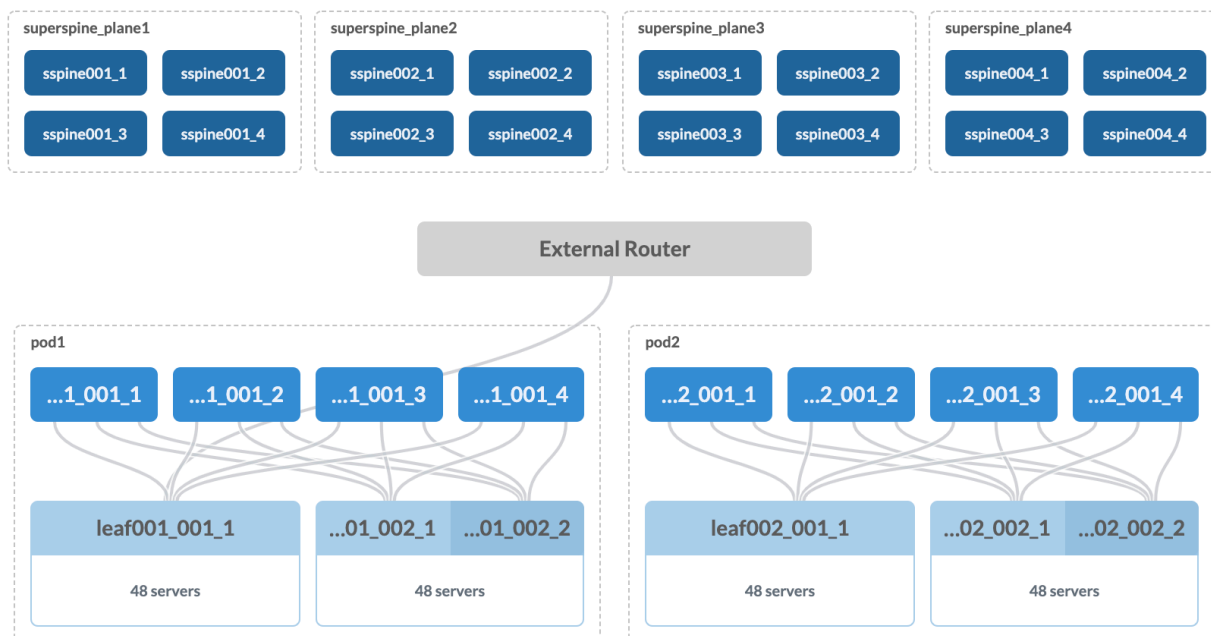
- [5-Stage Clos Overview | 802](#)
- [Create 5-Stage Clos Network | 804](#)
- [Modify 5-stage Clos Network | 805](#)

## 5-Stage Clos Overview

### IN THIS SECTION

- [5-Stage Clos Limitations | 803](#)
- [5-Stage Clos and EVPN | 804](#)

5-stage Clos architecture allows for large-scale topologies. With its additional aggregation layer, you can interconnect multiple **pods** into a single fabric. **Superspines** provide the additional layer that interconnects multiple pods. "**Planes**" on page 485 are groups of superspines. Each 5-stage topology consists of one or more planes. Each plane consists of one or more superspines. See below for an example.



Careful planning and consideration are required to build large 5-stage Clos networks. Refer to the limitations below when you're designing and validating your 5-stage topology. For assistance, contact "[Juniper Support](#)" on page 777.

### 5-Stage Clos Limitations

- You cannot change a 3-stage topology to a 5-stage topology.
- You must use the same overlay control protocol (static VXLAN or MP-EBGP-EVPN, specified during template creation) for all rack types in all pods.
- Root Cause Analysis is not supported.
- IPv6 / IPv4 support:
  - IPv6 support in the underlay depends on the NOS. See the "[feature matrix](#)" on page 853.
  - IPv6 for applications is not supported until Apstra 4.1.
  - The entire fabric across all pods must be either all IPv4, all IPv6 or all dual-stack
- Unsupported external connectivity implementations:
  - One generic system connecting to multiple pods

- EVPN with external generic systems on superspines
- External generic systems on spines and leafs in the same pod
- Unsupported blueprint modifications:
  - Add or remove superspine planes

## 5-Stage Clos and EVPN

EVPN networks can be extended across multiple pods within the same blueprint to provide the following added value:

- **Scaling:** provide any-to-any connectivity for applications distributed across multiple pods.
- **Workloads Redistribution:** Load-balance applications by migrating a group of applications from one pod to another pod while preserving application IP and MAC addresses.
- **Maintenance:** Perform pod maintenance by migrating all applications from one pod to another, while preserving the application IP and MAC addresses.
- **Active / Standby applications across sites / pods:** Deploy A/S applications across multiple pods to provide high availability at pod level, or as part of application migration tasks.
- **Facilitate external connectivity** for a virtual network from a remote pod without external connectivity.

5-stage Clos networks support the Junos QFX series of switches. You can use the ESI redundancy protocol, create templates from them, and then use those templates as pods in 5-stage Clos networks. For more information about working with Juniper devices with EVPN, see ["Juniper EVPN Support" on page 806](#).

Just like in other Apstra-managed networks, required configuration is rendered to bring up multi-pod networks, and with proprietary *Intent-based Networking* technology the networks are validated to ensure they operate as designed.

You can create cross-pod virtual networks in the same manner as for 3-stage networks. See ["Virtual Networks" on page 488](#) for a comprehensive guide.

## Create 5-Stage Clos Network

Creating a 5-stage Clos network follows the same workflow as for ["3-stage Clos networks" on page 1](#), with the addition of creating a pod-based template and adhering to the 5-stage requirements described in the workflow below:

1. Confirm that the global catalog includes ["logical devices" on page 84](#) (Design > Logical Devices) that meet the 5-stage requirements below; create them if necessary:

- Make sure that devices have a sufficient number of ports and port groups; the exact number depends on your design.
  - Spine logical devices require a leaf-facing port group, and if they will be facing a superspine device they also require a **Superspines** port role in that port group.
  - Superspines logical devices require a **Spine** port role in the port group.
2. Confirm that the global catalog includes ["interface maps" on page 91](#) (Design > Interface Maps) that map the logical devices to the correct ["device profiles" on page 300](#); create them if necessary. The required number of interface maps depends on your design; each device model used requires its own interface map. At a minimum, if you are using only one model, you need two interface maps as listed below:
    - Superspines logical device to device profile
    - Spine logical device to device profile
  3. Create one or more rack-based ["templates" on page 110](#), each including at least one link for **Superspines Connectivity**.
  4. Create a pod-based template that uses as the pod the rack-based template(s) created in the previous step. Pod-based templates are essentially templates of templates where one or more rack-based templates are combined into a larger topology. (If you don't see the rack-based template that you created in the previous step in the pods drop-down list, it's probably because you didn't include a superspine-to-spine link.)
  5. Create pools for resources (["ASNs" on page 391](#), ["IPv4 addresses" on page 395](#), ["IPv6 addresses" on page 397](#)) needed in the network.
  6. Create a ["blueprint" on page 400](#) using the pod-based template that you created in the previous step.
  7. Build the 5-stage Clos network in the same manner as for building a 3-stage Clos network.

## Modify 5-stage Clos Network

You can modify 5-stage blueprints in the same manner as for 3-stage networks, provided that you take into account the limitations described above. For information about rack changes, see [Racks](#). For information about adding and removing pods, or changing pod names, see ["Pods" on page 481](#), and for information about adding superspines to planes see [Planes. "Racks \(Staged\)" on page 477](#)

## Access Layer Switches

When creating and managing access switches, follow the general workflow for building a network while taking into account the following options and design considerations.

1. When creating ["logical devices" on page 84](#), on leaf switches facing an access switch, select the port role **access**, and configure ports in the access switch logical device.
2. Create an ["interface map" on page 91](#) per standard procedure.

3. Create a ["rack type" on page 101](#) with configured access switches.
4. Create a ["template" on page 110](#) that uses rack types with access switches.
5. Access switch configlets are not supported. If you require configlets in access switches, contact ["Juniper Support" on page 777](#).
6. Create a ["blueprint" on page 400](#) and ["build" on page 419](#) it following the general workflow.

You can perform the same tasks as for other blueprints such as:

- ["changing link speeds" on page 473](#)
- ["adding links" on page 437](#)
- ["deleting links" on page 445](#)

**NOTE: Access Switches in Apstra Versions 4.0.0 and 4.0.1** This feature has been classified as a Juniper Apstra Technology Preview feature. These features are "as is" and voluntary use. Juniper Support will attempt to resolve any issues that customers experience when using these features and create bug reports on behalf of support cases. However, Juniper may not provide comprehensive support services to Tech Preview features.

For additional information, refer to the ["Juniper Apstra Technology Previews" on page 1082](#) page or contact ["Juniper Support" on page 777](#).

## Juniper EVPN Support

### IN THIS SECTION

- [Overview | 807](#)
- [EVPN multi-homing Terminology and Concepts | 807](#)
- [Topology Specification | 808](#)
- [EVPN Services | 810](#)
- [Configuration Rendering | 811](#)

## Overview

The Junos EVPN ESI multi-homing feature enables you to directly connect end servers to leafs and provide redundant connectivity via multi-homing. This feature is supported only on LAGs that span two leafs on the fabric. EVPN ESI also removes the need for "peer-link", and hence facilitates clean leaf-spine design.

Blueprints using the **MP-EBGP EVPN** Overlay Control Protocol can use Juniper Junos devices. Racks with leaf-pair redundancy can implement **EVPN ESI multi-homing**.

EVPN ESI multi-homing helps to maintain EVPN service and traffic forwarding to and from the multi-homed site in the event of the following types of network failures and avoid single point of failure as per the scenarios below:

- Link failure from one of the leaf devices to end server device
- Failure of one of the leaf devices
- Fast convergence on the local VTEP by changing next-hop adjacencies and maintaining end host reachability across multiple remote VTEPs

## EVPN multi-homing Terminology and Concepts

The following terminology and concepts are used with EVPN multi-homing:

**EVI** - EVPN instance that spans between the leaf devices making up the EVPN. It's represented by the Virtual Network Identifier (VNI). EVI is mapped to VXLAN-type virtual networks (VN).

**MAC-VRF** - A virtual routing and forwarding (VRF) table to house MAC addresses on the VTEP leaf device (often called a "MAC table"). A unique route distinguisher and VRF target is configured per MAC-VRF.

**Ethernet Segment (ES)** - Ethernet links span from an end host to multiple ToR leafs and form ES. It constitutes a set of bundled links.

**Ethernet Segment Identifier (ESI)** - Represents each ES uniquely across the network. ESI is only supported on LAGs that span two leafs on the fabric.

ESI helps with end host level redundancy in an EVPN VXLAN-based blueprint. Ethernet links from each Juniper ToR leaf connected to the server are bundled as an aggregated Ethernet interface. LACP is enabled for each aggregated Ethernet interface of the Juniper devices. Multi-homed interfaces into the ES are identified using the ESI.

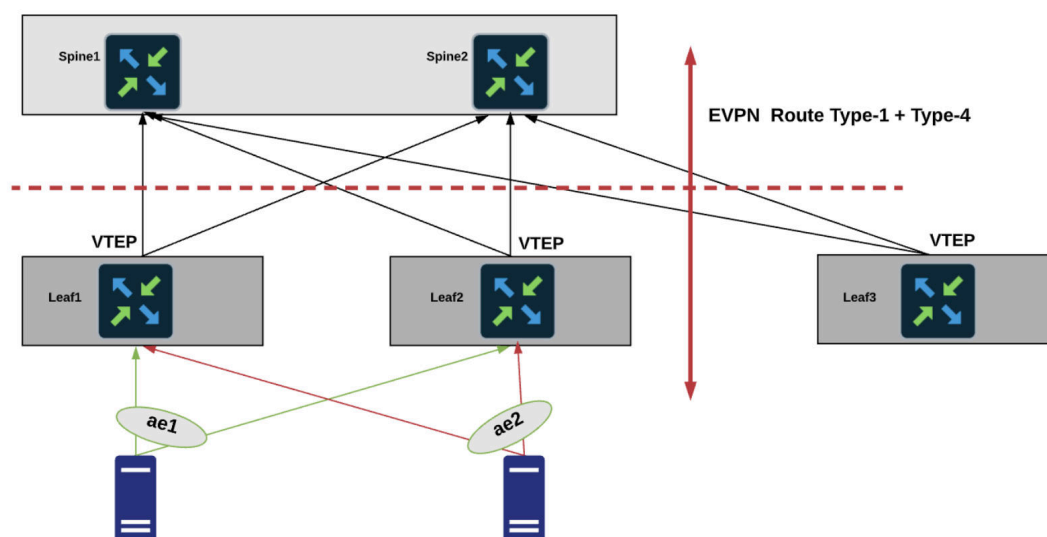
ESI has certain restrictions and requirements as listed below:

- ESI based ToR leafs cannot have any L2/L3 peer links as EVPN multi-homing eliminates peer links used by MLAG/vPC.

- A bond of two physical interfaces towards a single leaf is not supported in the ESI implementation (version 3.3.0); make sure the server with LAG in that rack type spans two leaves.
- ESI and MLAG/vPC-based rack types cannot be mixed in a single blueprint.
- L2 External Connectivity Points (ECPs) with an ESI-based rack type is not supported. Only L3 ECPs are supported.
- Per-leaf VN assignment - having different VLAN sets among individual leaves for an ESI-based port channel is not supported.
- Connecting a single server to a single leaf using a bond of two physical interfaces cannot use an ESI.
- ESI is supported only on LAGs (port-channels) and not directly on physical interfaces. This has no functional impact, as leaf local port-channels for multi-home links are automatically generated.
- Only ESI **active-active redundancy** mode is supported. Active-standby mode is not supported.
- **active-active** redundancy mode is only supported for Juniper EVPN multi-homing where each Juniper ToR leaf attached to an ES is allowed to forward traffic to and from a given VLAN.
- More than two leaves in one ESI segment using ESI-based rack types is not supported.
- Switching from an ESI to MLAG rack type or vice versa is not supported under Flexible Fabric Expansion (FFE) operations.

## Topology Specification

In the example below Leaf1 and Leaf2 are part of the same ES, and Leaf3 is the switch sending traffic towards the ES.



Juniper EVPN multi-homing uses five route types:

- Type 1 - Ethernet Auto-Discovery (EAD) Route
- Type 2 - MAC advertisement Route
- Type 3 - Inclusive Multicast Route
- Type 4 - Ethernet Segment Route
- Type 5 - IP Prefix Route

BGP EVPN running on Juniper devices use:

- Type 2 to advertise MAC and IP (host) information
- Type 3 to carry VTEP information
- Type 5 to advertise IP prefixes in a Network Layer Reachability Information (NLRI).

**NOTE:** In Junos MAC/IP Type 2 route type doesn't contain VNI and RT for the IP part of the route, it is derived from the accompanying Type 5 route type.

Type 1 routes are used for per-ES auto-discovery (A-D) to advertise EVPN multi-homing mode. Remote ToR leaf devices in the EVPN network use the EVPN Type 1 route type functionality to learn the EVPN Type 2 MAC routes from other leaf devices. In this route type ESI and the Ethernet Tag ID are considered to be part of the prefix in the NLRI. Upon a link failure between ToR leaf and end server VTEP withdraws Ethernet Auto-Discovery routes (Type 1) per ES. The Juniper EVPN multi-homing Ethernet Tag value is set to the VLAN ID for ES auto-discovery/ES route types.

**Mass Withdrawal** - Used for fast convergence during link failure scenarios between leaf devices to the end server using Type 1 EAD/ES routes.

**DF Election** - Used to prevent forwarding of the loops and the duplicates as only a single switch is allowed to decapsulate and forward the traffic for a given ES. Ethernet Segment Route is exported and imported when ESI is locally configured under the LAG. Type 4 NLRI is mainly used for designated forwarder(DF) elections and to apply Split Horizon Filtering.

**Split Horizon** - It is used to prevent forwarding of the loops and the duplicates for the Broadcast, Unknown-unicast and Multicast (BUM) traffic. Only the BUM traffic that originates from a remote site is allowed to be forwarded to a local site.

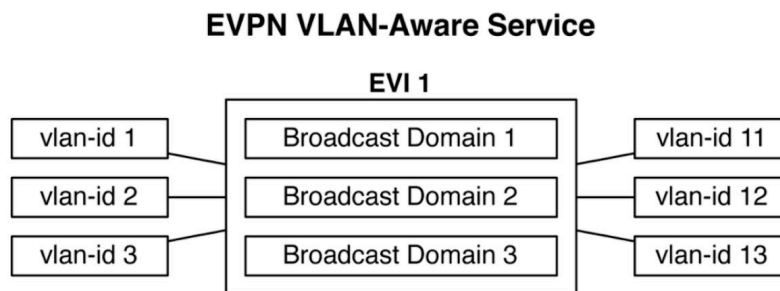
## EVPN Services

### IN THIS SECTION

- [EVPN VLAN-Aware | 810](#)
- [Create EVPN Network | 810](#)

### EVPN VLAN-Aware

At a high level, Ethernet Services can be (1) VLAN-based, (2) VLAN Bundle or (3) VLAN-Aware. Only VLAN-Aware is supported on Junos. With the EVPN VLAN-Aware Service each VLAN is mapped directly to its own EVPN instance (EVI). The mapping between VLAN, Bridge Domain (BD) and EVPN instance (EVI) is N:1:1. For example, N VLANs are mapped into a single BD mapped into a single EVI. In this model all VLAN IDs share the same EVI as shown below:



VLAN-aware Ethernet Services in Junos have a separate Route target for each VLAN (which is Juniper internal optimization), so each VLAN has a label to mimic VLAN-based implementations.

From the control plane perspective EVPN MAC/IP routes (Type 2) for VLAN-aware services carry VLAN ID in the Ethernet Tag ID attribute that is used to disambiguate MAC routes received.

From the data plane perspective - every VLAN is tagged with its own VNI that is used during packet lookup to place it onto the right Bridge Domain(BD)/VLAN.

### Create EVPN Network

Creating an EVPN network follows the same workflow as for other networks.

1. Create/Install off-box ["device agents" on page 184](#) for all switches. (On-box agents are not supported on Junos.)

2. Confirm that the global catalog includes ["logical devices" on page 84](#) (Design > Logical Devices) that meet Juniper device requirements; create them if necessary:
3. Confirm that the global catalog includes ["interface maps" on page 91](#) (Design > Interface Maps) that map the logical devices to the correct ["device profiles" on page 300](#) for the Juniper devices; create them if necessary.
4. Create a ["rack type" on page 101](#).
  - For single leaf racks, specify redundancy protocol **None** in the **Leaf** section.
  - For dual leaf racks
    - Specify redundancy protocol **ESI** in the **Leaf** section.
    - When specifying the end server in the **Server** section, specify attachment type as **Dual-Homed** towards ESI-based ToR leafs. EVPNs using ESs have a link aggregation option. Select the LAG mode **LACP (Active)**
5. Create a ["rack-based template" on page 110](#).
6. Create a generic system for an external router.
7. Create resource pools for ["ASNs" on page 391](#), ["IP addresses" on page 395](#), and ["VNI" on page 393](#).
8. Create a ["blueprint" on page 400](#) based on the ESI-based template, then build the EVPN-based network topology for the Juniper devices by assigning ["resources" on page 420](#), ["device profiles" on page 422](#), and ["device IDs" on page 423](#).

## Configuration Rendering

### IN THIS SECTION

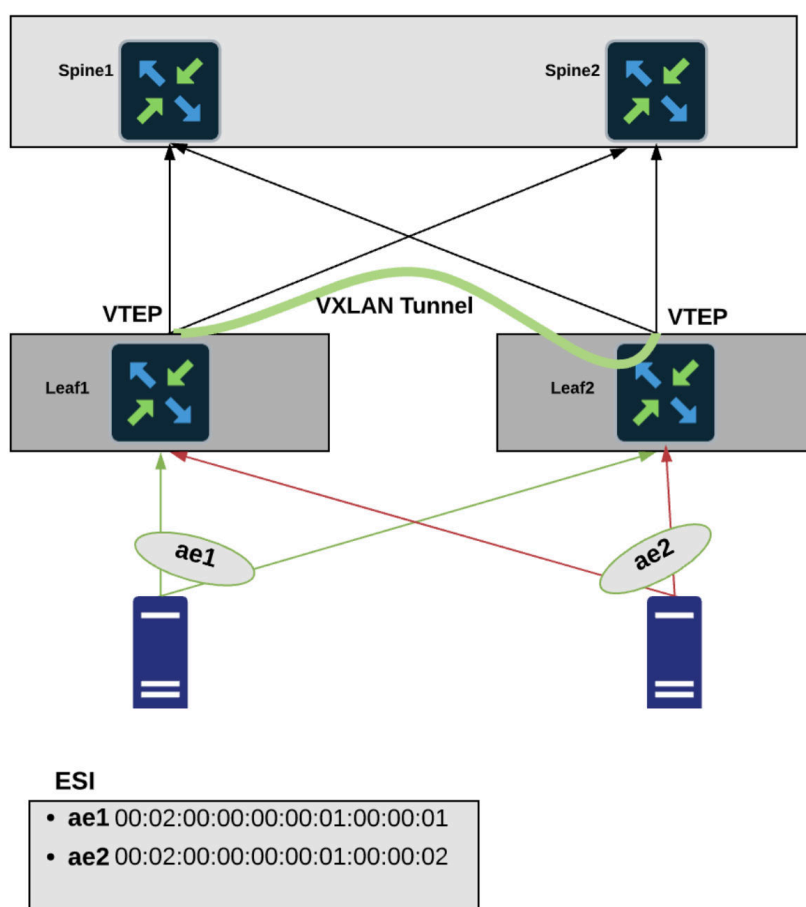
- [Reference Design | 811](#)
- [Limitations | 814](#)

### Reference Design

- **Underlay** - The underlay in the data center fabric is Layer-3 configured using standard eBGP over the physical interfaces of Juniper devices.
- **Overlay** - Overlay is configured eBGP over 100.0 address. EVPN VXLAN is used as an overlay protocol. All the ToR devices are enabled with L2 VN. Each one of these L2 VNs can have its default

gateway hosted on connected ToR Leafs. For the inter-VN traffic VXLAN routing is done in the fabric using L3 VNIs on the Border Leafs as per standard design.

- **VXLAN VTEPs** - On Juniper leafs one IP address on `100.0` is rendered which is used as VTEP address. The VTEP IP address is used to establish the VXLAN tunnel.
- **EVPN multi-homing LAG** - **Unique ESI value** and **LACP system IDs** are used per EVPN LAG. The multi-homed links are configured with an ESI and a LACP system identifier is specified for each link. The ESI is used to identify LAG groups and loop prevention. To support Active/Active and multi-homing for Juniper Leafs, they are configured with the same LACP parameter for a given ESI so that they appear as a single system.

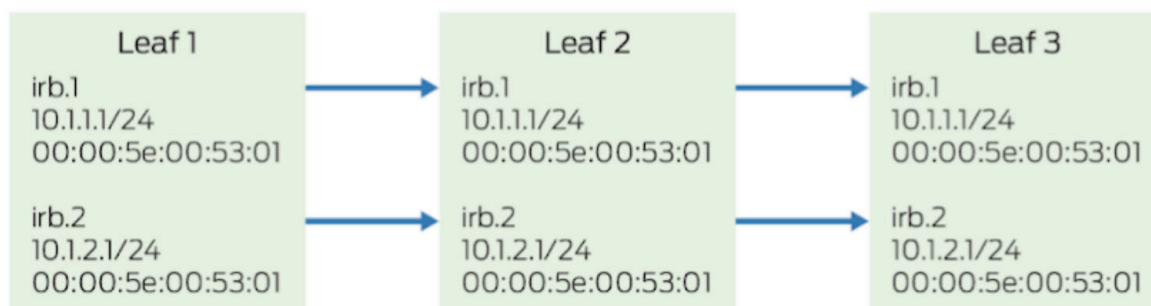


ESI MAC addresses are auto-generated internally. You can ["configure the value of the most significant byte"](#) on [page 603](#) used in the generated MAC. A new facade API is added to update the MSB value. A new node is added to the rack based template that contains the MAC MSB value. The default value of this byte is 2 and you can change it to any even number up to 254. Updating this value results in regeneration of all ESI MACs in the blueprint. This is exposed to address DCI use cases where ESIs must be unique across multiple blueprints (IP Fabrics).

- **L3VNIs** - L3VNI is rendered as a routing zone per VRF. Multi-tenancy functionality is available to ensure that workloads remain logically separated within a VN (overlay) construct using routing zone.
- **Route Target (RT) for L2/L3 VNIs** - Auto-generated for L2/L3 VNIs in the format VNI:1. There is 1 (fabric-wide) RT per MAC-VRF (that is, L3VNI). The value must be the same across all switches participating in one EVI. You can find the RT in the blueprint by navigating to **Staged > Virtual > Virtual Networks** and clicking the VN name. RT is in the parameters section.
- **Route Distinguisher (RD) for L2/L3 VNIs** - For Junos VLAN-Aware based model, the RD is per EVI (switch). There is no RD for each L2 VNI. RD exists only for routing zone VRF in the format {primary\_loopback}:vlan\_id.
- **Virtual Switch Configuration** - Under the *switch-options* hierarchy for Juniper devices the *vtep-source-interface* parameter is rendered, then the VTEP IP address used to establish the VXLAN tunnel is specified. Reachability to loopback interface (for example, lo0.0) is provided by the underlay. The RD here defines the EVI specific RD carried by Type 1, Type 2, Type 3 routes. RD for the global switch options is provided in the format {loopback\_id}:65534.

The RT here defines the global RT inherited by EVPN routes. It is used by Type 1 routes. A default RT value is rendered for it (100:100) for global switch options across all switches.

- **MTU** - The MTU values that are rendered for Juniper Devices:
  - L2 ports: 9100
  - L3 ports: 9216
  - Integrated Routing and Bridging (IRB) Interfaces: 9000
- **Anycast Gateway** - The same IP on IRB interfaces of all the leafs is configured and no virtual gateway is set. Every IRB interface that participates in the stretched L2 service has the same IP/MAC configured as below:



In this model, all default gateway IRB interfaces in an overlay subnet are configured with the same IP and MAC address. A benefit of this model is that only a single IP address is required per subnet for default gateway IRB interface addressing, which simplifies gateway configuration on end systems.

Here MAC address of the IRB is auto generated.

### Limitations

The following limitations apply to EVPN multi-homing topologies for Juniper devices as of version 3.3.0:

- Only two-way multi-homing is supported. More than two Juniper leafs in a multi-homed group is not supported.
- Juniper EVPN with EVPN on other network vendors in the same blueprint is not supported.
- No Static VXLAN support.
- IPv6-based fabrics do not support Junos.
- In Juniper EVPN multi-homing, L3 External Connectivity Points (ECP) towards generic systems are supported; L2 ECP is not supported.
- BGP routing from Junos leafs to Apstra-managed Layer 3 servers is not supported.

## Intent-Based Analytics with AOS-CLI Utility

### IN THIS SECTION

- [IBA with AOS-CLI Overview | 814](#)
- [Install AOS-CLI | 815](#)
- [Install Packages | 815](#)
- [Create Agent Profiles | 818](#)
- [Create Agents | 819](#)
- [Update Agents from AOS-CLI | 821](#)
- [Install IBA Probes | 822](#)
- [Apstra IBA Probes Examples | 824](#)

### IBA with AOS-CLI Overview

After you've deployed a blueprint you can get started with intent-based analytics (IBA). You can work with IBA using the Apstra GUI, or for non-production environments you can use the experimental aos-cli

utility. For information about how to use IBA probes from the GUI, see ["Probes" on page 410](#) in the Analytics section. This guide shows you how to use aos-cli.

**NOTE:** The aos-cli utility is an experimental tool and has limited support. Do not use it in production environments unless advised by Juniper Support. Some versions are not intended for certain Apstra releases. Some aos-cli commands may or may not work between different Apstra releases. It's always best to test a version of aos-cli with a specific Apstra release in a non-production environment, or contact ["Juniper Support" on page 777](#) for assistance.

The aos-cli utility enables you to extract information from the Apstra server for analytics (and other functionalities). The workflow for IBA probes is as follows:

1. Install aos-cli.
2. Install packages.
3. Create device agent profiles.
4. Install device agents.
5. Install IBA probes.

After probes are instantiated you can use Syslog to send messages to Syslog servers. See ["Syslog Configuration" on page 718](#) for details.

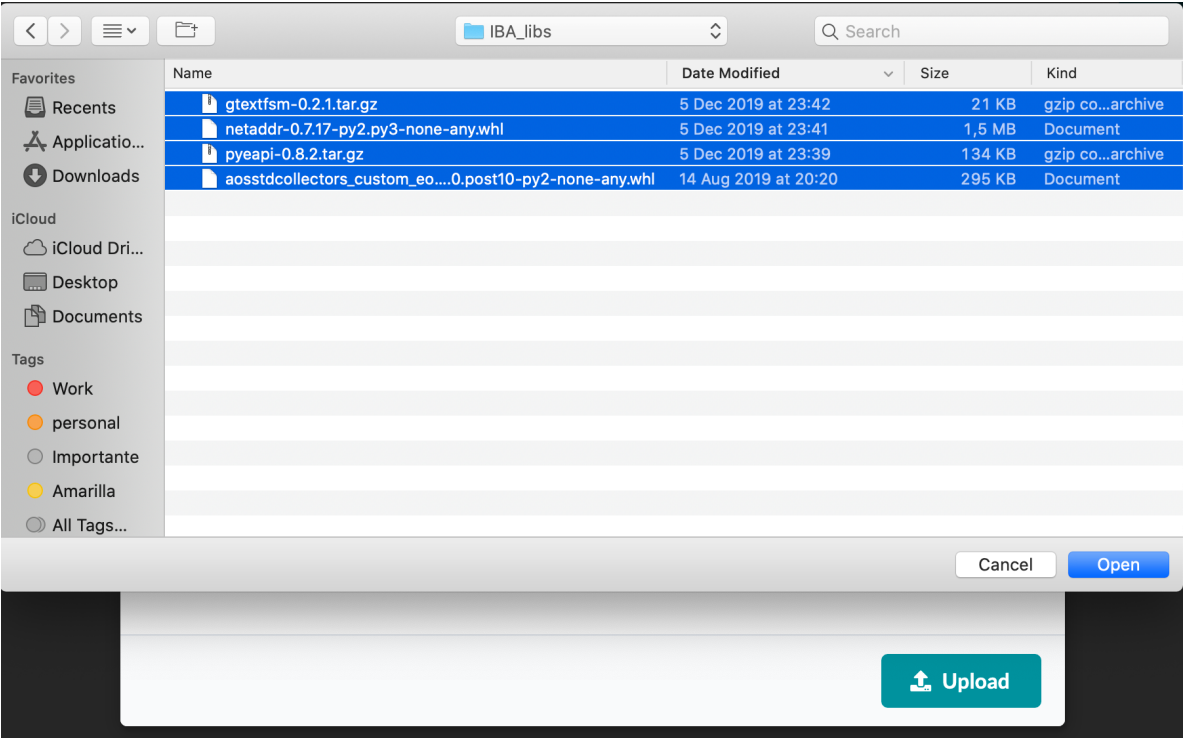
## Install AOS-CLI

Install the aos-cli utility per instructions in [Juniper Support Knowledge Base article KB36747](#).

## Install Packages

1. Download the latest Apstra SDK package from [Juniper Support Knowledge Base article KB37156](#).
2. Custom collector packages enable the collection of telemetry from devices. Extract the collector for your platform (for example, aosstdcollectors\_custom\_eos-0.1.0.post10-py2-none-any.whl where eos is the platform and 10 is the version).
3. Collectors require specific Python library packages. If the Apstra environment has Internet access, the files are automatically installed. If the environment doesn't have Internet access, download the following files from the official Python repository. Make sure to download the correct versions:
  - netaddr-0.7.17-py2.py3-none-any.whl
  - gtextfsm-0.2.1.tar.gz
  - pyeapi-0.8.2.tar.gz

4. From the left navigation menu in the Apstra GUI, navigate to **Devices > System Agents > Packages** and click **Upload Packages**.
5. Either click **Choose File** and navigate to the custom collector package (and if the Internet is inaccessible, the three (3) Python packages), or drag and drop the file(s) into the dialog window. See example below for Arista devices in an environment without Internet access:



Upload Packages

Drag and drop files here or click the button below.

Choose Files

aosstdcollectors_custom_eos-0.1.0.post10-py2-none-any.whl	295kB	✗
gtextfsm-0.2.1.tar.gz	21kB	✗
netaddr-0.7.17-py2.py3-none-any.whl	1.53MB	✗
pyeapi-0.8.2.tar.gz	134kB	✗

Upload

- 6. Click **Upload** to upload the packages to the Apstra server, then close the dialog to return to the list view.

Create Agent Profiles

With agent profiles you can specify packages once in the profile, then apply the profile to multiple agents at the same time. Let's create a profile that contains all four packages. (Remember, if your environment has Internet access, you only need to include the custom collector package.)

- 1. From the left navigation menu, navigate to **Devices > System Agents > Agent Profiles** and click **Create Agent Profile**.
- 2. For this example, select **EOS** from the platform drop-down list.

Create Agent Profile

Profile Parameters

Name \*

EOS-IBA

Platform

EOS

Username

☐ Set username?

Password

☐ Set password?

3. In the **Packages** section, select the four uploaded packages to associate them with the agent profile.

**Create Agent Profile**

Key	Value
No options	
<a href="#">Add an option</a>	

Packages **4**

Query: All 1-4 of 4 Page Size: 25

<input checked="" type="checkbox"/>	Name	Version
<input checked="" type="checkbox"/>	aosstdcollectors-custom-eos	0.1.0.post10
<input checked="" type="checkbox"/>	gtextfsm	0.2.1
<input checked="" type="checkbox"/>	netaddr	0.7.17
<input checked="" type="checkbox"/>	pyeapi	0.8.2

☐ Create Another? [Create](#)

4. Click **Create** to create the agent profile and return to the list view.

For more information about agent profiles, see ["Agent Profiles"](#) on page 260.

## Create Agents

Now let's create agents for Arista devices and use the agent profile to associate the packages to them. We recommend that you use agent profiles to associate custom collector packages so you can bulk update agents later, as needed, with a single command.

1. From the left navigation menu, navigate to **Devices > System Agents > Agents** and click **Create Onbox Agent(s)**.

2. Enter details for the agent and select the agent profile from the drop-down list as shown in the image below:

Create System Agent(s)

Device Addresses (25 max) \*

192.168.1.5-192.168.1.10

Comma-separated list of hostnames, individual IP addresses, and IP address ranges, e.g. '192.168.1.5-192.168.1.10,mydevice.local'

192.168.1.5

192.168.1.6

192.168.1.7

192.168.1.8

192.168.1.9

192.168.1.10

Operation Mode

☒ FULL CONTROL
 ☐ TELEMETRY ONLY

Username

☒ Set username?

Password

☒ Set password?

Agent Profile

EOS-IBA

Job to run after creation

☐ Check
 ☒ Install

☒ Install Requirements ⓘ

Create

3. To verify that packages have been successfully installed on agents, from the left navigation menu, navigate to **Devices > System Agents < Agents** and click the device address of the device to check. The **Config** section lists the installed packages. If you manually uploaded the Python packages (netaddr, gtextfsm and pyeapi) they are listed. If the Apstra server has Internet access, they were automatically uploaded and won't be listed here. (To see all packages installed on the device, log into

the device and check the /tmp/plugins folder.)

[Home](#) › [Devices](#) › [Agents](#) › 80f490f0-b909-435e-93ce-c6f28176a5d2

✓

Expanded View

Compact View

Config

Device Address	172.20.38.8
Operation Mode	FULL CONTROL
Profile	Not Selected
Packages	pyeapi==0.8.2 netaddr==0.7.17 gtextfsm==0.2.1 aossdcollectors-custom-eos==0.1.0.post10

Status

For more information about agents, see ["Agents" on page 184](#).

Update Agents from AOS-CLI

As of apstra build 423, you can update agents with a given agent profile, as needed, based on IP/ID or OS type (os\_type) (for example, EOS).

To update agents by IP range with a specific agent profile, use the command system-agents update-profile as shown in the example below. When setting the --profile option, aos-cli shows available agent profiles. To select, use the up and down arrow keys.

```
aos> system-agents update-profile --ip 172.20.120.6-11 --profile
EOS-IBA  EOS
```

For example.

```
aos> system-agents update-profile --ip 172.20.120.6-11 --profile 692bb0bb-c5e0-4d7e-a70c-
c24b0d5650a8
Successfully updated agent 172.20.120.9 with given profile
Successfully updated agent 172.20.120.6 with given profile
```

```

Successfully updated agent 172.20.120.11 with given profile
Successfully updated agent 172.20.120.7 with given profile
Successfully updated agent 172.20.120.10 with given profile
Successfully updated agent 172.20.120.8 with given profile
aos>

```

## Install IBA Probes

You can install IBA probes using the Apstra GUI, or for non-production environments you can use `aos-cli`. For information about how to create or instantiate predefined probes from the GUI, see ["Probes" on page 410](#) in the Analytics section. This section shows you how to use the `aos-cli` utility.

All probes described in this document are included in `aos-cli` build 412 and later. Probe `.j2` files may be made available if the probe file is not built into the `aos-cli` build.

Some of these probes require an updated service registry. Download the latest Apstra SDK and extract the `json-schemas.tar.gz` file. Copy the file to the `/home/admin` directory of the Apstra server so it is available in the `aos-cli/mytmp` directory.

```

aos> service-registry import-from --file /mytmp/json-schemas.tar.gz
Successfully imported service registry entry for interface_details
Successfully imported service registry entry for route_count
Successfully imported service registry entry for multicast_groups
Successfully imported service registry entry for sfp
Successfully imported service registry entry for resource_usage
Successfully imported service registry entry for mlag_domain
Successfully imported service registry entry for stp
Successfully imported service registry entry for vtep_counters
Successfully imported service registry entry for vlan
Successfully imported service registry entry for evpn_type5
Successfully imported service registry entry for ping
Successfully imported service registry entry for vxlan_info
Successfully imported service registry entry for pim_neighbor_count
Successfully imported service registry entry for lldp_details
Successfully imported service registry entry for evpn_type3
Successfully imported service registry entry for multicast_info
Successfully imported service registry entry for bgp_vrf
Successfully imported service registry entry for traceroute
Successfully imported service registry entry for vrf
Successfully imported service registry entry for table_usage
Successfully imported service registry entry for vxlan_address_table
Successfully imported service registry entry for acl_stats
Successfully imported service registry entry for device_info

```

```

Successfully imported service registry entry for power_supply
Successfully imported service registry entry for interface_buffer
Successfully imported service registry entry for pim_rp
Successfully imported service registry entry for anycast_rp
Successfully imported service registry entry for bgp_iba
Successfully imported service registry entry for interface_iba
aos>

```

To create probes, use the `probe create aos-cli` command. You'll be prompted for additional options.

```

aos> probe create
--blueprint      Id of the blueprint
--file           Filename of json file with probe data. Choose from dropdown or specify
custom path
--skip-service-check [Optional] By default, required telemetry services are checked and enabled
on target
--check-status    [Optional] Wait for probe to become operational. Default: False
--service-interval When skip-service-check is False and service is not already present, this
indicates

```

To select the blueprint ID, use `--blueprint` and tab-completion.

```

aos> probe create --blueprint 67cd936d-c2de-49f8-8708-df465f0cdc68
                                L2 Virtual  two_stage_l3clos

```

To list available probes supplied with `aos-cli`, use `--file` and tab-completion. Scroll through the list with the up and down arrow keys.

```

aos> probe create --blueprint 67cd936d-c2de-49f8-8708-df465f0cdc68 --file
                                                                    evpn.j2
                                                                    sfp.j2

memory_usage_threshold_anomalies.j2

bandwidth_utilization_history.j2                                                                    power_supply_anomalies.j2

virtual_infra_vlan_mismatch.j2

hardware_vtep_counters_enabled.j2

```

Some probes need additional Probe template variables.

```
aos> probe create --blueprint 67cd936d-c2de-49f8-8708-df465f0cdc68 --file /usr/local/lib/
python2.7/site-packages/aos_cli/resources/probes/memory_usage_threshold_anomalies.j2
--skip-service-check [Optional] By default, required telemetry services are checked and enabled
on target
--check-status       [Optional] Wait for probe to become operational. Default: False
--service-interval   When skip-service-check is False and service is not already present, this
indicates
--process            Probe template variable
--os_family          Probe template variable
```

To see installed IBA probes in the blueprint, navigate to **Analytics > Probes**.

Apstra IBA Probes Examples

IN THIS SECTION

- [Packet Drops | 824](#)
- [SFP Optics | 825](#)
- [Switch Memory Leak \(Arista EOS only\) | 825](#)
- [Fault Tolerance | 827](#)

The following section describes how to install some of the most interesting probes which are not available by default.

Packet Drops

Packet drop IBA probes detect an abnormal amount of packet drops on interfaces of devices managed by Apstra based on interface telemetry collected by device agents.

Filename	Description
pkt_discard_anomalies.j2	Detect Fabric interfaces having sustained packet discards

To install the `pkt_discard_anomalies.j2` IBA Probe:

```
aos> probe create --blueprint 67cd936d-c2de-49f8-8708-df465f0cdc68 --file /usr/local/lib/
python2.7/site-packages/aos_cli/resources/probes/pkt_discard_anomalies.j2
Ensuring needed telemetry services for probe are enabled...
Successfully created probe f472ba21-d60f-44dc-9f5d-8318c8b9c07b in blueprint 67cd936d-
c2de-49f8-8708-df465f0cdc68
aos>
```

## SFP Optics

SFP optic IBA probes detect high and/or low warning thresholds in SFP RX power, TX power, temperature, voltage, or current for compatible optical modules on interfaces of devices managed by Apstra based on SFP telemetry collected by device agents.

Filename	Description
sfp.j2	Detect high and/or low warning thresholds in SFP RX Power, TX Power, Temperature, Voltage, or Current

To install the `sfp.j2` IBA Probe:

```
aos> probe create --blueprint 67cd936d-c2de-49f8-8708-df465f0cdc68 --file /usr/local/lib/
python2.7/site-packages/aos_cli/resources/probes/sfp.j2
Ensuring needed telemetry services for probe are enabled...
Enabled service sfp on device l2-virtual-002-leaf1:172.20.60.11
Enabled service sfp on device l2-virtual-001-leaf1:172.20.60.9
Enabled service sfp on device spine2:172.20.60.8
Enabled service sfp on device spine1:172.20.60.6
Enabled service sfp on device l2-virtual-003-leaf1:172.20.60.10
Enabled service sfp on device l2-virtual-004-leaf1:172.20.60.7
Successfully created probe b0c32a46-636c-4f82-b026-e9925b696625 in blueprint 67cd936d-
c2de-49f8-8708-df465f0cdc68
aos>
```

## Switch Memory Leak (Arista EOS only)

Switch Memory Leak IBA probes detect abnormal memory leaks in specified processes on devices managed by Apstra based on system telemetry collected by device agents. This probe requires device

user credentials set in the device agent configuration that has login and access to the device BASH prompt.

Filename	Description
memory_usage_threshold_anomalies.j2	Detect memory leaks in specified process on all switches in the Fabric
system_memory_usage_threshold_anomalies.j2	Detect switches having potential memory leaks in the Fabric

The `memory_usage_threshold_anomalies.j2` IBA probe requires additional "Probe template variables" for `os_family` and `process`.

```
aos> probe create --blueprint 67cd936d-c2de-49f8-8708-df465f0cdc68 --file /usr/local/lib/
python2.7/site-packages/aos_cli/resources/probes/memory_usage_threshold_anomalies.j2
  --skip-service-check [Optional] By default, required telemetry services are checked and
enabled on target
  --check-status       [Optional] Wait for probe to become operational. Default: False
  --service-interval   When skip-service-check is False and service is not already present, this
indicates
  --process            Probe template variable
  --os_family          Probe template variable
```

The only option for `os_family` is `eos` for Arista EOS. The (2) options for `process` are `edac-poller` and `fastcapi` or `configagent`.

```
aos> probe create --blueprint 67cd936d-c2de-49f8-8708-df465f0cdc68 --file /usr/local/lib/
python2.7/site-packages/aos_cli/resources/probes/memory_usage_threshold_anomalies.j2 --os_family
eos --process fastcapi
Ensuring needed telemetry services for probe are enabled...
Enabled service resource_usage on device l2-virtual-002-leaf1:172.20.60.11
Enabled service resource_usage on device l2-virtual-001-leaf1:172.20.60.9
Enabled service resource_usage on device spine2:172.20.60.8
Enabled service resource_usage on device spine1:172.20.60.6
Enabled service resource_usage on device l2-virtual-003-leaf1:172.20.60.10
Enabled service resource_usage on device l2-virtual-004-leaf1:172.20.60.7
Successfully created probe 6a258d83-1053-42ad-935c-0550cc500b7d in blueprint 67cd936d-
```

```
c2de-49f8-8708-df465f0cdc68
```

```
aos>
```

```
aos> probe create --blueprint rack-based-blueprint-10990707 --file /usr/local/lib/python2.7/site-
packages/aos_cli/resources/probes/memory_usage_threshold_anomalies.j2 --os_family eos --process
configagent
```

```
Ensuring needed telemetry services for probe are enabled...
```

```
Successfully created probe ed2c6be1-b4b1-4e1b-bd07-da431e89eeec in blueprint rack-based-
blueprint-10990707
```

```
aos>
```

**NOTE:** "FastCapi" as service process is valid only for EOS version 4.18. For the newer version of EOS, for example 4.20 and later only ConfigAgent is valid. Take extra care that service name is in lowercase during probe creation. So it should be configagent instead of ConfigAgent.

To install the IBA probe for a second process, repeat the `probe create` command for the other process.

You can edit the IBA probe name to include the process name.

To install the `system_memory_usage_threshold_anomalies.j2` IBA probe:

```
aos> probe create --blueprint 67cd936d-c2de-49f8-8708-df465f0cdc68 --file /usr/local/lib/
python2.7/site-packages/aos_cli/resources/probes/system_memory_usage_threshold_anomalies.j2
```

```
Ensuring needed telemetry services for probe are enabled...
```

```
Successfully created probe a669ccf8-cba7-414b-ad46-a7d4b4ca3928 in blueprint 67cd936d-
c2de-49f8-8708-df465f0cdc68
```

```
aos>
```

## Fault Tolerance

These (2) probes require aos-cli build 430 or later.

Filename	Description
spine_fault_tolerance.j2	Find out if failure of given number of spines in the fabric is going to be tolerated. Raise anomaly if total traffic on all spines is more than the available spine capacity, with the specified number of spine failures.

(Continued)

Filename	Description
lag_link_fault_tolerance.j2	Find out if failure of one link in a server LAG is going to be tolerated. Monitors total traffic in each LAG against total available capacity of the bond, with one link failure. Raise anomaly for racks with more than 50% of such overused bonds, sustained for certain duration.

To install the spine\_fault\_tolerance.j2 IBA Probe:

```
aos> probe create --blueprint bf7a322c-ee3a-4dcf-aa20-df0560f538da --file /usr/local/lib/
python2.7/site-packages/aos_cli/resources/probes/spine_fault_tolerance.j2 --
number_of_faulty_spines_to_be_tolerated 1
Successfully created probe 0f0e9bf7-d9b3-43d7-906e-a9f0675e68f2 in blueprint bf7a322c-ee3a-4dcf-
aa20-df0560f538da
aos>
```

**NOTE:** number\_of\_faulty\_spines\_to\_be\_tolerated must be specified.

To install the lag\_link\_fault\_tolerance.j2 IBA Probe:

```
aos> probe create --blueprint bf7a322c-ee3a-4dcf-aa20-df0560f538da --file /usr/local/lib/
python2.7/site-packages/aos_cli/resources/probes/lag_link_fault_tolerance.j2
Successfully created probe 45ce5fe8-555f-41a9-b0ae-267125669d3f in blueprint bf7a322c-ee3a-4dcf-
aa20-df0560f538da
aos>
```

## AOSOM-Streaming Guide

### IN THIS SECTION

- [AOSOM-Streaming Overview | 829](#)
- [Configure Aosom-Streaming | 834](#)

- [Reconfigure Aosom-streaming after Apstra Server Upgrade | 836](#)
- [Build Aosom-Streaming VM \(Optional\) | 837](#)
- [Troubleshooting | 841](#)

## AOSOM-Streaming Overview

### IN THIS SECTION

- [Grafana | 830](#)
- [Prometheus | 831](#)
- [InfluxDB | 833](#)

**NOTE:** AOSOM streaming is demonstration software, not intended for production environments.

You can configure Apstra to generate Google Protocol Buffer (protobuf) streams for counter data (perfmon), alerts, and events. Each data type is sent to a streaming receiver over its own TCP socket. Even if all three data types are configured for the same streaming receiver, three connections are created between the Apstra server and the streaming receiver. This also allows for all three types to be sent to three different streaming receivers. You can choose from the many open-source projects, or develop your own solutions to capture, store and inspect the protobuf data. Apstra has developed a project available on GitHub called [AOSOM-Streaming](#) to demonstrate how this can be achieved using several open-source components. The AOSOM-Streaming project is meant to help you understand how you can consume the AOS protobuf stream. It is for demonstration purposes only, except for the Apstra Telegraf input plugin. This plugin is fully supported by Apstra for you to use as part of your streaming telemetry solution.

The Aosom Streaming project provides a packaged solution to collect and visualize telemetry streaming information coming from an Apstra server. This provides a web interface experience and example queries to handle alerts, counters, and Apstra events. This open-source project officially lives on Github at <https://github.com/Apstra/aosom-streaming>.

The packaged solution includes:

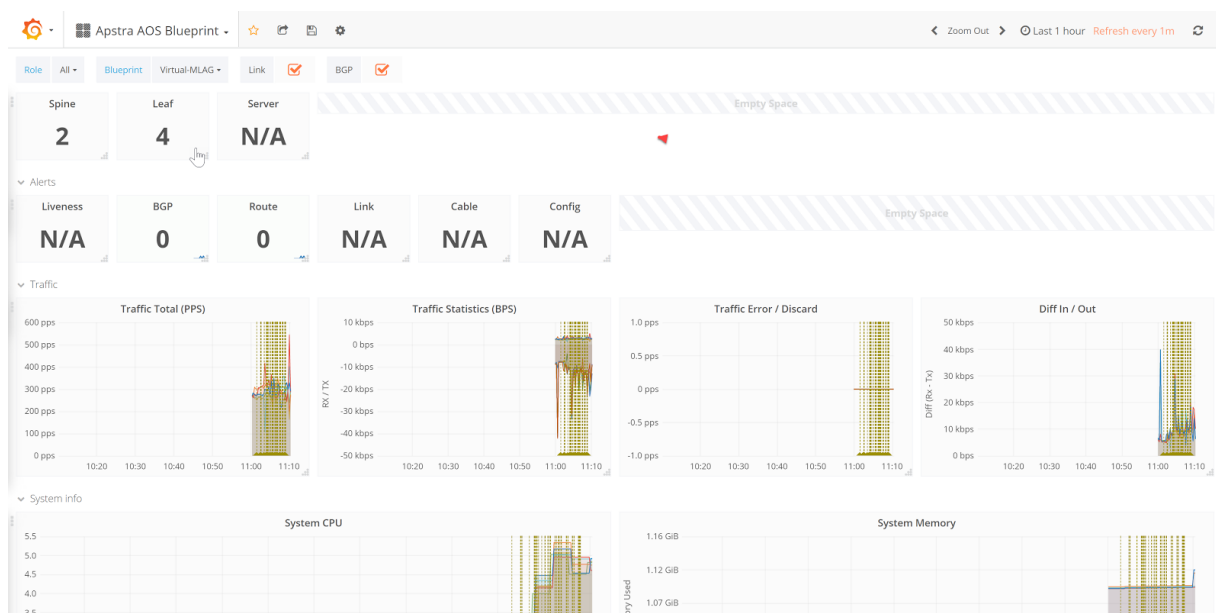
- A graphical Interface based on Grafana (port 3000)

- Prometheus for Counters and Alerts (port 9090)
- Influxdb for Events (port 8086)
- 2 Collectors, one for each database based on Telegraf.

## Grafana

From a web browser enter the URL **http://<aosom-streaming>:3000** and enter username **admin** (default) and password **admin** (default).

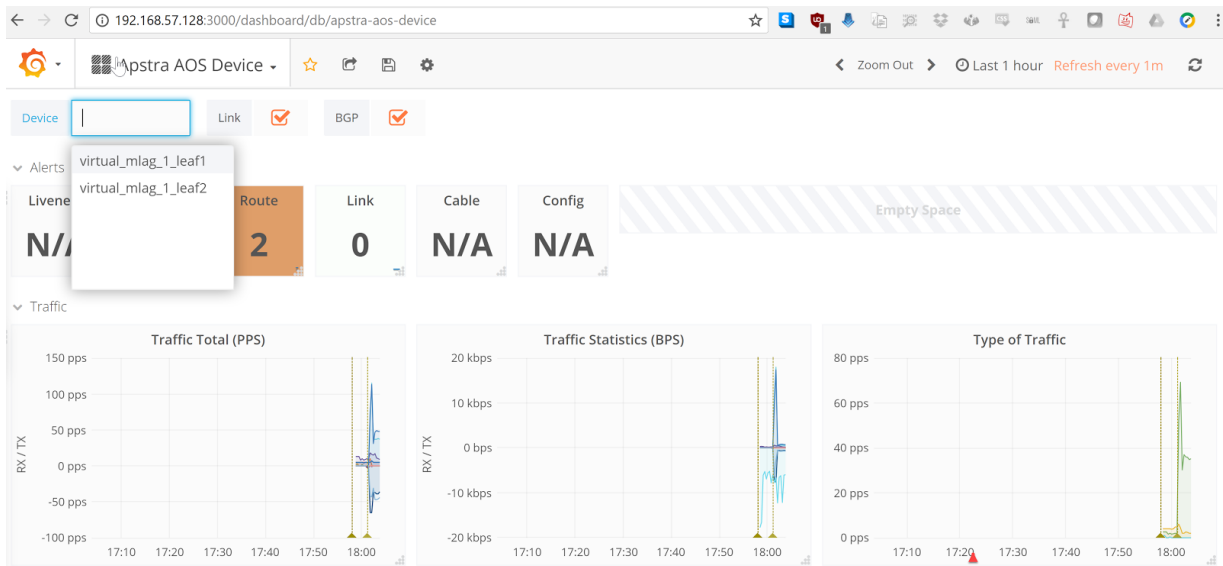
The grafana GUI includes two main sections (top left). **Apstra AOS Blueprint** describes overall telemetry alerts and traffic throughput, as well as individual devices for interface telemetry. Blueprints are learned automatically using the Apstra 'telegraf' Docker container; no further configuration is necessary.



In the screenshot above, we can observe traffic in the demo Apstra environment, and aggregate CPU, traffic, and errors.

To filter telemetry events based on specific and individual devices, change the dashboard at the top to **Apstra AOS Device**. Here we can observe there are two active route anomalies in the blueprint, and

Apstra has received telemetry for two leaf switches.



Scroll down to view device statistics such as CPU and Memory:

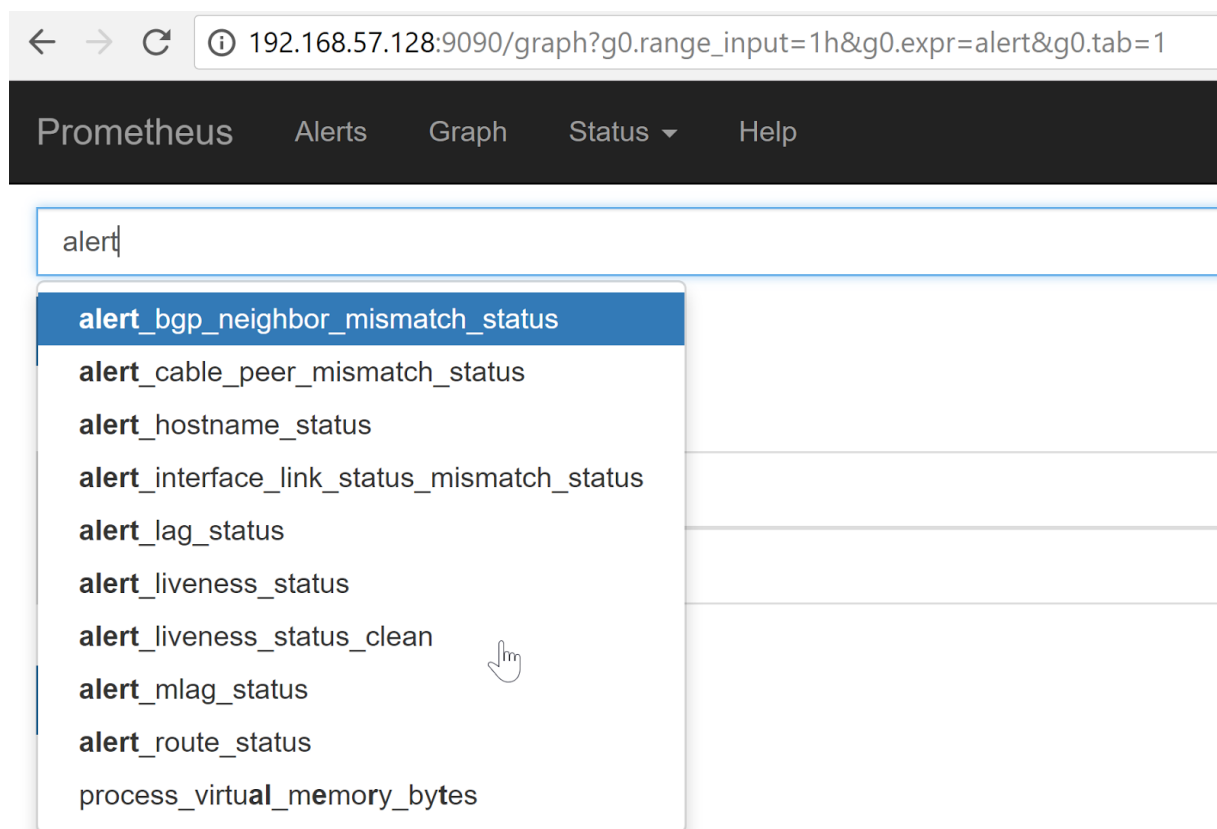


## Prometheus

Prometheus is used for alerts and device telemetry counter storage in the Aiosom-streaming appliance. From a web browser enter the URL <http://<aosom-streaming>:9090> to access the Prometheus GUI.

When incoming events appear, Apstra dynamically builds each of the queries. To see example query names, begin typing under 'execute'. Starting with 'alert' it tab-completes available alerts that

Prometheus has received from Apstra.



The screenshot shows the Prometheus web interface. The browser's address bar displays the URL `192.168.57.128:9090/graph?g0.range_input=1h&g0.expr=alert&g0.tab=1`. The navigation bar includes links for Prometheus, Alerts, Graph, Status, and Help. In the Alerts section, a search input field contains the text "alert". A dropdown menu is open, listing several alert metrics: `alert_bgp_neighbor_mismatch_status`, `alert_cable_peer_mismatch_status`, `alert_hostname_status`, `alert_interface_link_status_mismatch_status`, `alert_lag_status`, `alert_liveness_status`, `alert_liveness_status_clean`, `alert_mlag_status`, `alert_route_status`, and `process_virtual_memory_bytes`. A mouse cursor is positioned over the `alert_liveness_status_clean` option.

Here is an example of BGP Neighbors being offline.

PrometheusAlertsGraphStatus▼Help

alert\_bgp\_neighbor\_mismatch\_status

Execute

- insert metric at cursor -

GraphConsole

Element

alert\_bgp\_neighbor\_mismatch\_status(actual\_state="BGP\_SESSION\_DOWN",blueprint="Virtual-MLAG",device="virtual\_mlag\_1\_leaf2",device\_key="000C29CFDEAF",device\_name="virtual\_mlag\_1\_leaf2",expected\_state="BGP\_SESSION\_UP",host="429328fbb5ac",instance="telegraf-prom:9126",job="aos-streaming",lcl\_asn="101",lcl\_hostname="virtual-mlag-1-leaf2",lcl\_ipaddr="10.0.0.3",rmt\_asn="100",rmt\_ipaddr="10.0.0.2",role="leaf",severity="ALERT\_CRITICAL")

alert\_bgp\_neighbor\_mismatch\_status(actual\_state="BGP\_SESSION\_MISSING",blueprint="Virtual-MLAG",device="virtual\_mlag\_1\_leaf2",device\_key="000C29CFDEAF",device\_name="virtual\_mlag\_1\_leaf2",expected\_state="BGP\_SESSION\_UP",host="429328fbb5ac",instance="telegraf-prom:9126",job="aos-streaming",lcl\_asn="101",lcl\_ipaddr="10.0.0.15",rmt\_asn="3",rmt\_ipaddr="10.0.0.14",role="leaf",severity="ALERT\_CRITICAL")

alert\_bgp\_neighbor\_mismatch\_status(actual\_state="BGP\_SESSION\_MISSING",blueprint="Virtual-MLAG",device="virtual\_mlag\_1\_leaf2",device\_key="000C29CFDEAF",device\_name="virtual\_mlag\_1\_leaf2",expected\_state="BGP\_SESSION\_UP",host="429328fbb5ac",instance="telegraf-prom:9126",job="aos-streaming",lcl\_asn="101",lcl\_ipaddr="10.0.0.7",rmt\_asn="2",rmt\_ipaddr="10.0.0.6",role="leaf",severity="ALERT\_CRITICAL")

alert\_bgp\_neighbor\_mismatch\_status(actual\_state="BGP\_SESSION\_MISSING",blueprint="Virtual-MLAG",device="virtual\_mlag\_1\_leaf1",device\_key="000C297823FD",device\_name="virtual\_mlag\_1\_leaf1",expected\_state="BGP\_SESSION\_UP",host="429328fbb5ac",instance="telegraf-prom:9126",job="aos-streaming",lcl\_asn="100",lcl\_hostname="virtual-mlag-1-leaf1",lcl\_ipaddr="10.0.0.2",rmt\_asn="101",rmt\_ipaddr="10.0.0.3",role="leaf",severity="ALERT\_CRITICAL")

F

InfluxDB

InfluxDB is used to store Apstra events from telemetry streaming. From a web browser enter the URL <http://<aosom-streaming>.8083> to access InfluxDB.

We can show the available influxdb keys with queries, such as **show field keys** or **show measurements**.

←→↺192.168.57.128:8083

☆

InfluxDB

Write DataDocumentation

Database: aos⌵⚙

Query: show field keys

Generate Query URL

Query Templates⌵

event\_arp\_state

fieldKey	fieldType
event	"integer"

event\_bgp\_neighbor

fieldKey	fieldType
event	"integer"

event\_cable\_peer

fieldKey	fieldType
event	"integer"

Once we know a measurement, we can view the data and keys with `select * from <measurement>` -- In this case, we'll capture the LAG interface status.

The screenshot shows the InfluxDB web interface. The query bar contains `select * from event_lag_state`. Below the query bar, the results are displayed as a table with the following columns: `time`, `blueprint`, `device`, `device_key`, `device_name`, `event`, `host`, `interfacesup`, `interfacesupcount`, `lagname`, and `role`.

time	blueprint	device	device_key	device_name	event	host	interfacesup	interfacesupcount	lagname	role
2017-07-31T23:57:58.524206286Z	"Virtual-MLAG"	"virtual_mlag_1_leaf1"	"000C297823FD"	"virtual_mlag_1_leaf1"	1	"0a84241e1366"	"[]"	"0"	"port-channel3"	"leaf1"
2017-07-31T23:57:58.52422376Z	"Virtual-MLAG"	"virtual_mlag_1_leaf1"	"000C297823FD"	"virtual_mlag_1_leaf1"	1	"0a84241e1366"	"[]"	"1"	"port-channel3"	"leaf1"
2017-07-31T23:57:58.524249294Z	"Virtual-MLAG"	"virtual_mlag_1_leaf1"	"000C297823FD"	"virtual_mlag_1_leaf1"	1	"0a84241e1366"	"[]"	"2"	"port-channel3"	"leaf1"
2017-07-31T23:57:58.524272244Z	"Virtual-MLAG"	"virtual_mlag_1_leaf1"	"000C297823FD"	"virtual_mlag_1_leaf1"	1	"0a84241e1366"	"[]"	"0"	"port-channel1"	"leaf1"
2017-07-31T23:57:58.524299294Z	"Virtual-MLAG"	"virtual_mlag_1_leaf1"	"000C297823FD"	"virtual_mlag_1_leaf1"	1	"0a84241e1366"	"[]"	"1"	"port-channel1"	"leaf1"

**NOTE:** Developing an influx-db application is beyond the scope of this documentation.

## Configure Aosom-Streaming

To configure telemetry streaming as part of this project, you'll edit `variables.env`, run the `make start` file and restart the containers. No Apstra server configuration is required. Documentation for starting, stopping, and clearing data is available at <https://github.com/Apstra/aosom-streaming>

The telegraf project connects to the Apstra API and posts an IP:Port that Apstra uses to stream realtime telemetry data back to.

1. Copy `variables.default` to `variables.env`:

```
aosom@ubuntu:~/aosom-streaming$ cp variables.default variables.env
```

2. Configure `variables.env`.

```
AOS_SERVER=192.168.57.250
LOCAL_IP=192.168.57.128

INPUT_PORT_INFLUX=4444
INPUT_PORT_PROM=6666
AOS_LOGIN=admin
AOS_PASSWORD=admin
```

```
AOS_PORT=443
```

```
GRAFANA_LOGIN=admin
```

```
GRAFANA_PASSWORD=admin
```

- AOS\_SERVER - the IP address of the Apstra server that sends telemetry data to the aosom-streaming server.
- LOCAL\_IP - the IP address assigned to ens33 (first ethernet interface). In this case, it is learned via DHCP on this VM. See `ip addr show dev ens33`. GRAFANA configuration options to specify the username and password for the grafana web interface.
- AOS\_LOGIN, AOS\_PASSWORD, AOS\_PORT - You can customize username, port and password information.

3. Set up the project by running the command `make start`, or if you're making configuration changes, run `make update`.

```
aosom@ubuntu:~/aosom-streaming$ make start
-- Start all components --
Creating network "aosomstreaming_default" with the default driver
Creating volume "aosomstreaming_grafana_data_2" with default driver
Pulling telegraf-influx (apstra/telegraf:1.2)...
1.2: Pulling from apstra/telegraf
00d19003217b: Pull complete
72dd23d7de04: Pull complete
cf6581f43cce: Pull complete
Digest: sha256:1539d4b84618abb44bdfb1e0a27399a7272814be36535f4a7dfa04661d6e5f6
Status: Downloaded newer image for apstra/telegraf:1.2
Pulling prometheus (prom/prometheus:v1.5.2)...
v1.5.2: Pulling from prom/prometheus
557a0c95bfcd: Pull complete
a3ed95caeb02: Pull complete
caf4d0cf9832: Pull complete
ee054001e2db: Pull complete
b95bf6c4c81b: Pull complete
86503a6ba368: Pull complete
ff27c7b0b50e: Pull complete
534e30a17a42: Pull complete
475d41733562: Pull complete
Digest: sha256:e049c086e35c0426389cd2450ef193f6c18b3d0065b97e5f203fdb254716fa1c
Status: Downloaded newer image for prom/prometheus:v1.5.2
Pulling influxdb (influxdb:1.1.1-alpine)...
1.1.1-alpine: Pulling from library/influxdb
```

```

0a8490d0dfd3: Pull complete
5f0fd352f87d: Pull complete
873718bcf8aa: Pull complete
3fbaf3e4140e: Pull complete
Digest: sha256:e0184202151b2abb9ceee79e6523d9492fc3c632324eb6f7bf1a672dd130a3bb
Status: Downloaded newer image for influxdb:1.1.1-alpine
Pulling grafana (grafana/grafana:4.1.2)...
4.1.2: Pulling from grafana/grafana
43c265008fae: Pull complete
c2ab838d4052: Pull complete
e8a816c8f505: Pull complete
Digest: sha256:05d925bd64cd3f9d6f56a4353774ccec588586579ab738f933cd002b7f96aca3
Status: Downloaded newer image for grafana/grafana:4.1.2
Creating aosomstreaming_telegraf-influx_1
Creating aosomstreaming_prometheus_1
Creating aosomstreaming_telegraf-prom_1
Creating aosomstreaming_influxdb_1
Creating aosomstreaming_grafana_1

```

## Reconfigure Aosom-streaming after Apstra Server Upgrade

After you upgrade the Apstra server you must reconfigure to ensure a proper streaming connection.

1. If you upgraded the Apstra server onto a different VM (or if the server IP address is different for any reason), update the `variables.env` file with the new Apstra IP address.
2. Verify that the current **Telegraf** container image matches the proper version for the new Apstra release by running the `docker ps` command.

```

admin@aeon-ztps:~$ docker ps
CONTAINER ID IMAGE
4edf204e7be9 apstra/telegraf:latest

```

You can check the different Telegraf versions in the [Apstra Docker Hub](#).

3. If required, modify the `docker-compose.yml` file and point to the correct Docker image.
4. Restart the service by running the command `docker-compose up -d`.
5. Verify that the container is running with the new image by running the `docker ps` command.

**NOTE:** For assistance regarding which version to install or if you have any questions about the procedure, contact "[Juniper Support](#)" on page 777.

## Build Aosom-Streaming VM (Optional)

IN THIS SECTION

- [Install Ubuntu 16.04.2 | 837](#)
- [Install Packages | 837](#)
- [Set Container Restart Policy | 839](#)
- [Change System Hostname | 840](#)

You can build your own Aosom-streaming VM, which is a Docker container. This steps show you how to set up a basic Docker server.

### Install Ubuntu 16.04.2

Download the Ubuntu 16.04.2 ISO and provision a new VM. The default username is **aosom** and the password is **admin**.

For larger blueprints, we recommend changing RAM to at least 8GB and CPU to at least 2 vCPU. More disk space may also be required.

Resource	Quantity
RAM	8GB
CPU	2 vCPU
Network	1 vNIC

### Install Packages

Install required packages, based on Ubuntu 16.04.2.

```
apt-get update
```

Update the system to ensure all packages are up to date.

```
apt-get install docker docker-compose git make curl openssh-server
```

```
aosom@ubuntu:~$ sudo apt-get install docker docker-compose git make curl openssh-server
```

```
[sudo] password for aosom:
```

```
Reading package lists... Done
```

```
Building dependency tree
```

```
Reading state information... Done
```

```
The following additional packages will be installed:
```

```
bridge-utils cgroupfs-mount containerd dns-root-data dnsmasq-base docker.io
git-man liberror-perl libnetfilter-conntrack3 libperl5.22 libpython-stdlib
libpython2.7-minimal libpython2.7-stdlib libyaml-0-2 patch perl
perl-modules-5.22 python python-backports.ssl-match-hostname
python-cached-property python-cffi-backend python-chardet
python-cryptography python-docker python-dockerpty python-docopt
python-enum34 python-funcsigs python-functools32 python-idna
python-ipaddress python-jsonschema python-minimal python-mock
python-ndg-httpsclient python-openssl python-pbr python-pkg-resources
python-pyasn1 python-requests python-six python-texttable python-urllib3
python-websocket python-yaml python2.7 python2.7-minimal rename runc
ubuntu-fan xz-utils
```

```
Suggested packages:
```

```
mountall aufs-tools btrfs-tools debootstrap docker-doc rinse zfs-fuse
| zfsutils git-daemon-run | git-daemon-sysvinit git-doc git-el git-email
git-gui gitk gitweb git-arch git-cvs git-mediawiki git-svn diffutils-doc
perl-doc libterm-readline-gnu-perl | libterm-readline-perl-perl make
python-doc python-tk python-cryptography-doc python-cryptography-vectors
python-enum34-doc python-funcsigs-doc python-mock-doc python-openssl-doc
python-openssl-dbg python-setuptools doc-base python-ntlm python2.7-doc
binutils binfmt-support make
```

```
The following NEW packages will be installed:
```

```
bridge-utils cgroupfs-mount containerd dns-root-data dnsmasq-base docker
docker-compose docker.io git git-man liberror-perl libnetfilter-conntrack3
libperl5.22 libpython-stdlib libpython2.7-minimal libpython2.7-stdlib
libyaml-0-2 patch perl perl-modules-5.22 python
python-backports.ssl-match-hostname python-cached-property
python-cffi-backend python-chardet python-cryptography python-docker
python-dockerpty python-docopt python-enum34 python-funcsigs
python-functools32 python-idna python-ipaddress python-jsonschema
python-minimal python-mock python-ndg-httpsclient python-openssl python-pbr
```

```
python-pkg-resources python-pyasnl python-requests python-six
python-texttable python-urllib3 python-websocket python-yaml python2.7
python2.7-minimal rename runc ubuntu-fan xz-utils make
0 upgraded, 54 newly installed, 0 to remove and 3 not upgraded.
Need to get 32.4 MB of archives.
After this operation, 174 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

Add the aosom user to the Docker group. This allows 'aosom' to make Docker configuration changes without having to escalate to sudo.

```
aosom@ubuntu:~/aosom-streaming$ sudo usermod -aG docker aosom
Log out and log back in again for 'aosom' user to be properly added to the group.
```

Copy the Aosom-streaming Docker containers over with 'git clone'.

```
aosom@ubuntu:~$ git clone https://github.com/Apstra/aosom-streaming.git
Cloning into 'aosom-streaming'...
remote: Counting objects: 303, done.
remote: Total 303 (delta 0), reused 0 (delta 0), pack-reused 303
Receiving objects: 100% (303/303), 64.10 KiB | 0 bytes/s, done.
Resolving deltas: 100% (176/176), done.
Checking connectivity... done.
aosom@ubuntu:~$
```

### Set Container Restart Policy

The AOSOM-Streaming package does not set the Docker restart policy; this is up to your orchestration toolchain. Open aosom-streaming/docker-compose.yml and add restart: always to each of the service directives. This ensures that Docker containers are online after a service reboot.

```
git diff docker-compose.yml
```

```
aosom@ubuntu:~/aosom-streaming$ git diff docker-compose.yml
diff --git a/docker-compose.yml b/docker-compose.yml
index 799d4c5..0d0fcc2 100644
--- a/docker-compose.yml
+++ b/docker-compose.yml
```

```

@@ -16,6 +16,7 @@ services:
    - prometheus
    ports:
      - "3000:3000"
+   restart: always

# -----
# Prometheus -
@@ -30,6 +31,7 @@ services:
    - '-config.file=/etc/prometheus/prometheus.yml'
    ports:
      - '9090:9090'
+   restart: always

# -----
# influxdb
@@ -43,6 +45,7 @@ services:
    ports:
      - "8083:8083"
      - "8086:8086"
+   restart: always

# -----
# Telegraf - Prom
@@ -57,6 +60,7 @@ services:
    - /etc/localtime:/etc/localtime
    ports:
      - '6666:6666'
+   restart: always

# -----
# Telegraf - Influx
@@ -71,3 +75,4 @@ services:
    - /etc/localtime:/etc/localtime
    ports:
      - '4444:4444'
+   restart: always

```

Set up `variables.env` and start container per Aosom-Streaming application setup section.

### Change System Hostname

Modify `/etc/hostname` to `aosom`, and change the loopback IP in `/etc/hosts` to `aosom` from `ubuntu`.

## Troubleshooting

IN THIS SECTION

- Check for Logs from Apstra to Aosom-streaming | 841
- Ensure Containers are Running | 841

While most troubleshooting information is included in the Github main page at <https://github.com/Apstra/aosom-streaming>, you can run some simple commands to make sure the environment is healthy.

### Check for Logs from Apstra to Aosom-streaming

Run Docker logs aosomstreaming\_telegraf-influx\_1

You should see a blueprint ID, and some influxdb 'write' events when telemetry events occur on AOS - BGP, liveness, config deviation, etc.

```
GetBlueprints() - Id 0033cf3f-41ed-4ddc-91f5-ea68318fba9b
2017-07-31T23:59:13Z D! Finished to Refresh Data, will sleep for 20 sec
2017-07-31T23:59:15Z D! Output [influxdb] buffer fullness: 11 / 10000 metrics.
2017-07-31T23:59:15Z D! Output [influxdb] wrote batch of 11 metrics in 5.612057ms
2017-07-31T23:59:20Z D! Output [influxdb] buffer fullness: 4 / 10000 metrics.
2017-07-31T23:59:20Z D! Output [influxdb] wrote batch of 4 metrics in 5.349171ms
2017-07-31T23:59:25Z D! Output [influxdb] buffer fullness: 11 / 10000 metrics.
2017-07-31T23:59:25Z D! Output [influxdb] wrote batch of 11 metrics in 4.68295ms
2017-07-31T23:59:30Z D! Output [influxdb] buffer fullness: 4 / 10000 metrics.
2017-07-31T23:59:30Z D! Output [influxdb] wrote batch of 4 metrics in 5.007029ms
GetBlueprints() - Id 0033cf3f-41ed-4ddc-91f5-ea68318fba9b
2017-07-31T23:59:33Z D! Finished to Refresh Data, will sleep for 20 sec
```

### Ensure Containers are Running

To see and ensure that all the expected containers are running, run docker ps:

```
aosom@ubuntu:~/aosom-streaming$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED
STATUS	PORTS		NAMES
e03d003a2ef9	grafana/grafana:4.1.2	"/run.sh"	3 minutes ago
			Up 3

minutes	0.0.0.0:3000->3000/tcp	aosomstreaming_grafana_1
3042d45f1107	prom/prometheus:v1.5.2	"/bin/prometheus -con" 3 minutes ago Up 3
minutes	0.0.0.0:9090->9090/tcp	aosomstreaming_prometheus_1
429328fbb5ac	apstra/telegraf:1.2	"telegraf -debug" 3 minutes ago Up 3
minutes	0.0.0.0:6666->6666/tcp	aosomstreaming_telegraf-prom_1
0a84241e1366	apstra/telegraf:1.2	"telegraf -debug" 3 minutes ago Up 3
minutes	0.0.0.0:4444->4444/tcp	aosomstreaming_telegraf-influx_1
f4d2deb0e428	influxdb:1.1.1-alpine	"/entrypoint.sh influ" 3 minutes ago Up 3
minutes	0.0.0.0:8083->8083/tcp, 0.0.0.0:8086->8086/tcp	aosomstreaming_influxdb_1

## Mixed Uplink Speeds between Leafs and Spines

The leafs in your racks can have different uplink speeds to a spine. When designing for mixed speeds, make sure you plan sufficient ports for spine-to-leaf connections with mixed link speeds for Day 0, and for adding racks as a Day 2 operation. The spine logical device must have mixed port speeds defined that specify the port role as **Leaf** for the required number of ports. The following limitations apply:

- Parallel links between the same devices cannot have mixed speeds.
- You can't update logical devices of spines that are already being used by a blueprint. You could possibly use the experimental aos-cli utility for manual patching. Although it may not be able to provide a solution. For assistance, contact ["Juniper Support" on page 777](#).

The example below shows how to design rack types and templates with mixed speeds.

1. Create an **L3 Clos** rack type with logical devices **AOS-7x10-Leaf** and **AOS-40x10+6x40-1** for two leaf switches, having 10 GbE and 40GbE, respectively, as uplinks towards spines

Create Rack Type

Leaf

Name \*

10gig

Leaf Logical Device \*

AOS-7x10-Leaf

Links per spine (2 available) \*

1

Link speed \*

10 Gbps

Redundancy Protocol

☒ None ☐ MLAG ☐ ESI

Tags

Select...

10gig

1 x 10 Gbps Links per spine

2 x 10 Gbps Mesh Links

AOS-7x10-Leaf  
AOS-7x10-Leaf

40gig

1 x 40 Gbps Links per spine

AOS-40x10+6x40-1  
AOS-40x10+6x40-1

Leaf

Name \*

40gig

Leaf Logical Device \*

AOS-40x10+6x40-1

Logical Device Preview

Name

AOS-7x10-Leaf

PANEL #1

TOTAL

7 ports

PORT GROUPS

2 x 10 Gbps Spine • Leaf

2 x 10 Gbps Peer

2 x 10 Gbps Access • Generic

1 x 10 Gbps Generic

Connected to ▾

1 2 3 4 5 6 7

Logical Device Preview

Name

AOS-40x10+6x40-1

PANEL #1

TOTAL

40 ports

PORT GROUPS

40 x 10 Gbps Access • Peer • Generic

Connected to ▾

1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39

2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40

PANEL #2



4. As a Day 0 operation you can create a ["blueprint" on page 400](#) with one of the above templates; or as a Day 2 operation you can select a mixed speed rack type when ["adding a rack" on page 479](#) to an existing blueprint.

## Enable VXLAN Routing on Cumulus Tomahawk

### IN THIS SECTION

- [Overview | 845](#)
- [Create Device Profile for Hyperloop | 845](#)
- [Update Logical Device for Hyperloop | 847](#)
- [Create Interface Map for Hyperloop | 849](#)
- [Update Managed Devices for Hyperloop | 850](#)
- [Assign Hyperloop Device Profile | 851](#)
- [Verify Loopback Port | 851](#)

### Overview

Cumulus devices equipped with Tomahawk and Tomahawk+ ASICs do not natively support Routing In and Out of Tunnels (RIOT). Therefore, switch ports for VXLAN routing must be configured to use hyperloop interfaces. Hyperloop interfaces recirculate packets through the ingress pipeline. As packets enter the VXLAN tunnel they are encapsulated, as they exit the VXLAN tunnel they are decapsulated.

To set up the hyperloop interface, you'll create a device profile, make sure your logical device has an unused port (as applicable), create an interface map, update your managed devices, assign the new device profile to the device, and deploy the device.

### Create Device Profile for Hyperloop

1. You can create a ["device profile" on page 300](#) from scratch, but it's more efficient to clone an existing one and change details that are different. Choose a device profile to clone from that is based on the

device to be used for the hyperloop interface, and add 'hyperloop' to the name to distinguish it.

Devices / Device Profiles

Create Device Profile

Search criteria: Manufacturer = Accton or Edgecore

1-5 of 5 Page Size: 25

### Clone Device Profile

**Summary**

Label \*  
Edgecore AS7712-32X-hyperloop

Number of slots \*  
0

Start from ID  
0

Create

2. In the **Ports** section of the device profile that you are cloning, click the port that you want as the hyperloop port.

Devices / Device Profiles

### Clone Device Profile

**Summary**

**Selector**

**Capabilities**

**Ports**

**Panel #1** Select port(s) to fill in the details

Port breakout Autonegotiation

1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31
2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32

Edit selected port #32

Connector type \*  
qsfp28

**Transformations**

Default	Speed	Interfaces
<input checked="" type="radio"/>	100 Gbps	swp32
<input type="radio"/>	50 Gbps	swp32s0 swp32s1
<input type="radio"/>	40 Gbps	swp32
<input type="radio"/>	25 Gbps	swp32s0 swp32s1 swp32s2 swp32s3
<input type="radio"/>	10 Gbps	swp32s0 swp32s1 swp32s2 swp32s3

Add new transformation



**CAUTION:** The remaining ports must be set to their native / default port speeds. If they are not, RIOT (and therefore VXLAN routing) won't work. For example, if you're using a 4 x 25G SFP28 breakout and you configure ports 1 and 2 for hyperloop, then ports 3 and 4 must be set to the default of 25G. If you set them to 10G, you will have connectivity, but hyperloop will not work.

3. In the **Transformation** section, click the **Edit** button for that port.
4. In the **Settings** field, change the value for "speed" to "loopback", then click **Create**.

In the example below, for "full speed" transformation (100Gbps on QSFP28 port), the value for "speed" was changed from the default speed of "100G" to "loopback".

The screenshot shows the 'Clone Device Profile' interface. On the left is a sidebar with tabs: Summary, Selector, Capabilities, and Ports (selected). The main area is titled 'Panel #1 Select port(s) to fill in the details'. It shows a grid of ports from 1 to 32, with port 32 selected. Below the grid, it says 'Edit selected port #32'. The 'Connector type' is set to 'qsfp28'. Under the 'Transformations' section, there is a table with columns: Default, Speed, Active, Name template, Setting, and buttons. The first row is selected, showing '100 Gbps' speed, 'swp32' name template, and a setting field containing '["command": {"speed": "loopback"}]'. The other rows show different speeds (50 Gbps, 40 Gbps, 25 Gbps, 10 Gbps) with their respective interface names and settings.

Default	Speed	Active	Name template	Setting		
<input checked="" type="radio"/>	100 Gbps	<input checked="" type="checkbox"/>	swp32	["command": {"speed": "loopback"}]		
<input type="radio"/>	50 Gbps	<input type="checkbox"/>	swp32s0 x swp32s1 x			
<input type="radio"/>	40 Gbps	<input type="checkbox"/>	swp32			
<input type="radio"/>	25 Gbps	<input type="checkbox"/>	swp32s0 x swp32s1 x swp32s2 x swp32s3 x			
<input type="radio"/>	10 Gbps	<input type="checkbox"/>	swp32s0 x swp32s1 x swp32s2 x swp32s3 x			

## Update Logical Device for Hyperloop

If you have a "logical device" on page 84 that includes the hyperloop port, make sure the port is configured for an **Unused** role.

AOS-16x100-1

16 × 100 Gbps

1

16

Leaf • External Router

Clone Logical Device

Name

AOS-32x100-hyperloop

PANEL #1

TOTAL

2 × 16 ports

32 assigned • 0 available

PORT GROUPS

2 × 100 Gbps

Spine

26 × 100 Gbps

L2 Server • L3 Server

3 × 100 Gbps

External Router

1 × 100 Gbps

Unused

All roles

1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31
2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32

+

 Add Panel

Create

AOS-48x10+6x40-1

48 × 10 Gbps

2

54

6 × 100 Gbps

Spine • External Router

6 × 100 Gbps

Spine • External Router

Blueprints

Devices

Design

Resources

Platform

admin

## Create Interface Map

Label

AOS-32x100\_AOS-32x100-hyperloop

Logical device \*

AOS-32x100-hyperloop

Device profile \*

Edgecore AS7712-32X-hyperloop

### Map interfaces

Logical device port groups		Mapped/required number of interfaces	Device profile interfaces
Speed	Roles		
100 Gbps	Spine	2 / 2	▶ Select interfaces
100 Gbps	L2 Server • L3 Server	26 / 26	▶ Select interfaces
100 Gbps	External Router	3 / 3	▶ Select interfaces
100 Gbps	Unused	1 / 1	▼ Select interfaces

1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31
2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32

Transformation #1 (default)


swp32

Interface map preview

Click on interface to toggle the details

# Update Managed Devices for Hyperloop

- 1. Navigate to the managed device that will use hyperloop.




Blueprints

Devices ▾



Design ▾

Resources ▾

Platform ▾

 admin ▾

[/ Devices / Managed Devices / 771232X1633033](#)



Info

Telemetry

✓

Expanded View

Compact View

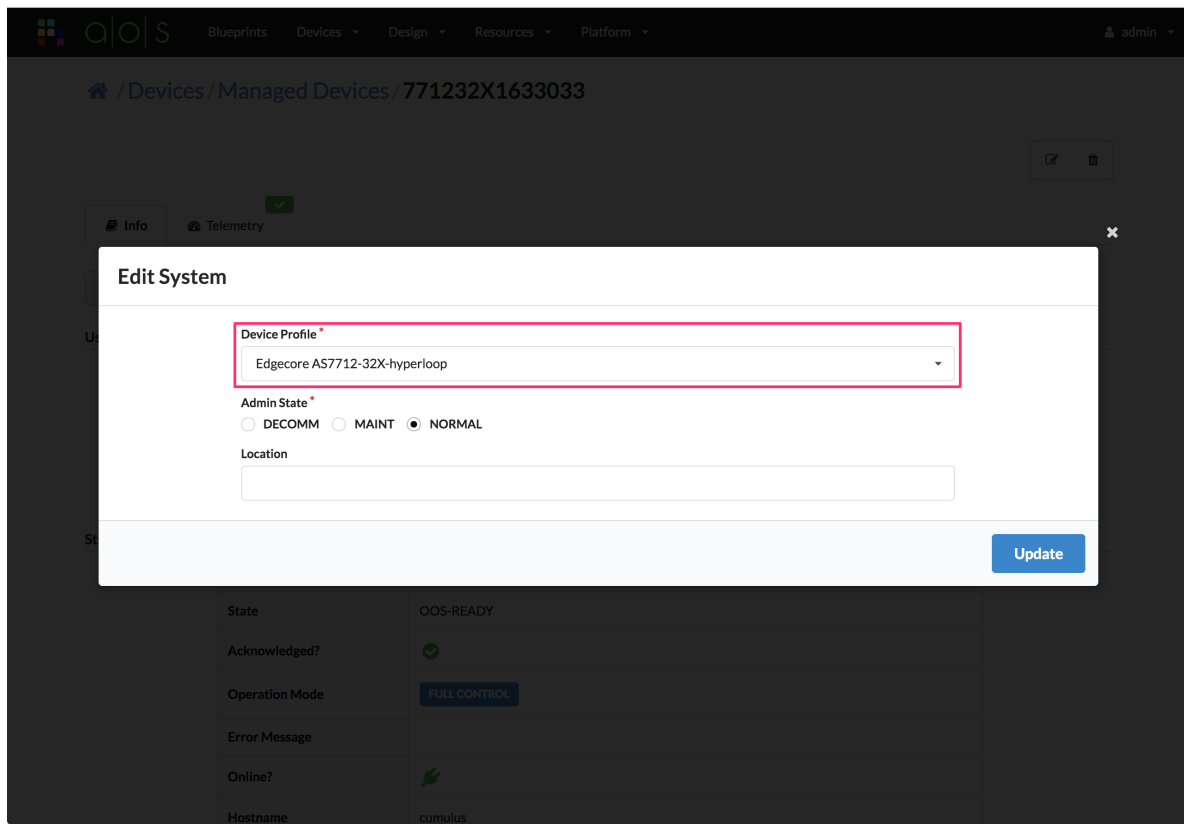
User Config

Device Profile	Edgecore AS7712-32X
Admin State	normal
Location	

Status

State	OOS-READY
Acknowledged?	✓
Operation Mode	FULL CONTROL
Error Message	
Online?	👤
Hostname	cumulus

2. Click the **Edit** button (top-right) and select the new hyperloop device profile from the drop-down list.



ODS Blueprints Devices Design Resources Platform admin

/Devices/Managed Devices/771232X1633033

Info Telemetry

**Edit System**

Device Profile \*

Edgecore AS7712-32X-hyperloop

Admin State \*

☐ DECOMM ☐ MAINT ☒ NORMAL

Location

Update

State	OOS-READY
Acknowledged?	<input checked="" type="checkbox"/>
Operation Mode	FULL CONTROL
Error Message	
Online?	<input checked="" type="checkbox"/>
Hostname	cumulus

3. Click **Update** to save your changes.

## Assign Hyperloop Device Profile

In the blueprint, ["assign the new device profile" on page 422](#) to the appropriate device, then deploy it.

## Verify Loopback Port

After deploying the device, verify that the hyperloop port is set to "loopback" by looking at the device file `/etc/cumulus/ports.conf`.

```
admin@border-1-leaf:mgmt-vrf:~$ sudo cat /etc/cumulus/ports.conf
sudo: unable to resolve host border-1-leaf
1=100G
2=100G
3=100G
4=100G
5=100G
6=100G
7=100G
```

```
8=100G
9=100G
10=100G
11=100G
12=100G
13=100G
14=100G
15=100G
16=100G
17=100G
18=100G
19=100G
20=100G
21=100G
22=100G
23=100G
24=100G
25=100G
26=100G
27=100G
28=100G
29=100G
30=100G
31=100G
32=loopback
admin@border-1-leaf:mgmt-vrf:~$
```

## References

### IN THIS SECTION

- [Apstra Feature Matrix | 853](#)
- [Qualified Device and NOS \(Devices\) | 894](#)
- [Agent Configuration File \(Devices\) | 901](#)
- [NOS Upgrade Paths \(Devices\) | 906](#)
- [Apstra EVPN Support Addendum | 909](#)

- [Predefined Dashboards \(Analytics\) | 919](#)
- [Predefined Probes \(Analytics\) | 921](#)
- [Probe Processors \(Analytics\) | 993](#)
- [Configlet Examples \(Design\) | 1057](#)
- [Graph | 1066](#)
- [Juniper Apstra Technology Previews \(Tech Previews\) | 1082](#)

## Apstra Feature Matrix

### IN THIS SECTION

- [Apstra 4.0.2 Feature Matrix | 853](#)
- [Apstra 4.0.1 Feature Matrix | 866](#)
- [Apstra 4.0.0 Feature Matrix | 880](#)

## Apstra 4.0.2 Feature Matrix

### IN THIS SECTION

- [Fabric Roles | 854](#)
- [Fabric Connectivity | 854](#)
- [Device Management | 855](#)
- [Connectivity \(from Leaf Layer\) | 856](#)
- [Connectivity \(from Access Layer\) | 856](#)
- [Routing Policies | 857](#)
- [Miscellaneous | 857](#)
- [Virtual Network CT Type | 858](#)
- [IP Link CT Type | 858](#)
- [Static Route CT Type | 859](#)

- [Custom Static Route CT Type | 859](#)
- [BGP to Generic CT Type | 860](#)
- [BGP to IP Endpoint CT Type | 863](#)
- [Dynamic BGP Peering CT Type | 864](#)
- [Routing Policy CT Type | 865](#)
- [BGP Attributes \(common to all BGP CTs\) | 865](#)

Fabric Roles

Table 33: Fabric Roles

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
Access Switch	No	No	No	No	Yes	No
Non-EVPN-VXLAN Leaf (IP forwarder only)	Yes	Yes	Yes	Yes	Yes	Yes
EVPN-VXLAN Leaf	Yes	Yes	Yes	Yes	Yes	No
Spine or Superspine	Yes	Yes	Yes	Yes	Yes	Yes

Fabric Connectivity

Table 34: Fabric Connectivity

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
3-stage Clos	Yes	Yes	Yes	Yes	Yes	Yes
5-stage Clos	Yes	Yes	Yes	Yes	Yes	Yes
IP fabric (non-EVPN-VXLAN)	Yes	Yes	Yes	Yes	Yes	Yes
EVPN-VXLAN fabric	Yes	Yes	Yes	Yes	Yes	Limited
IPv6 fabric RFC-5549 (non-EVPN)	Yes	Yes	Yes	Yes	No	No

Table 34: Fabric Connectivity (*Continued*)

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
3-stage Clos with access switch layer	No	No	No	No	Yes	Limited
Collapsed fabric	No	No	No	No	Yes	No

## Device Management

Table 35: Device Management

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
On-box agent	Yes	Yes	Yes	Yes	Not possible	No
Off-box agent	Yes	Yes	Contact Juniper Support	No	Yes	Yes
Telemetry extensibility	Yes	Yes	Yes	Yes	Yes	No
Apstra ZTP	Yes	Yes	Yes	Yes	Yes	Yes
OS upgrade	Yes	Yes	Yes	Contact Juniper Support	Yes	Yes
Traffic draining (spines/ superspines - maintenance mode)	Yes	Yes	Yes	Yes	Yes	Yes
Traffic draining (leafs)	Limited	Limited	Limited	Limited	Limited	Limited

## Connectivity (from Leaf Layer)

Table 36: Connectivity (from Leaf Layer)

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
LAG	Yes	Yes	Yes	Yes	Yes	Yes
MLAG/vPC	Yes	Yes	Yes	Yes	Not possible	Not possible
EVPN ESI (with LACP)	No	No	No	Not possible	Yes	No
802.1x	Yes	No	No	Not possible	No	No
VLANs	Yes	Yes	Yes	Yes	Yes	Yes
Overlay protocol: static VXLAN	Yes	Yes	Not possible	Not possible	No	No
Overlay protocol: EVPN (3-stage and 5-stage)	Yes	Yes	Yes	Yes	Yes	No
IPv4 DHCP relay	Yes	Yes	Yes	Yes	Yes	Yes
IPv6 DHCP relay	Yes	Yes	Yes	Yes	Yes	Yes
EVPN DCI	Yes	Yes	Yes	Yes	Yes	No
IPv6 for applications (with EVPN and IPv4 fabric)	Yes	Yes	Yes	Yes	Yes	No
Group-based policy (ACL on ToRs)	Yes	Yes	No	No	Yes	Yes

## Connectivity (from Access Layer)

Table 37: Connectivity (from Access Layer)

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
LAG	N/A	N/A	N/A	N/A	Yes	N/A

Table 37: Connectivity (from Access Layer) *(Continued)*

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
MLAG/vPC	N/A	N/A	N/A	N/A	No	N/A

## Routing Policies

Table 38: Routing Policies

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
Import all routes or default route or extra routes only	Yes	Yes	Yes	Yes	Yes	Yes
Export loopback, link and VN IP. Export extra routes	Yes	Yes	Yes	Yes	Yes	Yes
Export aggregate prefixes	Yes	Yes	Yes	Yes	Yes	No
Export L3 server link subnets	Yes	Yes	Yes	Yes	Yes	Yes
Route target import/export policies	Yes	Yes	Yes	Yes	Yes	No

## Miscellaneous

Table 39: Miscellaneous

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
Configlets	Yes	Yes	Yes	Yes	Yes	Yes
FFE: add racks/add links/change speed, etc.	Yes	Yes	Yes	Yes	Yes	Yes
Mixed leaf/spine link speed	Yes	Yes	Yes	Yes	Yes	Yes

## Virtual Network CT Type

Table 40: Virtual Network CT Type

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
Single Virtual Network	Yes	Yes	Yes	Yes	Yes	No
Multiple Virtual Network	Yes	Yes	Yes	Yes	Yes	No
VLAN (default VRF, non-VXLAN)	Yes	Yes	Yes	Yes	Yes	Yes

## IP Link CT Type

Table 41: IP Link CT Type

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
L3 Sub-interface on non-LAG physical interface (untagged/vlan tagged, default/non-default RZ, IPv4)	Yes	Yes	Yes	Yes	Yes	Yes - for default VRF only
L3 Sub-interface on non-LAG physical interface (untagged/vlan tagged, default/non-default RZ, IPv6)	Yes	Yes	Yes	Yes	Yes	No
L3 Sub-interface on LAG interface (untagged/vlan tagged, default/non-default RZ, IPv4)	Yes	Yes	Yes	Yes	Yes	No
L3 Sub-interface on LAG interface (untagged/vlan tagged, default/non-default RZ, IPv6)	Yes	Yes	Yes	Yes	Yes	No

## Static Route CT Type

**Table 42: Static Route CT Type**

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
Static Route (IPv4) applied on L3 Sub-interface	Yes	Yes	Yes	Yes	Yes	Yes - for default VRF only
Static Route (IPv6) applied on L3 Sub-interface	Yes	Yes	Yes	Yes	Yes	No
Static Route (IPv4) applied on SVI	Yes	Yes	Yes	Yes	Yes	No
Static Route (IPv6) applied on SVI	Yes	Yes	Yes	Yes	Yes	No
Static Route with Share IP Endpoint Enabled (IPv4)	Yes	Yes	Yes	Yes	Yes	No
Static Route with Share IP Endpoint Enabled (IPv6)	Yes	Yes	Yes	Yes	Yes	No

## Custom Static Route CT Type

**Table 43: Custom Static Route CT Type**

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
Custom Static Route (IPv4, default/non-default RZ)	Yes	Yes	Yes	Yes	Yes	Yes - for default VRF only
Custom Static Route (IPv6, default/non-default RZ)	Yes	Yes	Yes	Yes	Yes	No

## BGP to Generic CT Type

Table 44: BGP to Generic CT Type

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
BGP session on L3 Sub-interface towards generic (IPv4, default/non-default RZ)	Yes	Yes	Yes	Yes	Yes	Yes
BGP session on L3 Sub-interface towards generic (IPv6, default/non-default RZ)	Yes	Yes	Yes	Yes	Yes	No
BGP session on SVI towards generic (IPv4, default RZ)	Yes	Yes	Yes	Yes	Yes	No
BGP session on SVI towards generic (IPv4, non-default RZ)	Yes	Yes	Yes	Yes	Yes	No
BGP session on SVI towards generic (IPv6, non-default RZ)	Yes	Yes	Yes	Yes	Yes	No
BGP session on SVI towards generic (IPv6, default RZ)	Yes	Yes	Yes	Yes	Yes	No
BGP session on SVI (mlag) towards dual-homed generic using secondary IPs (IPv4, default VRF)	Yes	Yes	Yes	Not possible	Not possible	No
BGP session on SVI (mlag) towards dual-homed generic using secondary IPs (IPv4, non-default VRF)	Yes	Yes - with non-VXLAN VLAN in non-default VRF	Yes	Not possible	Yes	No
BGP session on SVI (mlag) towards dual-homed generic using secondary IPs (IPv6, default VRF)	Yes	Yes	Yes	Not possible	Not possible	No

Table 44: BGP to Generic CT Type *(Continued)*

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
BGP session on SVI (mld) towards dual-homed generic using secondary IPs (IPv6, non-default VRF)	Yes	Yes - with non-VXLAN VLAN in non-default VRF	Yes	Not possible	Yes	No
BGP session to generic with Share IP Endpoint Enabled (IPv4)	Yes	Yes - with non-VXLAN VLAN in non-default VRF	Yes	Yes	Yes	No
BGP session to generic with Share IP Endpoint Enabled (IPv6)	Yes	Yes - with non-VXLAN VLAN in non-default VRF	Yes	Yes	Yes	No
BGP session to generic with dynamic ASN (IPv4)	No	No	No	No	No	No
BGP session to generic with Static ASN (IPv4)	Yes	Yes	Yes	Yes	Yes	No
BGP session to generic with dynamic ASN (IPv6)	No	No	No	No	No	No
BGP session to generic with static ASN (IPv6)	Yes	Yes	Yes	Yes	Yes	No
BGP Unnumbered session (link-local peering) on L3 Sub-interface (BP has IPv6 app enabled, default VRF)	Yes	Yes	Yes	Yes	No	No
BGP Unnumbered session (link-local peering) on L3 Sub-interface (BP has IPv6 app enabled, non-default VRF)	No	Yes	Yes	Yes	No	No

Table 44: BGP to Generic CT Type *(Continued)*

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
BGP Unnumbered session (link-local peering) on SVI (BP has IPv6 app enabled, default VRF)	Yes	Yes	Yes	Yes	No	No
BGP Unnumbered session (link-local peering) on SVI (BP has IPv6 app enabled, non-default VRF)	No	Yes	Yes	Yes	No	No
BGP Unnumbered session (link-local peering) on L3 Sub-interface (default VRF, BP has IPv6 app disabled)	Yes	Yes	Yes	Yes	No	No
BGP Unnumbered session (link-local peering) on L3 Sub-interface (non-default VRF, BP has IPv6 app disabled)	No	Yes	Yes	Yes	No	No
BGP Unnumbered session (link-local peering) on SVI (BP has IPv6 app disabled, default VRF only)	Yes	Yes	Yes	Yes	No	No
BGP Peering combinations (Int to Int, Lo to Int, Int to Lo, Lo to Lo)	Yes	Yes	Yes	Yes	Yes	Yes - in default VRF only
BGP session (IPv6 addressed) with IPv4 SAFI (rfc5549) with static ASN (BP has IPv6 app enabled)	No	No	No	No	No	No
BGP session (IPv6 addressed) with IPv4 SAFI (rfc5549) with dynamic ASN (BP has IPv6 app enabled)	No	No	No	No	No	No

## BGP to IP Endpoint CT Type

Table 45: BGP to IP Endpoint CT Type

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
BGP session from L3 sub-interface to any IP endpoint in the network (IPv4, default/non-default RZ)	Yes	Yes	Yes	Yes	Yes	Yes - in default VRF only
BGP session from L3 sub-interface to any IP endpoint in the network (IPv6, default/non-default RZ)	Yes	Yes	Yes	Yes	Yes	No
BGP session from SVI to any IP endpoint in the network (IPv4, default/non-default RZ)	Yes	Yes	Yes	Yes	Yes	No
BGP session from SVI to any IP endpoint in the network (IPv6, non-default RZ)	Yes	Yes	Yes	Yes	Yes	No
BGP session from SVI to any IP endpoint in the network (IPv6, default RZ)	Yes	Yes	Yes	Yes	Yes	No
BGP session from Loopback to any IP endpoint in the network (IPv4, default/non-default RZ)	Yes	Yes	Yes	Yes	Yes	Yes
BGP session from Loopback to any IP endpoint in the network (IPv6, default/non-default RZ)	Yes	Yes	Yes	Yes	Yes	No
BGP session with specific peer IP and and Static ASN (IPv4)	Yes	Yes	Yes	Yes	Yes	Yes
BGP session with specific peer IP and and Static ASN (IPv6)	Yes	Yes	Yes	Yes	Yes	No
BGP session with specific peer IP and and dynamic ASN (IPv4)	No	No	No	No	No	No
BGP session with specific peer IP and and dynamic ASN (IPv6)	No	No	No	No	No	No

Table 45: BGP to IP Endpoint CT Type *(Continued)*

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
BGP session (IPv6 addressed) with IPv4 SAFI (rfc5549) with static ASN (BP has IPv6 app enabled)	No	No	No	No	No	No
BGP session (IPv6 addressed) with IPv4 SAFI (rfc5549) with dynamic ASN (BP has IPv6 app enabled)	No	No	No	No	No	No

### Dynamic BGP Peering CT Type

Table 46: Dynamic BGP Peering CT Type

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
Dynamic BGP prefix peering on SVI (IPv4), default VRF	Yes	Yes	Yes	Yes	No	No
Dynamic BGP prefix peering on SVI (IPv4), non-default VRF	Yes	Yes	Yes	Yes	No	No
Dynamic BGP prefix peering on SVI (IPv6), default VRF	Yes	Yes	Yes	Yes	No	No
Dynamic BGP prefix peering on SVI (IPv6), non-default VRF	Yes	Yes	Yes	Yes	No	No
Dynamic BGP prefix peering on L3 sub-interface (IPv4), default VRF	Yes	Yes	Yes	Yes	No	No
Dynamic BGP prefix peering on L3 sub-interface (IPv4), non-default VRF	Yes	Yes	Yes	Yes	No	No
Dynamic BGP prefix peering on L3 sub-interface (IPv6), default VRF	Yes	Yes	Yes	Yes	No	No

**Table 46: Dynamic BGP Peering CT Type (Continued)**

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
Dynamic BGP prefix peering on L3 sub-interface (IPv6), non-default VRF	Yes	Yes	Yes	Yes	No	No
Dynamic prefix peering (link-local prefix peering, rfc5549), (BP has IPv6 app disabled)	Yes	No	No	No	No	No
Dynamic prefix peering (IPv6 peering, IPv4 AFI, rfc5549), (BP has IPv6 app enabled)	No	No	No	No	No	No

**Routing Policy CT Type****Table 47: Routing Policy CT Type**

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
Routing Policy on a BGP session with import/export IPv4 prefixes	Yes	Yes	Yes	Yes	Yes	Yes - for BGP sessions in default VRF
Routing Policy on a BGP session with import/export IPv6 prefixes	Yes	Yes	Yes	Yes	Yes	No
Routing Policy on a BGP session with IPv4 aggregate prefixes	Yes	Yes	Yes	Yes	Yes	No
Routing Policy on a BGP session with IPv6 aggregate prefixes	Yes	Yes	Yes	Yes	Yes	No

**BGP Attributes (common to all BGP CTs)****Table 48: BGP Attributes (common to all BGP CTs)**

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
BGP: enable Password/MD5 based authentication	Yes	Yes	Yes	Yes	Yes	No

Table 48: BGP Attributes (common to all BGP CTs) *(Continued)*

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
BGP: Custom BGP timers (Keep Alive timer, Hold timer)	Yes	Yes	Yes	Yes	Yes	Yes
BGP: Custom TTL	Yes	Yes	Yes	Yes	Yes	Yes
BGP: Enable Single-hop BFD	Yes	Yes	Yes	Yes	Yes	Yes

## Apstra 4.0.1 Feature Matrix

### IN THIS SECTION

- [Fabric Roles | 867](#)
- [Fabric Connectivity | 867](#)
- [Device Management | 868](#)
- [Connectivity \(from Leaf Layer\) | 868](#)
- [Connectivity \(from Access Layer\) | 869](#)
- [Routing Policies | 870](#)
- [Miscellaneous | 870](#)
- [Virtual Network CT Type | 870](#)
- [IP Link CT Type | 871](#)
- [Static Route CT Type | 871](#)
- [Custom Static Route CT Type | 872](#)
- [BGP to Generic CT Type | 872](#)
- [BGP to IP Endpoint CT Type | 876](#)
- [Dynamic BGP Peering CT Type | 878](#)
- [Routing Policy CT Type | 879](#)
- [BGP Attributes \(common to all BGP CTs\) | 880](#)

## Fabric Roles

**Table 49: Fabric Roles**

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
Access Switch	No	No	No	No	Yes	No
Leaf	Yes	Yes	Yes	Yes	Yes	No
Spine or Superspine	Yes	Yes	Yes	Yes	Yes	Yes

## Fabric Connectivity

**Table 50: Fabric Connectivity**

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
3-stage Clos	Yes	Yes	Yes	Yes	Yes	Yes
5-stage Clos	Yes	Yes	Yes	Yes	Yes	Yes
IP fabric (non-EVPN)	Yes	Yes	Yes	Yes	Yes	Yes
EVPN fabric	Yes	Yes	Yes	Yes	Yes	Yes
IPv6 fabric RFC-5549 (non-EVPN)	Yes	Yes	Yes	Yes	Not supported	Not supported
3-stage Clos with access switch layer	Not supported	Not supported	Not supported	Not supported	Limited	Not possible
Collapsed fabric	Not supported	Not supported	Not supported	Not supported	Limited	Not possible

## Device Management

**Table 51: Device Management**

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
On-box agent	Yes	Yes	Yes	Yes	Not possible	Not supported
Off-box agent	Yes	Yes	Contact support	Not supported	Yes	Yes
Telemetry extensibility	Yes	Yes	Yes	Yes	Yes	Not supported
Apstra ZTP	Yes	Yes	Yes	Yes	Yes	Yes
OS upgrade	Yes	Yes	Yes	Yes	Yes	Yes
Traffic draining (spines/ superspines - maintenance mode)	Yes	Yes	Yes	Yes	Yes	Yes
Traffic draining (leafs)	Limited	Limited	Limited	Limited	Limited	Limited

## Connectivity (from Leaf Layer)

**Table 52: Connectivity (from Leaf Layer)**

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
LAG	Yes	Yes	Yes	Yes	Yes	N/A
MLAG/vPC	Yes	Yes	Yes	Yes	Not possible	N/A
EVPN ESI (with LACP)	Not supported	Not supported	Not supported	Not possible	Yes	N/A
802.1x	Yes	Not supported	Not supported	Not possible	Not supported	N/A
VLANs	Yes	Yes	Yes	Yes	Yes	N/A

Table 52: Connectivity (from Leaf Layer) *(Continued)*

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
Overlay protocol: static VXLAN	Yes	Yes	Not possible	Not possible	Not supported	N/A
Overlay protocol: EVPN (3-stage and 5-stage)	Yes	Yes	Yes	Yes	Yes	N/A
IPv4 DHCP relay	Yes	Yes	Yes	Yes	Yes	N/A
IPv6 DHCP relay	Yes	Yes	Yes	Yes	Yes	N/A
EVPN DCI	Yes	Yes	Yes	Yes	Yes	N/A
IPv6 for applications (with EVPN and IPv4 fabric)	Yes	Yes	Yes	Yes	Yes	N/A
Security Policies (ACL on ToRs)	Yes	Yes	Not supported	Not supported	Limited	N/A

## Connectivity (from Access Layer)

Table 53: Connectivity (from Access Layer)

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
LAG	N/A	N/A	N/A	N/A	Yes	N/A
MLAG/vPC	N/A	N/A	N/A	N/A	Not possible	N/A

## Routing Policies

**Table 54: Routing Policies**

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
Import all routes or default route or extra routes only	Yes	Yes	Yes	Yes	Yes	Yes
Export loopback, link and VN IP. Export extra routes	Yes	Yes	Yes	Yes	Yes	Yes
Export aggregate prefixes	Yes	Yes	Yes	Yes	Yes	Yes
Export L3 server link subnets	Yes	Yes	Yes	Yes	Yes	Yes
Route target import/export policies	Yes	Yes	Yes	Yes	Yes	N/A

## Miscellaneous

**Table 55: Miscellaneous**

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
Configlets	Yes	Yes	Yes	Yes	Limited	Limited
FFE: add racks/add links/change speed, etc.	Yes	Yes	Yes	Yes	Yes	Yes
Mixed leaf/spine link speed	Yes	Yes	Yes	Yes	Yes	Yes

## Virtual Network CT Type

**Table 56: Virtual Network CT Type**

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
Single Virtual Network	Yes	Yes	Yes	Yes	Yes	Not supported
Multiple Virtual Network	Yes	Yes	Yes	Yes	Yes	Not supported

## IP Link CT Type

Table 57: IP Link CT Type

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
L3 Sub-interface on non-LAG physical interface (untagged/vlan tagged, default/non-default RZ, IPv4)	Yes	Yes	Yes	Not possible	Yes	Not supported
L3 Sub-interface on non-LAG physical interface (untagged/vlan tagged, default/non-default RZ, IPv6)	Yes	Yes	Yes	Not possible	Yes	Not supported
L3 Sub-interface on LAG interface (untagged/vlan tagged, default/non-default RZ, IPv4)	Yes	Yes	Yes	Not possible	Yes	Not supported
L3 Sub-interface on LAG interface (untagged/vlan tagged, default/non-default RZ, IPv6)	Yes	Yes	Yes	Not possible	Yes	Not supported

## Static Route CT Type

Table 58: Static Route CT Type

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
Static Route (IPv4) applied on L3 Sub-interface	Yes	Yes	Yes	Not possible	Yes	Not supported
Static Route (IPv6) applied on L3 Sub-interface	Yes	Yes	Yes	Not possible	Yes	Not supported
Static Route (IPv4) applied on SVI	Yes	Yes	Yes	Yes	Yes	Not supported
Static Route (IPv6) applied on SVI	Yes	Yes	Yes	Yes	Yes	Not supported
Static Route with Share IP Endpoint Enabled (IPv4)	Yes	Yes	Yes	Yes	Yes	Not supported

Table 58: Static Route CT Type *(Continued)*

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
Static Route with Share IP Endpoint Enabled (IPv6)	Yes	Yes	Yes	Yes	Yes	Not supported

### Custom Static Route CT Type

Table 59: Custom Static Route CT Type

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
Custom Static Route (IPv4, default/non-default RZ)	Yes	Yes	Yes	Yes	Yes	Not supported
Custom Static Route (IPv6, default/non-default RZ)	Yes	Yes	Yes	Yes	Yes	Not supported

### BGP to Generic CT Type

Table 60: BGP to Generic CT Type

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
BGP session on L3 Sub-interface towards generic (IPv4, default/non-default RZ)	Yes	Yes	Yes	Not possible	Yes	Not supported
BGP session on L3 Sub-interface towards generic (IPv6, default/non-default RZ)	Yes	Yes	Yes	Not possible	Yes	Not supported
BGP session on SVI towards generic (IPv4, default RZ)	Yes	Yes	Yes	Yes	Yes	Not supported

Table 60: BGP to Generic CT Type *(Continued)*

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
BGP session on SVI towards generic (IPv4, non-default RZ)	Yes	Yes	Yes	Yes	Yes	Not supported
BGP session on SVI towards generic (IPv6, non-default RZ)	Yes	Yes	Yes	Yes	Yes	Not supported
BGP session on SVI towards generic (IPv6, default RZ)	Yes	Yes	Yes	Yes	Yes	Not supported
BGP session on SVI (mlag) towards dual-homed generic using secondary IPs (IPv4, default VRF)	Yes	Yes	Yes	Not possible	Not possible	Not supported
BGP session on SVI (mlag) towards dual-homed generic using secondary IPs (IPv4, non-default VRF)	Yes	Not supported	Yes	Not possible	Yes	Not supported
BGP session on SVI (mlag) towards dual-homed generic using secondary IPs (IPv6, default VRF)	Yes	Yes	Yes	Not possible	Not possible	Not supported
BGP session on SVI (mlag) towards dual-homed generic using secondary IPs (IPv6, non-default VRF)	Yes	Not supported	Yes	Not possible	Yes	Not supported
BGP session to generic with Share IP Endpoint Enabled (IPv4)	Yes	Not supported	Yes	Yes	Yes	Not supported

Table 60: BGP to Generic CT Type *(Continued)*

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
BGP session to generic with Share IP Endpoint Enabled (IPv6)	Yes	Not supported	Yes	Yes	Yes	Not supported
BGP session to generic with dynamic ASN (IPv4)	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported
BGP session to generic with Static ASN (IPv4)	Yes	Yes	Yes	Yes	Yes	Not supported
BGP session to generic with dynamic ASN (IPv6)	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported
BGP session to generic with static ASN (IPv6)	Yes	Yes	Yes	Yes	Yes	Not supported
BGP Unnumbered session (link-local peering) on L3 Sub-interface (BP has IPv6 app enabled, default VRF)	Yes	Yes	Yes	Yes	Not supported	Not supported
BGP Unnumbered session (link-local peering) on L3 Sub-interface (BP has IPv6 app enabled, non-default VRF)	Not supported	Yes	Yes	Yes	Not supported	Not supported
BGP Unnumbered session (link-local peering) on SVI (BP has IPv6 app enabled, default VRF)	Yes	Yes	Yes	Yes	Not supported	Not supported

Table 60: BGP to Generic CT Type *(Continued)*

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
BGP Unnumbered session (link-local peering) on SVI (BP has IPv6 app enabled, non-default VRF)	Not supported	Yes	Yes	Yes	Not supported	Not supported
BGP Unnumbered session (link-local peering) on L3 Sub-interface (default VRF, BP has IPv6 app disabled)	Yes	Yes	Yes	Not possible	Not supported	Not supported
BGP Unnumbered session (link-local peering) on L3 Sub-interface (non-default VRF, BP has IPv6 app disabled)	Not supported	Yes	Yes	Not possible	Not supported	Not supported
BGP Unnumbered session (link-local peering) on SVI (BP has IPv6 app disabled, default VRF only)	Yes	Yes	Yes	Yes	Not supported	Not supported
BGP Peering combinations (Int to Int, Lo to Int, Int to Lo, Lo to Lo)	Yes	Yes	Yes	Yes	Yes	Not supported
BGP session (IPv6 addressed) with IPv4 SAFI (rfc5549) with static ASN (BP has IPv6 app enabled)	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported

Table 60: BGP to Generic CT Type *(Continued)*

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
BGP session (IPv6 addressed) with IPv4 SAFI (rfc5549) with dynamic ASN (BP has IPv6 app enabled)	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported

**BGP to IP Endpoint CT Type**

Table 61: BGP to IP Endpoint CT Type

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
BGP session from L3 sub-interface to any IP endpoint in the network (IPv4, default/non-default RZ)	Yes	Yes	Yes	Yes	Yes	Not supported
BGP session from L3 sub-interface to any IP endpoint in the network (IPv6, default/non-default RZ)	Yes	Yes	Yes	Yes	Yes	Not supported
BGP session from SVI to any IP endpoint in the network (IPv4, default/non-default RZ)	Yes	Yes	Yes	Yes	Yes	Not supported
BGP session from SVI to any IP endpoint in the network (IPv6, non-default RZ)	Yes	Yes	Yes	Yes	Yes	Not supported

**Table 61: BGP to IP Endpoint CT Type (Continued)**

Feature	EOS	NX-OS	Cumulus	SONIC	Junos OS	Junos OS Evolved
BGP session from SVI to any IP endpoint in the network (IPv6, default RZ)	Yes	Yes	Yes	Yes	Yes	Not supported
BGP session from Loopback to any IP endpoint in the network (IPv4, default/non-default RZ)	Yes	Yes	Yes	Yes	Yes	Not supported
BGP session from Loopback to any IP endpoint in the network (IPv6, default/non-default RZ)	Yes	Yes	Yes	Yes	Yes	Not supported
BGP session with specific peer IP and Static ASN (IPv4)	Yes	Yes	Yes	Yes	Yes	Not supported
BGP session with specific peer IP and Static ASN (IPv6)	Yes	Yes	Yes	Yes	Yes	Not supported
BGP session with specific peer IP and dynamic ASN (IPv4)	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported
BGP session with specific peer IP and dynamic ASN (IPv6)	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported
BGP session (IPv6 addressed) with IPv4 SAFI (rfc5549) with static ASN (BP has IPv6 app enabled)	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported

Table 61: BGP to IP Endpoint CT Type *(Continued)*

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
BGP session (IPv6 addressed) with IPv4 SAFI (rfc5549) with dynamic ASN (BP has IPv6 app enabled)	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported

### Dynamic BGP Peering CT Type

Table 62: Dynamic BGP Peering CT Type

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
Dynamic BGP prefix peering on SVI (IPv4), default VRF	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported
Dynamic BGP prefix peering on SVI (IPv4), non-default VRF	Yes	Yes	Yes	Yes	Not supported	Not supported
Dynamic BGP prefix peering on SVI (IPv6), default VRF	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported
Dynamic BGP prefix peering on SVI (IPv6), non-default VRF	Yes	Yes	Yes	Yes	Not supported	Not supported
Dynamic BGP prefix peering on L3 sub-interface (IPv4), default VRF	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported

Table 62: Dynamic BGP Peering CT Type *(Continued)*

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
Dynamic BGP prefix peering on L3 sub-interface (IPv4), non-default VRF	Yes	Yes	Yes	Not possible	Not supported	Not supported
Dynamic BGP prefix peering on L3 sub-interface (IPv6), default VRF	Not supported	Not supported	Not supported	Not possible	Not supported	Not supported
Dynamic BGP prefix peering on L3 sub-interface (IPv6), non-default VRF	Yes	Yes	Yes	Not possible	Not supported	Not supported
Dynamic prefix peering (link-local prefix peering, rfc5549), (BP has IPv6 app disabled)	Yes	Not supported	Not supported	Not supported	Not supported	Not supported
Dynamic prefix peering (IPv6 peering, IPv4 AFI, rfc5549), (BP has IPv6 app enabled)	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported

## Routing Policy CT Type

Table 63: Routing Policy CT Type

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
Routing Policy on a BGP session with import/export IPv4 prefixes	Yes	Yes	Yes	Yes	Yes	Not supported
Routing Policy on a BGP session with import/export IPv6 prefixes	Yes	Yes	Yes	Yes	Yes	Not supported

Table 63: Routing Policy CT Type *(Continued)*

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
Routing Policy on a BGP session with IPv4 aggregate prefixes	Yes	Yes	Yes	Yes	Yes	Not supported
Routing Policy on a BGP session with IPv6 aggregate prefixes	Yes	Yes	Yes	Yes	Yes	Not supported

**BGP Attributes (common to all BGP CTs)**

Table 64: BGP Attributes (common to all BGP CTs)

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
BGP: enable Password/MD5 based authentication	Yes	Yes	Yes	Yes	Yes	Not supported
BGP: Custom BGP timers (Keep Alive timer, Hold timer)	Yes	Yes	Yes	Yes	Yes	Not supported
BGP: Custom TTL	Yes	Yes	Yes	Yes	Yes	Not supported
BGP: Enable Single-hop BFD	Yes	Yes	Yes	Yes	Yes	Not supported

**Apstra 4.0.0 Feature Matrix****IN THIS SECTION**

- [Fabric Connectivity | 881](#)
- [Device Management | 882](#)
- [Connectivity \(from Leaf Layer\) | 882](#)
- [Routing Policies | 883](#)
- [Miscellaneous | 884](#)
- [Virtual Network CT Type | 884](#)
- [IP Link CT Type | 885](#)

- [Static Route CT Type | 885](#)
- [Custom Static Route CT Type | 886](#)
- [BGP to Generic CT Type | 886](#)
- [BGP to IP Endpoint CT Type | 890](#)
- [Dynamic BGP Peering CT Type | 892](#)
- [Routing Policy CT Type | 893](#)
- [BGP Attributes \(common to all BGP CTs\) | 894](#)

### Fabric Connectivity

**Table 65: Fabric Connectivity**

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
3-stage Clos	Yes	Yes	Yes	Yes	Yes	Yes
5-stage Clos	Yes	Yes	Yes	Yes	Yes	Yes
IPv6 fabric RFC-5549 (non-EVPN)	Yes	Yes	Yes	Yes	Not supported	Not supported
3-stage Clos with access switch layer	Not supported	Not supported	Not supported	Not supported	Limited	Limited
Collapsed fabric	Not supported	Not supported	Not supported	Not supported	Limited	Limited

## Device Management

**Table 66: Device Management**

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
On-box agent	Yes	Yes	Yes	Yes	Not possible	Not supported
Off-box agent	Yes	Yes	Contact support	Not supported	Yes	Yes
Telemetry extensibility	Yes	Yes	Yes	Yes	Yes	Not supported
Apstra ZTP	Yes	Yes	Yes	Yes	Yes	Not supported
OS upgrade	Yes	Yes	Yes	Yes	Yes	Not supported
Traffic draining (spines/ superspines - maintenance mode)	Yes	Yes	Yes	Yes	Yes	Yes
Traffic draining (leafs)	Limited	Limited	Limited	Limited	Limited	Limited

## Connectivity (from Leaf Layer)

**Table 67: Connectivity (from Leaf Layer)**

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
Access layer of switches	Not supported	Not supported	Not supported	Not supported	Limited	N/A
LAG	Yes	Yes	Yes	Yes	Yes	N/A
MLAG/vPC	Yes	Yes	Yes	Yes	N/A	N/A
EVPN ESI (with LACP)	Not supported	Not supported	Not supported	Not possible	Yes	N/A

Table 67: Connectivity (from Leaf Layer) *(Continued)*

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
802.1x	Yes	Not supported	Not supported	Not possible	Not supported	N/A
VLANs	Yes	Yes	Yes	Yes	Yes	N/A
Overlay protocol: static VXLAN	Yes	Yes	Not possible	Not possible	Not supported	N/A
Overlay protocol: EVPN (3-stage and 5-stage)	Yes	Yes	Yes	Yes	Yes	N/A
IPv4 DHCP relay	Yes	Yes	Yes	Yes	Yes	N/A
IPv6 DHCP relay	Yes	Yes	Yes	Yes	Not possible	N/A
EVPN DCI	Yes	Yes	Yes	Yes	Limited	N/A
IPv6 for applications (with EVPN and IPv4 fabric)	Yes	Yes	Yes	Yes	Not supported	N/A
Group-based policy (ACL on ToRs)	Yes	Yes	Not supported	Not supported	Not supported	N/A

## Routing Policies

Table 68: Routing Policies

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
Import all routes or default route or extra routes only	Yes	Yes	Yes	Yes	Yes	Yes

Table 68: Routing Policies *(Continued)*

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
Export loopback, link and VN IP. Export extra routes	Yes	Yes	Yes	Yes	Yes	Yes
Export aggregate prefixes	Yes	Yes	Yes	Yes	Not supported	Not supported
Export L3 server link subnets	Yes	Yes	Yes	Yes	Yes	Yes
Route target import/export policies	Yes	Yes	Yes	Yes	Not supported	Not supported

## Miscellaneous

Table 69: Miscellaneous

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
Configlets	Yes	Yes	Yes	Yes	Limited	Limited
FFE: add racks/add links/change speed, and so on	Yes	Yes	Yes	Yes	Yes	Yes
Mixed leaf/spine link speed	Yes	Yes	Yes	Yes	Yes	Limited

## Virtual Network CT Type

Table 70: Virtual Network CT Type

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
Single Virtual Network	Yes	Yes	Yes	Yes	Yes	Not supported
Multiple Virtual Network	Yes	Yes	Yes	Yes	Yes	Not supported

## IP Link CT Type

**Table 71: IP Link CT Type**

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
L3 Sub-interface on non-LAG physical interface (untagged/vlan tagged, default/non-default RZ, IPv4)	Yes	Yes	Yes	Not possible	Yes	Not supported
L3 Sub-interface on non-LAG physical interface (untagged/vlan tagged, default/non-default RZ, IPv6)	Yes	Yes	Yes	Not possible	Not supported	Not supported
L3 Sub-interface on LAG interface (untagged/vlan tagged, default/non-default RZ, IPv4)	Yes	Yes	Yes	Not possible	Yes	Not supported
L3 Sub-interface on LAG interface (untagged/vlan tagged, default/non-default RZ, IPv6)	Yes	Yes	Yes	Not possible	Not supported	Not supported

## Static Route CT Type

**Table 72: Static Route CT Type**

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
Static Route (IPv4) applied on L3 Sub-interface	Yes	Yes	Yes	Not possible	Yes	Not supported
Static Route (IPv6) applied on L3 Sub-interface	Yes	Yes	Yes	Not possible	Not supported	Not supported
Static Route (IPv4) applied on SVI	Yes	Yes	Yes	Yes	Yes	Not supported
Static Route (IPv6) applied on SVI	Yes	Yes	Yes	Yes	Not supported	Not supported

Table 72: Static Route CT Type *(Continued)*

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
Static Route with Share IP Endpoint Enabled (IPv4)	Yes	Yes	Yes	Yes	Yes	Not supported
Static Route with Share IP Endpoint Enabled (IPv6)	Yes	Yes	Yes	Yes	Not supported	Not supported

### Custom Static Route CT Type

Table 73: Custom Static Route CT Type

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
Custom Static Route (IPv4, default/non-default RZ)	Yes	Yes	Yes	Yes	Yes	Not supported
Custom Static Route (IPv6, default/non-default RZ)	Yes	Yes	Yes	Yes	Not supported	Not supported

### BGP to Generic CT Type

Table 74: BGP to Generic CT Type

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
BGP session on L3 Sub-interface towards generic (IPv4, default/non-default RZ)	Yes	Yes	Yes	Not possible	Yes	Not supported
BGP session on L3 Sub-interface towards generic (IPv6, default/non-default RZ)	Yes	Yes	Yes	Not possible	Not supported	Not supported

Table 74: BGP to Generic CT Type *(Continued)*

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
BGP session on SVI towards generic (IPv4, default RZ)	Yes	Yes	Yes	Yes	Not supported	Not supported
BGP session on SVI towards generic (IPv4, non-default RZ)	Yes	Yes	Yes	Yes	Yes	Not supported
BGP session on SVI towards generic (IPv6, non-default RZ)	Yes	Yes	Yes	Yes	Not supported	Not supported
BGP session on SVI towards generic (IPv6, default RZ)	Yes	Yes	Yes	Yes	Not supported	Not supported
BGP session on SVI (mlag) towards dual-homed generic using secondary IPs (IPv4, default VRF)	Yes	Yes	Yes	Not possible	Not supported	Not supported
BGP session on SVI (mlag) towards dual-homed generic using secondary IPs (IPv4, non-default VRF)	Yes	Not supported	Yes	Not possible	Yes	Not supported
BGP session on SVI (mlag) towards dual-homed generic using secondary IPs (IPv6, default VRF)	Yes	Yes	Yes	Not possible	Not supported	Not supported
BGP session on SVI (mlag) towards dual-homed generic using secondary IPs (IPv6, non-default VRF)	Yes	Not supported	Yes	Not possible	Not supported	Not supported

Table 74: BGP to Generic CT Type *(Continued)*

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
BGP session to generic with Share IP Endpoint Enabled (IPv4)	Yes	Not supported	Yes	Yes	Yes	Not supported
BGP session to generic with Share IP Endpoint Enabled (IPv6)	Yes	Not supported	Yes	Yes	Not supported	Not supported
BGP session to generic with dynamic ASN (IPv4)	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported
BGP session to generic with Static ASN (IPv4)	Yes	Yes	Yes	Yes	Yes	Not supported
BGP session to generic with dynamic ASN (IPv6)	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported
BGP session to generic with static ASN (IPv6)	Yes	Yes	Yes	Yes	Not supported	Not supported
BGP Unnumbered session (link-local peering) on L3 Sub-interface (BP has IPv6 app enabled, default VRF)	Yes	Yes	Yes	Yes	Not supported	Not supported
BGP Unnumbered session (link-local peering) on L3 Sub-interface (BP has IPv6 app enabled, non-default VRF)	Not supported	Yes	Yes	Yes	Not supported	Not supported
BGP Unnumbered session (link-local peering) on SVI (BP has IPv6 app enabled, default VRF)	Yes	Yes	Yes	Yes	Not supported	Not supported

Table 74: BGP to Generic CT Type *(Continued)*

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
BGP Unnumbered session (link-local peering) on SVI (BP has IPv6 app enabled, non-default VRF)	Not supported	Yes	Yes	Yes	Not supported	Not supported
BGP Unnumbered session (link-local peering) on L3 Sub-interface (default VRF, BP has IPv6 app disabled)	Yes	Yes	Yes	Not possible	Not supported	Not supported
BGP Unnumbered session (link-local peering) on L3 Sub-interface (non-default VRF, BP has IPv6 app disabled)	Not supported	Yes	Yes	Not possible	Not supported	Not supported
BGP Unnumbered session (link-local peering) on SVI (BP has IPv6 app disabled, default VRF only)	Yes	Yes	Yes	Yes	Not possible	Not supported
BGP Peering combinations (Int to Int, Lo to Int, Int to Lo, Lo to Lo)	Yes	Yes	Yes	Yes	Yes	Not supported
BGP session (IPv6 addressed) with IPv4 SAFI (rfc5549) with static ASN (BP has IPv6 app enabled)	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported

Table 74: BGP to Generic CT Type *(Continued)*

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
BGP session (IPv6 addressed) with IPv4 SAFI (rfc5549) with dynamic ASN (BP has IPv6 app enabled)	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported

**BGP to IP Endpoint CT Type**

Table 75: BGP to IP Endpoint CT Type

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
BGP session from L3 sub-interface to any IP endpoint in the network (IPv4, default/non-default RZ)	Yes	Yes	Yes	Not possible	Yes	Not supported
BGP session from L3 sub-interface to any IP endpoint in the network (IPv6, default/non-default RZ)	Yes	Yes	Yes	Not possible	Not supported	Not supported
BGP session from SVI to any IP endpoint in the network (IPv4, default/non-default RZ)	Yes	Yes	Yes	Yes	Yes	Not supported
BGP session from SVI to any IP endpoint in the network (IPv6, non-default RZ)	Yes	Yes	Yes	Yes	Not supported	Not supported

Table 75: BGP to IP Endpoint CT Type *(Continued)*

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
BGP session from SVI to any IP endpoint in the network (IPv6, default RZ)	Yes	Yes	Yes	Yes	Not supported	Not supported
BGP session from Loopback to any IP endpoint in the network (IPv4, default/non-default RZ)	Yes	Yes	Yes	Yes	Yes	Not supported
BGP session from Loopback to any IP endpoint in the network (IPv6, default/non-default RZ)	Yes	Yes	Yes	Yes	Not supported	Not supported
BGP session with specific peer IP and Static ASN (IPv4)	Yes	Yes	Yes	Yes	Yes	Not supported
BGP session with specific peer IP and Static ASN (IPv6)	Yes	Yes	Yes	Yes	Not supported	Not supported
BGP session with specific peer IP and dynamic ASN (IPv4)	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported
BGP session with specific peer IP and dynamic ASN (IPv6)	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported
BGP session (IPv6 addressed) with IPv4 SAFI (rfc5549) with static ASN (BP has IPv6 app enabled)	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported

Table 75: BGP to IP Endpoint CT Type *(Continued)*

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
BGP session (IPv6 addressed) with IPv4 SAFI (rfc5549) with dynamic ASN (BP has IPv6 app enabled)	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported

### Dynamic BGP Peering CT Type

Table 76: Dynamic BGP Peering CT Type

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
Dynamic BGP prefix peering on SVI (IPv4), default VRF	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported
Dynamic BGP prefix peering on SVI (IPv4), non-default VRF	Yes	Yes	Yes	Yes	Not supported	Not supported
Dynamic BGP prefix peering on SVI (IPv6), default VRF	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported
Dynamic BGP prefix peering on SVI (IPv6), non-default VRF	Yes	Yes	Yes	Yes	Not supported	Not supported
Dynamic BGP prefix peering on L3 sub-interface (IPv4), default VRF	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported

Table 76: Dynamic BGP Peering CT Type *(Continued)*

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
Dynamic BGP prefix peering on L3 sub-interface (IPv4), non-default VRF	Yes	Yes	Yes	Not possible	Not supported	Not supported
Dynamic BGP prefix peering on L3 sub-interface (IPv6), default VRF	Not supported	Not supported	Not supported	Not possible	Not supported	Not supported
Dynamic BGP prefix peering on L3 sub-interface (IPv6), non-default VRF	Yes	Yes	Yes	Not possible	Not supported	Not supported
Dynamic prefix peering (link-local prefix peering, rfc5549), (BP has IPv6 app disabled)	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported
Dynamic prefix peering (IPv6 peering, IPv4 AFI, rfc5549), (BP has IPv6 app enabled)	Not supported	Not supported	Not supported	Not supported	Not supported	Not supported

## Routing Policy CT Type

Table 77: Routing Policy CT Type

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
Routing Policy on a BGP session with import/export IPv4 prefixes	Yes	Yes	Yes	Yes	Yes	Not supported
Routing Policy on a BGP session with import/export IPv6 prefixes	Yes	Yes	Yes	Yes	Not supported	Not supported

Table 77: Routing Policy CT Type *(Continued)*

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
Routing Policy on a BGP session with IPv4 aggregate prefixes	Yes	Yes	Yes	Yes	Not supported	Not supported
Routing Policy on a BGP session with IPv6 aggregate prefixes	Yes	Yes	Yes	Yes	Not supported	Not supported

**BGP Attributes (common to all BGP CTs)**

Table 78: BGP Attributes (common to all BGP CTs)

Feature	EOS	NX-OS	Cumulus	SONiC	Junos OS	Junos OS Evolved
BGP: enable Password/MD5 based authentication	Yes	Yes	Yes	Yes	Yes	Not supported
BGP: Custom BGP timers (Keep Alive timer, Hold timer)	Yes	Yes	Yes	Yes	Yes	Not supported
BGP: Custom TTL	Yes	Yes	Yes	Yes	Yes	Not supported
BGP: Enable Single-hop BFD	Yes	Yes	Yes	Yes	Yes	Not supported

**Qualified Device and NOS (Devices)****IN THIS SECTION**

- [Apstra Release 4.0.2 | 895](#)
- [Apstra Release 4.0.1 | 897](#)
- [Apstra Release 4.0.0 | 899](#)
- [Juniper Junos OS Evolved on Apstra Versions 4.0.1 and 4.0.0 | 901](#)

Recommended qualified NOS versions and device (series) are listed below. For information about additional devices and NOS versions that may be available, contact ["Juniper Support" on page 777](#). Additional NOS versions may be supported on a case-by-case basis.

To request support for NOS not listed, contact your Juniper Apstra Sales representative.

## Apstra Release 4.0.2

**Table 79: Apstra Release Version 4.0.2 - Supported NOS**

Device Operating System	Qualified NOS Versions	Supported Device (Series)
Juniper Junos OS	• 21.2R3	• QFX5100 - Don't use as leaf with Layer 3 VNI
	• 21.2R1-S2.2	• QFX5110 - Can't be used as border leaf. It can't route between VXLAN IRB and L3 interface.
	• 20.4R3-S1.3	• QFX5120
		• QFX5200
		• QFX5210
		• QFX10002
		• QFX10008
		• EX4300-48MP - Can't be used as border leaf. It can't route between VXLAN IRB and L3 interface.
		• EX4650-48Y
Juniper Junos OS	21.2R1-S2.2	• EX4400-48F
		• EX4400-48T
		• EX4400-24T
Juniper Junos OS Evolved (spine, superspine and non-EVPN-VXLAN leaf)	• 21.2R2-S2-EVO	• QFX5220
	• 20.4R3-S1.3-EVO	• QFX5130
		• PTX10001-36MR

Table 79: Apstra Release Version 4.0.2 - Supported NOS *(Continued)*

Device Operating System	Qualified NOS Versions	Supported Device (Series)
Enterprise SONiC	<ul style="list-style-type: none"> <li>Broadcom Version: SONiC-OS-3.3.0-GA-Enterprise_Advanced</li> <li>Broadcom Version: SONiC-OS-3.3.0-GA-Enterprise_Base</li> </ul>	<ul style="list-style-type: none"> <li>Dell Z9432F-ON (spine role)</li> <li>Dell Z9332F-ON</li> <li>Dell Z9264F-ON</li> <li>Dell Z9100-ON</li> <li>Dell S5296F-ON</li> <li>Dell S5248F-ON</li> <li>Dell S5232F-ON</li> <li>Dell S5212F-ON</li> <li>Edgecore/Accton AS7816-64X</li> <li>Edgecore/Accton AS7726-32X</li> <li>Edgecore/Accton S7712-32X</li> <li>Edgecore/Accton AS7326-56X</li> <li>Edgecore/Accton AS5712-54X</li> </ul>
Cisco NX-OS	<ul style="list-style-type: none"> <li>10.1(2)</li> <li>9.3(8)</li> </ul>	Nexus 3000 or 9000 Platform
Arista EOS	<ul style="list-style-type: none"> <li>4.25.3.1M</li> <li>4.24.5M</li> <li>4.23.6M</li> <li>4.22.9M</li> </ul>	DCS-7000 Series
Cumulus Linux	Contact Juniper Support	Cumulus Linux-supported platforms with x86 CPU processor

## Apstra Release 4.0.1

Table 80: Apstra Release Version 4.0.1 - Supported NOS

Device Operating System	Qualified NOS Versions	Supported Device (Series)
Juniper Junos OS	<ul style="list-style-type: none"> <li>21.2R1-S1.3</li> <li>20.2R3-S2.5</li> </ul>	<ul style="list-style-type: none"> <li>QFX5100 - Don't use as leaf with Layer 3 VNI</li> <li>QFX5110 - Can't be used as border leaf. It can't route between VXLAN IRB and L3 interface.</li> <li>QFX5120</li> <li>QFX5130</li> <li>QFX5200</li> <li>QFX5210</li> <li>QFX5220</li> <li>QFX10002</li> <li>QFX10008</li> <li>EX4300-48MP - Can't be used as border leaf. It can't route between VXLAN IRB and L3 interface.</li> <li>EX4650-48Y</li> </ul>
Juniper Junos OS	<ul style="list-style-type: none"> <li>21.2R1-S1.3</li> </ul>	<ul style="list-style-type: none"> <li>EX4400-48F</li> <li>EX4400-48T</li> <li>EX4400-24T</li> </ul>
Juniper Junos OS Evolved - Tech Preview - see section below	20.4R2-S2.10-EVO	<ul style="list-style-type: none"> <li>QFX5220</li> <li>QFX5130</li> </ul>

Table 80: Apstra Release Version 4.0.1 - Supported NOS *(Continued)*

Device Operating System	Qualified NOS Versions	Supported Device (Series)
Juniper Junos OS Evolved - Tech Preview - see section below	20.4R2-S1.4-EVO	<ul style="list-style-type: none"> <li>• QFX5220</li> <li>• QFX5130</li> <li>• PTX10001-36MR</li> </ul>
Enterprise SONiC	<ul style="list-style-type: none"> <li>• SONiC-OS-3.3.0-Enterprise_Advanced</li> <li>• SONiC-OS-3.3.0-Enterprise_Base</li> </ul>	<ul style="list-style-type: none"> <li>• Del Z9432F-ON (spine role)</li> <li>• Dell Z9332F-ON</li> <li>• Dell Z9264F-ON</li> <li>• Dell Z9100-ON</li> <li>• Dell S5296F-ON</li> <li>• Dell S5248F-ON</li> <li>• Dell S5232F-ON</li> <li>• Dell S5212F-ON</li> <li>• Edgecore/Accton AS7816-64X</li> <li>• Edgecore/Accton AS7726-32X</li> <li>• Edgecore/Accton S7712-32X</li> <li>• Edgecore/Accton AS7326-56X</li> <li>• Edgecore/Accton AS5712-54X</li> </ul>
Cisco NX-OS	<ul style="list-style-type: none"> <li>• 10.1(2)</li> <li>• 9.3(8)</li> <li>• 9.3(7)</li> </ul>	Nexus 3000 or 9000 Platform

Table 80: Apstra Release Version 4.0.1 - Supported NOS *(Continued)*

Device Operating System	Qualified NOS Versions	Supported Device (Series)
Arista EOS	<ul style="list-style-type: none"> <li>• 4.25.3.1M</li> <li>• 4.24.5M</li> <li>• 4.23.6M</li> <li>• 4.22.9M</li> </ul>	DCS-7000 Series
Cumulus Linux	Contact Juniper Support	Cumulus Linux-supported platforms with x86 CPU processor

## Apstra Release 4.0.0

Table 81: Apstra Release Version 4.0.0 - Supported NOS

Device Operating System	Qualified NOS Versions	Supported Device (Series)
Juniper Junos OS	20.2R2-S3.5	<ul style="list-style-type: none"> <li>• QFX5110</li> <li>• QFX5120</li> <li>• QFX5130</li> <li>• QFX5200</li> <li>• QFX5210</li> <li>• QFX5220</li> <li>• QFX10002</li> <li>• QFX10008</li> <li>• EX4300-48MP - Can't be used as border leaf. It can't route between VXLAN IRB and L3 interface.</li> <li>• EX4650-48Y</li> </ul>

Table 81: Apstra Release Version 4.0.0 - Supported NOS (*Continued*)

Device Operating System	Qualified NOS Versions	Supported Device (Series)
Juniper Junos OS	18.4R2	QFX5100 - Can't be used as border leaf. It can't route between VXLAN IRB and L3 interface.
Juniper Junos OS Evolved - Tech Preview - see section below	20.4R2-S1.4-EVO	<ul style="list-style-type: none"> <li>• QFX5220</li> <li>• QFX5130</li> </ul>
Enterprise SONiC	<ul style="list-style-type: none"> <li>• SONiC-OS-3.2.0-Enterprise_Advanced</li> <li>• SONiC-OS-3.2.0-Enterprise_Base</li> </ul>	<ul style="list-style-type: none"> <li>• Dell Z9332F-ON</li> <li>• Dell Z9264F-ON</li> <li>• Dell Z9100-ON</li> <li>• Dell S5296F-ON</li> <li>• Dell S5248F-ON</li> <li>• Dell S5232F-ON</li> <li>• Dell S5212F-ON</li> <li>• Edgecore/Accton AS7816-64X</li> <li>• Edgecore/Accton AS7726-32X</li> <li>• Edgecore/Accton S7712-32X</li> <li>• Edgecore/Accton AS7326-56X</li> <li>• Edgecore/Accton AS5712-54X</li> </ul>
Cisco NX-OS	<ul style="list-style-type: none"> <li>• 9.3(7)</li> <li>• 7.0(3)I7(9)</li> </ul>	Nexus 3000 or 9000 Platform

Table 81: Apstra Release Version 4.0.0 - Supported NOS *(Continued)*

Device Operating System	Qualified NOS Versions	Supported Device (Series)
Arista EOS	<ul style="list-style-type: none"> <li>4.24.5M</li> <li>4.22.9M</li> <li>4.21.12M</li> </ul>	DCS-7000 Series
Cumulus Linux	3.7.13 - For the latest NVIDIA Cumulus Linux support information, contact Juniper Support <support>.	Cumulus Linux-supported platforms with x86 CPU processor

## Juniper Junos OS Evolved on Apstra Versions 4.0.1 and 4.0.0

**NOTE:** This feature has been classified as a Juniper Apstra Technology Preview feature. These features are "as is" and for voluntary use. Juniper Support will attempt to resolve any issues that customers experience when using these features and create bug reports on behalf of support cases. However, Juniper may not provide comprehensive support services to Tech Preview features. For additional information, refer to the ["Juniper Apstra Technology Previews" on page 1082](#) or contact ["Juniper Support" on page 777](#).

On Apstra release versions 4.0.1 and 4.0.0, limited support is provided for Junos OS Evolved on selected devices on spine and super spine roles only (not leaf roles). On EVPN fabrics, devices running Junos OS Evolved can be used as spine or superspine only. For other details and limitations see Juniper Device Profiles <juniper\_device\_profile>.

## Agent Configuration File (Devices)

### IN THIS SECTION

- [Controller Section | 902](#)
- [Service Section | 903](#)
- [Logrotate Section | 904](#)

- Device Info Section | 905
- Device Profile Section | 905

## Controller Section

```
[controller]
# <metadb> provides directory service for AOS. It must be configured properly
# for a device to connect to AOS controller.
metadb = tbt://aos-server:29731
# Use <web> to specify AOS web server IP address or name. This is used by
# device to make REST API calls to AOS controller. It is assumed that AOS web
# server is running on the same host as metadb if this option is not specified
web =
# <interface> is used to specify the management interface. This is currently
# being used only on server devices and the AOS agent on the server device will
# not come up unless this is specified.
interface =
```

### metadb

Agent Server Discovery is a client-server model. The Apstra Device agent registers directly to the Apstra server via the `metadb` connection. The Apstra server can be discovered from static IP or DNS.

**Dynamic DNS** - By default, Apstra device agents point to the DNS entry **aos-server**, relying on dhcp-provided DNS resolution and hostname resolution. On the Apstra server, if the *metadb* connection entry points to a DNS entry, then the Apstra agents must be able to resolve that DNS entry as well. DNS must be configured so `aos-server` resolves to an interface on the Apstra server itself, and so the agents are configured with `metadb = tbt://aos-server:29731`

**Static DNS** - We can add a static DNS entry pointing directly to the IP of `aos-server`. Add a static DNS entry, or use a DNS Nameserver configuration on the device.

Arista and Cisco Static Hostname

```
localhost(config)#ip host aos-server 192.168.25.250
```

## Cumulus and Linux Static Hostnames /etc/hosts

```
192.168.25.250 aos-server
```

## Obtaining IP from Apstra Server

```
admin@aos-server:~# ip addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 08:00:27:8a:39:05 brd ff:ff:ff:ff:ff:ff
    inet 192.168.59.250/24 brd 192.168.59.255 scope global eth0
    inet6 fe80::a00:27ff:fe8a:3905/64 scope link
    valid_lft forever preferred_lft forever
```

Then the agents will be configured with `metadb = tbt://aos-server:29731`.

## web

In a future release, the Apstra REST API will be able to run on a separate server from the Apstra server itself. This feature is for Apstra internal usage only.

## interface

The device agent source interface applies to Linux servers only (Ubuntu, CentOS). This source IP is the server interface that the device agent uses when registering with Apstra. For example, on a server, to bind the device agent to `eth1` instead of the default `eth0`, specify `interface = eth1`.

## Service Section

```
[service]
# AOS device agent by default starts in "telemetry-only" mode.Set following
# variable to 1 if you want AOS agent to manage the configuration of your
# device.
enable_configuration_service = 0
# When managing device configuration AOS agent will restore backup config if it
# fails to connect to AOS controller in <backup_config_restoration_timeout>,
# specified as <hh:mm:ss>. Set it to 00:00:00 to disable backup restoration
backup_config_restoration_timeout = 00:00:00
```

The service section manages specific agent configuration related to configuration rendering and telemetry services.

## enable\_configuration\_service

This field specifies the operation mode of the device agent: telemetry only or full control.

`enable_configuration_service = 0` Leaving the default value of 0 allows Apstra to push telemetry (alerts) only. Configuration files are not modified. This ensures that Apstra does not modify any configurations unless specified by the network administrator.

`enable_configuration_service = 1` Setting this field to 1 allows Apstra to fully manage the device agent configuration, including pushing discovery and full intent-based configuration.

## backup\_config\_restoration\_timeout

By design, configuration is not *stored* on the device. This prevents a device from booting up and immediately participating in fabric that may not be properly configured yet. The Apstra device agent is configured after the discovery phase completes.

`backup_restoration_timeout = 00:00:00` This disabled state (default) keeps the Apstra device agent from replacing the running configuration if it cannot contact the Apstra server. Any previous configuration state is not restored.

`backup_restoration_timeout = 00:15:00` Any value other than the default `00:00:00` enables the Apstra agent to boot and replace the running configuration with the most known previous state after the specified period of time (fifteen minutes in this example). Specifically, the files from `/.aos/rendered/` are restored to the system after the configuration restore period expires.

## Logrotate Section

```
[logrotate]

# AOS has builtin log rotate functionality. You can disable it by setting
# <enable_log_rotate> to 0 if you want to use linux logrotate utility to manage
# your log files. AOS agent reopens log file on SIGHUP
enable_log_rotate = 1
# Log file will be rotated when its size exceeds <max_file_size>
max_file_size = 1M
# The most recent <max_kept_backups> rotated log files will be saved. Older
# ones will be removed. Specify 0 to not save rotated log files, i.e. the log
# file will be removed as soon as its size exceeds limit.
max_kept_backups = 5
# Interval, specified as <hh:mm:ss>, at which log files are checked for
# rotation.
check_interval = 1:00:00
```

Apstra logs to the `/var/log/aos` folder under a series of files. Apstra implements its own method of log rotation to prevent `/var/log/aos` from filling up. You can enable (2) or disable (1) log rotation. Each individual log file is rotated when it approaches the appropriate maximum size. Log rotation occurs by default every hour.

## Device Info Section

```
[device_info]
# <model> is used to specify the device's hardware model to be reported to AOS
# device manager. This is only used by servers, so can be ignored for non-
# server devices such as switches. By default a server reports "Generic Model"
# which matches a particular HCL entry's selector::model value in AOS. Specify
# another model for the server to be classified as a different HCL entry.
model = Generic Model
```

### model

The device info section is used to modify the default device model of servers as they register to Apstra. For example, `Server 2x10G` changes the server to a dual-attached L3 server. All valid options for `model` include:

- Generic Model
- Server 2x10G
- Server 1x25G
- Server 1x40G
- Server 4x10G

## Device Profile Section

```
# <device_profile_id> is used to specify the device profile to be associated to
# the device. Selector in the specified device profile should match the
# reported device facts.
device_profile_id =
[credential]
username = admin
```

# NOS Upgrade Paths (Devices)

## IN THIS SECTION

- [Apstra Release 4.0.2 | 906](#)
- [Apstra Release 4.0.1 | 907](#)
- [Apstra Release 4.0.0 | 908](#)

Network operating system (NOS) upgrade paths can be from a recommended NOS release in a previous Apstra release to a recommended NOS release in a newer Apstra release. They can also be between NOS releases on the same Apstra release. See the sections below for supported paths.

For information about other upgrade paths that may be available, or to request support for a specific upgrade path, contact ["Juniper Support" on page 777](#).

## Apstra Release 4.0.2

Table 82: Apstra Release 4.0.2 - Supported OS Upgrade/Downgrade Paths

Network Operating System	From	To
Juniper Junos OS	20.2R3-S2.5	20.4R3-S1.3
Juniper Junos OS	20.4R3-S1.3	21.2R1-S2.2
Juniper Junos OS	21.2R1-S2.2	20.4R3-S1.3
Juniper Junos OS Evolved	20.4R3-S1.3-Evolved	20.4R2-S2.10-Evolved
Cisco NX-OS	7.0.3.I7.9	9.3.8
Cisco NX-OS	9.2.3	9.3.8
Cisco NX-OS	9.3.3	9.3.8
Cisco NX-OS	9.3.8	10.1.2
Arista EOS	4.20.11M	4.25.3.1M

Table 82: Apstra Release 4.0.2 - Supported OS Upgrade/Downgrade Paths *(Continued)*

Network Operating System	From	To
Arista EOS	4.20.11M	4.24.5M
Arista EOS	4.21.14M	4.23.6M
Arista EOS	4.22.9M	4.24.5M
Arista EOS	4.22.9M	4.23.6M
Arista EOS	4.23.6M	4.22.9M
Arista EOS	4.24.5M	4.22.9M
Cumulus Linux	Contact Juniper Support	Contact Juniper Support

## Apstra Release 4.0.1

Table 83: Apstra Release 4.0.1 - Supported OS Upgrade/Downgrade Paths

Network Operating System	From	To
Juniper Junos OS	18.4R2-S8	20.2R3-S2.5
Juniper Junos OS	20.2R3-S2.5	21.2R1-S1.3
Juniper Junos OS	21.2R1-S1.3	<ul style="list-style-type: none"> <li>20.2R3-S2.5</li> <li>21.2R1.10</li> </ul>
Juniper Junos OS Evolved	20.4R2-S1.4-Evolved	20.4R2-S2.10-Evolved
Cisco NX-OS	9.3.3	9.3.7
Cisco NX-OS	9.3.7	10.1.2
Arista EOS	4.10.11M	4.23.6M

Table 83: Apstra Release 4.0.1 - Supported OS Upgrade/Downgrade Paths *(Continued)*

Network Operating System	From	To
Arista EOS	4.20.11M	<ul style="list-style-type: none"> <li>• 4.22.9M</li> <li>• 4.24.5M</li> </ul>
Arista EOS	4.21.5.1F	4.23.6M
Arista EOS	4.22.8M	4.25.3.1M
Arista EOS	4.23.6M	4.22.9M
Arista EOS	4.24.5M	4.22.9M
Cumulus Linux	Contact Juniper Support	Contact Juniper Support

## Apstra Release 4.0.0

Table 84: Apstra Release 4.0.0 - Supported OS Upgrade/Downgrade Paths

Network Operating System	From	To
Juniper Junos OS	18.4R2-S5.4	20.2R2-S3.5
Juniper Junos OS	20.2R2-S3.5	18.4R2-S5.4
SONiC	3.1.2-GA-adv	3.2.0-GA-adv
SONiC	3.2.0-GA-adv	3.1.2-GA-adv
Cisco NX-OS	7.0.3.I7.9	9.3.7
Cisco NX-OS	9.2.2	7.0.3.I7.9
Cisco NX-OS	9.3.3	9.3.7

Table 84: Apstra Release 4.0.0 - Supported OS Upgrade/Downgrade Paths *(Continued)*

Network Operating System	From	To
Cisco NX-OS	9.3.7	<ul style="list-style-type: none"> <li>7.0.3.17.9</li> <li>9.2.2</li> </ul>
Arista EOS	4.20.11M (not supported)	<ul style="list-style-type: none"> <li>4.22.9M</li> <li>4.23.6M</li> <li>4.24.5M</li> </ul>
Arista EOS	4.21.5.1F (not supported)	4.23.6M
Arista EOS	4.23.6M	4.22.9M
Arista EOS	4.24.5M	4.22.9M
Cumulus Linux	3.7.5	3.7.13
Cumulus Linux	3.7.13	4.2.1
Cumulus Linux	4.2.1	3.7.13

## Apstra EVPN Support Addendum

### IN THIS SECTION

- [Qualified Vendor and NOS | 910](#)
- [Limitations | 911](#)
- [Cumulus RN-766 Support | 912](#)
- [TCAM Carving in NX-OS | 913](#)
- [Arista EOS VxLAN Routing | 914](#)
- [Graph Node VTEP Types | 915](#)

When deploying EVPN on Apstra-supported devices and NOSs, you must be aware of several caveats and limitations. Even though EVPN is a standard, vendors implement protocols in very different manners. Also, different ASICs support varying feature sets that impact EVPN BGP VXLAN implementations (Routing In and Out of Tunnels (RIOT) for example). The following sections describe supported EVPN deployment implementations.

## Qualified Vendor and NOS

Apstra software supports EVPN on the following hardware. For recommended NOS versions, see ["Qualified Device and NOS" on page 894](#).

## Hardware ASIC Support

Apstra supports EVPN on the following hardware ASICs:

- Cisco Cloudscale
- Mellanox Spectrum A1
- Trident Trident2 (see below)
- Trident Trident2+ (see below)
- Trident Trident3 (see below)
- Trident Tomahawk (see below)
- Juniper Q5

**Table 85: Apstra EVPN ASIC Support**

ASIC	Example Switches	Notes
Arista Trident2	Arista DCS-7050	Can be used as Spine, Leaf, or Border Leaf. Must set up EOS Recirculation interface(s) to be used as a Layer3 Leaf (see <a href="#">Arista VXLAN documentation</a> for more information).
Arista Trident3	DCS-7050CX3	Can be used as Spine, Leaf, or Border Leaf.
Arista XP80	Arista DCS-7160	Can be used as Spine, Leaf, or Border Leaf.
Arista Jericho	DCS-7280R	Can be used as Spine, Leaf, or Border Leaf.
Cisco Cloudscale	Cisco 93180YC-EX	Can be used as Spine, Leaf, or Border Leaf

Table 85: Apstra EVPN ASIC Support *(Continued)*

ASIC	Example Switches	Notes
Cisco Trident2 with ALE	Cisco 9396PX, 9372PX, 9332PQ, 9504	Can be used as Spine, Leaf, or Border Leaf (see TCAM Carving in NXOS section).
Cisco Trident2+	Cisco 3132Q-V	Cannot be used as Border Leaf
Cumulus Trident2+	Dell S4048T-ON, S6010-ON, EdgeCore AS5812-54X, AS6812-32X	Can be used as Spine, Leaf, or Border Leaf (see Cumulus RN-766 Support) in Cumulus.
Cumulus Maverick	Dell S4148F-ON, S4148T-ON	Can be used as Spine, Leaf, or Border Leaf (see Cumulus RN-766 Support) in Cumulus.
Cumulus Mellanox A1	Mellanox MSN2010, MSN2100, MSN2410, MSN2700 (Spectrum A1)	Can be used as Spine, Leaf, or Border Leaf. Spectrum A0 not supported.
Cumulus Tomahawk	Dell Z9100-ON, EdgeCore AS7712-32X	Can be used as Spine, Leaf, or Border Leaf (see Cumulus RN-766 Support). Must set up <a href="#">"Hyperloop interface(s)" on page 845</a> to be used as a Layer3 Leaf in Cumulus.
Juniper Q5	Juniper QFX10002	Can be used as Spine, Leaf, or Border Leaf
Juniper Trident2	Juniper QFX5100	Can be used as Spine or Layer2 Leaf
Juniper Trident2+	Juniper QFX5110	Can be used as Spine, Leaf, or Border Leaf
Juniper Trident3	Juniper QFX5120	Can be used as Spine, Leaf, or Border Leaf

For recommended NOS versions, refer to Device and NOS Support <device\_support>.

## Limitations

### IN THIS SECTION

- [EVPN Layer2 Limitations | 912](#)
- [EVPN Layer3 Limitations | 912](#)

### EVPN Layer2 Limitations

- VLAN (Rack-local) Virtual networks must be in the default routing zone.
- VxLAN (Inter-rack) Virtual networks cannot be part of the default routing zone.

### EVPN Layer3 Limitations

- Generic systems with BGP peering to non-default routing zones must connect to leaf devices.
- Generic systems with BGP peering only to the default routing zone can connect to leafs, spines or superspines.
- Multi-zone security segmentations only support up to 16 routing zones (VRFs) on Arista (HW Limitation)
- Inter routing zone (VRF) routing must be handled on a generic system (EVPN type 5 route leaking)
- All BGP sessions and loopback addresses are part of the default routing zone.

### Cumulus RN-766 Support

In all current versions of Cumulus Linux; when using Broadcom Trident II+, Trident3, and Maverick platforms in an external VXLAN routing environment; the switch does not rewrite MAC addresses and TTL, so packets are dropped by the next hop. See [Cumulus Linux Release Notes for RN-766](#) for more information.

Work-arounds are automatically implemented for Cumulus Linux RN-766 for the following criteria:

- Device profiles with "ASIC" field set to 'T2+', 'T3', or 'maverick' assigned to border-leaf device(s) in the blueprint
- **Overlay Control Protocol** set to **MP-EBGP EVPN**
- External Connectivity Point (ECP) configured for default or new routing zone
- External Connectivity Point (ECP) set to L2 mode, with Layer 2 VLAN based BGP peering with the generic system
- External Connectivity Point (ECP) configured with VLAN ID, SVI Subnet, and SVI IPs for border-leaf(s) and router(s).
- User assigns VNI from a VNI pool to the blueprint

Additional VNI and configuration is automatically allocated for the Cumulus Linux RN-766 work-around.



**CAUTION:** The RN-766 workaround is not supported on blueprints with MLAG L3 peer links deployed.



**CAUTION:** RN-766 workaround is not supported for blueprints with **Overlay Control Protocol** set to **Static VXLAN** and any L3 External Connectivity Points (ECP).

## TCAM Carving in NX-OS

To successfully deploy EVPN on Cisco Nexus devices other than Cisco Cloudscale, you must first configure Cisco NXOS TCAM carving. These other devices may include Cisco NXOSv, or Cisco Nexus "Trident2" devices such as 9396PX, 9372PX, 9332PQ, or 9504. On Cisco NXOS the ARP Suppression feature is used in order to minimize ARP flooding.

For details, see [Juniper Support Knowledge Base article KB36733](#)

Before installing the device agent, we recommend that you apply TCAM Carving during device management setup or during Cisco Power-on Auto Provisioning (POAP). TCAM Carving requires a device reboot.

Alternatively, you can apply TCAM Carving with configlets when you deploy the blueprint. You must manually reboot devices.

Use `show hardware access-list tcam region` to show and verify TCAM allocation on Cisco NX-OS.

### Cisco NXOSv TCAM Carving

```
hardware access-list tcam region vacl 0
hardware access-list tcam region racl 0
hardware access-list tcam region arp-ether 256
```

```
no hardware access-list tcam region arp-ether 256
no hardware access-list tcam region racl 0
no hardware access-list tcam region vacl 0
```

## Cisco Trident2 TCAM Carving

```
hardware access-list tcam region l3qos 0
hardware access-list tcam region arp-ether 256 double-wide
```

```
no hardware access-list tcam region l3qos 0
no hardware access-list tcam region arp-ether 256 double-wide
```

## Arista EOS VxLAN Routing

### IN THIS SECTION

- [Recirculation Interface for Arista Trident2 Devices | 914](#)
- [VxLAN Routing System Profile for Arista Jericho Devices | 915](#)

### Recirculation Interface for Arista Trident2 Devices

VxLAN Routing for Trident2 devices (for example, 7050QX-32) is supported but requires assigning EOS recirculation interfaces to unused physical interfaces on the device. You can use configlets to deploy this to all devices that require this configuration.

```
interface Recirc-Channel501
  switchport recirculation features vxlan
interface Ethernet35
  traffic-loopback source system device mac
  channel-group recirculation 501
interface Ethernet36
  traffic-loopback source system device mac
  channel-group recirculation 501
```

```
interface Ethernet35
```

```

no traffic-loopback source system device mac
no channel-group recirculation 501
interface Ethernet36
no traffic-loopback source system device mac
no channel-group recirculation 501
no interface Recirc-Channel501

```

### VxLAN Routing System Profile for Arista Jericho Devices

We recommend when using VxLAN Routing for Jericho devices (for example, 7280SR-48C6) that you assign EOS VxLAN Routing System Profile on the device.

Before installing the device agent, we recommend that you apply the Arista TCAM system profile during the device management setup or during Arista Zero-Touch Provisioning (ZTP). TCAM system profile requires a device reboot.

Alternatively, you can use configlets to deploy this to all devices requiring this configuration and manually reboot the devices.

```

hardware tcam
system profile vxlan-routing

```

```

hardware tcam
no system profile vxlan-routing

```

## Graph Node VTEP Types

### IN THIS SECTION

- [Unicast VTEPs | 915](#)
- [Logical VTEPs | 916](#)
- [Anycast VTEP | 918](#)

### Unicast VTEPs

Unicast VTEPs do not apply to Cumulus and Arista.

## Cisco Unicast VTEPs - Vendor Definition: Anycast VTEP

### Apstra IP Allocation

Unique per leaf in MLAG pair

Not allocated to singleton switches

### MLAG Configuration

```
interface loopback1
  IP address 10.0.0.1/32
  IP address 10.0.0.3/32 secondary
interface nve1
  source-interface loopback1
```

```
interface loopback1
  IP address 10.0.0.2/32
  IP address 10.0.0.3/32 secondary
interface nve1
  source-interface loopback1
```

### Single Switch Configuration

```
interface loopback1
  IP address 10.0.0.1/32
interface nve1
  source-interface loopback1
```

## Logical VTEPs

### Arista Logical VTEPs

#### Apstra IP Allocation

Logical VTEP configured as primary IP on loopback1 interface for both MLAG and singleton switches

All top of rack nodes share same logical VTEP IP:

- MLAG leafs share same logical VTEP IP

- Singleton leaf gets its own VTEP IP

### MLAG Configuration

```
interface loopback1
  IP address: 10.0.0.1/32
  IP address: 10.0.0.4/32 secondary
interface vxlan1
  vxlan source-interface loopback1
```

```
interface loopback1
  IP address: 10.0.0.1/32
  IP address: 10.0.0.4/32 secondary
interface vxlan1
  vxlan source-interface loopback1
```

### Single Switch Configuration

```
interface loopback1
  IP address: 10.0.0.5/32
  IP address 10.0.0.4/32 secondary
interface vxlan1
  vxlan source-interface loopback1
```

### Cumulus Logical VTEPs

#### Apstra IP Allocation

For MLAG (clagd) leafs, shared as clagd-vxlan-anycast-ip on lo interface, shared on leaf1 and leaf2

For singleton leafs, configured as additional IP alongside loopback ip on iface lo

### MLAG Configuration

```
auto lo
iface lo inet loopback
```

```
IP address: 10.0.0.6/32
clagd-vxlan-anycast-ip 10.0.0.4/32
```

```
auto lo
iface lo inet loopback
    IP address: 10.0.0.6/32
    clagd-vxlan-anycast-ip 10.0.0.4/32
```

### Single Switch Configuration

```
auto lo
iface lo inet loopback
    IP address: 10.0.0.6/32
    IP address: 10.0.0.7/32
```

### Anycast VTEP

Anycast VTEPs do not apply to Cisco and Cumulus.

### Arista Anycast VTEPs

#### Apstra IP Allocation

One anycast VTEP for entire blueprint, shared between all Arista leafs

Configured as secondary IP on loopback1 interface

### MLAG Configuration

```
interface loopback1
    IP address 10.0.0.1/32
    IP address 10.0.0.5/32 secondary
interface vxlan1
    vxlan source-interface loopback1
```

```
interface loopback1
    IP address 10.0.0.1/32
    IP address 10.0.0.5/32 secondary
```

```
interface vxlan1
  vxlan source-interface loopback1
```

Single Switch Configuration

```
interface loopback1
  IP address 10.0.0.5/32
  IP address 10.0.0.4/32 secondary
interface vxlan1
  vxlan source-interface loopback1
```

Predefined Dashboards (Analytics)

IN THIS SECTION

- [Device Health Summary Dashboard | 919](#)
- [Drain Validation Dashboard | 920](#)
- [Throughput Health MLAG Dashboard | 920](#)
- [Traffic Trends Dashboard | 920](#)
- [Virtual Infra Fabric Health Check Dashboard | 920](#)
- [Virtual Infra Redundancy Check Dashboard | 921](#)

Device Health Summary Dashboard

Ensure that the same metric is not collected twice from the same device.

Goal	Present utilization data for system CPU, system memory and maximum disk utilization of a partition on every system present
Trigger	Presence of at least one deployed system

- |                  |  |
|------------------|--|
| Widgets / Probes | <ul style="list-style-type: none"> <li>• Systems with high cpu utilization / Device System Health</li> <li>• Systems with high memory utilization / Device System Health</li> <li>• Systems with high disk utilization / Device System Health</li> </ul> |
|------------------|--|

## Drain Validation Dashboard

- |                  |  |
|------------------|--|
| Goal             | Ensure drained switches are indeed drained of traffic by ensuring total bandwidth is minimal |
| Trigger          | Presence of at least one drained switch  |
| Widgets / Probes | Drained Switches Excess Traffic / Drain Traffic Anomaly                                      |

## Throughput Health MLAG Dashboard

- |                  |  |
|------------------|--|
| Goal             | Find issues in physical infrastructure that affect the available throughput caused by issues such as imbalanced traffic over a group of L3 (ECMP) or L2 (LAG) links                                |
| Trigger          | Created on blueprints with no redundancy groups or MLAG blueprint  |
| Widgets / Probes | <ul style="list-style-type: none"> <li>• LAG Imbalance / LAG Imbalance</li> <li>• MLAG Imbalance / MLAG Imbalance</li> <li>• Fabric ECMP Imbalance / ECMP Imbalance (Fabric Interfaces)</li> </ul> |

## Traffic Trends Dashboard

- |                  |   |
|------------------|---|
| Goal             | Visualize traffic trends for general insights into fabric usage |
| Trigger          | Grouped Ingress Traffic last 1 hour / Bandwidth Utilization     |
| Widgets / Probes | Grouped Egress Traffic last 1 hour / Bandwidth Utilization      |

## Virtual Infra Fabric Health Check Dashboard

Goal	Find problems in physical or virtual infrastructure that affect workload connectivity
Trigger	Presence of at least one virtual infra manager in the blueprint
Widgets / Probes	<ul style="list-style-type: none"> <li>• Hypervisor VLANs missing in Fabric / Hypervisor &amp; Fabric VLAN Config Mismatch</li> <li>• Hypervisor PNIC LAG Status / Hypervisor &amp; Fabric LAG Config Mismatch</li> <li>• Hypervisor Low MTU anomalies / Hypervisor MTU Threshold Check</li> <li>• Critical Services affected by VLAN misconfig / VMs Without Fabric Configured VLANs</li> <li>• Hypervisor has inconsistent MTU / Hypervisor MTU Mismatch</li> </ul>

## Virtual Infra Redundancy Check Dashboard

Goal	Find single points of failure in physical or virtual infrastructure that affect high availability and available bandwidth for workloads
Trigger	Presence of at least one virtual infra manager in the blueprint
Widgets / Probes	<ul style="list-style-type: none"> <li>• Hypervisors without ToR switch redundancy / Hypervisor Redundancy Checks</li> <li>• Virtual Infra Networks without link redundancy / Hypervisor Redundancy Checks</li> </ul>

## Predefined Probes (Analytics)

### IN THIS SECTION

- [Bandwidth Utilization Probe | 923](#)
- [Critical Services: Utilization, Trending, Alerting Probe \(new in 4.0.1\) | 925](#)
- [Device System Health Probe | 926](#)
- [Device Traffic Probe | 928](#)
- [Drain Traffic Anomaly Probe | 932](#)
- [ECMP Imbalance \(External Interfaces\) Probe | 933](#)

- [ECMP Imbalance \(Fabric Interfaces\) Probe | 935](#)
- [ECMP Imbalance \(Spine to Superspine Interfaces\) Probe | 938](#)
- [ESI Imbalance Probe | 940](#)
- [EVPN VXLAN Type-3 Route Validation Probe | 942](#)
- [EVPN VXLAN Type-5 Route Validation Probe | 944](#)
- [External Routes Probe | 946](#)
- [Hot/Cold Interface Counters \(Fabric Interfaces\) Probe | 946](#)
- [Hot/Cold Interface Counters \(Specific Interfaces\) Probe | 951](#)
- [Hot/Cold Interface Counters \(Spine to Superspine Interfaces\) Probe | 953](#)
- [Hypervisor & Fabric LAG Config Mismatch Probe \(Virtual Infra\) | 955](#)
- [Hypervisor & Fabric VLAN Config Mismatch Probe \(Virtual Infra\) | 956](#)
- [Hypervisor MTU Mismatch Probe \(Virtual Infra\) | 963](#)
- [Hypervisor MTU Threshold Check Probe \(Virtual Infra\) | 964](#)
- [Hypervisor Missing LLDP Config Probe \(Virtual Infra\) | 965](#)
- [Hypervisor Redundancy Checks Probe \(Virtual Infra\) | 965](#)
- [Interface Flapping \(Fabric Interfaces\) Probe | 966](#)
- [Interface Flapping \(Specific Interfaces\) Probe | 968](#)
- [Interface Flapping \(Specific Interfaces\) Probe | 970](#)
- [Interface Policy 802.1x Probe | 972](#)
- [LAG Imbalance Probe | 973](#)
- [Leafs Hosting Critical Services: Utilization, Trending, Alerting Probe \(new in 4.0.1\) | 975](#)
- [Link Fault Tolerance in Leaf and Access LAGs Probe | 976](#)
- [MLAG Imbalance Probe | 978](#)
- [Multiagent Detector Probe | 982](#)
- [Packet Discard Percentage Probe | 983](#)
- [Spine Fault Tolerance Probe | 985](#)
- [Total East/West Traffic Probe | 986](#)
- [VMs without Fabric Configured VLANs Probe \(Virtual Infra\) | 988](#)
- [VXLAN Flood List Validation Probe | 991](#)

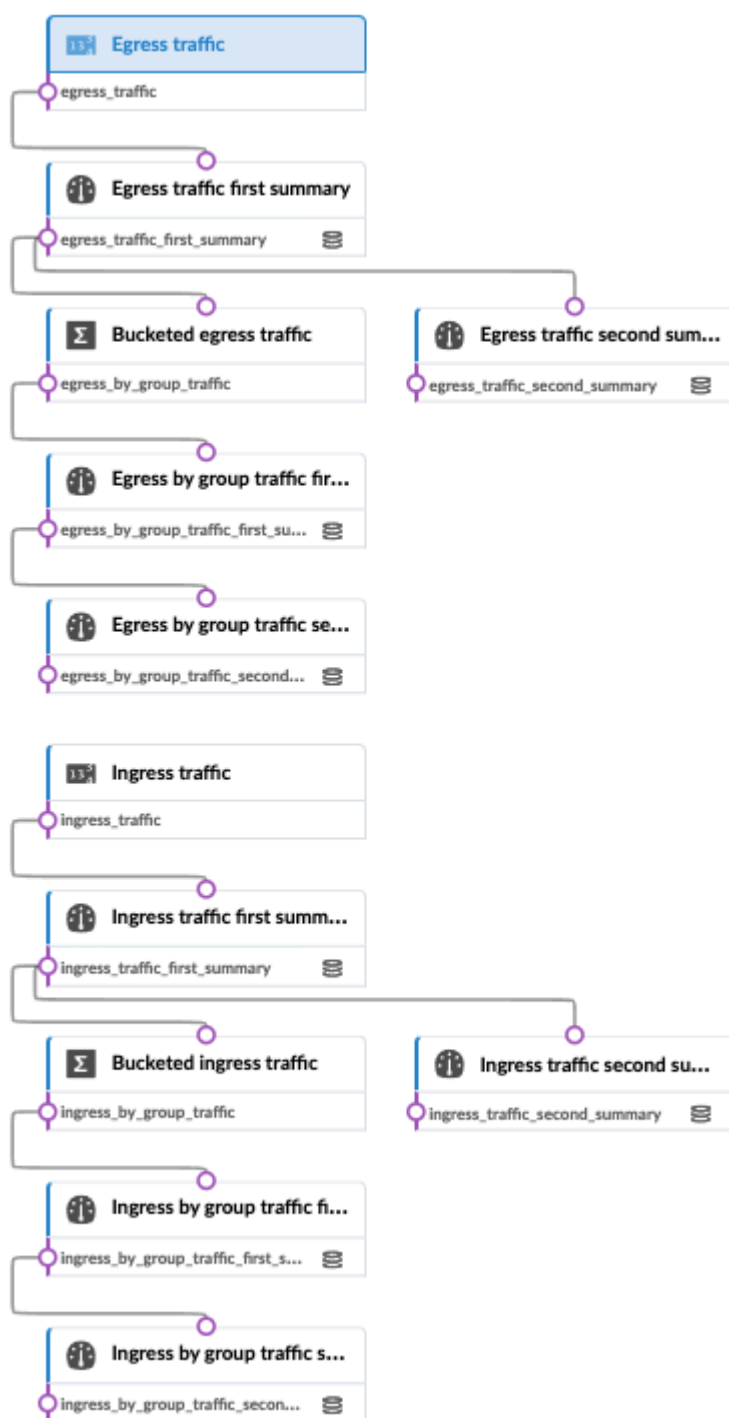
Apstra software ships with many predefined probes that you can instantiate.

## Bandwidth Utilization Probe

The bandwidth utilization probe calculates bandwidth utilization. It captures history of bandwidth utilization trends at differing levels of aggregation.

### Instantiate Predefined Probe

<div>Predefined Probe <sup>*</sup></div> <div>Bandwidth Utilization ▼</div> <div>Probe Label <sup>*</sup></div> <div>Bandwidth Utilization</div> <div>First summary average period</div> <div>2 Minutes ▼</div> <div>First summary history duration</div> <div>1 Hour ▼</div> <div>Second summary average period</div> <div>1 Hour ▼</div> <div>Second summary history duration</div> <div>30 Days ▼</div>	<div>Generate a probe to calculate bandwidth utilization</div> <div>This probe captures history of bandwidth utilization trends at differing levels of aggregation.</div>
--	---



For more information about this probe, from the blueprint, navigate to **Analytics > Probes**, click **Create Probe**, then select **Instantiate Predefined Probe** from the drop-down list. Select the probe from the **Predefined Probe** drop-down list to see details specific to the probe.

### Critical Services: Utilization, Trending, Alerting Probe (new in 4.0.1)

The critical services probe monitors critical services identified by user *tags* and provides trending data for interfaces hosting the generic systems tag. Users are proactively notified of issues from potential bandwidth contention. Additionally, historical data is persisted for trending analysis for troubleshooting or assisting in right-sizing future deployments. By default, the probe displays 1h/1d/30day average information and alerts if any individual interface with the specified tag reaches utilization threshold.

#### Instantiate Predefined Probe

**Predefined Probe \***

Critical Services: Utilization, Trending, Alerting

**Probe Label \***

Critical Services: Utilization, Trending, Alerting

**Generic System Tags**

No tags

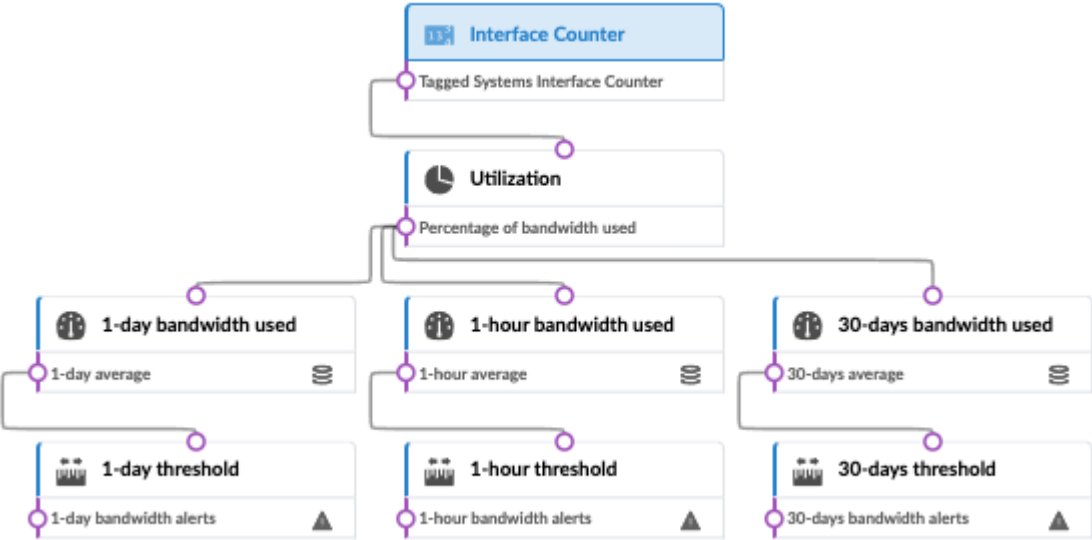
**Utilization threshold**

80

Bandwidth utilization is monitored for leaf and access switch interfaces facing generic systems that have at least one of specified tags assigned, and also for leaf interfaces facing access switches that is connected to tagged generics.

If percentage bandwidth utilization reaches the threshold, an anomaly is raised.

Monitors critical services identified by user "tags" and provides trending data for interfaces hosting the generic systems tag. Users are proactively notified of issues from potential bandwidth contention. Additionally, historical data is persisted for trending analysis for troubleshooting or assisting in right-sizing future deployments. By default, the probe will display 1h/1d/30day average information and will alert if any individual interface with the specified tag reaches utilization threshold.



For more information about this probe, from the blueprint, navigate to **Analytics > Probes**, click **Create Probe**, then select **Instantiate Predefined Probe** from the drop-down list. Select the probe from the **Predefined Probe** drop-down list to see details specific to the probe.

### **Device System Health Probe**

The device system health probe alerts if the system health parameters (CPU, memory and disk usage) exceed their specified thresholds for the specified duration.

Instantiate Predefined Probe

Predefined Probe \*

Device System Health

Probe Label \*

Device System Health

CPU utilization threshold

80

If percentage CPU utilization exceeds the threshold, an anomaly is raised

Memory utilization threshold

80

If percentage memory utilization exceeds the threshold, an anomaly is raised

Disk utilization threshold

80

If percentage disk utilization exceeds the threshold, an anomaly is raised

Duration

11 minutes

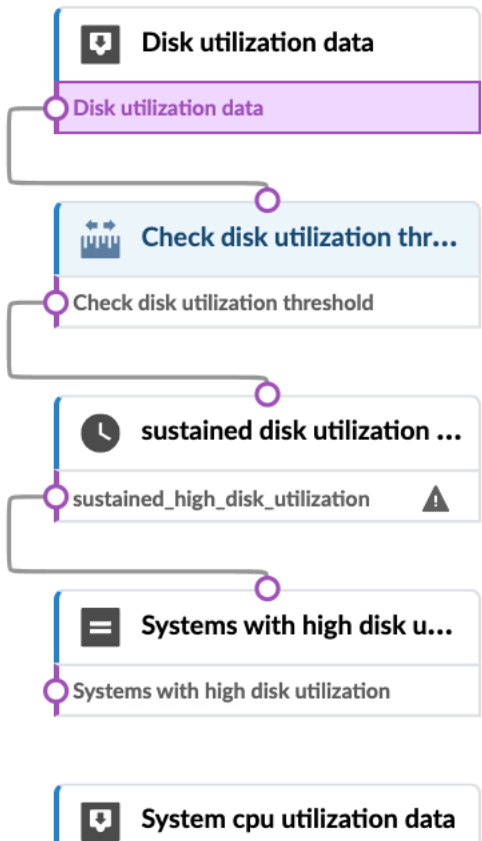
Time period in recent-history over which utilization data will be considered

Threshold Duration

6 minutes

Total amount of time in recent-history during which the utilization has to be high for anomaly to be raised

This probe alerts if the system health parameters (CPU, memory and disk usage) exceed their specified thresholds for the specified duration.



For more information about this probe, from the blueprint, navigate to **Analytics > Probes**, click **Create Probe**, then select **Instantiate Predefined Probe** from the drop-down list. Select the probe from the **Predefined Probe** drop-down list to see details specific to the probe.

## Device Traffic Probe

The device traffic probe (previously known as headroom probe) provides insights about link capacity between two points in the network. It provides multiple interface counters (rx, tx, discard, errors and so on) for all managed devices. It displays all interface counters available for the system, their utilization on a per-port and aggregated utilization per-system basis. If rules are violated, it raises anomalies.

Instantiate Predefined Probe

Predefined Probe \*

Device Traffic

Probe Label \*

Device Traffic

Interface counters average period

2 Minutes

The average period duration for interface counters

☒ Enable interface counters history

Maintain historical interface counters data

Interface counters history retention period

30 Days

Duration to maintain historical interface counters data

☒ Enable system counters history

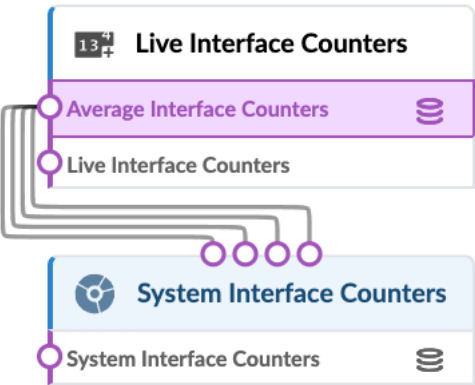
Maintain historical system interface counters data

System interface counters history retention period

30 Days

Duration to maintain historical system interface counters data

This probe displays the all the interface counters available for the system, their utilizations and utilizations aggregated on a per system basis.



**NOTE:** You can change probe inputs, but if you change the probe processors then the probe is not a **predefined** probe anymore and the traffic layer view is not available in the active topology. For more information about the traffic layer view, see "[Neighbors View](#)" on page 434.

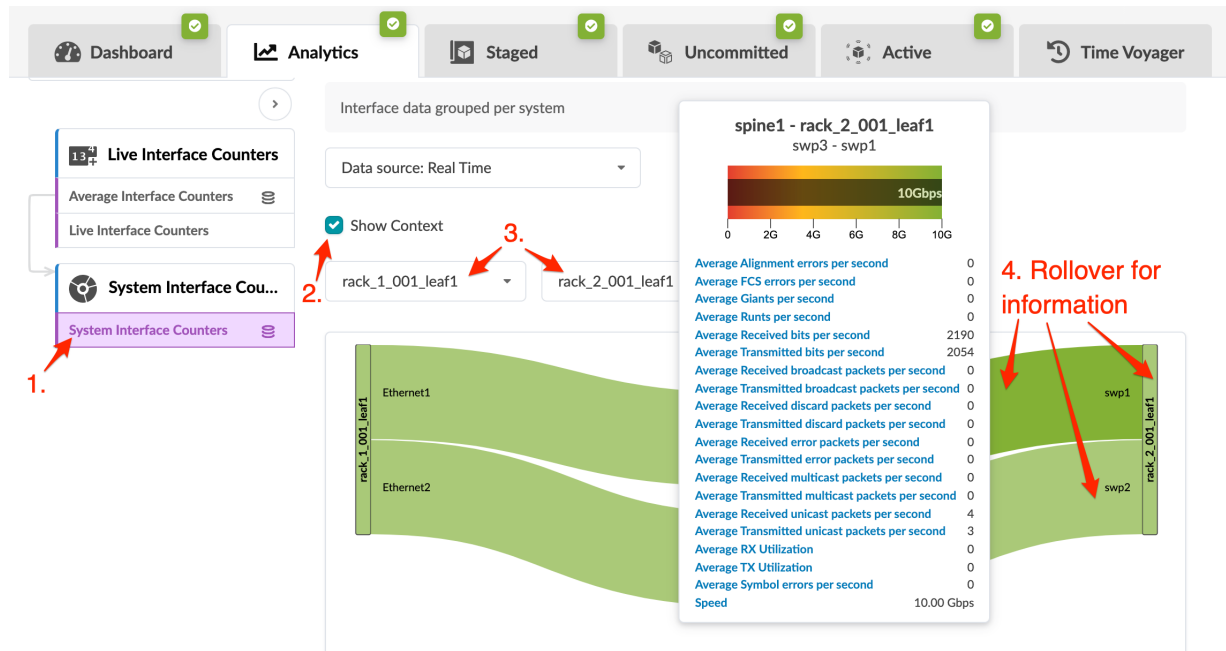
Source Processor	Live Interface Counters ( <a href="#">"Traffic Monitor" on page 1053</a> )	Purpose: Wires in Interface traffic counters every 5 seconds (by default) for all managed devices and keeps historical data based on retention period specified during probe creation.		
		Output Stages	Average Interface Counters	Set of interface counters samples, for each port of each managed device, based on specified average time with historical data.
			Live Interface Counters	Set of live interface counter samples for each port of each managed device

Additional Processor(s)	System interface counters ( <a href="#">"System Utilization" on page 1047</a> )	Purpose: This processor consumes in 'Average Interface Counters' for calculating interface counters per system with historical data. It uses properties rx_bps_average, rx_utilization_average, tx_bps_average, and tx_utilization_average to compute the system TX and RX utilization and to compute headroom between the specified source and destination systems.		
		Input Stage: Average Interface counters		

Output Stage: System Interface Counters	Set of system interface counters samples (for each device of managed devices) indicating Aggregated TX/RX, Aggregated TX/RX %, and Max interface TX/RX utilization %. The system level RX/TX calculation aggregates the Tx/RX of all the device interfaces that are "up". The max interface RX/TX calculation is the device interface with the highest Rx and the device interface with highest Tx.
---	---

To see traffic between a particular source and destination from the device traffic probe, click **System Interface Counters**, check the **Show Context** check box, then select a source and destination from the drop-down lists. Roll over different sections to display relevant information. Different colors represent link capacity, where green means plenty of capacity and red means that the link is running out of

capacity.



For more information about this probe, from the blueprint, navigate to **Analytics > Probes**, click **Create Probe**, then select **Instantiate Predefined Probe** from the drop-down list. Select the probe from the **Predefined Probe** drop-down list to see details specific to the probe.

## Drain Traffic Anomaly Probe

The drain traffic anomaly probe raises anomalies when excess traffic is on a node that is being drained.

### Instantiate Predefined Probe

Predefined Probe \*

Drain Traffic Anomaly

Probe Label \*

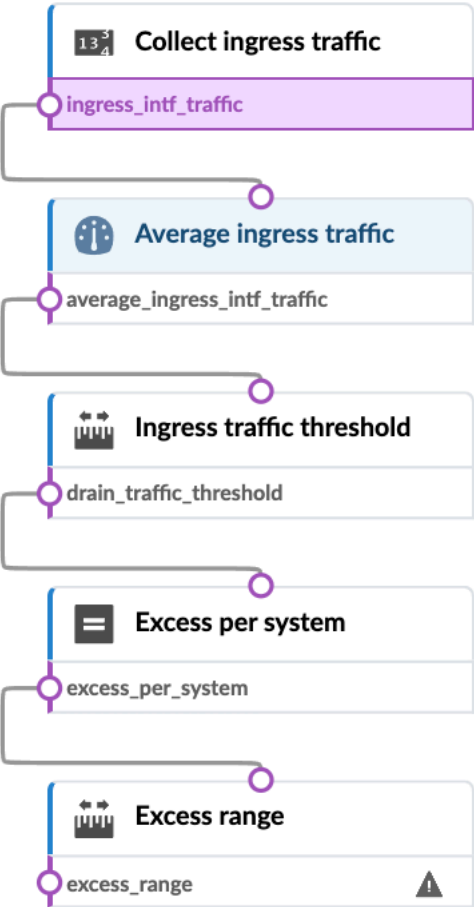
Drain Traffic Anomaly

Threshold

100000

Traffic threshold in bits per second. An anomaly will be raised if a traffic on some interface is in excess of this value.

Generate a probe to raise anomaly when there is excess traffic on a node that is being drained.



For more information about this probe, from the blueprint, navigate to **Analytics > Probes**, click **Create Probe**, then select **Instantiate Predefined Probe** from the drop-down list. Select the probe from the **Predefined Probe** drop-down list to see details specific to the probe.

### ECMP Imbalance (External Interfaces) Probe

**Purpose** This probe calculates ECMP imbalance on generic system-facing ports. The set of external-facing links (keyed by common system\_id) is determined to be imbalanced if the standard deviation of the tx\_bytes counter (averaged periodically over the specified period) for the involved interfaces is above "Max Standard Deviation". If such imbalance is observed for more than "Threshold Duration" over the last "Duration" time period, an anomaly is raised. The last "Anomaly History Count" anomaly state changes are stored for observation. If more than "Max Imbalanced Systems" systems are imbalanced, an anomaly is raised. We maintain for inspection the number of imbalanced systems over the last "System Imbalance History Count" samples.

When instantiating this probe, external router tag(s) must be specified (new in version 4.0).

<b>Source Processor</b>	<b>external interface traffic (Interface Counters)</b>	Purpose: wires in interface traffic samples (measured in transmitted bytes per second) from each interface connected to the generic systems.
		Output Stage: external_int_traffic
<b>Additional Processor(s)</b>	<b>external interface traffic avg (Periodic Average)</b>	Purpose: Calculate average traffic during period specified by average_period facade parameter. Unit is bytes per second.  Input Stage: external_int_traffic
		<b>Output Stage:</b> <b>external_int_traffic_avg</b> Set of traffic average values (for each generic system-facing interface). Each set member has the following keys to identify it: label (human-readable name of the system), system_id (id of the system, usually serial number), interface (name of the interface).
	<b>external interface std-dev (Standard Deviation)</b>	Purpose: calculate standard deviation for a set consisting of traffic averages for each generic system-facing interface on a given system. Grouping per system is achieved using 'group_by' property set to 'system_id' and 'label'.  Input Stage: external_int_traffic_avg

	<p><b>Output Stage:</b> <b>ext_int_std_dev</b></p> <p>Set of values, each indicating standard deviation (as a measure of ECMP imbalance) for traffic averages for each generic system-facing interface on a given system. Each set member has 'system_id' and 'label' key to identify system whose ECMP imbalance the value represents.</p>
<p><b>std-dev percentage (Ratio)</b></p>	<p>Input Stage: ext_int_std_dev</p> <p>Output Stage: std_dev_percentage</p>
<p><b>live ecmp imbalance (Range)</b></p>	<p>Purpose: Evaluate if standard deviation between generic system-facing interfaces on each system is within acceptable range. In this case acceptable range is between 0 and std_max facade parameter (in bytes per second unit).</p> <p>Input Stage: std_dev_percentage</p> <p><b>Output Stage:</b> <b>live_ecmp_imbalance</b></p> <p>Set of true/false values, each indicating if standard deviation (as a measure of ECMP imbalance) for traffic averages for each external router-facing interface on a given leaf is within acceptable range. Each set member has system_id key to identify system whose ECMP imbalance the value represents.</p>
<p><b>links imbalanced percentage (Match Percentage)</b></p>	<p>Input Stage: live_ecmp_imbalance</p> <p>Output Stage: links_imbalanced_percentage</p>
<p><b>systems imbalanced (Range)</b></p>	<p>Input Stage: links_imbalanced_percentage</p> <p>Output Stage: systems_imbalanced</p>
<p><b>sustained ecmp imbalance (Time in State)</b></p>	<p>Purpose: Evaluate if standard deviation between generic system-facing interfaces on each leaf has been outside acceptable range, (as defined by 'live ecmp imbalance' processor) for more than 'threshold_duration' seconds during last 'total_duration' seconds. These two parameters are part of facade specification.</p> <p>Input Stage: systems_imbalanced</p>

	<p><b>Output Stage:</b> <b>sustained_ecmp_imbalance</b></p> <p>Set of true/false values, each indicating if standard deviation (as a measure of ECMP imbalance) for traffic averages for each external router-facing interface on a given system has been outside acceptable range for more than specified period of time. Each set member has system_id key to identify system whose ECMP imbalance the value represents.</p>
<p><b>systems imbalanced count (Match Count)</b></p>	<p>Purpose: Count how many systems have external ecmp imbalance anomaly true at any instant in time.</p> <p>Input Stage: sustained_ecmp_imbalance</p> <p><b>Output Stage:</b> <b>system_tx_imbalance_count</b></p> <p>Number of systems with external ecmp imbalance.</p>
<p><b>live system imbalanced (Range)</b></p>	<p>Purpose: Evaluate if the number of imbalanced systems is within acceptable range, which in this instance means less than 'max_systems_imbalanced' value which is a facade parameter</p> <p>Input Stage: system_tx_imbalance_count</p> <p><b>Output Stage:</b> <b>live_system_imbalance_count</b></p> <p>Boolean indicating if the number of imbalanced systems is within accepted range, i.e. less than 'max_systems_imbalanced' which is a facade parameter</p>

For more information about this probe, from the blueprint, navigate to **Analytics > Probes**, click **Create Probe**, then select **Instantiate Predefined Probe** from the drop-down list. Select the probe from the **Predefined Probe** drop-down list to see details specific to the probe.

### ECMP Imbalance (Fabric Interfaces) Probe

**Purpose** This probe calculates ECMP imbalance on fabric ports.

A given set of ECMP links (only calculated on leaf-to-spine links), identified by common system\_id, is determined to be imbalanced if the standard-deviation of the tx\_bytes counter (averaged periodically over the specified period) for the involved leaf-interfaces is above "Max Standard Deviation".

If such imbalance is observed for more-than "Threshold Duration" over the last "Duration" time period, we raise an anomaly.

The last "Anomaly History Count" anomaly state-changes are stored for observation.

If more-than "Max Imbalanced Systems" systems are imbalanced, we raise a distinct anomaly.

We maintain for inspection the number of imbalanced systems over the last "System Imbalance History Count" samples.

<b>Source Processor</b>	<b>leaf fabric interface traffic (Interface Counters-)</b>	Purpose: wires in interface traffic samples (measured in bytes per second) from each spine-facing interface on each leaf.	<b>Output Stage:</b> <b>leaf_fabric_int_traffic</b> Set of traffic samples (for each spine-facing interface on each leaf). Each set member has the following keys to identify it: label (human-readable name of the leaf), system_id (id of the leaf system, usually serial number), interface (name of the interface).
<b>Additional Processor(s)</b>	<b>leaf fabric interface traffic avg (Periodic Average)</b>	Purpose: Calculate average traffic during period specified by average_period facade parameter. Unit is bytes per second.	<b>Input Stage:</b> leaf_fabric_int_traffic  <b>Output Stage:</b> <b>leaf_fabric_int_tx_avg</b> Set of traffic average values (for each spine-facing interface on each leaf). Each set member has the following keys to identify it: label (human-readable name of the leaf), system_id (id of the leaf system, usually serial number), interface (name of the interface).
	<b>leaf fabric interface std-dev (Standard Deviation)</b>	Purpose: calculate standard deviation for a set consisting of traffic averages for each spine-facing interface on a given leaf. Grouping per leaf is achieved using 'group_by' property set to 'system_id'.	<b>Input Stage:</b> leaf_fabric_int_tx_avg  <b>Output Stage:</b> <b>leaf_fab_int_std_dev</b> Set of values, each indicating standard deviation (as a measure of ECMP imbalance) for traffic averages for each spine-facing interface on a given leaf. Each set member has

system\_id key to identify leaf whose ECMP imbalance the value represents.

**std-dev  
percentage  
(Ratio)**

Input Stage: leaf\_fab\_int\_std\_dev

Output Stage: std\_dev\_percentage

**live ecmp  
imbalance  
(Range)**

Purpose: Evaluate if standard deviation between spine-facing interfaces on each leaf is within acceptable range. In this case acceptable range is between 0 and std\_max facade parameter (in bytes per second unit).

Input Stage: std\_dev\_percentage

**Output Stage:  
live\_ecmp\_imbalance**

Set of true/false values, each indicating if standard deviation (as a measure of ECMP imbalance) for traffic averages for each spine-facing interface on a given leaf is within acceptable range. Each set member has system\_id key to identify leaf whose ECMP imbalance the value represents.

**sustained  
ecmp  
imbalance  
(Time in  
State)**

Purpose: Evaluate if standard deviation between spine-facing interfaces on each leaf has been outside acceptable range, (as defined by 'live ecmp imbalance' processor) for more than 'threshold\_duration' seconds during last 'total\_duration' seconds. These two parameters are part of facade specification.

Input Stage: live\_ecmp\_imbalance

Output Stage: system\_imbalance

**systems  
imbalanced  
count (Match  
Count)**

Purpose: Count how many systems have ecmp imbalance anomaly true at any instant in time.

Input Stage: system\_imbalance

**Output Stage: system\_imbalance\_count**

Number of systems with ecmp imbalance.

**imbalanced  
system count  
out of range  
(Range)**

Purpose: Evaluate if the number of imbalanced systems is within acceptable range, which in this instance means less than 'max\_systems\_imbalanced' value which is a facade parameter.

Input Stage: system\_imbalanced\_count

<b>Output Stage:</b> <b>imbalanced_system_count_out_of_range</b>	Boolean indicating if the number of imbalanced systems is within accepted range, i.e. less than 'max_systems_imbalanced" which is a facade parameter.
---	---

For more information about this probe, from the blueprint, navigate to **Analytics > Probes**, click **Create Probe**, then select **Instantiate Predefined Probe** from the drop-down list. Select the probe from the **Predefined Probe** drop-down list to see details specific to the probe.

### ECMP Imbalance (Spine to Superspine Interfaces) Probe

The ECMP imbalance (spine to superspine interfaces) probe calculates ECMP imbalance on spine-to-superspine ports. A given set of ECMP links (only calculated on spine-to-superspine links), identified by common system\_id, is determined to be imbalanced if the standard-deviation of the tx\_bytes counter (averaged periodically over the specified period) for the involved spine interfaces is above "Max Standard Deviation". If such imbalance is observed for more-than "Threshold Duration" the last "Duration" period, we raise an anomaly. The last "Anomaly History Count" anomaly state-changes are stored for observation. If more-than "Max Imbalanced Systems" systems are imbalanced, we raise a distinct anomaly. We maintain for inspection the number of imbalanced systems over the last "System Imbalance History Count" samples.

Instantiate Predefined Probe

Predefined Probe \*

ECMP Imbalance (Spine to Superspine Interfaces) ▾

Probe Label \*

ECMP Imbalance (Spine to Superspine Interfaces)

Max Standard Deviation

20

Maximum standard deviation in bps across a set of ECMP paths on a given system (in percents of link bandwidth). If this standard deviation is exceeded, we consider that system to be imbalanced

Average Period

30 seconds ▾

Period over which to average input bps counter samples

Threshold Duration

2 minutes 10 seconds ▾

Total amount of time in recent-history during which set of ECMP links must be unbalanced for anomaly to be raised

Duration

5 Minutes ▾

Time period in recent-history over which we will consider ECMP imbalance

Max Imbalanced Systems

1

If this number of total imbalanced systems is exceeded, an anomaly is raised

Generate a probe to calculate ECMP imbalance on spine to superspine ports.

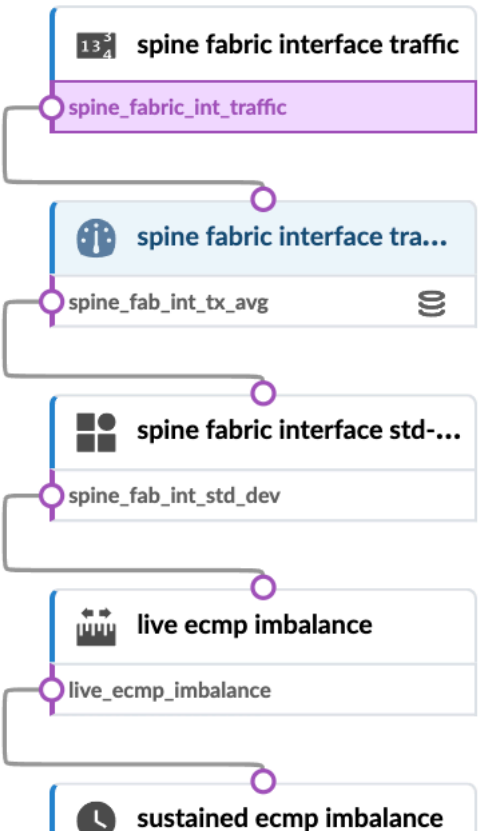
A given set of ECMP links (only calculated on spine to superspine links), identified by common system\_id, is determined to be imbalanced if the standard-deviation of the tx\_bytes counter (averaged periodically over the specified period) for the involved spine interfaces is above "Max Standard Deviation".

If such imbalance is observed for more-than "Threshold Duration" the last "Duration" period, we raise an anomaly.

The last "Anomaly History Count" anomaly state-changes are stored for observation.

If more-than "Max Imbalanced Systems" systems are imbalanced, we raise a distinct anomaly.

We maintain for inspection the number of imbalanced systems over the last "System Imbalance History Count" samples.



For more information about this probe, from the blueprint, navigate to **Analytics > Probes**, click **Create Probe**, then select **Instantiate Predefined Probe** from the drop-down list. Select the probe from the **Predefined Probe** drop-down list to see details specific to the probe.

### **ESI Imbalance Probe**

The ESI imbalance probe calculate ESI imbalance. It calculates the standard deviation across links for all ESIs in the network. If any are over the specified threshold in the last specified time period, an anomaly is raised. It also calculates percentage of ESIs in each rack in this state.

Instantiate Predefined Probe

Predefined Probe \*

ESI Imbalance

Probe Label \*

ESI Imbalance

Max Standard Deviation

20

Maximum standard deviation used for imbalance detection (in percents of link bandwidth).

Duration

1 Minute

Time period in recent-history over which average traffic will be considered

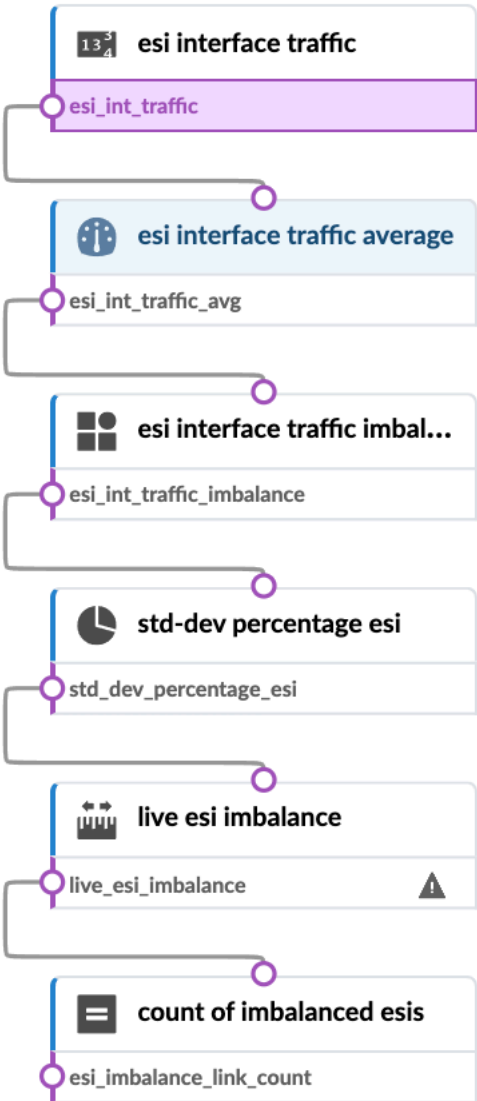
History duration

1 Hour

Time period during which the data of deviation will be retained

Generate a probe to calculate ESI imbalance

Calculates std deviation across links for all ESIs in the network. If any are over the specified threshold in the last specified time period, an anomaly is raised. Also calculates percentage of ESIs in each rack in this state.



For more information about this probe, from the blueprint, navigate to **Analytics > Probes**, click **Create Probe**, then select **Instantiate Predefined Probe** from the drop-down list. Select the probe from the **Predefined Probe** drop-down list to see details specific to the probe.

## EVPN VXLAN Type-3 Route Validation Probe

The EVPN VXLAN Type-3 route validation probe validates EVPN Type-3 routes on every leaf in the network. It collects appropriate telemetry data, compares it to the set of Type-3 routes expected to be present and alerts if expected routes are missing on any device.

You can configure the following parameters:

- **Probe Label:** Name to identify the probe.
- **Anomaly Time Window :** Average period duration for interface counters.
- **Anomaly Threshold (in %):** If routes are missing for more than or equal to percentage of Anomaly Time Window, an anomaly is raised. If Anomaly Time Window ATW, and Anomaly Threshold is AT. It calculates  $Z = (ATW * AT)/100$  in seconds. E.g. If ATW = 20 seconds, AT = 5%, then  $Z = (20 * 5)/100 = 1$  second. When the route is in Missing state for Z seconds from total ATW duration, anomaly is raised.
- **Collection period:** All these probes are polling-based so they have a polling period.
- **Monitored VNs:** Specify the virtual networks to be monitored. Either list of desired VN's e.g. "1-3,6,8,10-13" or " \* " to monitor all virtual networks.

The route labels include the following:

- **Expected:** This route is expected on the device as per service defined.
- **Missing:** This route is missing on the device when compared to the expected route set.
- **Unexpected:** There are no expectations rendered (by AOS) for this route.

This probe is created with an empty **Monitored VNs** (monitored\_vn) list, which means that the probe does not monitor any virtual networks by default. When you instantiate this probe you must specify a list of virtual networks (up to ten) for which routes are collected, or you can specify " \* " in which case all virtual networks are monitored.



**CAUTION:** Specifying " \* " in the **Monitored VNs** field may result in high cpu/memory/ network I/O overhead associated with BGP routing table iteration on the device side.

Instantiate Predefined Probe

Predefined Probe \*

EVPN VXLAN Type-3 Route Validation

Probe Label \*

EVPN VXLAN Type-3 Route Validation

Anomaly Time Window

11 minutes

Anomaly Threshold (in %)

100

If routes are missing for more than or equal to percentage of Anomaly Time Window, an anomaly will be raised.

Collection period

10 Minutes

Telemetry collection interval.

Monitored VNs

What VNs are to be monitored. Specify "" to monitor all the VNs or list the desired ones, e.g. "1-3,6,8,10-13". Number of VNs can not be more than 10.

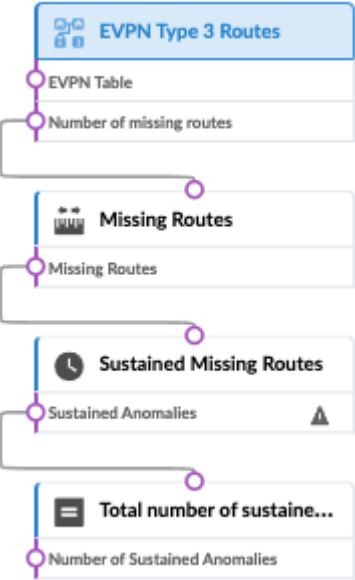
This probe validates EVPN Type-3 routes on every leaf in the network. It collects appropriate telemetry data, compares it to the set of Type-3 routes expected to be present and alerts if expected routes are missing on any device.

Route Labels

Expected: This route is expected on the device as per service defined.

Missing: This route is missing on the device when compared to the expected route set.

Unexpected: There are no expectations rendered (by AOS) for this route.



**NOTE:** Auto-enabling the **EVPN VXLAN Route Summary** analytics dashboard enables the **EVPN VXLAN Type-3 Route Validation** and **EVPN Flood List Validation** probes automatically (but not the EVPN VXLAN Type-5 Route Validation probe). See [Configuring Auto-Enabled Dashboards<configure\\_dashboard>](#) for information about enabling the dashboard.

For more information about this probe, from the blueprint, navigate to **Analytics > Probes**, click **Create Probe**, then select **Instantiate Predefined Probe** from the drop-down list. Select the probe from the **Predefined Probe** drop-down list to see details specific to the probe.

## EVPN VXLAN Type-5 Route Validation Probe

The EVPN VXLAN Type-5 route validation probe validates the EVPN Type 5 routes on every leaf. The collected data is matched against the graph data to ascertain any missing routes on any system.

You can configure the following parameters:

- **Probe Label:** Name to identify the probe.
- **Anomaly Time Window :** Average period duration for interface counters.
- **Anomaly Threshold (in %):** If routes are missing for more than or equal to percentage of Anomaly Time Window, an anomaly is raised. If Anomaly Time Window ATW, and Anomaly Threshold is AT. It calculates  $Z = (ATW * AT)/100$  in seconds. E.g. If ATW = 20 seconds, AT = 5%, then  $Z = (20 * 5)/100 = 1$  second. When the route is in Missing state for Z seconds from total ATW duration, anomaly is raised.
- **Collection period:** All these probes are polling-based so they have a polling period.

The route labels include the following:

- **Expected:** This route is expected on the device as per service defined.
- **Missing:** This route is missing on the device when compared to the expected route set.
- **Unexpected:** There are no expectations rendered (by AOS) for this route.

If this probe is enabled it monitors all virtual networks from all devices. It does not provide the “monitored VN list” configuration option like the VXLAN Type-3 probe does.

Instantiate Predefined Probe

**Predefined Probe** \*

EVPN VXLAN Type-5 Route Validation

**Probe Label** \*

EVPN VXLAN Type-5 Route Validation

**Anomaly Time Window**

11 minutes

**Anomaly Threshold (in %)**

100

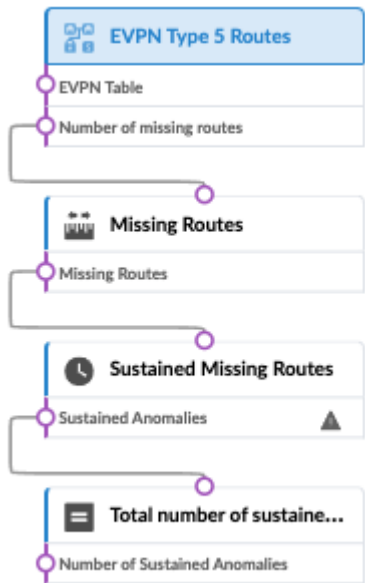
If routes are missing for more than or equal to percentage of Anomaly Time Window, an anomaly will be raised.

**Collection period**

10 Minutes

Telemetry collection interval.

This probe validates the EVPN Type 5 routes on every leaf. The collected data is matched against the graph data to ascertain any missing routes on any system.



**NOTE:** Auto-enabling the **EVPN VXLAN Route Summary** analytics dashboard enables the **EVPN VXLAN Type-3 Route Validation** and **EVPN Flood List Validation** probes automatically (but not

the EVPN VXLAN Type-5 Route Validation probe). See [Configuring Auto-Enabled Dashboards<configure\\_dashboard>](#) for information about enabling the dashboard.

For more information about this probe, from the blueprint, navigate to **Analytics > Probes**, click **Create Probe**, then select **Instantiate Predefined Probe** from the drop-down list. Select the probe from the **Predefined Probe** drop-down list to see details specific to the probe.

## External Routes Probe

**Purpose** The External Routes probe automatically activates the collection of received or advertised routes across all BGP sessions established with generic systems into a single stage output table (mixing received, used and advertised routes). This probe assists with troubleshooting external network connectivity problems.

**Parameters** The External Routes probe parameters below can be configured at time of creation or anytime afterwards.

AFI: Address Family Identifiers - IPv4 or IPv6

Type: advertised-routes or received-routes

Routing Zone (VRF): All or specific name

Prefix: Only routes matching the prefix

Filter options: exact or longer

More-specific prefixes mask: Match more-specific prefixes from a parent prefix, up until le\_mask prefix length.

Less-specific prefixes mask: Match less-specific prefixes from a parent prefix, up from ge\_mask to the prefix length of the route.

For more information about this probe, from the blueprint, navigate to **Analytics > Probes**, click **Create Probe**, then select **Instantiate Predefined Probe** from the drop-down list. Select the probe from the **Predefined Probe** drop-down list to see details specific to the probe.

## Hot/Cold Interface Counters (Fabric Interfaces) Probe

**Purpose** This probe determines hot/cold interface counters. It determines if interface counters are hot (too high) or cold (too low). A given interface (considering only leaf fabric interfaces) is considered to be in a hot state if its average counter value is greater than "Max". A given interface (considering only leaf fabric interfaces) is considered to be in a cold state if its average counter value is less than "Min". If such undesired state is observed for more-

than "Threshold Duration" over the last "Duration" period, an anomaly is raised. Distinct anomalies are raised for hot and cold states. If more than "Max Hot Interface Percentage" percent of interfaces on a given device are hot, we raise an anomaly. If more than "Max Cold Interface Percentage" percent of interfaces on a given device are cold, we raise an anomaly. Finally, the last "Anomaly History Count" anomaly state-changes are stored for observation.

<b>Source Processor</b>	<b>leaf interface traffic (Interface Counters)</b>	Purpose: wires in interface traffic samples (measured in bytes per second) from each spine facing interface on each leaf.
	<b>Output Stage: leaf_int_traffic</b>	Set of traffic samples (for each spine-facing interface on each leaf). Each set member has the following keys to identify it: system_id (id of the leaf system, usually serial number), interface (name of the interface), role (role of the interface, such as 'fabric').
<b>Additional Processor(s)</b>	<b>leaf interface tx avg (Periodic Average)</b>	Purpose: Calculate average traffic during period specified by average_period facade parameter. Unit is bytes per second.  Input Stage: leaf_int_traffic
	<b>Output Stage: leaf_int_tx_avg</b>	Set of traffic average values (for each spine-facing interface on each leaf). Each set member has the following keys to identify it: system_id (id of the leaf system, usually serial number), interface (name of the interface), role (role of the interface, such as 'fabric').
	<b>interface sum per device (Sum)</b>	Purpose: Sum average traffic for all interface under consideration per device.  Input Stage: leaf_int_tx_avg
	<b>Output Stage: if_counter_sum_per_device</b>	Set of numbers, each indicating the total average traffic for all interface under consideration per device, expressed in bytes per second. Each set member has the following key to identify it: system_id (id of the leaf system, usually serial number).
	<b>interface sum per device</b>	Purpose: Sum average traffic for all interface under consideration per device, per interface role.

<b>per link role (Sum)</b>	Input Stage: leaf_int_tx_avg	<b>Output Stage:</b> <b>if_counter_sum_per_device_role</b> Set of numbers, each indicating the total average traffic for all interface under consideration per device, expressed in bytes per second. Each set member has the following keys to identify it: system_id (id of the leaf system, usually serial number), role (role of the interface, such as 'fabric').
<b>live leaf interface cold (Range)</b>	Purpose: Evaluate if the average traffic on spine facing interfaces on each leaf is within acceptable range. In this case acceptable range means larger than min facade parameter (in bytes per second unit).  Input Stage: leaf_int_tx_avg	<b>Output Stage:</b> <b>live_leaf_int_cold</b> Set of true/false values, each indicating if traffic averages for each spine-facing interface on each leaf is within acceptable range. Each set member has the following keys to identify it: system_id (id of the leaf system, usually serial number), interface (name of the interface) role (role of the interface, such as 'fabric'). Samples unit is bytes per second.
<b>live leaf interface hot (Range)</b>	Purpose: Evaluate if the average traffic on spine-facing interfaces on each leaf is within acceptable range. In this case acceptable range is between 0 and max facade parameter (in bytes per second unit).  Input Stage: leaf_int_tx_avg	<b>Output Stage:</b> <b>live_leaf_int_hot</b> Set of true/false values, each indicating if traffic averages for each spine-facing interface on each leaf is within acceptable range. Each set member has the following keys to identify it: system_id (id of the leaf system, usually serial number), interface (name of the interface) role (role of the interface, such as 'fabric'). Samples unit is bytes per second.
<b>sustained cold leaf interface (Time in State)</b>	Purpose: Evaluate if the average traffic spine facing interfaces on each leaf has been outside acceptable range, (as defined by 'live leaf interface cold' processor) for more than 'threshold_duration' seconds during the	

last 'total\_duration' seconds. These two parameters are part of facade specification.

Input Stage: live\_leaf\_int\_cold

**Output Stage:** Set of true/false values, each indicating if the traffic average for each spine-facing interface on each leaf has been in 'cold' range for more than specified period of time. Each set member has the following keys to identify it: system\_id (id of the leaf system, usually serial number), interface (name of the interface) role (role of the interface, such as 'fabric'). Samples unit is bytes per second.

**sustained hot leaf interface (Time in State)** Evaluate if the average traffic spine facing interfaces on each leaf has been outside acceptable range, (as defined by 'live leaf interface hot' processor) for more than 'threshold\_duration' seconds during the last 'total\_duration' seconds. These two parameters are part of facade specification.

Input Stage: live\_leaf\_int\_hot

**Output Stage:** Set of true/false values, each indicating if the traffic average for each spine-facing interface on each leaf has been in 'hot' range for more than specified period of time. Each set member has the following keys to identify it: system\_id (id of the leaf system, usually serial number), interface (name of the interface) role (role of the interface, such as 'fabric'). Samples unit is bytes per second.

**system percent cold (Match Percentage)** Purpose: Calculate percentage of interfaces that are cold on any given device under consideration.

Input Stage: cold\_leaf\_int

**Output Stage:** Set of numbers, each indicating the the percentage of cold interfaces on any given device under consideration. Each set member has the following key to identify it: system\_id (id of the leaf system, usually serial number).

<b>system percent hot (Match Percentage)</b>	<p>Purpose: Calculate percentage of interfaces that are hot on any given device under consideration.</p> <p>Input Stage: hot_leaf_int</p> <p><b>Output Stage:</b> <b>system_perc_hot</b> Set of numbers, each indicating the the percentage of hot interfaces on any given device under consideration. Each set member has the following key to identify it: system_id (id of the leaf system, usually serial number).</p>
<b>device cold (Range)</b>	<p>Purpose: Evaluate if the percentage of cold interfaces on a specific device is outside the acceptable range, where acceptable range in his case means less than 'max_cold_interface_percentage', which is a facade parameter.</p> <p>Input Stage: system_perc_cold</p> <p><b>Output Stage:</b> <b>device_cold_anomalous</b> Set of boolean values, each indicating if the the percentage of cold interfaces on any given device was out of acceptable range. Each set member has the following key to identify it: system_id (id of the leaf system, usually serial number).</p>
<b>device hot (Range)</b>	<p>Purpose: Evaluate if the percentage of hot interfaces on a specific device is outside the acceptable range, where acceptable range in his case means less than 'max_hot_interface_percentage', which is a facade parameter.</p> <p>Input Stage: system_perc_hot</p> <p><b>Output Stage:</b> <b>device_hot_anomalous</b> Set of boolean values, each indicating if the the percentage of hot interfaces on any given device was out of acceptable range. Each set member has the following key to identify it: system_id (id of the leaf system, usually serial number).</p>

For more information about this probe, from the blueprint, navigate to **Analytics > Probes**, click **Create Probe**, then select **Instantiate Predefined Probe** from the drop-down list. Select the probe from the **Predefined Probe** drop-down list to see details specific to the probe.

## Hot/Cold Interface Counters (Specific Interfaces) Probe

The hot/cold interface counters (specific interfaces) probe determines hot/cold specific interface counters. It determines if interface counters averaged over "Average Period" are hot (too high) or cold (too low). A given interface (out of the specified list) is considered to be in a hot state if its average counter value is greater than "Max". A given interface (out of the specified list) is considered to be in a cold state if its average counter value is less than "Min". If such undesired state is observed for more-than "Threshold Duration" over the last "Duration" time period, we raise an anomaly. Distinct anomalies are raised for hot and cold states. If more than "Max Hot Interface Percentage" percent of interfaces on a given device are hot, we raise an anomaly. If more than "Max Cold Interface Percentage" percent of interfaces on a given device are cold, we raise an anomaly. Finally, the last "Anomaly History Count" anomaly state-changes are stored for observation.

Instantiate Predefined Probe

Predefined Probe \*

Hot/Cold Interface Counters (Specific Interfaces) ▾

Probe Label \*

Hot/Cold Interface Counters (Specific Interfaces)

Interfaces \*

No interfaces specified.

+ Add Interface

Counter Type \*

▾

A type of an interface counter.

Min

0

Minimum level of counter

Max

10

Maximum level of counter

Max Cold Interface Percentage

30

Maximum percentage of cold interfaces on a device

Max Hot Interface Percentage

30

Maximum percentage of hot interfaces on a device

Average Period

1 Minute ▾

Period over which to average input counter samples

Threshold Duration

10 seconds ▾

Total amount of time in recent-history during which interface must be hot/cold for anomaly to be raised

Duration

1 Minute ▾

Time period in recent-history over which interface counter hot/cold status will be considered

Generate a probe to determine hot/cold specific interface counters

This probe determines if interface counters averaged over "Average Period" are hot (too high) or cold (too low).

A given interface (out of the specified list) is considered to be in a hot state if its average counter value is greater than "Max"

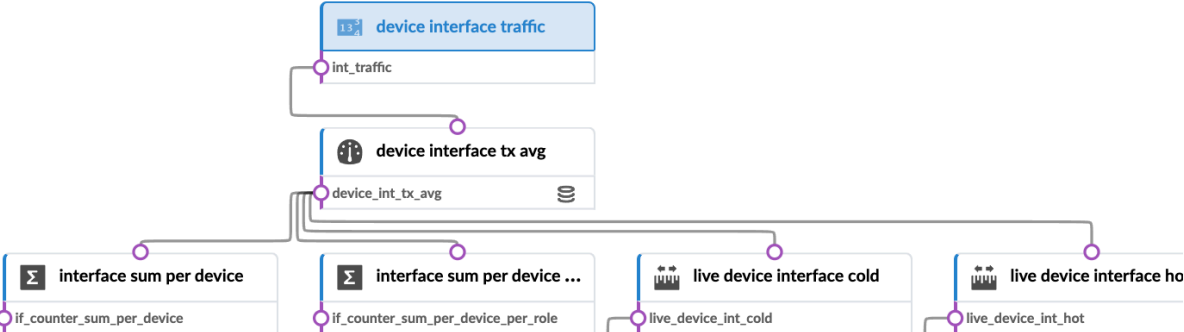
A given interface (out of the specified list) is considered to be in a cold state if its average counter value is less than "Min"

If such undesired state is observed for more-than "Threshold Duration" over the last "Duration" time period, we raise an anomaly. Distinct anomalies are raised for hot and cold states.

If more than "Max Hot Interface Percentage" percent of interfaces on a given device are hot, we raise an anomaly.

If more than "Max Cold Interface Percentage" percent of interfaces on a given device are cold, we raise an anomaly.

Finally, the last "Anomaly History Count" anomaly state-changes are stored for observation.



For more information about this probe, from the blueprint, navigate to **Analytics > Probes**, click **Create Probe**, then select **Instantiate Predefined Probe** from the drop-down list. Select the probe from the **Predefined Probe** drop-down list to see details specific to the probe.

### **Hot/Cold Interface Counters (Spine to Superspine Interfaces) Probe**

The hot/cold interface counters (spine-to-superspine interfaces) probe calculates ECMP imbalance on spine-to-superspine ports. A given set of ECMP links (only calculated on spine-to-superspine links), identified by common system\_id, is determined to be imbalanced if the standard-deviation of the tx\_bytes counter (averaged periodically over the specified period) for the involved spine interfaces is above "Max Standard Deviation". If such an imbalance is observed for more-than "Threshold Duration" the last "Duration" period, we raise an anomaly. The last "Anomaly History Count" anomaly state-changes are stored for observation. If more-than "Max Imbalanced Systems" systems are imbalanced, we raise a distinct anomaly. We maintain for inspection the number of imbalanced systems over the last "System Imbalance History Count" samples.

Instantiate Predefined Probe

Predefined Probe \*

Hot/Cold Interface Counters (Spine to Superspine Interfaces) ▾

Probe Label \*

Hot/Cold Interface Counters (Spine to Superspine Interfaces)

Counter Type \*

▾

A type of an interface counter.

Min

0

Minimum level of counter

Max

10

Maximum level of counter

Max Cold Interface Percentage

30

Maximum percentage of cold interfaces on a device

Max Hot Interface Percentage

30

Maximum percentage of hot interfaces on a device

Average Period

1 Minute ▾

Period over which to average input counter samples

Threshold Duration

10 seconds ▾

Total amount of time in recent-history during which interface must be hot/cold for anomaly to be raised

Duration

1 Minute ▾

Time period in recent-history over which interface counter hot/cold status will be considered

Generate a probe to determine hot/cold spine to superspine interface counters.

This probe determines if interface counters are hot (too high) or cold (too low).

A given interface (considering only spine to superspine interfaces) is considered to be in a hot state if its average counter value is greater than "Max"

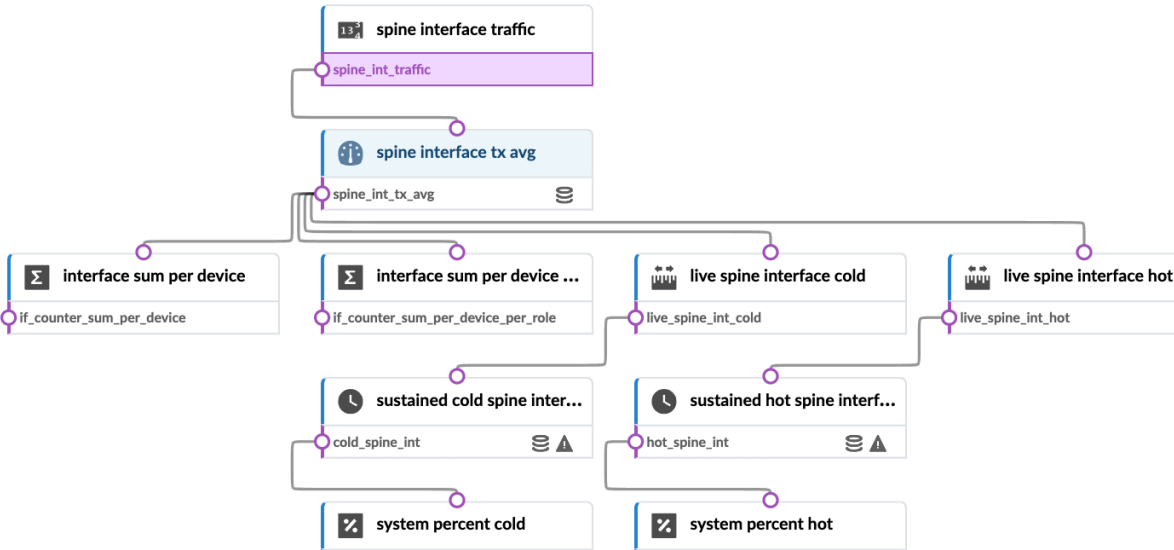
A given interface (considering only spine to superspine interfaces) is considered to be in a cold state if its average counter value is less than "Min"

If such undesired state is observed for more-than "Threshold Duration" over the last "Duration" time period, we raise an anomaly. Distinct anomalies are raised for hot and cold states.

If more than "Max Hot Interface Percentage" percent of interfaces on a given device are hot, we raise an anomaly.

If more than "Max Cold Interface Percentage" percent of interfaces on a given device are cold, we raise an anomaly.

Finally, the last "Anomaly History Count" anomaly state-changes are stored for observation.



For more information about this probe, from the blueprint, navigate to **Analytics > Probes**, click **Create Probe**, then select **Instantiate Predefined Probe** from the drop-down list. Select the probe from the **Predefined Probe** drop-down list to see details specific to the probe.

### Hypervisor & Fabric LAG Config Mismatch Probe (Virtual Infra)

**Purpose** Detect inconsistent LAG configs between fabric and virtual infra and calculate LAGs missing on hypervisors and managed leafs connected to hypervisors.

**Source Processor** **Hypervisor NICs with LAG (generic graph collector)** output stage: Hypervisor NICs LAG Intent Status (discrete state set) (generated from graph)

**Additional Processor(s)** **Hypervisor NIC LAG anomalies (state)** input stage: Hypervisor NICs LAG Intent Status  
output stage: Hypervisor NIC LAG Mismatch Anomaly (discrete state set)

**Example Usage** **vSphere Integration** - This probe detects inconsistent LAG configs between fabric LAG dual-leafs and ESXi hosts. LACP mode information is collected from the fabric LAG dual-leafs and also connects to vCenter API and collects LAG groups and members per hypervisor.

**NOTE:** Current validation is done on vCenter virtual Distributed Switches only, not on virtual Standard Switches. LLDP must be enabled on vCenter vDS switches.

Anomalies are raised if any of the following occurs:

- LAG member ports on ToR are connected to non-LAG physical ports on ESXi.
- Non-LAG member ports on ToR are connected to LAG physical ports on ESXi.

**NSX Integration** - Enabling this probe activates a continuous LAG validation between NSX-T transport nodes and data center fabric. It validate that LAGs are properly configured between fabric LAG dual-leafs and NSX-T transport nodes. The NSX-T uplink profile defines the network interface configuration facing the fabric in terms of LAG and LACP config. Network interface misconfiguration between the transport node and the ToR switch is validated and detected.

Anomalies are raised in the following circumstances:

- NSX-T transport nodes are not configured for LAG but ToR has LAG member ports in the fabric.
  - ESXi hosts are dual-attached to ToR leafs but corresponding NSX-T transport nodes are “single-attached” or they are using “NIC-teaming” using active-standby or load-balanced config.
1. Add NSX-T API user as a Virtual Infra.
  2. Add NSX-T Manager in the blueprint (External Systems > Virtual Infra Managers).
  3. Enable this probe (Hypervisor and Fabric LAG config mismatch).

Let's say in the NSX-T uplink profile, LAG is deleted but the fabric has LAG in terms of ToR leafs having LAG member ports. As a result in a blueprint after enabling this probe LAG mismatch anomalies are raised.

Fabric Interface	Fabric Lag	Hypervisor	Leaf	Pnic	Pnic Lag	Anomaly	Value	Updated
swp3	bond1	zz-karun-nsx-t.cvx.2485377892354-3839439666-TN-2	leaf-2-52540005BE0B	eth1		Anomalous value: mismatch Actual value: mismatch	true	a day ago
swp4	bond1	zz-karun-nsx-t.cvx.2485377892354-3839439666-TN-2	leaf-2-52540005BE0B	eth2		Anomalous value: mismatch Actual value: mismatch	true	a day ago

Since the LAG on the NSX-T transport nodes has been deleted, there is a mismatch between physical network adapter (pnic) on ESXi host LAG configuration and LAG configuration on ToR leafs.

For more information about this probe, from the blueprint, navigate to **Analytics > Probes**, click **Create Probe**, then select **Instantiate Predefined Probe** from the drop-down list. Select the probe from the **Predefined Probe** drop-down list to see details specific to the probe.

## Hypervisor & Fabric VLAN Config Mismatch Probe (Virtual Infra)

### IN THIS SECTION

- [Hypervisor & Fabric VLAN Config Mismatch Probe Overview | 957](#)
- [Usage with NSX-T Integration | 958](#)
- [Usage with VCenter Integration | 963](#)

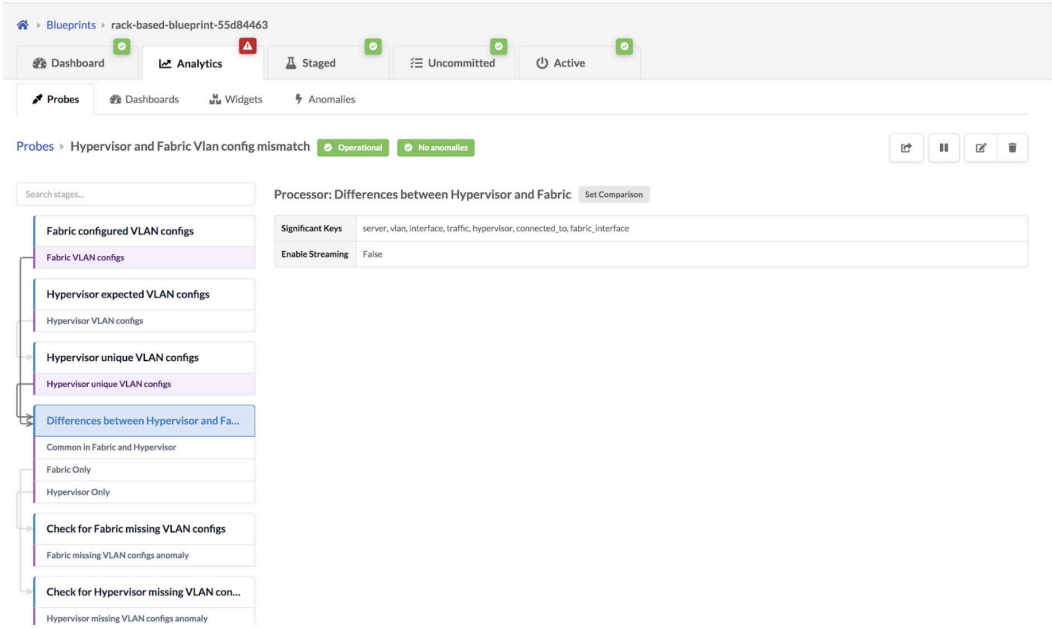
## Hypervisor & Fabric VLAN Config Mismatch Probe Overview

<b>Purpose</b>	Calculate VLAN mismatch between configured virtual networks on leafs and VLANs needed by VMs running on the hypervisors attached to leafs. (Formerly known as Virtual Infra VLAN Match).		
<b>Source Processors</b>	<b>Fabric configured VLAN configs (generic graph collector)</b>	output stage: Fabric VLAN configs (number set) (generated from graph)	
	<b>Hypervisor expected VLAN configs (generic graph)</b>	output stage: Hypervisor VLAN configs (number set)	
<b>Additional Processor(s)</b>	<b>Hypervisor unique VLAN configs (set count)</b>	input stage: Hypervisor VLAN configs output stage: Hypervisor unique VLAN configs (number set)	
	<b>Differences between Hypervisor and Fabric (set comparison)</b>	<b>input stages:</b>	Hypervisor unique VLAN configs
			Fabric VLAN configs
		<b>output stages:</b>	Common in Fabric and Hypervisor (number set)
			Fabric Only (number set)
			Hypervisor Only (number set)
	<b>Fabric missing VLAN configs accumulator (accumulate)</b>	input stage: Hypervisor Only output stage: Hypervisor Only TimeSeries (number set time series)	
	<b>Hypervisor missing VLAN configs accumulator (accumulate)</b>	input stage: Fabric Only output stage: Fabric Only TimeSeries (number set time series)	
	<b>Check for Fabric missing VLAN configs (range)</b>	input stage: Hypervisor Only TimeSeries output stage: Fabric missing VLAN configs anomaly (discrete state set)	
	<b>Check for Hypervisor missing VLAN configs (range)</b>	input stage: Fabric Only TimeSeries	

output stage: Hypervisor missing VLAN configs anomaly  
(discrete state set)

Usage with NSX-T Integration

- 1. From the blueprint, navigate to **Analytics > Probes** and click **Hypervisor & Fabric VLAN Config Mismatch** in the probe name list to go to its details. When the VLANs between the data center fabric and the NSX-T transport nodes match, then the probe looks similar to the image below:



- Click the **Fabric VLAN Configs** stage to show the VLANs tagged towards NSX-T transport nodes on fabric ToR leafs as shown below:

Blueprints > rack-based-blueprint-55d84463

Dashboard Analytics Staged Uncommitted Active

Probes > Hypervisor and Fabric Vlan config mismatch Operational No anomalies

Search stages...

Stage: Fabric VLAN configs Number Set

Search stage data...

1-3 of 3 Page Size: 25

Connected To	Fabric Interface	Hypervisor	Interface	Server	Traffic	Vl
leaf-2-52540005BE0B	bond1	zz-karun-nsxt.csv.2485377892354-3839439666-TN-2	a5bb8183-90ef-497f-a988-3ef571066261	rack2_001_server001	tagged	10
leaf-2-52540005BE0B	bond1	zz-karun-nsxt.csv.2485377892354-3839439666-TN-2	a5bb8183-90ef-497f-a988-3ef571066261	rack2_001_server001	tagged	10
leaf-2-52540005BE0B	bond1	zz-karun-nsxt.csv.2485377892354-3839439666-TN-2	a5bb8183-90ef-497f-a988-3ef571066261	rack2_001_server001	tagged	20

Left sidebar menu:

- Fabric configured VLAN configs
  - Fabric VLAN configs
- Hypervisor expected VLAN configs
  - Hypervisor VLAN configs
- Hypervisor unique VLAN configs
  - Hypervisor unique VLAN configs
- Differences between Hypervisor and ...
  - Common in Fabric and Hypervisor
  - Fabric Only
  - Hypervisor Only

- Click the **Common in Fabric and Hypervisor** stage to show that VLANs in the NSX-T transport nodes and the fabric match.

Blueprints > rack-based-blueprint-55d84463

Dashboard Analytics Staged Uncommitted Active

Probes Dashboards Widgets Anomalies

Probes > Hypervisor and Fabric Vlan config mismatch Operational No anomalies

Search stages...

Stage: Common in Fabric and Hypervisor Number Set

Search stage data...

1-3 of 3 Page Size: 25

Connected To	Fabric Interface	Hypervisor	Interface	Server	Traffic	Vlan	Value
52540005BE0B	bond1	zz-karun-nsxt.csv.2485377892354-3839439666-TN-2	a5bb8183-90ef-497f-a988-3ef571066261	rack2_001_server001	tagged	100	1
52540005BE0B	bond1	zz-karun-nsxt.csv.2485377892354-3839439666-TN-2	a5bb8183-90ef-497f-a988-3ef571066261	rack2_001_server001	tagged	1000	1
52540005BE0B	bond1	zz-karun-nsxt.csv.2485377892354-3839439666-TN-2	a5bb8183-90ef-497f-a988-3ef571066261	rack2_001_server001	tagged	2000	1

Left sidebar menu:

- Fabric configured VLAN configs
  - Fabric VLAN configs
- Hypervisor expected VLAN configs
  - Hypervisor VLAN configs
- Hypervisor unique VLAN configs
  - Hypervisor unique VLAN configs
- Differences between Hypervisor and Fa...
  - Common in Fabric and Hypervisor
  - Fabric Only
  - Hypervisor Only
- Check for Fabric missing VLAN configs
  - Fabric missing VLAN configs anomaly
- Check for Hypervisor missing VLAN con...
  - Hypervisor missing VLAN configs anomaly

If the VLAN defined in the Uplink Transport Zone used for BGP peering is modified in the NSX-T Manager, then VLAN mismatch anomalies are raised.

The screenshot displays the NSX-T Manager interface with the 'Probes' tab selected. The breadcrumb trail shows 'Probes > Hypervisor and Fabric Vlan config mismatch'. The status bar indicates 'Operational' and '2 anomalies'. The left sidebar contains a tree view with categories like 'Fabric configured VLAN configs', 'Hypervisor expected VLAN configs', and 'Differences between Hypervisor and Fabric'. The main content area shows the 'Stage: Fabric missing VLAN configs anomaly' with a 'Discrete State Set' button. Below this is an 'Anomaly Remediation' section with a 'Remediate Anomalies' button. A search bar and 'Anomalies Only' checkbox are present. A table displays the anomaly details:

Connected To	Fabric Interface	Hypervisor	Interface	Server	Traffic	Vlan	A
leaf-2-52540005BE08	bond1	zz-karun-nsxt.cvx.2485377892354-3839439666-TN-2	a5bb8183-90ef-497f-a988-3ef571066261	rack2_001_server001	tagged	99	A

Some other reasons for mismatching include the following:

- If the configured VLAN NSX-T transport node is missing in the fabric.
- If the configured VLAN NSX-T transport node is in the fabric, but the end VMs or servers are not part of this virtual network or VLAN.
- If a segment is created in NSX-T for either an overlay or VLAN-based transport zone. It could be that the configured VLAN spanning the logical switch/segment on the transport node is missing on the fabric.
- If L2 bridging for VMs in different overlay logical segments is broken because one VM exists in one logical switch/segment and the other VM exists in a separate uplink logical switch/segment.

As an example, a VLAN is missing in NSX-T 3.0 Host Transport node on the Overlay segment connected to ToR leafs and respective VXLAN VN is present in Juniper Apstra Fabric and ports towards

Hypervisors are assigned in a **Virtual Network** based Connectivity Template as below:

Assign Tagged VxLAN 'overlay-tep-pool-vn' ✕

ae4 -> muc_leaf_5100_001_sys004 (Interface)			<input type="checkbox"/>
▼ muc_leaf_5110_001 (Rack)			<input type="checkbox"/> ⚙
▼ muc_leaf_5110_001_leaf1 / muc_leaf_5110_001_leaf2 (Leaf-pair)			<input type="checkbox"/> ⚙
ae1 -> muc_leaf_5110_001_sys001 (Interface)			<input type="checkbox"/>
ae2 -> muc_leaf_5110_001_sys002 (Interface)			<input type="checkbox"/>
ae3 -> muc_leaf_5110_001_sys003 (Interface)			<input type="checkbox"/>
ae4 -> muc_leaf_5110_001_sys004 (Interface)			<input checked="" type="checkbox"/>
▼ muc_leaf_5120_001 (Rack)			<input type="checkbox"/> ⚙
▼ muc_leaf_5120_001_leaf1 / muc_leaf_5120_001_leaf2 (Leaf-pair)			<input type="checkbox"/> ⚙
ae1 -> muc_leaf_5120_001_sys001 (Interface)			<input type="checkbox"/>
ae2 -> muc_leaf_5120_001_sys002 (Interface)			<input type="checkbox"/>
ae3 -> muc_leaf_5120_001_sys003 (Interface)			<input type="checkbox"/>
ae4 -> muc_leaf_5120_001_sys004 (Interface)			<input checked="" type="checkbox"/>
▼ rack_border_001 (Rack)			<input type="checkbox"/> ⚙
▼ muc_rack_border_001_leaf1 (Leaf)			<input type="checkbox"/> ⚙
et-0/0/32 -> MX_LINK1 (Interface)			<input type="checkbox"/>
et-0/0/33 -> muc_rack_border_001_sys006 (Interface)			<input type="checkbox"/>
xe-0/0/0-0 -> muc_rack_border_001_sys001 (Interface)			<input type="checkbox"/>
► muc_rack_border_001_leaf1 / muc_rack_border_001_leaf2 (Leaf-pair)			<input type="checkbox"/> ⚙
► muc_rack_border_001_leaf2 (Leaf)			<input type="checkbox"/> ⚙

Assign

A **Hypervisor missing VLAN Configs** anomaly is raised as shown below:

Stage: Hypervisor missing VLAN configs anomaly ⌵

**Anomaly Remediation**  
It is possible to automatically fix the anomalies.

Remediate Anomalies

☐ Anomalies Only

Query: All >\_ 1-10 of 10 < Page Size: 25

false ☐ true

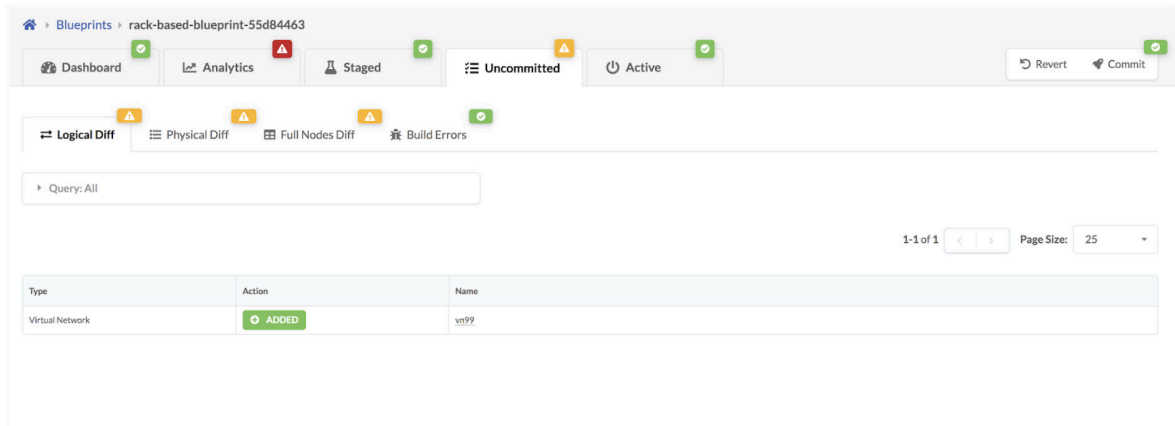
connected To	Fabric Interface	Hypervisor	Interface	Server	Traffic	Vlan	Anomaly	Value	Updated
muc_leaf_5100_001_leaf_pair1	ae2	10.6.1.37	011e8004-942f-4029-ac0a-3fe7be017324	muc_leaf_5100_001_sys002	tagged	150	Anomalous value: ≥ 1 Actual value: 12	true	14 minutes ago
muc_leaf_5100_001_leaf_pair1	ae2	10.6.1.37	011e8004-942f-4029-ac0a-3fe7be017324	muc_leaf_5100_001_sys002	tagged	50	Anomalous value: ≥ 1 Actual value: 12	true	14 minutes ago
muc_leaf_5110_001_leaf_pair1	ae4	10.6.1.35	e3f6918a-8619-4e36-8cef-4f8146732a23	muc_leaf_5110_001_sys004	tagged	150	Anomalous value: ≥ 1 Actual value: 21	true	14 minutes ago
muc_leaf_5110_001_leaf_pair1	ae4	10.6.1.35	e3f6918a-8619-4e36-8cef-4f8146732a23	muc_leaf_5110_001_sys004	tagged	50	Anomalous value: ≥ 1 Actual value: 21	true	14 minutes ago
muc_leaf_5120_001_leaf_pair1	ae4	10.6.1.31	a857436e-66a6-468b-99eb-a198fe0fb0ad	muc_leaf_5120_001_sys004	tagged	150	Anomalous value: ≥ 1 Actual value: 27	true	14 minutes ago
muc_leaf_5120_001_leaf_pair1	ae4	10.6.1.31	a857436e-66a6-468b-99eb-a198fe0fb0ad	muc_leaf_5120_001_sys004	tagged	50	Anomalous value: ≥ 1 Actual value: 24	true	14 minutes ago
muc_rack_border_001_leaf_pair1	ae3	10.6.1.42	01b956ba-4815-42e0-879f-1987aafc7071	muc_rack_border_001_sys003	tagged	150	Anomalous value: ≥ 1 Actual value: 24	true	14 minutes ago
muc_rack_border_001_leaf_pair1	ae3	10.6.1.42	01b956ba-4815-42e0-879f-1987aafc7071	muc_rack_border_001_sys003	tagged	50	Anomalous value: ≥ 1 Actual value: 24	true	14 minutes ago

In some scenarios, a VLAN mismatch anomaly can be remediated. If so, the **Remediate Anomalies** button appears on the probe details page as shown in the screenshot above. Example scenarios include:

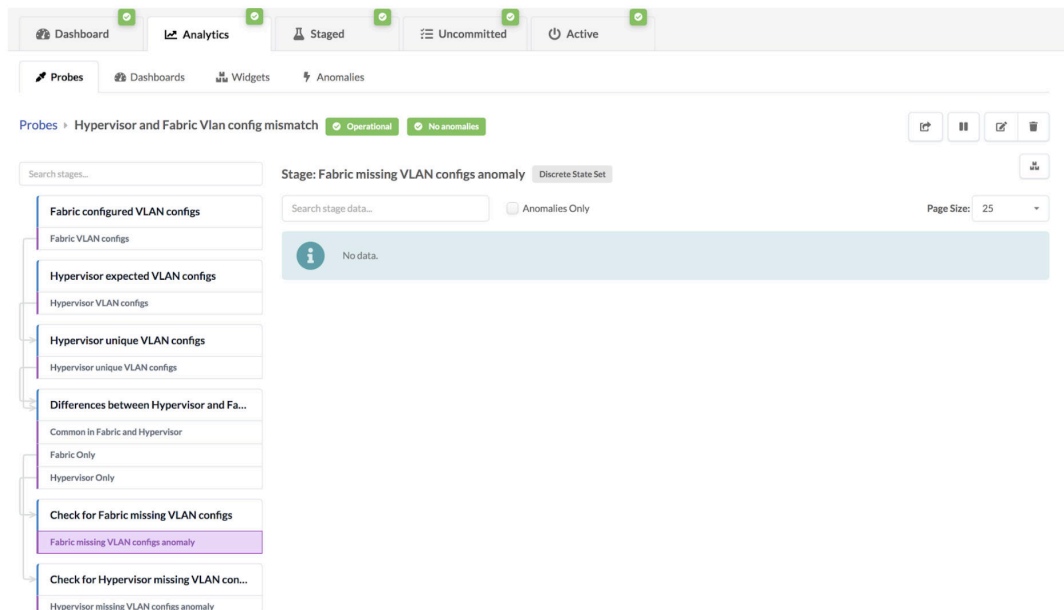
- NSX-T transport nodes use an uplink profile to define transport VLAN over which overlay tunnel comes up. Fabric could be missing the rack-local VN for transport VLAN on hypervisors. One-click remediation can be provided by creating a new rack-local virtual network with the proper VLAN ID in the fabric.

- A rack-local virtual network is defined with VLAN ID Y, however, the connected virtual infra nodes (i.e hypervisors) do not have the VLAN ID in the logical segment/switch. One-click remediation can be provided by removing the endpoint from the affected VLAN ID.

If the **Remediate Anomalies** button appears under the stage name, you can click it to automatically stage the changes required to remediate the anomaly. You can see the staged changes on the **Uncommitted** tab.



Review the staged configuration, add any necessary resources (such as IP subnet address, virtual gateway IP, as so on), then commit the configuration. The probe shows that the anomaly is no longer present.



### Usage with VCenter Integration

Some anomalies, that are raised because of a VLAN config mismatch between vCenter and the fabric, can automatically be remediated, such as the following.

- If the vCenter Distributed Virtual Switch (vDS) port group does not have a corresponding rack-local VN (VLAN) for VLAN ID X. With one-click remediation, a new rack-local virtual network (VLAN) with the proper VLAN ID is created.
- If endpoint X in a rack-local VN with VLAN ID Y, does not have a corresponding dVS port group. With one-click remediation, the endpoint is removed from the affected VLAN ID.

#### Note

vCenter vDS must be used with VLAN specific ID allocation on the port group for L2 network segmentation at the hypervisor level.

A VLAN-based rack-local virtual network is extending each VLAN segment defined on the vDS, across servers within the same rack. For example, vDS port group VLAN 10 = rack-local virtual network with VLAN 10.

For more information about this probe, from the blueprint, navigate to **Analytics > Probes**, click **Create Probe**, then select **Instantiate Predefined Probe** from the drop-down list. Select the probe from the **Predefined Probe** drop-down list to see details specific to the probe.

### Hypervisor MTU Mismatch Probe (Virtual Infra)

<b>Purpose</b>	Detect maximum transmission unit (MTU) value deviations across hypervisor physical network adapters (pnics).	
<b>Source Processor</b>	<b>Interface MTU (generic graph collector)</b>	output stage: Interface MTU (number set) (generated from graph)
<b>Additional Processor(s)</b>	<b>Check MTU mismatch between hypervisors (standard deviation)</b>	input stage: Interface MTU output stage: Hypervisor MTU Deviation (number set)
	<b>MTU Mismatch (range)</b>	input stage: Hypervisor MTU Deviation (number set) output stage: MTU Mismatch (discrete state set)
<b>Example Usage</b>	<b>NSX Integration</b> - If validation fails between NSX-T nodes and the controller in terms of mismatch of minimum configured MTU to support Geneve encapsulation or if the VLANs defined on NSX-T nodes are not configured on ToR leaf interfaces connecting an NSX node to the fabric, then anomalies are raised.	

For more information about this probe, from the blueprint, navigate to **Analytics > Probes**, click **Create Probe**, then select **Instantiate Predefined Probe** from the drop-down list. Select the probe from the **Predefined Probe** drop-down list to see details specific to the probe.

### Hypervisor MTU Threshold Check Probe (Virtual Infra)

Purpose

Detect virtual infra interfaces with maximum transmission units (MTU) below a specified threshold (default: 1600).

Source Processor

Interface MTU (generic graph collector)

output stage: Interface MTU (number set) (generated from graph)

Additional Processor(s)

MTU below threshold (range)

input stage: Interface MTU

output stage: MTU below threshold (discrete state set)

**Example Usage** **NSX Integration** - To carry VXLAN-encapsulated overlay traffic, an MTU greater than 1600 is recommended. NSX-T transport nodes connected to ToR leafs that are below the specified threshold are detected.

To support Geneve encapsulation, the MTU configuration on NSX-T nodes involved in an overlay transport zone must have a valid MTU setting on the ESXi host. The image (from a previous Apstra version) below shows hypervisors with the MTU above the threshold.

Blueprints

rack-based-blueprint-87a66f1c

Dashboard

Analytics

Staged

Uncommitted

Active

Probes

Hypervisor MTU threshold check

Operational

No anomalies

Search stages...

Interface MTU

Interface MTU

MTU below threshold

MTU below threshold

Stage: MTU below threshold

Discrete State Set

Search stage data...

Anomalies Only

1-15 of 15

Page Size: 25

Hypervisor	Pnic Name	Vtep	Anomaly	Value	Updated
172.20.64.6	vmnic1	vmk10	No anomaly	false	10 days ago
demo-NSX-Node1	eth1	nsx-vtep0.0	No anomaly	false	10 days ago
demo-NSX-Node1	eth2	nsx-vtep0.0	No anomaly	false	10 days ago
demo-NSX-Node2	eth1	nsx-vtep0.0	No anomaly	false	10 days ago
demo-NSX-Node2	eth2	nsx-vtep0.0	No anomaly	false	10 days ago
zz-aaron-nsxtveos.2485377892354-1782299148-TN-2	eth1	nsx-vtep0.0	No anomaly	false	10 days ago
zz-aaron-nsxtveos.2485377892354-1782299148-TN-2	eth2	nsx-vtep0.0	No anomaly	false	10 days ago
zz-addbp-nsxt.cvx.2485377892354-1122158190-TN-1	eth1	nsx-vtep0.0	No anomaly	false	2 days ago
zz-addbp-nsxt.cvx.2485377892354-1122158190-TN-1	eth2	nsx-vtep0.0	No anomaly	false	2 days ago
zz-addbp-nsxt.cvx.2485377892354-1122158190-TN-2	eth1	nsx-vtep0.0	No anomaly	false	2 days ago

If any of the hypervisors were below the threshold, the expected value would change to **true** and an anomaly would be raised.

For more information about this probe, from the blueprint, navigate to **Analytics > Probes**, click **Create Probe**, then select **Instantiate Predefined Probe** from the drop-down list. Select the probe from the **Predefined Probe** drop-down list to see details specific to the probe.

### Hypervisor Missing LLDP Config Probe (Virtual Infra)

<b>Purpose</b>	Detect virtual infra hosts that are not configured for LLDP. (Formerly known as Virtual Infra missing LLDP config).	
<b>Source Processor</b>	<b>Hypervisor NIC LLDP Config (generic graph)</b>	output stage: Hypervisor NIC LLDP config (discrete state set) (generated from graph)
<b>Additional Processor(s)</b>	<b>LLDP config by switch (match count)</b>	input stage: Hypervisor NIC LLDP config output stage: LLDP config by switch (number set)
	<b>Switches missing LLDP config (range)</b>	input stage: LLDP config by switch output stage: Switches missing LLDP config anomaly (discrete state set)
<b>Example Usage</b>	<b>VMware Integration</b> - If LLDP information is missing on ToR connected to physical ports on ESXi, an anomaly is raised.	

For more information about this probe, from the blueprint, navigate to **Analytics > Probes**, click **Create Probe**, then select **Instantiate Predefined Probe** from the drop-down list. Select the probe from the **Predefined Probe** drop-down list to see details specific to the probe.

### Hypervisor Redundancy Checks Probe (Virtual Infra)

<b>Purpose</b>	Detect hypervisor redundancy.	
<b>Source Processors</b>	<b>Hypervisor and connected leaf (generic graph)</b>	output stage: hypervisor_and_leaf (text set) (generated from graph)
	<b>Hypervisor pnic and vnet (generic graph collector)</b>	output stage: hypervisor_pnic_vnet (text set) (generated from graph)
<b>Additional Processor(s)</b>	<b>Hypervisor and leaf count (set count)</b>	input stage: hypervisor_and_leaf output stage: hypervisors_leaf_count (number set)
	<b>Hypervisor vnet pnic count (set count)</b>	input stage: hypervisors_pnic_vnet

output stage: hypervisors\_vnet\_pnic\_count (number set)

- Hypervisor without ToR Switch redundancy (range)

input stage: hypervisors\_leaf\_count

output stage: hypervisor\_single\_leaf\_anomaly (discrete state set)

Networks without link redundancy (range)

input stage: hypervisors\_vnet\_pnic\_count

output stage: hypervisor\_vnet\_single\_pnic\_anomaly (discrete state set)

Example Usage

- NSX-T Integration** - an anomaly is raised in cases without redundancy or a single point of failure (SPOF) in hypervisor connectivity. Examples include:
- NSX-T transport nodes with a single non-LAG uplink towards ToR Leafs in the fabric can result in a single point of failure (SPOF) for overlay traffic.
  - NSX-T transport nodes with a single LAG uplink with both members going to a single ToR leaf can result in a single point of failure (SPOF).
  - Lack of redundancy between fabric LAG dual-leafs and ESXi hosts.

Detect single point of failures in hypervisor connectivity

Search stages...

Hypervisor and connected leaf

hypervisor\_and\_leaf

Hypervisor pnic and vnet

hypervisor\_pnic\_vnet

Hypervisor and leaf count

hypervisors\_leaf\_count

Hypervisor vnet pnic count

hypervisors\_vnet\_pnic\_count

Hypervisors without ToR Switch redundancy

hypervisor\_single\_leaf\_anomaly

Stage: hypervisor\_single\_leaf\_anomaly

Discrete State Set

Search stage data...

Anomalies Only

1 of 1

Page Size: 25

Hypervisor	Hypervisor Id	Rack	Anomaly	Value	Updated
zz-karun-nst-cnx-2485377892354-3839439666-TN-2	859D779-d85a-4380-a256-5c2eb00141c	rack2_001	Anomalous value: ≤ 1 Actual value: 1	true	39 minutes ago

For more information about this probe, from the blueprint, navigate to **Analytics > Probes**, click **Create Probe**, then select **Instantiate Predefined Probe** from the drop-down list. Select the probe from the **Predefined Probe** drop-down list to see details specific to the probe.

Interface Flapping (Fabric Interfaces) Probe

**Purpose** This probe determines if fabric interfaces are flapping. A given interface (considering only fabric interfaces) is considered to be flapping if it transitions state more than "Threshold" times over the last "Duration". Such flapping will cause an anomaly to be raised. If more than "Max Flapping Interfaces Percentage" percent of interfaces on a given device are

flapping, an anomaly will be raised for that device. Finally, the last "Anomaly History Count" anomaly state-changes are stored for observation.

<b>Source Processor</b>	<b>leaf fab int status (Service Data Collector)</b>	Purpose: wires in interface status telemetry for all fabric interfaces on the leafs.
		<b>Output Stage:</b> <b>leaf_if_status</b> Set of operational states ("up" or "down"). Each set member corresponds to a leaf fabric interface and has the following keys to identify it: system_id (id of the leaf system, usually serial number), interface (name of the interface).
<b>Additional Processor(s)</b>	<b>leaf fabric interface status history (Accumulate)</b>	Purpose: create recent history time series for each interface status In terms of the number of samples, the time series will hold the smaller of: 1024 samples or samples collected during the last 'total_duration' seconds (facade parameter).
		Input Stage: leaf_if_status
		<b>Output Stage:</b> <b>leaf_fab_int_status_accumulate</b> Set of interface status time series (for each spine facing interface on each leaf). Each set member has the following keys to identify it: system_id (id of the leaf system, usually serial number), interface (name of the interface).
	<b>leaf fabric interface flapping (Range)</b>	Purpose: Count the number of state changes in the leaf_fab_int_status_accumulate ("up" to "down" and "down" to "up"). If the count is higher than 'threshold' facade parameter return "true", otherwise "false".
		Input Stage: leaf_fab_int_status_accumulate
		<b>Output Stage:</b> <b>if_status_flapping</b> Set of statuses (for each spine facing interface on each leaf), indicating if the interface has been flapping or not. Each set member has the following keys to identify it: system_id (id of the leaf system, usually serial number), interface (name of the interface).

percentage flapping per device interfaces (MatchPercentage)	Input Stage: if_status_flapping
	Output Stage: flapping_fab_int_perc
system anomalous flapping (Range)	Input Stage: flapping_fab_int_perc
	<b>Output Stage: system_flapping</b> Set of statuses for each leaf, indicting if the leaf has higher then acceptable percentage of flapping interfaces. Each set member has the following key to identify it: system_id (id of the leaf system, usually serial number).

For more information about this probe, from the blueprint, navigate to **Analytics > Probes**, click **Create Probe**, then select **Instantiate Predefined Probe** from the drop-down list. Select the probe from the **Predefined Probe** drop-down list to see details specific to the probe.

### Interface Flapping (Specific Interfaces) Probe

The interface flapping (specific interfaces) probe determines if specific interfaces are flapping. A given interface (considering only those specified) is considered to be flapping if it transitions state more than "Threshold" times over the last "Duration". Such flapping causes an anomaly to be raised. If more-than "Max Flapping Interfaces Percentage" percent of interfaces on a given device are flapping, an anomaly is raised for that device. Finally, the last "Anomaly History Count" anomaly state-changes are stored for

observation.

Instantiate Predefined Probe

Predefined Probe \*

Interface Flapping (Specific Interfaces) ▼

Probe Label \*

Interface Flapping (Specific Interfaces)

Interfaces \*

No interfaces specified.

+ Add Interface

Max Flapping Interfaces Percentage

10

Maximum percentage of flapping interfaces on a device

Threshold

5

Sum total of number of flaps in recent-history for which an anomaly will be raised

Duration

1 Minute ▼

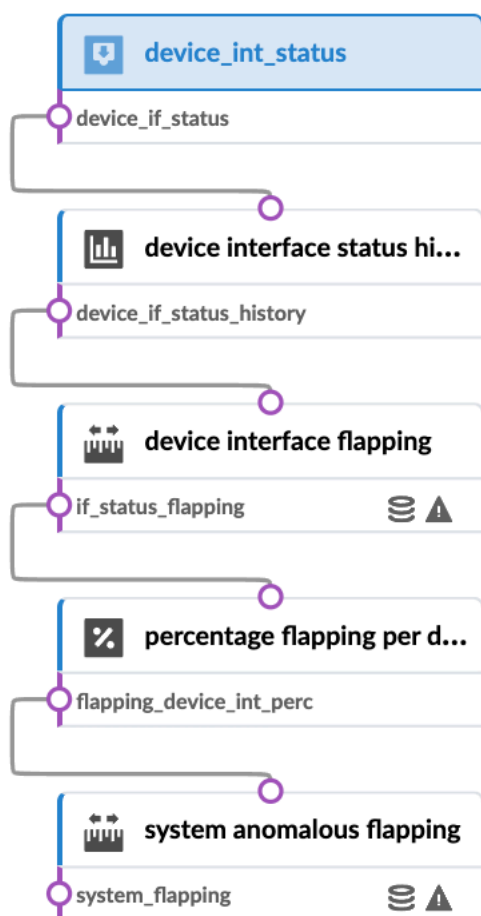
Time period in recent-history in which interface flapping will be considered

Generate a probe to determine if specific interfaces are flapping

A given interface (considering only those specified) is considered to be flapping if it transitions state more than "Threshold" times over the last "Duration". Such flapping will cause an anomaly to be raised.

If more-than "Max Flapping Interfaces Percentage" percent of interfaces on a given device are flapping, an anomaly will be raised for that device.

Finally, the last "Anomaly History Count" anomaly state-changes are stored for observation.



For more information about this probe, from the blueprint, navigate to **Analytics > Probes**, click **Create Probe**, then select **Instantiate Predefined Probe** from the drop-down list. Select the probe from the **Predefined Probe** drop-down list to see details specific to the probe.

### Interface Flapping (Specific Interfaces) Probe

The interface flapping (specific interfaces) probe determines if specific interfaces are flapping. A given interface (considering only those specified) is considered to be flapping if it transitions state more than "Threshold" times over the last "Duration". Such flapping causes an anomaly to be raised. If more-than "Max Flapping Interfaces Percentage" percent of interfaces on a given device are flapping, an anomaly is raised for that device. Finally, the last "Anomaly History Count" anomaly state-changes are stored for

observation.

Instantiate Predefined Probe

Predefined Probe \*

Interface Flapping (Specific Interfaces) ▼

Probe Label \*

Interface Flapping (Specific Interfaces)

Interfaces \*

No interfaces specified.

+ Add Interface

Max Flapping Interfaces Percentage

10

Maximum percentage of flapping interfaces on a device

Threshold

5

Sum total of number of flaps in recent-history for which an anomaly will be raised

Duration

1 Minute ▼

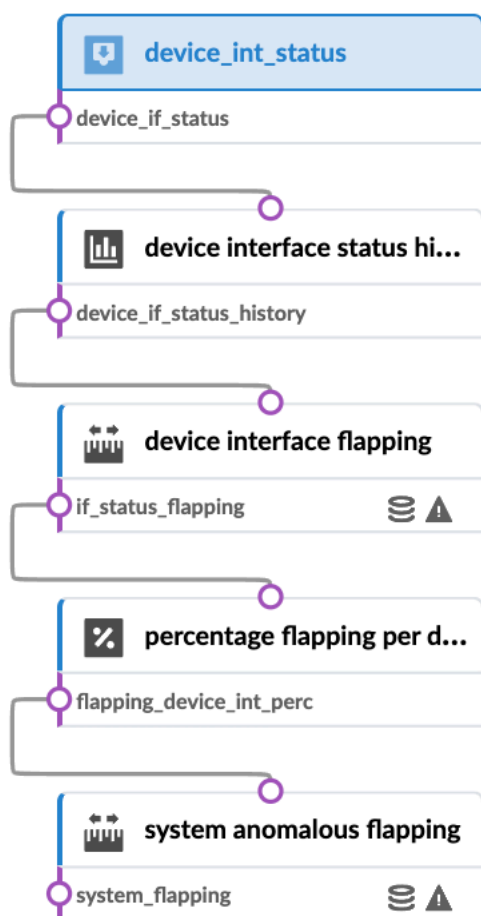
Time period in recent-history in which interface flapping will be considered

Generate a probe to determine if specific interfaces are flapping

A given interface (considering only those specified) is considered to be flapping if it transitions state more than "Threshold" times over the last "Duration". Such flapping will cause an anomaly to be raised.

If more-than "Max Flapping Interfaces Percentage" percent of interfaces on a given device are flapping, an anomaly will be raised for that device.

Finally, the last "Anomaly History Count" anomaly state-changes are stored for observation.



For more information about this probe, from the blueprint, navigate to **Analytics > Probes**, click **Create Probe**, then select **Instantiate Predefined Probe** from the drop-down list. Select the probe from the **Predefined Probe** drop-down list to see details specific to the probe.

### Interface Policy 802.1x Probe

The Interface Policy predefined probe is used to monitor 802.1X supplicants and interface authentication. You can instantiate this probe to maintain 802.1X networks. The 802.1X hosts probe gives a fast view of network 802.1X MAC addresses, authorization status, ports, and dynamic VLAN

information.

Dashboard

Analytics

Staged

Uncommitted

Active

Time Voyager

Obtain telemetry status for interfaces that are activated for interface policies defining 802.1x port control.

Search stages...

802.1x Authorizatio...

802.1x Authorization status

802.1x Authorized ...

802.1x Authorized MACs

802.1x Interface stat...

802.1x Interface status

802.1x hosts

802.1x hosts

802.1x Expected aut...

Processor: 802.1x Authorization status

Extensible Service Data Collector

Properties

Data Type	Text
Graph Query	<pre>match(   node('interface_policy', name='interface_policy', dot1x_port_control=is_in(['auto', 'force_unauthorized']))   .in_('interface_policy')   .node('interface', name='interface'),   node('system', name='system', deploy_mode='deploy', role=is_in(['leaf', 'access']))   .out_('hosted_interfaces')   .node('interface', name='interface')   .out_('link')   .node('link')   .in_('link')   .node('interface')   .in_('hosted_interfaces')   .node('system', name='remote_system', role='generic') ) .ensure_different('system', 'remote_system')</pre>
Ingestion filter	

For more information about interface policies, see Interface Policies <interface\_policies>.

For more information about this probe, from the blueprint, navigate to **Analytics > Probes**, click **Create Probe**, then select **Instantiate Predefined Probe** from the drop-down list. Select the probe from the **Predefined Probe** drop-down list to see details specific to the probe.

LAG Imbalance Probe

The LAG imbalance probe calculates LAG imbalance. It calculates the standard deviation across physical links for all LAGs in the network.

Instantiate Predefined Probe

Predefined Probe \*

LAG Imbalance

Probe Label \*

LAG Imbalance

Max Standard Deviation

20

Maximum standard deviation used for imbalance detection (in percents of link bandwidth).

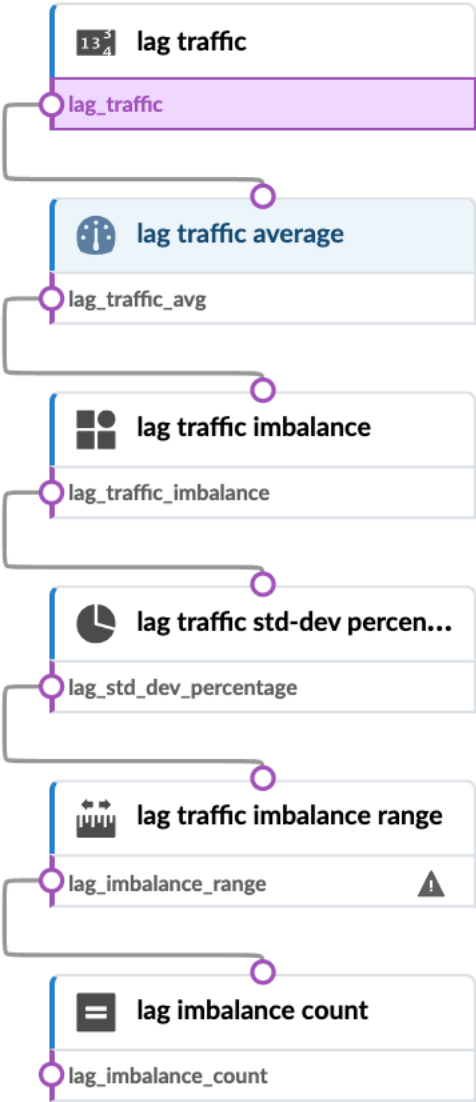
Duration

1 Minute

Time period in recent-history over which imbalance will be considered

Generate a probe to calculate LAG imbalance

Calculates std deviation across physical links for all LAGs in the network.



For more information about this probe, from the blueprint, navigate to **Analytics > Probes**, click **Create Probe**, then select **Instantiate Predefined Probe** from the drop-down list. Select the probe from the **Predefined Probe** drop-down list to see details specific to the probe.

### Leafs Hosting Critical Services: Utilization, Trending, Alerting Probe (new in 4.0.1)

Monitors leaf devices hosting critical services identified by user "tags" and provides trending data for fabric-facing interfaces and alerts if bandwidth utilization reaches a threshold (80%). Users are proactively notified of issues from potential bandwidth contention. Additionally, historical data is persisted for trending analysis for troubleshooting or assisting in right-sizing future deployments. By default, the probe will display the total fabric interface as well as the total percentage of bandwidth used for each tagged leaf device for the past one day (1-day). An anomaly will be raised if the used bandwidth from the tagged leaf reaches 80% of the total available uplink bandwidth.

Predefined Probe \*

Leafs Hosting Critical Services: Utilization, Trending, Alerting

Probe Label \*

Leafs Hosting Critical Services: Utilization, Trending, Alerting

Leaf Tags

No tags

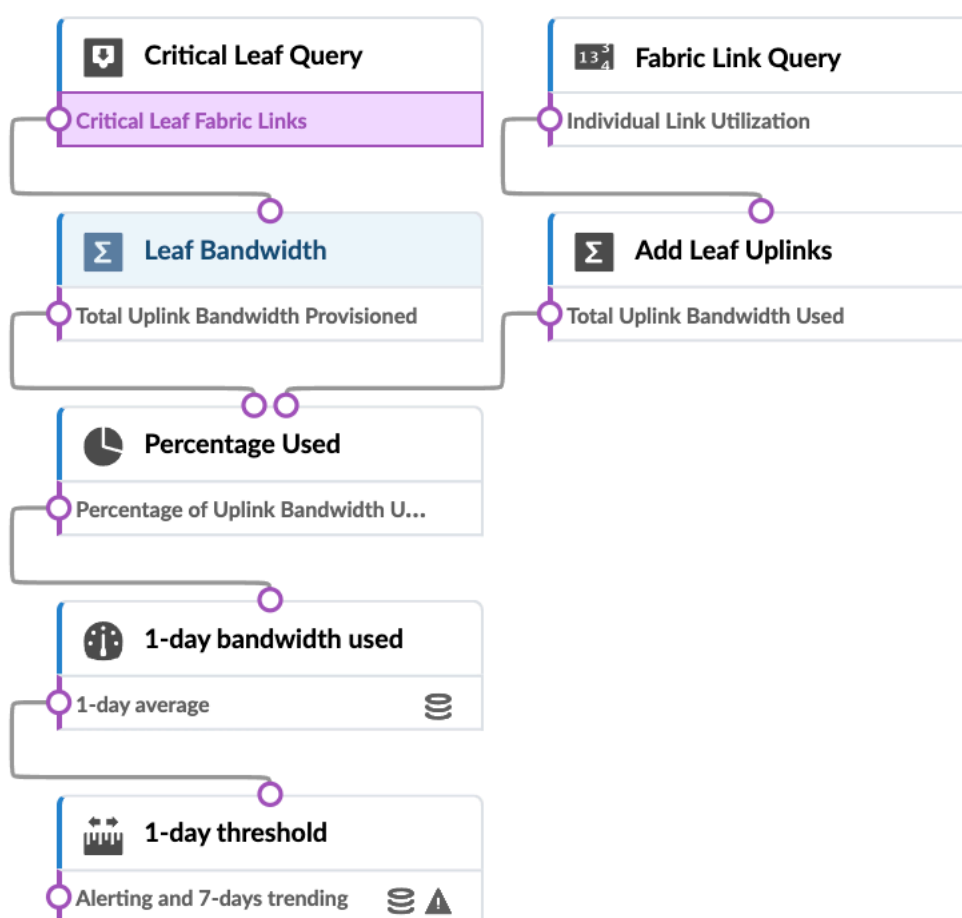
Bandwidth utilization is monitored for fabric interfaces hosted by leaf that have at least one of specified tags assigned.

Utilization threshold

80

If percentage bandwidth utilization reaches the threshold, an anomaly is raised.

Monitors leaf devices hosting critical services identified by user "tags" and provides trending data for fabric-facing interfaces and alerts if bandwidth utilization reaches a threshold (default 80%). Users are proactively notified of issues from potential bandwidth contention. Additionally, historical data is persisted for trending analysis for troubleshooting or assisting in right-sizing future deployments. By default, the probe will display the total fabric interface as well as the total percentage of bandwidth used for each tagged leaf device for the past one day (1-day). An anomaly will be raised if the used bandwidth from the tagged leaf reaches threshold of the total available uplink bandwidth.



For more information about this probe, from the blueprint, navigate to **Analytics > Probes**, click **Create Probe**, then select **Instantiate Predefined Probe** from the drop-down list. Select the probe from the **Predefined Probe** drop-down list to see details specific to the probe.

### Link Fault Tolerance in Leaf and Access LAGs Probe

The link fault tolerance in leaf and access LAG probe monitors LAG fault tolerance issues from a capacity viewpoint.

Instantiate Predefined Probe

Predefined Probe \*

Link Fault Tolerance in Leaf and Access LAGs

Probe Label \*

Link Fault Tolerance in Leaf and Access LAGs

History Duration

12 Hours

Time period of history to maintain

Duration

10 Minutes

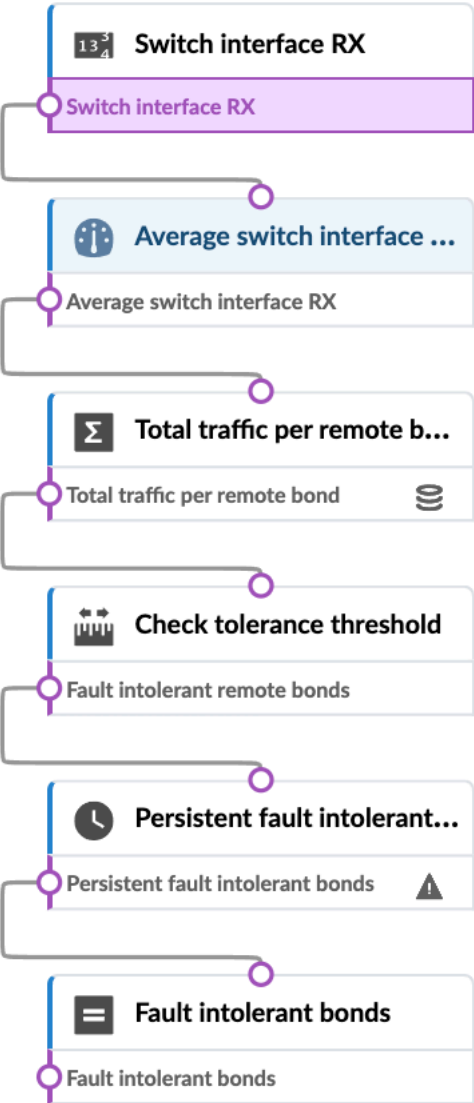
Time period in recent-history over which anomaly intolerant bonds will be considered

Threshold Duration

9 minutes

Total amount of time in recent-history during which bonds with traffic exceeding tolerance threshold is observed for anomaly to be raised

Generate a probe to monitor LAG fault tolerance issues from capacity viewpoint



For more information about this probe, from the blueprint, navigate to **Analytics > Probes**, click **Create Probe**, then select **Instantiate Predefined Probe** from the drop-down list. Select the probe from the **Predefined Probe** drop-down list to see details specific to the probe.

### MLAG Imbalance Probe

The MLAG Imbalance probe calculates MLAG imbalance. It calculates standard deviation across links for all MLAGs in the network. If any are over the specified threshold in the last specified time period, an anomaly is raised. It calculates the percentage of MLAGs in each rack in this state. It calculates standard deviation across port-channels for all port-channels in all MLAGs in the network. If any are over the specified threshold in the last specified time period, an anomaly is raised. It also calculates the percentage of MLAGs in each rack in this state. Finally, it calculates standard deviation of port-channels across their containing MLAGs. If the standard deviation for any of these MLAGs is over the specified threshold, an anomaly is raised. Finally, we calculate the percentage of port-channels in each rack in this state.

<b>Source Processor</b>	<b>mlag interface traffic (Interface Counters)</b>	Purpose: wires in interface traffic samples (measured in bytes per second) all leaf interfaces that are part of an MLAG. Unit is bytes per second.
	<b>Output stage: mlag_int_traffic</b>	Set of traffic samples (for each mlag interface on each leaf). Each set member has the following keys to identify it: mlag_id, server (label of the server node), leaf (label of the leaf node), rack (label of the rack), system_id (leaf serial number), interface (name of the interface).
<b>Additional Processor(s)</b>	<b>mlag interface traffic average (Periodic Average)</b>	Purpose: Calculate average traffic during period specified by average_period facade parameter. Unit is bytes per second.  Input Stage: mlag_int_traffic  <b>Output Stage: mlag_int_traffic_avg</b> Set of traffic average values (for each spine-facing interface on each leaf). Each set member has the following keys to identify it: mlag_id, server (label of the server node), leaf (label of the leaf node), rack (label of the rack), system_id (leaf serial number), interface (name of the interface). Unit is bytes per second.
	<b>mlag interface traffic imbalance</b>	Purpose: Calculate standard deviation between traffic averages on all interfaces belonging to a given MLAG. Unit is bytes per second.  Input Stage: mlag_int_traffic_avg

(Standard Deviation)	<b>Output Stage:</b> <b>mlag_int_traffic_imbalance</b>	Set of numbers, one for each mlag_id, each indicating standard deviation of the average traffic on each interface that is part of this MLAG. Each set member has the following keys to identify it: rack, mlag_id. Unit is bytes per second.
<b>port-channel interface std-dev (Standard Deviation)</b>	Purpose: Calculate standard deviation between traffic averages on all interfaces belonging to a port channel. Unit is bytes per second.  Input Stage: mlag_int_traffic_avg  <b>Output Stage:</b> <b>port_channel_int_std_dev</b>	Set of numbers, one for each port channel identified by mlag_id, leaf pair. Each number each indicates standard deviation of the average traffic on each interface that is part of this port channel. Each set member has the following keys to identify it: rack, mlag_id, leaf. Unit is bytes per second.
<b>port-channel total traffic (Sum)</b>	Purpose: Calculate total traffic per port channel. Unit is byte per second.  Input Stage: mlag_int_traffic_avg  <b>Output Stage:</b> <b>mlag_port_channel_total</b>	Set of numbers, each indicating total traffic for each port channel. Each set member has the following key to identify it: rack, mlag_id, leaf. Unit is byte per second.
<b>mlag port-channel traffic std-dev (Standard Deviation)</b>	Purpose: Calculate standard deviation between traffic averages on both port channels belonging to an MLAG. Unit is bytes per second.  Input Stage: mlag_port_channel_total  <b>Output Stage:</b> <b>mlag_port_channel_imbalance</b>	Set of numbers, one for each MLAG identified by mlag_id, rack pair. Each number indicates standard deviation of the average traffic on each port channel that is part of this MLAG. Each set member has the following keys to identify it: rack, mlag_id. Unit is bytes per second.

<b>std-dev percentage mlag (Ratio)</b>	Input Stage: mlag_int_traffic_imbalance Output Stage: std_dev_percentage_mlag
<b>std-dev percentage port-channel (Ratio)</b>	Input Stage: port_channel_int_std_dev Output Stage: std_dev_percentage_pc
<b>live mlag imbalance (Range)</b>	<p>Purpose: Evaluate if the MLAG imbalance as measured by standard deviation for the average traffic on each member interface is within acceptable range. In this case acceptable range is between 0 and std_max facade parameter (in bytes per second unit).</p> <p>Input Stage: std_dev_percentage_mlag</p> <p><b>Output Stage:</b> <b>live_mlag_imbalance</b> Set of true/false values, each indicating if MLAG imbalance for the average traffic on each member interface is within acceptable range for each mlag. Each set member has the following keys to identify it: rack, mlag_id.</p>
<b>live port- channel imbalance (Range)</b>	<p>Purpose: Evaluate if the port channel imbalance as measured by standard deviation for the average traffic on each member interface is within acceptable range. In this case acceptable range is between 0 and std_max facade parameter (in bytes per second unit).</p> <p>Input Stage: std_dev_percentage_pc</p> <p><b>Output Stage:</b> <b>live_port_channel_imbalance</b> Set of true/false values, each indicating if port channel imbalance for the average traffic on each member interface is within acceptable range for each mlag. Each set member has the following keys to identify it: rack, mlag_id, leaf.</p>
<b>std-dev percentage mlag port- channel (Ratio)</b>	Input Stage: mlag_port_channel_imbalance Output Stage: std_dev_percentage_mlag_pc
<b>live mlag port-channel imbalance (Range)</b>	<p>Purpose: Evaluate if the mlag imbalance as measured by standard deviation for the average traffic on each member port channel is within acceptable range. In this case acceptable range is between 0 and std_max facade parameter (in bytes per second unit).</p>

	Input Stage: std_dev_percentage_mlag_pc	
	<b>Output Stage:</b> <b>mlag_port_channel_imbalance_out_of_range</b>	Set of true/false values, each indicating if MLAG imbalance between the average traffic on each member port channel is within acceptable range for each mlag. Each set member has the following keys to identify it: rack, mlag_id.
<b>mlag imbalance per link count (Match Count)</b>	Input Stage: live_mlag_imbalance  Output Stage: mlag_imbalance_link_count	
<b>port-channel imbalance per rack (Match Percentage)</b>	Purpose: Calculate percentage of port channels on a given rack that have imbalance anomaly. Input Stage: live_port_channel_imbalance  <b>Output Stage:</b> <b>port_channel_imbalance_per_rack</b>	Set of numbers, each indicating the percentage of port channels with imbalance on each rack. Each set member has the following key to identify it: rack, mlag_id, leaf.
<b>mlag port-channel imbalance per rack (Match Percentage)</b>	Purpose: Calculate percentage of MLAGs on a given rack that have port channel imbalance anomaly.  Input Stage: mlag_port_channel_imbalance_out_of_range  <b>Output Stage:</b> <b>mlag_port_channel_imbalance_anomaly_per_rack</b>	Set of numbers, each indicating the percentage of port channels with imbalance on each rack. Each set member has the following key to

identify it: rack,  
mlag\_id.

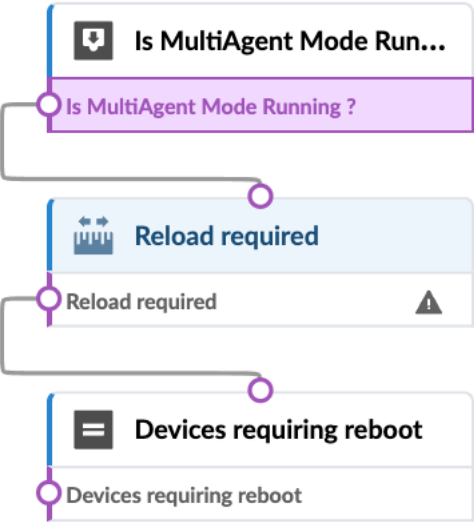
For more information about this probe, from the blueprint, navigate to **Analytics > Probes**, click **Create Probe**, then select **Instantiate Predefined Probe** from the drop-down list. Select the probe from the **Predefined Probe** drop-down list to see details specific to the probe.

Multiagent Detector Probe

The multiagent detector probe raises an anomaly if EOS is not running in multiagent mode, indicating that a reboot is required.

Instantiate Predefined Probe

<p>Predefined Probe *</p> <div>Multiagent Detector</div>	<p>This probe raises an anomaly if EOS is not running in multiagent mode, indicating reboot is required.</p>
<p>Probe Label *</p> <div>Multiagent Detector</div>	



For more information about this probe, from the blueprint, navigate to **Analytics > Probes**, click **Create Probe**, then select **Instantiate Predefined Probe** from the drop-down list. Select the probe from the **Predefined Probe** drop-down list to see details specific to the probe.

# Packet Discard Percentage Probe

The packet discard percentage probe raises visibility into issues related to physical interfaces.

## Instantiate Predefined Probe

Predefined Probe \*

Packet Discard Percentage

Probe Label \*

Packet Discard Percentage

Ingress History Duration

12 Hours

Time period in recent-history for discard ingress packets and total rx packets

Discard Percent Threshold

1

Discard percentage threshold. Consider the discard percentage is too high if it is greater than this threshold

Duration

1 minute 30 seconds

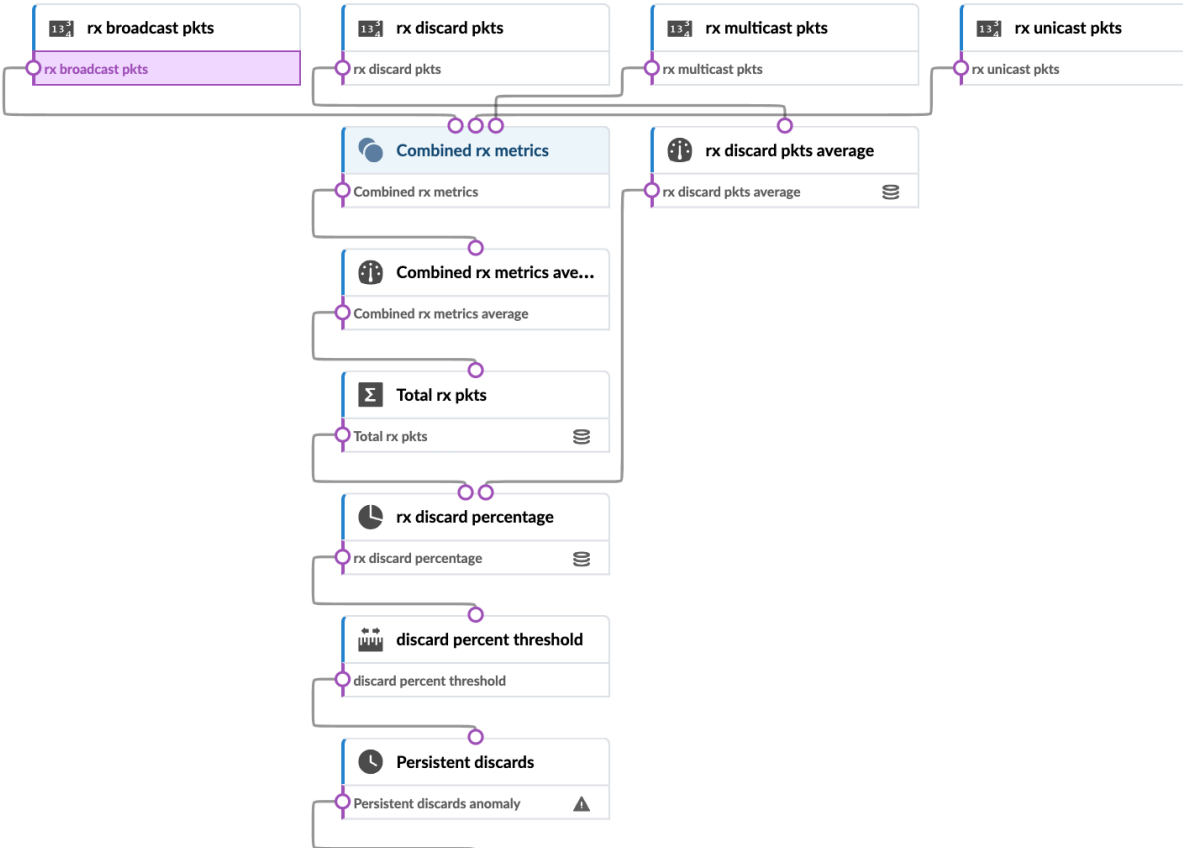
Time period in recent-history over which discard percentage data will be considered

Threshold Duration

20 seconds

Total amount of time in recent-history during which the discard percentage is higher than threshold for anomaly to be raised

Generate a probe to raise visibility into issues related to physical interfaces



For more information about this probe, from the blueprint, navigate to **Analytics > Probes**, click **Create Probe**, then select **Instantiate Predefined Probe** from the drop-down list. Select the probe from the **Predefined Probe** drop-down list to see details specific to the probe.

## Spine Fault Tolerance Probe

The spine fault tolerance probe monitors spine fault tolerance issues from a capacity viewpoint.

### Instantiate Predefined Probe

Predefined Probe \*

Spine Fault Tolerance

Probe Label \*

Spine Fault Tolerance

History Duration

12 Hours

Time period of history to maintain

Number of Faulty Spines

1

Number of faulty spine used to monitor link fault tolerance

Duration

10 Minutes

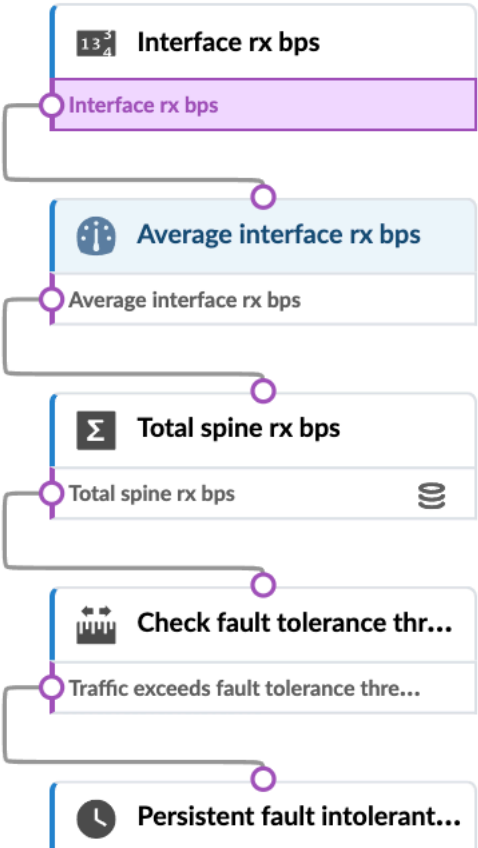
Time period in recent-history in which anomaly intolerant traffic will be considered

Threshold Duration

9 minutes

Total amount of time in recent-history during which total rx traffic of spines exceeding tolerance threshold is observed for anomaly to be raised

Generate a probe to monitor spine fault tolerance issues from capacity viewpoint



For more information about this probe, from the blueprint, navigate to **Analytics > Probes**, click **Create Probe**, then select **Instantiate Predefined Probe** from the drop-down list. Select the probe from the **Predefined Probe** drop-down list to see details specific to the probe.

### Total East/West Traffic Probe

**Purpose** The Total East/West Traffic probe calculates total east/west traffic. This probe takes the sum of all traffic to leafs from their directly-attached servers and subtracts from that the sum of all traffic to external routers (all traffic values in this calculation are averaged periodically over "Average Period"). The result of this is the total east/west traffic. Time series of length "History Sample Count" is maintained for the sum of server traffic, the sum of external traffic, and the total east/west traffic.

When instantiating this probe, external router tag(s) must be specified (new in version 4.0).

<b>Source Processors</b>	<b>external router south-north link traffic (Interface Counters)</b>	Purpose: wires in interface traffic samples (measured in bytes per second) for traffic sent to external routers	
		Output Stage: ext_router_interface_traffic	
	<b>leaf server traffic counters (Interface Counters)</b>	Purpose: wires in interface traffic samples (measured in bytes per second) for traffic received on leafs from the servers	
		<b>Output Stage:</b> <b>server_traffic_counters</b>	Set of traffic samples (for each server-facing interface on each leaf) in the receive direction. Each set member has the following keys to identify it: system_id (id of the leaf system, usually serial number), interface (name of the interface).
<b>Additional Processor(s)</b>	<b>external router south-north links traffic average (Periodic Average)</b>	Purpose: Calculate average traffic for each interface-facing external router traffic during period specified by average_period facade parameter. Unit is bytes per second.	
		Input Stage: ext_router_interface_traffic	
		<b>Output Stage:</b> <b>ext_router_interface_traffic_avg</b>	Set of traffic average values (for each external router-facing interface on each device). Each set member has the following keys to identify

it: system\_id (id of the leaf system, usually serial number), interface (name of the interface).

<b>server traffic average (Periodic Average)</b>	<p>Purpose: Calculate average server traffic during period specified by average_period facade parameter. Unit is bytes per second.</p> <p>Input Stage: server_traffic_counters</p> <p><b>Output Stage:</b> <b>server_traffic_avg</b> Set of traffic average values (for each server-facing interface on each leaf) in the receive direction. Each set member has the following keys to identify it: system_id (id of the leaf system, usually serial number), interface (name of the interface).</p>
<b>south-north traffic (Sum)</b>	<p>Purpose: Calculate total traffic by summing average traffic on each interface-facing external router. Unit is bytes per second.</p> <p>Input Stage: ext_router_interface_traffic_avg</p> <p><b>Output Stage:</b> <b>total_outgoing_traffic</b> Total south-north traffic average in bytes per second.</p>
<b>total server traffic (Sum)</b>	<p>Purpose: Calculate total server traffic by summing average traffic on each interface attached to servers in receive direction. Unit is bytes per second.</p> <p>Input Stage: server_traffic_avg</p> <p><b>Output Stage:</b> <b>total_server_traffic</b> Total server traffic average in bytes per second.</p>
<b>outgoing_traffic_average (Periodic Average)</b>	<p>Purpose: Calculate total south-north traffic over average_period seconds, which is a facade parameter. Unit is bytes per second.</p> <p>Input Stage: total_outgoing_traffic</p>

	<b>Output Stage:</b> <b>total_outgoing_traffic_average</b>	Total south-north traffic average in bytes per second.
<b>server generated traffic average (Periodic Average)</b>	Purpose: Calculate total average server traffic over average_period seconds, which is a facade parameter. Unit is bytes per second.  Input Stage: total_server_traffic	
	<b>Output Stage:</b> <b>total_server_traffic_history</b>	Time series showing total average server traffic over recent history. Unit is bytes per second.
<b>east-west traffic (Subtract)</b>	Purpose: create recent history time series showing how total average east-west traffic changed over time. In terms of the number of samples, the time series holds history_sample_count values (facade parameter). Unit is bytes per second.  Input Stages: total_outgoing_traffic_average and total_server_traffic_history	
	<b>Output Stage:</b> <b>eastwest_traffic_history</b>	Time series showing how total average east-west traffic changed over recent history. Unit is bytes per second

**VMs without Fabric Configured VLANs Probe (Virtual Infra)**

<b>Purpose</b>	Calculate VMs missing a VLAN and calculate VMs not backed by VLANs on managed leafs connected to hypervisors.	
<b>Source Processors</b>	<b>VMs backed by Fabric VLANs (generic graph collector)</b>	output stage: VMs backed by Fabric VLANs (number set) (generated from graph)
	<b>VMs on hypervisors connected to Fabric (generic)</b>	output stage: VMs on hypervisors connected to Fabric (number set)

Additional Processor(s)	Differences between Fabric and Hypervisor (set comparison<processor_set_comparison>)	input stage(s):	VMs backed by Fabric VLANs (number set)
			VMs on hypervisors connected to Fabric (number set)
		output stage:	VMs not backed by Fabric VLANs (number set)
	Affected VM Anomalies (range <processor_range>)	input stage:	VMs not backed by Fabric VLANs
		output stage:	Affected VM Anomalies (discrete state set)

**Example Usage**

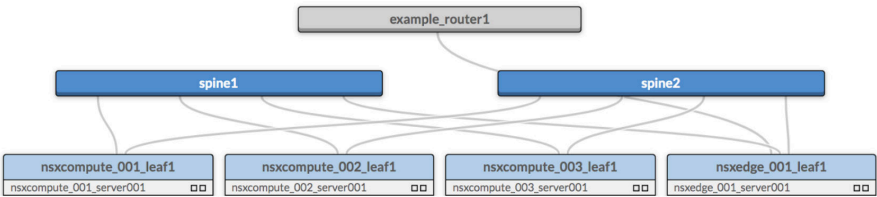
**NSX-T Integration** - VMs participating in a particular network are attached to an NSX logical switch. In NSX transport zone controls to which hypervisors or ESXi host an NSX logical switch can span. To have VXLAN connectivity for these VMs they need to be part of the same transport zone. This predefined anomaly helps validate that all VLAN backend interfaces defined for NSX-T nodes are also configured on the ToR interfaces connecting that node to the fabric.

VLAN probe anomaly checks for VLAN specification in case of NSX-T via one of the two methods below:

Method One: When you have VMs that are connected to the NSX-T overlay, you can configure a bridge-backed logical switch to provide layer 2 connectivity with other devices or VMs. So via VLAN specification on NSX-T layer 2 bridges and fabric if respective VXLAN VN is not there, then an anomaly is raised.

Method Two: Edge uplinks go out through VLAN logical switches. So let's say if the uplink VLAN logical switch has a particular VLAN ID and respective VLAN on ToR port connected to the hypervisor host is not configured then also this VLAN probe will raise anomalies and help detect such misconfiguration.

The following is a simple topology where nsxcompute\_001\_server\_001 and nsxedge\_001\_server001 are ESXi hosting VMs that are connected to the NSX-T overlay network.



There is one VM on each ESXi host that needs a VXLAN VN endpoint on each leaf, i.e. nsxcompute\_001\_leaf1 and nsxedge\_001\_leaf1 to communicate on the overlay network.

When VXLAN VNs assigned to ToR leafs are deleted, VLAN misconfig anomalies are raised as below under Fabric Health in the dashboard.

#### Critical services affected by VLAN misconfig

Hypervisor	Virtual Machine	Virtual Machine Ip
nsxtcomputehost01	webtier010	192.168.1.10
nsxtcomputehost01	webtier011	192.168.1.30
nsxtedgehost01	webtier020	192.168.1.20
<a href="#">View stage</a>		

**VMs not backed by Fabric VLANs** shows VMs with VLAN missing.

Probes > VMs without Fabric configured VLANs Operational 3 anomalies

Search stages...

- VMs backed by Fabric VLANs
- VMs on hypervisors connected to...
- Differences between Fabric and ...
- Affected VM Anomalies

Stage: VMs not backed by Fabric VLANs Output: B - A Type: Number Set

Search stage data... ☐ Spotlight View 1-3 of 3 Page Size: 25

Hypervisor	Interface	Virtual Machine	Virtual Machine Ip	Vlan	Vnet	Vnic
nsxtcomputehost01	33c307a5-5895-4252-8e60-aef5d8ccd4c2	webtier010	192.168.1.10	No value	benefitswebtier	Network adapter 1
nsxtcomputehost01	33c307a5-5895-4252-8e60-aef5d8ccd4c2	webtier011	192.168.1.30	No value	benefitswebtier	Network adapter 1
nsxtedgehost01	f0286797-26d1-4e00-b8af-5260e3b671ac	webtier020	192.168.1.20	No value	benefitswebtier	Network adapter 1

**Affected VM Anomalies** shows VLAN missing in the fabric.

Probes > VMs without Fabric configured VLANs Operational 3 anomalies

Search stages...

- VMs backed by Fabric VLANs
- VMs on hypervisors connected to...
- Differences between Fabric and ...
- Affected VM Anomalies

Stage: Affected VM Anomalies Output: out Type: Discrete State Set

Search stage data... ☐ Spotlight View Anomalies only 1-3 of 3 Page Size: 25

	Virtual Machine	Virtual Machine Ip	Vlan	Vnet	Vnic	Anomaly	Value	Updated
252-8e60-aef5d8ccd4c2	webtier010	192.168.1.10	No value	benefitswebtier	Network adapter 1	Expected value: 0 Actual value: 1	true	8 hours ago
252-8e60-aef5d8ccd4c2	webtier011	192.168.1.30	No value	benefitswebtier	Network adapter 1	Expected value: 0 Actual value: 1	true	8 hours ago
e00-b8af-5260e3b671ac	webtier020	192.168.1.20	No value	benefitswebtier	Network adapter 1	Expected value: 0 Actual value: 1	true	8 hours ago

## VXLAN Flood List Validation Probe

The VXLAN flood list validation probe validates the VXLAN flood list entries on every leaf in the network. It collects appropriate telemetry data, compares it to the set of flood list forwarding entries expected to be present and alerts if expected entries are missing on any device.

You can configure the following parameters:

- **Probe Label:** Name to identify the probe.
- **Anomaly Time Window :** Average period duration for interface counters.
- **Anomaly Threshold (in %):** If routes are missing for more than or equal to percentage of Anomaly Time Window, an anomaly is raised. If Anomaly Time Window ATW, and Anomaly Threshold is AT. It calculates  $Z = (ATW * AT)/100$  in seconds. E.g. If ATW = 20 seconds, AT = 5%, then  $Z = (20 * 5)/100 = 1$  second. When the route is in Missing state for Z seconds from total ATW duration, anomaly is raised.
- **Collection period:** All these probes are polling-based so they have a polling period.

The route labels include the following:

- **Expected:** This route is expected on the device as per service defined.
- **Missing:** This route is missing on the device when compared to the expected route set.

- **Unexpected:** There are no expectations rendered (by AOS) for this route.

**Instantiate Predefined Probe**

**Predefined Probe \***

VXLAN Flood List Validation

**Probe Label \***

VXLAN Flood List Validation

**Anomaly Time Window**

6 minutes

**Anomaly Threshold (in %)**

100

If routes are missing for more than or equal to percentage of Anomaly Time Window, an anomaly will be raised.

**Collection period**

5 Minutes

Telemetry collection interval.

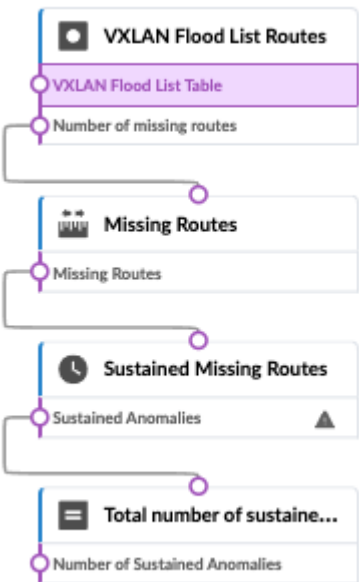
This probe validates the VXLAN flood list entries on every leaf in the network. It collects appropriate telemetry data, compares it to the set of flood list forwarding entries expected to be present and alerts if expected entries are missing on any device.

**Route Labels**

**Expected:** This route is expected on the device as per service defined.

**Missing:** This route is missing on the device when compared to the expected route set.

**Unexpected:** There are no expectations rendered (by AOS) for this route.



**NOTE:** Auto-enabling the **EVPN VXLAN Route Summary** analytics dashboard enables the **EVPN VXLAN Type-3 Route Validation** and **EVPN Flood List Validation** probes automatically (but not the EVPN VXLAN Type-5 Route Validation probe). See [Configuring Auto-Enabled Dashboards<configure\\_dashboard>](#) for information about enabling the dashboard.

For more information about this probe, from the blueprint, navigate to **Analytics > Probes**, click **Create Probe**, then select **Instantiate Predefined Probe** from the drop-down list. Select the probe from the **Predefined Probe** drop-down list to see details specific to the probe.

## Probe Processors (Analytics)

### IN THIS SECTION

- [Processor: Accumulate | 994](#)
- [Processor: Average | 998](#)
- [Processor: Comparison | 999](#)
- [Processor: EVPN Type 3 | 1000](#)
- [Processor: EVPN Type 5 | 1001](#)
- [Processor: Extensible Service Data Collector | 1002](#)
- [Processor: Generic Graph Collector | 1005](#)
- [Processor: Generic Service Data Collector | 1008](#)
- [Processor: Interface Counters | 1011](#)
- [Processor: Logical Operator | 1014](#)
- [Processor: Match Count | 1015](#)
- [Processor: Match Percentage | 1017](#)
- [Processor: Match String | 1019](#)
- [Processor: Max | 1022](#)
- [Processor: Min | 1024](#)
- [Processor: Periodic Average | 1026](#)
- [Processor: Range | 1029](#)
- [Processor: Ratio | 1032](#)
- [Processor: Service Data Collector | 1034](#)
- [Processor: Set Comparison | 1037](#)
- [Processor: Set Count | 1039](#)
- [Processor: Standard Deviation | 1040](#)
- [Processor: State | 1042](#)
- [Processor: Subtract | 1045](#)

- Processor: Sum | 1046
- Processor: System Utilization | 1047
- Processor: Time in State | 1048
- Processor: Traffic Monitor | 1053
- Processor: Union | 1055
- Processor: VXLAN Floodlist | 1057

## Processor: Accumulate

### IN THIS SECTION

- Example: Accumulate | 996

The Accumulate processor used in IBA probes creates one number or discrete state time-series on output for each input with the same properties; each time the input changes, it takes its timestamp and value and appends them to the corresponding output series. If total duration (total\_duration) is set and the length of the output time series in time is greater than duration, it removes old samples from the time series until this is no longer the case. If max samples (max\_samples) is set and the length of the output time series in terms of number of samples is greater than max\_samples, it removes old samples from the time series until this is no longer the case.

Parameters	Description
Input Types	Table (number or discrete state)
Output Types	Table (number or discrete state, accumulate=True)
Max Samples (max_samples)	Limits the maximum number of samples or an expression that evaluates to number of samples (default:1024)
Total Duration (total_duration)	Limits the number of samples by their total duration. (in seconds) or an expression that evaluates to number of seconds (default:0)
Graph Query (graph_query)	One or more queries on graph specified as strings, or a list of such queries. (String will be deprecated in a future release.) Multiple queries should provide all the named nodes referenced by the expression fields (including additional_properties). Graph query is executed on the "operation" graph. Results of the queries can be accessed using the "query_result" variable with

Parameters	Description
	<p>the appropriate index. For example, if querying property set nodes under name "ps", the result will be available as "query_result[0]["ps"]".</p> <p>In collector processors (*_collector, if_counter) it is used to choose a set of nodes for further processing (for example, all leafs, or all interfaces between leaf and spines)</p> <p>In other processors it is used for general parameterization and it is only supported as a list of queries.</p> <pre>graph_query: "node("system", role="leaf", name="system").     out("hosted_interfaces").     node("interface", name="iface").out("link").     node("link", role="spine_leaf")"</pre> <pre>graph_query: ["node("system", role="leaf", name="system")",     "node("system", role="spine", name="system")"]</pre> <p>Non-collector processors containing the graph_query configuration parameter, can be parameterized to use data from arbitrary nodes in the graph, such as property set nodes. Property sets allow you to parameterize macro level SLAs for individual business units. In the example below, graph_query matches a node of type property_set with label probe_propset. It's accessed using the special query_result variable, where Index 0 means it's the first node in query results. If a query returned N nodes, they could be accessed using indices starting from 0 to N-1. ps is what the actual node is referred to in the query; the rest depends on the structure of the node. The int() casting is required because values of property_set nodes are strings. Here it's assumed that a property set node has the label probe_propset and that the value accumulate_duration was already created.</p> <pre>graph_query: [node("property_set", label="probe_propset", name="ps")] duration: int(query_result[0]["ps"].values["accumulate_duration"])</pre> <p>Another example is a that probes can validate a compliance requirement; the compliance value may change over time and/or it can be used by more than one probe. Also, a probe can validate NOS versions on devices. In this case,</p>

Parameters	Description
	property sets can be used to define the current NOS version requirement. If it changes tomorrow: change the property set value, instead of going under the probe stage.
Enable Streaming (enable_streaming)	Makes samples of output stages streamed if enabled. An optional boolean that defaults to False. If set to True, all output stages of this processor are streamed in the generic protobuf schema.

### Example: Accumulate

Assume a configuration of

```
max_samples: 3
total_duration: 0
```

Assume the following input at time t=1

```
[if_name=eth0] : "up"
[if_name=eth1] : "down"
[if_name=eth3] : "up"
```

We have the following output at time t=1

```
[if_name=eth0] : [{"up", 1 second}]
[if_name=eth1] : [{"down", 1 second}]
[if_name=eth3] : [{"up", 1 second}]
```

Assume the following input at time t=2

```
[if_name=eth0] : "down"
[if_name=eth1] : "down"
[if_name=eth3] : "up"
```

We have the following output at time t=2

```
[if_name=eth0] : [{"up", 1 second}, {"down", 2 seconds}]
[if_name=eth1] : [{"down", 1 second}]
[if_name=eth3] : [{"up", 1 second}]
```

Assume the following input at time t=3

```
[if_name=eth0] : "up"
[if_name=eth1] : "down"
[if_name=eth3] : "up"
```

We have the following output at time t=3

```
[if_name=eth0] : [{"up", 1 second}, {"down", 2 seconds}, {"up", 3 seconds}]
[if_name=eth1] : [{"down", 1 second}]
[if_name=eth3] : [{"up", 1 second}]
```

Assume the following input at time t=4

```
[if_name=eth0] : "down"
[if_name=eth1] : "down"
[if_name=eth3] : "up"
```

We have the following output at time t=4

```
[if_name=eth0] : [{"down", 2 seconds}, {"up", 3 seconds}, {"down", 4 seconds}]
[if_name=eth1] : [{"down", 1 second}]
[if_name=eth3] : [{"up", 1 second}]
```

If the expressions are used for `max_samples` or `total_duration`, then they are evaluated for each input item and the corresponding key is added for each output item.

```
max_samples: context.ref_max_samples * 2
total_duration: context.ref_duration * 2
```

Sample input:

```
[if_name=eth0, ref_max_samples=10, ref_duration=60] : "up"
[if_name=eth1, ref_max_samples=20, ref_duration=120] : "down"
```

Output

```
[if_name=eth0, max_samples=20, duration=120] : "up"
[if_name=eth1, max_samples=40, duration=240] : "down"
```

### Processor: Average

The Average processor groups as described by **Group by**, then calculates averages and outputs one average for each group.

Parameter	Description
Input Types	Table (number), Table (number, accumulate=True)
Output Types	Table(number)
Group by (group_by)	<p>Accepts a list of property names to group input items into output items, produces only one output group for the empty list. Most processors take input and produce output. Many of them produce one output per input (for example, if input is a DSS, output is a DSS of same size). However, some processors reduce the size of the output relative to the size of the input. Effectively, they partition the input into groups, run some calculation on each of the groups that produce a single value per each group, and use that as output. Clearly, the size of the output set depends on the grouping scheme. We call such processors <b>grouping processors</b> and they all take the <b>Group by</b> configuration parameter.</p> <p>In the case of an empty list, the input is considered to be a single group; thus, the output is of size 1 and either N, DS, or TS. If a list of property names is specified, for example ["system_id", "iface_role"], or a single property is specified, for example ["system_id"], we divide the input into groups such that for each group, every item in the group has the same values for the given list of property names. See the <a href="#">"standard deviation processor" on page 1040</a> example for how this works.</p> <p>The output type of a processor depends on a value of the group_by parameter; for an empty list, a processor produces a single value result, such as N, DS, or T, and for grouping by one or more properties it returns a set result, such as NS, DSS, or TS.</p>

(Continued)

Parameter	Description
Enable Streaming (enable_streaming)	Makes samples of output stages streamed if enabled. An optional boolean that defaults to False. If set to True, all output stages of this processor are streamed in the generic protobuf schema.

### Example: Average

See ["standard deviation" on page 1040](#) example. It's the same except we calculate average instead of standard deviation.

### Processor: Comparison

IN THIS SECTION

Example: Comparison | 1000

The Comparison processor takes two Table(number) inputs: 'A' and 'B'. It then matches corresponding items from the inputs by their keys, and performs a comparison operation defined by the 'operation' configuration property. If the inputs have different sets of keys, the 'significant\_keys' configuration property should be set, which is a list of keys used to map items from the inputs. Otherwise, if the inputs set of keys are different, no items will be matched and an empty result is returned. Also, inputs and significant\_keys (if specified) must allow only 1:1 item mapping from 'A' to 'B'. If it allows to match one item from 'A' to more than one item from 'B' and vice versa, the probe goes into error state.

Parameters	Description
Input Types	Tablev(number)
Output Types	Table (discrete state): true or false
Comparison Operation (operation)	Operation for comparing operands. le (less than or equal), ne (not equal), ge (greater than or equal), gt (greater than), lt (less than), eq (equal)

(Continued)

Parameters	Description
Significant Keys (significant_keys)	List of keys to map items from the inputs for applying the specified operation. It is typically used by processors that take multiple inputs and perform operations on them. When inputs have the same sets of keys it does not need to be specified. When inputs have different sets of keys, it must be specified and it must allow only 1:1 items mapping from the given inputs, otherwise the probe will go into error state.
Enable Streaming (enable_streaming)	Makes samples of output stages streamed if enabled. An optional boolean that defaults to False. If set to True, all output stages of this processor are streamed in the generic protobuf schema.

Example: Comparison

```
significant_keys: ["system_id", "interface"]
operation: "ge"
```

Input A:

```
[system_id=leaf1,interface=eth0,counter_type=tx_bytes]: 34
[system_id=leaf1,interface=eth1,counter_type=tx_bytes]: 58
```

Input B:

```
[system_id=leaf1,interface=eth0,counter_type=rx_bytes]: 15
[system_id=leaf1,interface=eth1,counter_type=rx_bytes]: 73
```

Output (Discrete-State-Set):

```
[system_id=leaf1,interface=eth0]: "true"
[system_id=leaf1,interface=eth1]: "false"
```

Processor: EVPN Type 3

The EVPN Type 3 processor generates a configuration containing expectations of EVPN type 3 routes.

Parameter	Description
Input Types	Number-Set (NS), Discrete-State-Set (DSS)
Output Types	NSTS, DSSTS
Execution count	Number of times the data is collected
Monitored VNs	The VNs to be monitored. Specify * to monitor all the VNs or list the desired ones, e.g. "1-3,6,8,10-13".
Service Interval	Telemetry collection interval in seconds.
Service name	Name of the custom collector service.
Enable Streaming (enable_streaming)	Makes samples of output stages streamed if enabled. An optional boolean that defaults to False. If set to True, all output stages of this processor are streamed in the generic protobuf schema.

## Processor: EVPN Type 5

The EVPN Type 5 processor generates a configuration containing expectations of EVPN type 5 routes.

Parameter	Description
Input Types	Number-Set (NS), Discrete-State-Set (DSS)
Output Types	NSTS, DSSTS
Execution count	Number of times the data collection is done.
Service input	Data to pass to telemetry collectors, if any.
Service Interval	Telemetry collection interval in seconds.
Service name	Name of the custom collector service.
Enable Streaming (enable_streaming)	Makes samples of output stages streamed if enabled. An optional boolean that defaults to False. If set to True, all output stages of this processor are streamed in the generic protobuf schema.

## Processor: Extensible Service Data Collector

The Extensible Service Data Collector processor collects data supplied by a custom service that is not 'lldp', 'bgp' or 'interface'.

Parameter	Description
Input Types	No inputs. This is a source processor.
Output Types	NSTS, DSSTS
Data Type	Type of data the service collects: numbers (ns) (such as device temperature), discrete states (dss) (such as device status), text or tables
Graph Query (graph_query)	<p>One or more queries on graph specified as strings, or a list of such queries. (String will be deprecated in a future release.) Multiple queries should provide all the named nodes referenced by the expression fields (including additional_properties). Graph query is executed on the "operation" graph. Results of the queries can be accessed using the "query_result" variable with the appropriate index. For example, if querying property set nodes under name "ps", the result will be available as "query_result[0] ["ps"]".</p> <p>In collector processors (*_collector, if_counter) it is used to choose a set of nodes for further processing (for example, all leafs, or all interfaces between leaf and spines)</p> <p>In other processors it is used for general parameterization and it is only supported as a list of queries.</p> <pre>graph_query: "node("system", role="leaf", name="system").     out("hosted_interfaces").     node("interface", name="iface").out("link").     node("link", role="spine_leaf")"</pre> <pre>graph_query: ["node("system", role="leaf", name="system")",     "node("system", role="spine", name="system)"]</pre> <pre>graph_query: [node("property_set", label="probe_propset", name="ps")] duration: int(query_result[0]["ps"].values["accumulate_duration"])</pre>

*(Continued)*

Parameter	Description
Ingestion filter ( <code>ingestion_filter</code> )	<p>New (reserved) key. Ingestion filter determines what metrics from the collector make it into the probe. We support a degenerate case of ingestion filter, that is, probe specifies full identities of all metrics that need to be ingested. With this feature, you can ingest metrics that satisfy a criterion that is expressed using an ingestion filter.</p> <p>Ingestion filter is authored by probe authors, evaluated by the controller component that is responsible for ingesting raw telemetry into stage outputs within the probes. It is also propagated as a collection filter to the telemetry collector plugins.</p> <p>Keys available to express in the filter are same as the metric identity keys.</p> <ul style="list-style-type: none"> <li>• No metric identity key can exist directly under "properties". If any metric identity key is mistakenly specified directly under properties, a validation error is raised.</li> <li>• Any missing metric identity key under "ingestion_filter" is assumed to match.</li> <li>• Only explicitly specified keys under "ingestion_filter" can be referenced by the rest of the probe configuration. This is to enhance probe readability and allow better overall validation.</li> <li>• The <code>data_type</code> must be one of the table data types.</li> <li>• Existing reserved key "keys" is now made optional and can be omitted. The key names should exactly match those specified in the schema of the corresponding service definition.</li> </ul>
Keys (keys)	List of keys that are significant for specifying data elements for this service
Query Expansion	For every path, originally returned by graph queries, passed to each generator the latter one produces a set of items and for each item it produces a new path extended by a corresponding property name which value is set of a value of the produced item.

(Continued)

Parameter	Description
Query Group by (query_group_by)	<p>List (of strings) of node and relationship names used in the graph query to group query results by. Each element in this list represents a named node or relationship matcher in the graph_query field.It is not an expression to be consistent with existing group_by field in grouping processors. Non-expression is simple and more intuitive.</p> <p>When grouping is active (query_group_by is not null), query results are d by the specified list of names, where one output item is created per each group. In this case, the expressions can only access matcher names specified in query_group_by and the query results for each group are accessed using a new group_items variable. The group_items variable is a list of query results, where each result has named nodes/relationships, not present in query_group_by.</p> <p>The following list describes the behavior for various values of this field:</p> <ul style="list-style-type: none"> <li>• Value of query_group_by field - Semantics</li> <li>• Omitted or provided as json null (ala None in Python) - No grouping is done. This is equivalent to current behavior of extensible_data_collector. Using 'group_items' in this case is not permitted and results in probe error state.</li> <li>• Empty list ([]) - Produces one group containing all the query results.</li> <li>• One or more matcher names - The query results are grouped by the specified nodes or relationships. If this list covers all available matchers in the query, the number of groups is equal to the number of query results.</li> </ul>
Query Tag Filter (query_tag_filter)	Filters named nodes in the graph queries by assigned tags.
Value Map	<p>A mapping of discrete-state values to human readable strings. A dictionary with all possible Discrete-State-Set states mapped to human-readable representation; applicable for Discrete-State-Set data (that is, when data_type is 'dss') only.</p> <pre>{   "0": "unknown",   "1": "down",   "2": "up",   "3": "missing" }</pre>

*(Continued)*

Parameter	Description
Service name (service_name)	Name of the custom collector service.
System ID	Expression mapping from graph query to a system_id, e.g. "system.system_id" if "system" is a name in the graph query.
Execution count	Number of times the data is collected.
Service input (service_input)	Data to pass to telemetry collectors, if any. Can be an expression.
Service interval (service_interval)	Telemetry collection interval in seconds. Can be an expression.
Additional Keys	Each additional key/value pair is used to extend properties of output stages where value is considered as an expression executed in context of the graph query and its result is used as a property value with respective key. The value of this property is evaluated for each item to associate items with metrics provided by a corresponding collector service. The association is done by keys because each collector reports a set of metrics where each metric is identified by a key in a format that is specific for each collector.
Enable Streaming (enable_streaming)	Makes samples of output stages streamed if enabled. An optional boolean that defaults to False. If set to True, all output stages of this processor are streamed in the generic protobuf schema.

## Processor: Generic Graph Collector

### IN THIS SECTION

- [Example: Generic Graph Collector | 1008](#)

The Generic Graph Collector processor imports data from the graph into the output stage, depending on the configuration (a graph query).

'graph query' and 'additional properties' behave as in other source processors. Importantly, the expression in the 'value' field yields a value per each item. Thus, unique to this source processor, values come from the graph rather than from device telemetry.

Parameter	Description
Input Types	No inputs. This is a source processor.
Output Types	Table(discrete state or number or text)
Data Type	Type of data the service collects: numbers (ns) (such as device temperature), discrete states (dss) (such as device status), text or tables
Graph Query (graph_query)	<p>One or more queries on graph specified as strings, or a list of such queries. (String will be deprecated in a future release.) Multiple queries should provide all the named nodes referenced by the expression fields (including additional_properties). Graph query is executed on the "operation" graph. Results of the queries can be accessed using the "query_result" variable with the appropriate index. For example, if querying property set nodes under name "ps", the result will be available as "query_result[0]["ps"]".</p> <p>In collector processors (*_collector, if_counter) it is used to choose a set of nodes for further processing (for example, all leafs, or all interfaces between leaf and spines)</p> <p>In other processors it is used for general parameterization and it is only supported as a list of queries.</p> <pre>graph_query: "node("system", role="leaf", name="system").     out("hosted_interfaces").     node("interface", name="iface").out("link").     node("link", role="spine_leaf")"</pre> <pre>graph_query: ["node("system", role="leaf", name="system")",     "node("system", role="spine", name="system)"]</pre>
Query Expansion	For every path, originally returned by graph queries, passed to each generator the latter one produces a set of items and for each item it produces a new path extended by a corresponding property name which value is set of a value of the produced item.

*(Continued)*

Parameter	Description
Query Group by (query_group_by)	<p>List (of strings) of node and relationship names used in the graph query to group query results by. Each element in this list represents a named node or relationship matcher in the graph_query field. It is not an expression to be consistent with existing group_by field in grouping processors. Non-expression is simple and more intuitive.</p> <p>When grouping is active (query_group_by is not null), query results are divided by the specified list of names, where one output item is created per each group. In this case, the expressions can only access matcher names specified in query_group_by and the query results for each group are accessed using a new group_items variable. The group_items variable is a list of query results, where each result has named nodes/relationships, not present in query_group_by.</p> <p>The following list describes the behavior for various values of this field:</p> <ul style="list-style-type: none"> <li>• Value of query_group_by field - Semantics</li> <li>• Omitted or provided as json null (ala None in Python) - No grouping is done. This is equivalent to current behavior of extensible_data_collector. Using 'group_items' in this case is not permitted and results in probe error state.</li> <li>• Empty list ([]) - Produces one group containing all the query results.</li> <li>• One or more matcher names - The query results are grouped by the specified nodes or relationships. If this list covers all available matchers in the query, the number of groups is equal to the number of query results.</li> </ul>
Query Tag Filter (query_tag_filter)	Filters named nodes in the graph queries by assigned tags.
Value Map	<p>A mapping of discrete-state values to human readable strings. A dictionary with all possible Discrete-State-Set states mapped to human-readable representation; applicable for Discrete-State-Set data (that is, when data_type is 'dss') only.</p> <pre> {   "0": "unknown",   "1": "down",   "2": "up",   "3": "missing" } </pre>

*(Continued)*

Parameter	Description
Value (value)	Expression evaluated per query result to collect value. (integer for NS and string for TS/DSS)
Additional Keys	Each additional key/value pair is used to extend properties of output stages where value is considered as an expression executed in context of the graph query and its result is used as a property value with respective key. The value of this property is evaluated for each item to associate items with metrics provided by a corresponding collector service. The association is done by keys because each collector reports a set of metrics where each metric is identified by a key in a format that is specific for each collector.
Enable Streaming (enable_streaming)	Makes samples of output stages streamed if enabled. An optional boolean that defaults to False. If set to True, all output stages of this processor are streamed in the generic protobuf schema.

### Example: Generic Graph Collector

```
graph_query: "node("system", role="leaf", name="system").
    out("hosted_interfaces").
    node("interface", name="iface").out("link").
    node("link", role="spine_leaf")"
system_id: "system.system_id"
interface: "iface.if_name"
value: "iface.if_type"
data_type: "dss"
value_map: {0: "ip", 1: "loopback", ...}
```

Sample output (DSS):

```
[system_id=leaf1,interface=eth0]: "ip"
[system_id=leaf1,interface=eth1]: "ip"
```

### Processor: Generic Service Data Collector

The Generic Service Data Collector processor collects data supplied by a custom service that is not 'lldp', 'bgp' or 'interface'. Service name is specified as 'service\_name', service specific key is specified as 'key',

'data\_type' to specifies if the collected data is numbers or discrete state values, and 'value\_map' for the specific data could be specified as well.

Parameter	Description
Input Types	No inputs. This is a source processor.
Output Types	Discrete-State-Set (DSS), Number-Set (NS), TS (based on data_type)
Data Type	Type of data the service collects: numbers (ns) (such as device temperature), discrete states (dss) (such as device status), text or tables
Graph Query (graph_query)	<p>One or more queries on graph specified as strings, or a list of such queries. (String will be deprecated in a future release.) Multiple queries should provide all the named nodes referenced by the expression fields (including additional_properties). Graph query is executed on the "operation" graph. Results of the queries can be accessed using the "query_result" variable with the appropriate index. For example, if querying property set nodes under name "ps", the result will be available as "query_result[0] ["ps"]".</p> <p>In collector processors (*_collector, if_counter) it is used to choose a set of nodes for further processing (for example, all leafs, or all interfaces between leaf and spines)</p> <p>In other processors it is used for general parameterization and it is only supported as a list of queries.</p> <pre>graph_query: "node("system", role="leaf", name="system").     out("hosted_interfaces").     node("interface", name="iface").out("link").     node("link", role="spine_leaf")"</pre> <pre>graph_query: ["node("system", role="leaf", name="system")",     "node("system", role="spine", name="system)"]</pre>
Query Expansion	For every path, originally returned by graph queries, passed to each generator the latter one produces a set of items and for each item it produces a new path extended by a corresponding property name which value is set of a value of the produced item.

*(Continued)*

Parameter	Description
Query Group by (query_group_by)	<p>List (of strings) of node and relationship names used in the graph query to group query results by. Each element in this list represents a named node or relationship matcher in the graph_query field. It is not an expression to be consistent with existing group_by field in grouping processors. Non-expression is simple and more intuitive.</p> <p>When grouping is active (query_group_by is not null), query results are divided by the specified list of names, where one output item is created per each group. In this case, the expressions can only access matcher names specified in query_group_by and the query results for each group are accessed using a new group_items variable. The group_items variable is a list of query results, where each result has named nodes/relationships, not present in query_group_by.</p> <p>The following list describes the behavior for various values of this field:</p> <ul style="list-style-type: none"> <li>• Value of query_group_by field - Semantics</li> <li>• Omitted or provided as json null (ala None in Python) - No grouping is done. This is equivalent to current behavior of extensible_data_collector. Using 'group_items' in this case is not permitted and results in probe error state.</li> <li>• Empty list ([]) - Produces one group containing all the query results.</li> <li>• One or more matcher names - The query results are grouped by the specified nodes or relationships. If this list covers all available matchers in the query, the number of groups is equal to the number of query results.</li> </ul>
Query Tag Filter (query_tag_filter)	Filters named nodes in the graph queries by assigned tags.
Value Map	<p>A mapping of discrete-state values to human readable strings. A dictionary with all possible Discrete-State-Set states mapped to human-readable representation; applicable for Discrete-State-Set data (that is, when data_type is 'dss') only.</p> <pre>{   "0": "unknown",   "1": "down",   "2": "up",   "3": "missing" }</pre>

*(Continued)*

Parameter	Description
Key (key)	Expression mapping from graph query to whatever key is necessary for the service.
Service name (service_name)	Name of the custom collector service.
System ID	Expression mapping from graph query to a system_id, e.g. "system.system_id" if "system" is a name in the graph query.
Execution count	Number of times the data collection is done.
Service input (service_input)	Data to pass to telemetry collectors, if any. Can be an expression.
Service interval (service_interval)	Telemetry collection interval in seconds. Can be an expression.
Additional Keys	Each additional key/value pair is used to extend properties of output stages where value is considered as an expression executed in context of the graph query and its result is used as a property value with respective key. The value of this property is evaluated for each item to associate items with metrics provided by a corresponding collector service. The association is done by keys because each collector reports a set of metrics where each metric is identified by a key in a format that is specific for each collector.
Enable Streaming (enable_streaming)	Makes samples of output stages streamed if enabled. An optional boolean that defaults to False. If set to True, all output stages of this processor are streamed in the generic protobuf schema.

## Processor: Interface Counters

### IN THIS SECTION

- [Example: Interface Counter | 1014](#)

The Interface Counters processor selects interfaces according to the configuration and outputs counter stats of the specified types (such as 'tx\_bytes').

Parameter	Description
Input Types	No inputs. This is a source processor.
Output Types	Table(number)
Counter Type (counter_type)	A type of an interface counter. enum of: tx_unicast_packets, tx_broadcast_packets, tx_multicast_packets, tx_bytes, tx_error_packets, tx_discard_packets, rx_unicast_packets, rx_broadcast_packets, rx_multicast_packets, rx_bytes, rx_error_packets, rx_discard_packets.
Graph Query (graph_query)	<p>One or more queries on graph specified as strings, or a list of such queries. (String will be deprecated in a future release.) Multiple queries should provide all the named nodes referenced by the expression fields (including additional_properties). Graph query is executed on the "operation" graph. Results of the queries can be accessed using the "query_result" variable with the appropriate index. For example, if querying property set nodes under name "ps", the result will be available as "query_result[0]["ps"]".</p> <p>In collector processors (*_collector, if_counter) it is used to choose a set of nodes for further processing (for example, all leafs, or all interfaces between leaf and spines)</p> <p>In other processors it is used for general parameterization and it is only supported as a list of queries.</p> <pre>graph_query: "node("system", role="leaf", name="system").     out("hosted_interfaces").     node("interface", name="iface").out("link").     node("link", role="spine_leaf")"</pre> <pre>graph_query: ["node("system", role="leaf", name="system")",     "node("system", role="spine", name="system)"]</pre>
Query Expansion	For every path, originally returned by graph queries, passed to each generator the latter one produces a set of items and for each item it produces a new path extended by a corresponding property name which value is set of a value of the produced item.

*(Continued)*

Parameter	Description
Query Group by (query_group_by)	<p>List (of strings) of node and relationship names used in the graph query to group query results by. Each element in this list represents a named node or relationship matcher in the graph_query field. It is not an expression to be consistent with existing group_by field in grouping processors. Non-expression is simple and more intuitive.</p> <p>When grouping is active (query_group_by is not null), query results are divided by the specified list of names, where one output item is created per each group. In this case, the expressions can only access matcher names specified in query_group_by and the query results for each group are accessed using a new group_items variable. The group_items variable is a list of query results, where each result has named nodes/relationships, not present in query_group_by.</p> <p>The following list describes the behavior for various values of this field:</p> <ul style="list-style-type: none"> <li>• Value of query_group_by field - Semantics</li> <li>• Omitted or provided as json null (ala None in Python) - No grouping is done. This is equivalent to current behavior of extensible_data_collector. Using 'group_items' in this case is not permitted and results in probe error state.</li> <li>• Empty list ([]) - Produces one group containing all the query results.</li> <li>• One or more matcher names - The query results are grouped by the specified nodes or relationships. If this list covers all available matchers in the query, the number of groups is equal to the number of query results.</li> </ul>
Query Tag Filter (query_tag_filter)	Filters named nodes in the graph queries by assigned tags.
Interface (interface)	Expression mapping from graph query to interface name, e.g. "iface.if_name" if "iface" is a name in the graph query.
System ID	Expression mapping from graph query to a system_id, e.g. "system.system_id" if "system" is a name in the graph query.
Additional Keys	Each additional key/value pair is used to extend properties of output stages where value is considered as an expression executed in context of the graph query and its result is used as a property value with respective key. The value of this property is evaluated for each item to associate items with metrics provided by a corresponding collector service. The association is done by keys because each collector reports a set of metrics where each metric is identified by a key in a format that is specific for each collector.

*(Continued)*

Parameter	Description
Enable Streaming (enable_streaming)	Makes samples of output stages streamed if enabled. An optional boolean that defaults to False. If set to True, all output stages of this processor are streamed in the generic protobuf schema.

**Example: Interface Counter**

```

graph_query: "node(\"system\", name=\"system\").out(\"hosted_interfaces\").
              node(\"interface\", name=\"iface\").out(\"link\").
              node(\"link\", role=\"spine_leaf\")"
counter_type: "rx_bytes"
system_id: "system.system_id"
interface: "interface.if_name"
role: "system.role"

```

In this example, we create a NSS that has an entry for rx\_bytes (per second) per every interface in the system. Each entry is implicitly tagged by "system\_id" and "interface". Furthermore, as we have specified an additional property, each entry is also tagged by role of the system.

```

[system_id=spine1,role=spine,key=eth0]: 10
[system_id=spine2,role=spine,key=eth1]: 11
[system_id=leaf0,role=leaf, key=swp1]: 12

```

**Processor: Logical Operator**

(New in version 4.0) The Logical Operator processor calculates the logical operation of inputs. It takes two or more inputs that represent boolean values.

The property 'operation' specifies the logical operation. The property 'input\_columns' specifies column names that input items should be taken from.

Parameter	Description
Input Types	Tables that contain discrete_state type column according to the 'input_columns' property or Table (discrete_state) if the 'input_columns' is not specified.

*(Continued)*

Parameter	Description
Output Types	Table (discrete state)
Operation	Logical operation type that is used for processing the input data
Significant Keys (significant_keys)	List of keys to map items from the inputs for applying the specified operation. It is typically used by processors that take multiple inputs and perform operations on them. When inputs have the same sets of keys it does not need to be specified. When inputs have different sets of keys, it must be specified and it must allow only 1:1 items mapping from the given inputs, otherwise the probe will go into error state.
Enable Streaming (enable_streaming)	Makes samples of output stages streamed if enabled. An optional boolean that defaults to False. If set to True, all output stages of this processor are streamed in the generic protobuf schema.

## Processor: Match Count

### IN THIS SECTION

- [Example: Match Count | 1016](#)

For each input group, the Match Count processor creates a single output that is the number of items in the input group that are equal to the reference. The 'total\_count' key is added into output item keys where the value is a number of items in an input group.

Parameter	Description
Input Types	Table(text or discrete state)
Output Types	NS

*(Continued)*

Parameter	Description
Group by (group_by)	<p>Accepts a list of property names to group input items into output items, produces only one output group for the empty list. Most processors take input and produce output. Many of them produce one output per input (for example, if input is a DSS, output is a DSS of same size). However, some processors reduce the size of the output relative to the size of the input. Effectively, they partition the input into groups, run some calculation on each of the groups that produce a single value per each group, and use that as output. Clearly, the size of the output set depends on the grouping scheme. We call such processors <b>grouping processors</b> and they all take the <b>Group by</b> configuration parameter.</p> <p>In the case of an empty list, the input is considered to be a single group; thus, the output is of size 1 and either N, DS, or TS. If a list of property names is specified, for example ["system_id", "iface_role"], or a single property is specified, for example ["system_id"], we divide the input into groups such that for each group, every item in the group has the same values for the given list of property names. See the standard deviation processor&lt;processor_standard_deviation&gt; example for how this works.</p> <p>The output type of a processor depends on a value of the group_by parameter; for an empty list, a processor produces a single value result, such as N, DS, or T, and for grouping by one or more properties it returns a set result, such as NS, DSS, or TS.</p>
Reference State (reference_state)	DS or TS value which is used as a reference state to match input samples. discrete-state value
Enable Streaming (enable_streaming)	Makes samples of output stages streamed if enabled. An optional boolean that defaults to False. If set to True, all output stages of this processor are streamed in the generic protobuf schema.

**Example: Match Count**

Assume a configuration of:

```
reference_state: "false"
group_by: []
```

Sample Input:

```
[if_name=eth0] : "true"
[if_name=eth1] : "true"
[if_name=eth3] : "false"
```

Sample Output:

```
[] : 1
```

In the above example, we have 1 as the output because 1 element of the input group matches the reference value of "false".

Processor: Match Percentage

IN THIS SECTION

[Example: Match Percentage | 1018](#)

For each input group, the Match Percentage processor creates a single output that is the percentage of items in the input group that are equal to the reference.

Parameter	Description
Input Types	Table(text or discrete state)
Output Types	Table(number)

*(Continued)*

Parameter	Description
Group by (group_by)	<p>Accepts a list of property names to group input items into output items, produces only one output group for the empty list. Most processors take input and produce output. Many of them produce one output per input (for example, if input is a DSS, output is a DSS of same size). However, some processors reduce the size of the output relative to the size of the input. Effectively, they partition the input into groups, run some calculation on each of the groups that produce a single value per each group, and use that as output. Clearly, the size of the output set depends on the grouping scheme. We call such processors <b>grouping processors</b> and they all take the <b>Group by</b> configuration parameter.</p> <p>In the case of an empty list, the input is considered to be a single group; thus, the output is of size 1 and either N, DS, or TS. If a list of property names is specified, for example ["system_id", "iface_role"], or a single property is specified, for example ["system_id"], we divide the input into groups such that for each group, every item in the group has the same values for the given list of property names. See the standard deviation processor &lt;processor_standard_deviation&gt; example for how this works.</p> <p>The output type of a processor depends on a value of the group_by parameter; for an empty list, a processor produces a single value result, such as N, DS, or T, and for grouping by one or more properties it returns a set result, such as NS, DSS, or TS.</p>
Reference State (reference_state)	DS or TS value which is used as a reference state to match input samples.
Enable Streaming (enable_streaming)	Makes samples of output stages streamed if enabled. An optional boolean that defaults to False. If set to True, all output stages of this processor are streamed in the generic protobuf schema.

**Example: Match Percentage**

Assume a configuration of:

```
reference_state: "false"
group_by: []
```

Sample Input:

```
[if_name=eth0] : "true"
[if_name=eth1] : "true"
[if_name=eth3] : "false"
```

Sample Output:

```
[] : 33
```

In the above example, we have 33% as the output because 33% of the input group match the reference value of "false".

Processor: Match String

IN THIS SECTION

[Example: Match String | 1021](#)

The Max String processor checks that a string matches a regular expression. It accepts text series on input, for each series it configures a check that verifies if the input value matches the configured regular expression. Regular expression syntax is PCRE-compatible. Note that regexp matching is done in a partial mode, so if the full match is needed, regular expression needs to be specified accordingly. The output series contains anomaly values, such as 'false' and 'true'.

Parameter	Description
Input Types	Time-Series (TS), TSTS
Output Types	Table(discrete state)

*(Continued)*

Parameter	Description
Graph Query (graph_query)	<p>One or more queries on graph specified as strings, or a list of such queries. (String will be deprecated in a future release.) Multiple queries should provide all the named nodes referenced by the expression fields (including additional_properties). Graph query is executed on the "operation" graph. Results of the queries can be accessed using the "query_result" variable with the appropriate index. For example, if querying property set nodes under name "ps", the result will be available as "query_result[0]["ps"]".</p> <p>In collector processors (*_collector, if_counter) it is used to choose a set of nodes for further processing (for example, all leafs, or all interfaces between leaf and spines)</p> <p>In other processors it is used for general parameterization and it is only supported as a list of queries.</p> <pre>graph_query: "node("system", role="leaf", name="system").               out("hosted_interfaces").               node("interface", name="iface").out("link").               node("link", role="spine_leaf")"</pre> <pre>graph_query: ["node("system", role="leaf", name="system")",               "node("system", role="spine", name="system)"]</pre> <p>Non-collector processors containing the graph_query configuration parameter, can be parameterized to use data from arbitrary nodes in the graph, such as property set nodes. Property sets allow you to parameterize macro level SLAs for individual business units. In the example below, graph_query matches a node of type property_set with label probe_propset. It's accessed using the special query_result variable, where Index 0 means it's the first node in query results. If a query returned N nodes, they could be accessed using indices starting from 0 to N-1. ps is what the actual node is referred to in the query; the rest depends on the structure of the node. The int() casting is required because values of property_set nodes are strings. Here it's assumed that a property set node has the label probe_propset and that the value accumulate_duration was already created.</p> <pre>graph_query: [node("property_set", label="probe_propset", name="ps")] duration: int(query_result[0]["ps"].values["accumulate_duration"])</pre>

*(Continued)*

Parameter	Description
	Another example is a that probes can validate a compliance requirement; the compliance value may change over time and/or it can be used by more than one probe. Also, a probe can validate NOS versions on devices. In this case, property sets can be used to define the current NOS version requirement. If it changes tomorrow: change the property set value, instead of going under the probe stage.
Regular Expression (regexp)	Expression that evaluates to a PCRE-compatible regular expression.
Anomaly MetricLog Retention Duration	Retain anomaly metric data in MetricDb for specified duration in seconds
Anomaly MetricLog Retention Size	Maximum allowed size, in bytes of anomaly metric data to store in MetricDB
Anomaly Metric Logging	Enable metric logging for anomalies
Enable Streaming (enable_streaming)	Makes samples of output stages streamed if enabled. An optional boolean that defaults to False. If set to True, all output stages of this processor are streamed in the generic protobuf schema.
Raise Anomaly (raise_anomaly)	Outputs "true" and "false" values, "true" meaning an appropriate item is anomalous, and "false" meaning the item is not anomalous. When Raise Anomaly is set to True, an actual anomaly is generated in addition to a sample in the output.

**Example: Match String**

```
regexp: "os_version_pattern"
```

**Sample Input (TS)**

```
[device=leaf1,os_version_pattern=^4.[7-9].[0-9]+$] : 4.1
[device=leaf2,os_version_pattern=^4.[7-9].[0-9]+$] : 4.7
```

Sample Output (DSS):

```
[device=leaf1,os_version_pattern=^4.[7-9].[0-9]+$ ,regex=^4.[7-9].[0-9]+$] : "true"
[device=leaf2,os_version_pattern=^4.[7-9].[0-9]+$ ,regex=^4.[7-9].[0-9]+$] : "false"
```

Processor: Max

IN THIS SECTION

Example: Max | 1023

The Max processor groups as described by **Group by**, then finds the maximum value and outputs it for each group.

Parameter	Description
Input Types	Table (number), Table (number, accumulate=True)
Output Types	Table (number)

*(Continued)*

Parameter	Description
Group by (group_by)	<p>Accepts a list of property names to group input items into output items, produces only one output group for the empty list. Most processors take input and produce output. Many of them produce one output per input (for example, if input is a DSS, output is a DSS of same size). However, some processors reduce the size of the output relative to the size of the input. Effectively, they partition the input into groups, run some calculation on each of the groups that produce a single value per each group, and use that as output. Clearly, the size of the output set depends on the grouping scheme. We call such processors <b>grouping processors</b> and they all take the <b>Group by</b> configuration parameter.</p> <p>In the case of an empty list, the input is considered to be a single group; thus, the output is of size 1 and either N, DS, or TS. If a list of property names is specified, for example ["system_id", "iface_role"], or a single property is specified, for example ["system_id"], we divide the input into groups such that for each group, every item in the group has the same values for the given list of property names. See the <a href="#">"standard deviation processor" on page 1040</a> example for how this works.</p> <p>The output type of a processor depends on a value of the group_by parameter; for an empty list, a processor produces a single value result, such as N, DS, or T, and for grouping by one or more properties it returns a set result, such as NS, DSS, or TS.</p>
Enable Streaming (enable_streaming)	Makes samples of output stages streamed if enabled. An optional boolean that defaults to False. If set to True, all output stages of this processor are streamed in the generic protobuf schema.

**Example: Max**

Assume a configuration of:

```
group_by: ["system_id"]
```

Sample Input:

```
[system_id=leaf0,if_name=swp40] : 10
[system_id=leaf0,if_name=swp41] : 11
[system_id=leaf0,if_name=swp42] : 15
[system_id=spine0,if_name=eth15] : 32
```

```
[system_id=spine0,if_name=eth16] : 30
[system_id=spine0,if_name=eth17] : 36
```

Output "out":

```
[system_id=leaf0] : 15
[system_id=spine0] : 36
```

Processor: Min

IN THIS SECTION

Example: Min | 1025

The Min processor groups as described in **Group by**, then finds the minimum value and outputs it for each group.

Parameter	Description
Input Types	Table (number), Table (number, accumulate=True)
Output Types	Table (number)

*(Continued)*

Parameter	Description
Group by (group_by)	<p>Accepts a list of property names to group input items into output items, produces only one output group for the empty list. Most processors take input and produce output. Many of them produce one output per input (for example, if input is a DSS, output is a DSS of same size). However, some processors reduce the size of the output relative to the size of the input. Effectively, they partition the input into groups, run some calculation on each of the groups that produce a single value per each group, and use that as output. Clearly, the size of the output set depends on the grouping scheme. We call such processors <b>grouping processors</b> and they all take the <b>Group by</b> configuration parameter.</p> <p>In the case of an empty list, the input is considered to be a single group; thus, the output is of size 1 and either N, DS, or TS. If a list of property names is specified, for example ["system_id", "iface_role"], or a single property is specified, for example ["system_id"], we divide the input into groups such that for each group, every item in the group has the same values for the given list of property names. See the <a href="#">"standard deviation processor" on page 1040</a> example for how this works.</p> <p>The output type of a processor depends on a value of the group_by parameter; for an empty list, a processor produces a single value result, such as N, DS, or T, and for grouping by one or more properties it returns a set result, such as NS, DSS, or TS.</p>
Enable Streaming (enable_streaming)	Makes samples of output stages streamed if enabled. An optional boolean that defaults to False. If set to True, all output stages of this processor are streamed in the generic protobuf schema.

**Example: Min**

Assume a configuration of:

```
group_by: ["system_id"]
```

Sample Input:

```
[system_id=leaf0,if_name=swp40] : 10
[system_id=leaf0,if_name=swp41] : 11
[system_id=leaf0,if_name=swp42] : 15
[system_id=spine0,if_name=eth15] : 32
```

```
[system_id=spine0,if_name=eth16] : 30
[system_id=spine0,if_name=eth17] : 36
```

Output "out":

```
[system_id=leaf0] : 10
[system_id=spine0] : 30
```

Processor: Periodic Average

IN THIS SECTION

[Example: Periodic Average](#) | 1028

One number is created on output for each input. Each <period>, the output is set to the average of the input over the last <period>. This is not a weighted average.

Parameter	Description
Input Types	Table (number)
Output Types	Table (number)
Period	Size of the averaging period. (time in seconds, integer, or an expression that evaluates to time in seconds integer value)

*(Continued)*

Parameter	Description
Graph Query (graph_query)	<p>One or more queries on graph specified as strings, or a list of such queries. (String will be deprecated in a future release.) Multiple queries should provide all the named nodes referenced by the expression fields (including additional_properties). Graph query is executed on the "operation" graph. Results of the queries can be accessed using the "query_result" variable with the appropriate index. For example, if querying property set nodes under name "ps", the result will be available as "query_result[0]["ps"]".</p> <p>In collector processors (*_collector, if_counter) it is used to choose a set of nodes for further processing (for example, all leafs, or all interfaces between leaf and spines)</p> <p>In other processors it is used for general parameterization and it is only supported as a list of queries.</p> <pre>graph_query: "node("system", role="leaf", name="system").               out("hosted_interfaces").               node("interface", name="iface").out("link").               node("link", role="spine_leaf")"</pre> <pre>graph_query: ["node("system", role="leaf", name="system"),               "node("system", role="spine", name="system)"]</pre> <p>Non-collector processors containing the graph_query configuration parameter, can be parameterized to use data from arbitrary nodes in the graph, such as property set nodes. Property sets allow you to parameterize macro level SLAs for individual business units. In the example below, graph_query matches a node of type property_set with label probe_propset. It's accessed using the special query_result variable, where Index 0 means it's the first node in query results. If a query returned N nodes, they could be accessed using indices starting from 0 to N-1. ps is what the actual node is referred to in the query; the rest depends on the structure of the node. The int() casting is required because values of property_set nodes are strings. Here it's assumed that a property set node has the label probe_propset and that the value accumulate_duration was already created.</p> <pre>graph_query: [node("property_set", label="probe_propset", name="ps")] duration: int(query_result[0]["ps"].values["accumulate_duration"])</pre>

*(Continued)*

Parameter	Description
	Another example is a that probes can validate a compliance requirement; the compliance value may change over time and/or it can be used by more than one probe. Also, a probe can validate NOS versions on devices. In this case, property sets can be used to define the current NOS version requirement. If it changes tomorrow: change the property set value, instead of going under the probe stage.
Enable Streaming (enable_streaming)	Makes samples of output stages streamed if enabled. An optional boolean that defaults to False. If set to True, all output stages of this processor are streamed in the generic protobuf schema.

### Example: Periodic Average

```
period: 2
```

Assume the following input at time t=1

```
[if_name=eth0] : 10
[if_name=eth1] : 20
[if_name=eth3] : 30
```

And following input at time t=1.5

```
[if_name=eth0] : 20
[if_name=eth1] : 30
[if_name=eth3] : 40
```

And the following at time t=2.1

```
[if_name=eth0] : 40
[if_name=eth1] : 50
[if_name=eth3] : 60
```

We would now have the following output:

```
[if_name=eth0] : 15
[if_name=eth1] : 25
[if_name=eth3] : 35
```

This output is the average over the last discrete period of 2 seconds (time=0 to time=2). Notice that the average is not weighted by time; frequently-occurring closely-spaced samples will bias the average.

The next time the output would be updated would be at time t=4, in which case it would contain the average of the input over the range [t=2, t=4], a period of the configured two seconds.

## Processor: Range

### IN THIS SECTION

- [Example: Range | 1031](#)

The Range processor checks that a value is in a range. According to the specified range, it configures a check for the input series. This check returns an anomaly value if a series aggregation value, such as a last value, sum, avg etc., is in the range. This aggregation type is configured by the 'property' attribute, which is set to 'value' if not specified. The output series contains anomaly values, such as 'true' and 'false'. (Previously called 'not\_in\_range' and 'range\_check'.) The range processor generates the output of True when the input matches the specified criteria.

Parameter	Description
Input Types	Table (number), Table (number, accumulate=True)
Output Types	Table (discrete state)
Property	A property of input items which is used to check against the range. Enum of either value, sample_count, sum, avg
Anomalous Range (range)	Numeric range, either min or max is optional. Float type is acceptable only with property "std_dev", other property values require integers. Min and max can be expressions evaluated into numeric values.

*(Continued)*

Parameter	Description
Graph Query (graph_query)	<p>One or more queries on graph specified as strings, or a list of such queries. (String will be deprecated in a future release.) Multiple queries should provide all the named nodes referenced by the expression fields (including additional_properties). Graph query is executed on the "operation" graph. Results of the queries can be accessed using the "query_result" variable with the appropriate index. For example, if querying property set nodes under name "ps", the result will be available as "query_result[0]["ps"]".</p> <p>In collector processors (*_collector, if_counter) it is used to choose a set of nodes for further processing (for example, all leafs, or all interfaces between leaf and spines)</p> <p>In other processors it is used for general parameterization and it is only supported as a list of queries.</p> <pre>graph_query: "node("system", role="leaf", name="system").     out("hosted_interfaces").     node("interface", name="iface").out("link").     node("link", role="spine_leaf")"</pre> <pre>graph_query: ["node("system", role="leaf", name="system")",     "node("system", role="spine", name="system")"]</pre> <p>Non-collector processors containing the graph_query configuration parameter, can be parameterized to use data from arbitrary nodes in the graph, such as property set nodes. Property sets allow you to parameterize macro level SLAs for individual business units. In the example below, graph_query matches a node of type property_set with label probe_propset. It's accessed using the special query_result variable, where Index 0 means it's the first node in query results. If a query returned N nodes, they could be accessed using indices starting from 0 to N-1. ps is what the actual node is referred to in the query; the rest depends on the structure of the node. The int() casting is required because values of property_set nodes are strings. Here it's assumed that a property set node has the label probe_propset and that the value accumulate_duration was already created.</p> <pre>graph_query: [node("property_set", label="probe_propset", name="ps")] duration: int(query_result[0]["ps"].values["accumulate_duration"])</pre>

*(Continued)*

Parameter	Description
	Another example is a that probes can validate a compliance requirement; the compliance value may change over time and/or it can be used by more than one probe. Also, a probe can validate NOS versions on devices. In this case, property sets can be used to define the current NOS version requirement. If it changes tomorrow: change the property set value, instead of going under the probe stage.
Anomaly MetricLog Retention Duration	Retain anomaly metric data in MetricDb for specified duration in seconds
Anomaly MetricLog Retention Size	Maximum allowed size, in bytes of anomaly metric data to store in MetricDB
Anomaly Metric Logging	Enable metric logging for anomalies
Enable Streaming (enable_streaming)	Makes samples of output stages streamed if enabled. An optional boolean that defaults to False. If set to True, all output stages of this processor are streamed in the generic protobuf schema.
Raise Anomaly (raise_anomaly)	Outputs "true" and "false" values, "true" meaning an appropriate item is anomalous, and "false" meaning the item is not anomalous. When Raise Anomaly is set to True, an actual anomaly is generated in addition to a sample in the output.

**Example: Range**

```
range: {"min": 35, "max": 45}
property: "value"
```

**Sample Input (NS)**

```
[if_name=eth0] : 23
[if_name=eth1] : 55
[if_name=eth3] : 37
```

### Sample Output (DSS)

```
[if_name=eth0] : "false"
[if_name=eth1] : "false"
[if_name=eth3] : "true"
```

If expressions are used for min or max fields of the range property, then they are evaluated for each input item which results into item-specific thresholds. Properties of the respective output item are extended by range\_min or range\_max properties with calculated values.

```
range: {"max": "speed * 0.7"}
property: "value"
```

### Sample Input (NS)

```
[if_name=eth0,speed=10000000000] : 800000000
[if_name=eth1,speed=10000000000] : 800000000
```

### Sample Output (DSS)

```
if_name=eth0,speed=10000000000,range_max=7000000000] : "false"
[if_name=eth1,speed=10000000000,range_max=7000000000] : "true"
```

## Processor: Ratio

### IN THIS SECTION

- [Example: Ratio Output | 1033](#)

The Ratio processor calculates the ratio of inputs. It takes two inputs: numerator and denominator. Denominator is optional and could be specified as 'denominator' configuration property instead. It could be either an integer or an expression that evaluates to an integer. It should not be '0'.

When 'denominator' is specified as an input, 'numerator' and 'denominator' input items must allow only 1:1 mapping. If that is not the case, 'significant\_keys' configuration property should be specified to list keys that will allow such mapping.

It also supports 'multiplier' configuration property, which is an integer value greater than one to multiply numerator by before calculating ratio. This allows it to overcome limitations of dealing with integers. Default value is 100.

Parameter	Description
Input Types	Table (number)
Output Types	Table (number)
Denominator	Integer or an expression that evaluates to integer that is used as denominator. Optional denominator value if it's not specified as input; should be non-zero integer or an expression that evaluates to non-zero integer.
Significant Keys (significant_keys)	List of keys to map items from the inputs for applying the specified operation. It is typically used by processors that take multiple inputs and perform operations on them. When inputs have the same sets of keys it does not need to be specified. When inputs have different sets of keys, it must be specified and it must allow only 1:1 items mapping from the given inputs, otherwise the probe will go into error state.
Multiplier	Multiply numerator by a given value before calculating ratio. Optional. Default is 100.
Enable Streaming (enable_streaming)	Makes samples of output stages streamed if enabled. An optional boolean that defaults to False. If set to True, all output stages of this processor are streamed in the generic protobuf schema.

### Example: Ratio Output

Simple scenario with a static denominator.

```
denominator: 100
multiplier: 1
```

Input 'numerator':

```
[system_id=spine1,role=spine,interface=eth0]: 300
[system_id=spine2,role=spine,interface=eth1]: 500
```

Output:

```
[system_id=spine1,role=spine,interface=eth0]: 3
[system_id=spine1,role=spine,interface=eth1]: 5
```

Configuration where numerator and denominator are coming from inputs, and 'multiplier' value is the default 100:

```
significant_keys: ['system_id', 'interface']
```

Input 'numerator':

```
[system_id=spine1,role=spine,interface=eth0]: 300
[system_id=spine2,role=spine,interface=eth1]: 750
```

Input 'denominator':

```
[system_id=spine1,role=spine,interface=eth0]: 150
[system_id=spine2,role=spine,interface=eth1]: 250
```

Output:

```
[system_id=spine1,interface=eth0]: 200
[system_id=spine1,interface=eth1]: 300
```

## Processor: Service Data Collector

### IN THIS SECTION

- [Example: Service Data Collector | 1037](#)

The Service Data Collector processor collects data from the specified service. For example, 'bgp' service would be the status of BGP sessions. Objects to be monitored are configured via the graph query and key. In the BGP example, key should evaluate to localIp, localAs, remoteIp, or remote As. For interface-based services such as 'interface' and 'lldp', key is an interface name.

Parameter	Description
Input Types	No inputs. This is a source processor.
Output Types	Table (number or discrete state)
Graph Query (graph_query)	<p>One or more queries on graph specified as strings, or a list of such queries. (String will be deprecated in a future release.) Multiple queries should provide all the named nodes referenced by the expression fields (including additional_properties). Graph query is executed on the "operation" graph. Results of the queries can be accessed using the "query_result" variable with the appropriate index. For example, if querying property set nodes under name "ps", the result will be available as "query_result[0]["ps"]".</p> <p>In collector processors (*_collector, if_counter) it is used to choose a set of nodes for further processing (for example, all leafs, or all interfaces between leaf and spines)</p> <p>In other processors it is used for general parameterization and it is only supported as a list of queries.</p> <pre>graph_query: "node("system", role="leaf", name="system").     out("hosted_interfaces").     node("interface", name="iface").out("link").     node("link", role="spine_leaf")"</pre> <pre>graph_query: ["node("system", role="leaf", name="system")",     "node("system", role="spine", name="system)"]</pre>
Service name (service_name)	Name of the custom collector service.
Keys	<p>List of property names which values will be used as a key parameters for the service. Expression mapping from graph query to whatever key is necessary for the service. For lldp it's a string with interface name. For bgp it's a tuple like (src_addr, src_asn, dst_addr, dst_asn, vrf_name, addr_family), where addr_family should be one of ipv4, ipv6, or evpn. For interface it is a string with interface name.</p>
Query Expansion	For every path, originally returned by graph queries, passed to each generator the latter one produces a set of items and for each item it produces a new path extended by a corresponding property name which value is set of a value of the produced item.

*(Continued)*

Parameter	Description
Query Group by (query_group_by)	<p>List (of strings) of node and relationship names used in the graph query to group query results by. Each element in this list represents a named node or relationship matcher in the graph_query field. It is not an expression to be consistent with existing group_by field in grouping processors. Non-expression is simple and more intuitive.</p> <p>When grouping is active (query_group_by is not null), query results are divided by the specified list of names, where one output item is created per each group. In this case, the expressions can only access matcher names specified in query_group_by and the query results for each group are accessed using a new group_items variable. The group_items variable is a list of query results, where each result has named nodes/relationships, not present in query_group_by.</p> <p>The following list describes the behavior for various values of this field:</p> <ul style="list-style-type: none"> <li>• Value of query_group_by field - Semantics</li> <li>• Omitted or provided as json null (ala None in Python) - No grouping is done. This is equivalent to current behavior of extensible_data_collector. Using 'group_items' in this case is not permitted and results in probe error state.</li> <li>• Empty list ([]) - Produces one group containing all the query results.</li> <li>• One or more matcher names - The query results are grouped by the specified nodes or relationships. If this list covers all available matchers in the query, the number of groups is equal to the number of query results.</li> </ul>
Query Tag Filter (query_tag_filter)	Filters named nodes in the graph queries by assigned tags.
System ID	Expression mapping from graph query to a system_id, e.g. "system.system_id" if "system" is a name in the graph query.
Additional Keys	Each additional key/value pair is used to extend properties of output stages where value is considered as an expression executed in context of the graph query and its result is used as a property value with respective key. The value of this property is evaluated for each item to associate items with metrics provided by a corresponding collector service. The association is done by keys because each collector reports a set of metrics where each metric is identified by a key in a format that is specific for each collector.

*(Continued)*

Parameter	Description
Enable Streaming (enable_streaming)	Makes samples of output stages streamed if enabled. An optional boolean that defaults to False. If set to True, all output stages of this processor are streamed in the generic protobuf schema.

**Example: Service Data Collector**

```
ode("system", name="system").out("hosted_interfaces").
    node("interface", name="iface").out("link").
    node("link", role="spine_leaf")"
system_id: "system.system_id"
key: "interface.if_name"
role: "system.role"
```

In this example, we create a DSS that has an entry for every fabric interface in the system. Each entry is implicitly tagged by "system\_id" and "key" (where key happens to be the interface name for the interface service). Furthermore, as we have specified an additional property "role", each entry is also tagged by system role.

```
[system_id=spine1,role=spine,key=eth0]: "up"
[system_id=spine2,role=spine,key=eth1]: "down"
[system_id=leaf0,role=leaf, key=swp1]: "up"
```

**Processor: Set Comparison****IN THIS SECTION**

- [Example: Set Comparison | 1038](#)

The Set Comparison processor does a set-comparison of input stages.

Accept two DS or NS inputs, called "A" and "B". There are three outputs: A stage "A - B" that contains the items that are only in stage "A," a stage "B - A" that contains the items that are only in stage "B," and a stage "A & B" that contains the items that are in both stage "A" and stage "B."

When conducting the above operations, we first normalize all items in each stage by dropping all the keys that are not in "significant\_keys." It is an error if a key in "significant\_keys" is not present in either stage "A" or "B."

Furthermore, only the keys of each normalized item are considered; values are preserved (and kept from stage "A" in the intersection output), but not considered in the comparison operations.

Results are undefined if, when normalizing items in either stage\_A or stage\_B, there is more-than-one item with a given set of key-value pairs.

Parameter	Description
Input Types	Table (number or discrete state)
Significant Keys (significant_keys)	List of keys to map items from the inputs for applying the specified operation. It is typically used by processors that take multiple inputs and perform operations on them. When inputs have the same sets of keys it does not need to be specified. When inputs have different sets of keys, it must be specified and it must allow only 1:1 items mapping from the given inputs, otherwise the probe will go into error state.
Enable Streaming (enable_streaming)	Makes samples of output stages streamed if enabled. An optional boolean that defaults to False. If set to True, all output stages of this processor are streamed in the generic protobuf schema.

### Example: Set Comparison

Consider we have inputs with device temperature information.

Input A:

```
[system_id=leaf1]: 45
[system_id=leaf2]: 52
[system_id=leaf3]: 61
```

Input B:

```
[system_id=leaf2]: 52
[system_id=leaf4]: 64
```

Outputs will be the following.

A - B:

```
[system_id=leaf1]: 45
[system_id=leaf3]: 61
```

B - A:

```
[system_id=leaf4]: 64
```

A & B:

```
[system_id=leaf2]: 52
```

Processor: Set Count

IN THIS SECTION

Example: Set Count | 1040

The Set Count processor groups as described in **Group by**, then calculates the number of items in each group.

Parameter	Description
Input Types	Table (number or text or discrete state)
Output Types	Table (number)

(Continued)

Parameter	Description
Group by (group_by)	<p>Accepts a list of property names to group input items into output items, produces only one output group for the empty list. Most processors take input and produce output. Many of them produce one output per input (for example, if input is a DSS, output is a DSS of same size). However, some processors reduce the size of the output relative to the size of the input. Effectively, they partition the input into groups, run some calculation on each of the groups that produce a single value per each group, and use that as output. Clearly, the size of the output set depends on the grouping scheme. We call such processors <b>grouping processors</b> and they all take the <b>Group by</b> configuration parameter.</p> <p>In the case of an empty list, the input is considered to be a single group; thus, the output is of size 1 and either N, DS, or TS. If a list of property names is specified, for example ["system_id", "iface_role"], or a single property is specified, for example ["system_id"], we divide the input into groups such that for each group, every item in the group has the same values for the given list of property names. See the <a href="#">"standard deviation processor" on page 1040</a> example for how this works.</p> <p>The output type of a processor depends on a value of the group_by parameter; for an empty list, a processor produces a single value result, such as N, DS, or T, and for grouping by one or more properties it returns a set result, such as NS, DSS, or TS.</p>
Enable Streaming (enable_streaming)	<p>Makes samples of output stages streamed if enabled. An optional boolean that defaults to False. If set to True, all output stages of this processor are streamed in the generic protobuf schema.</p>

Example: Set Count

See ["standard deviation" on page 1040](#) example. It's the same except we calculate the number of stage items.

Processor: Standard Deviation

IN THIS SECTION

Example: Standard Deviation | 1041

The Standard Deviation processor groups as described by **Group by**, calculates the standard deviation, then outputs one standard deviation per group.

Parameter	Description
Input Types	Table (number), Table (number, accumulate=True)
Output Types	Table (number)
Group by (group_by)	<p>Accepts a list of property names to group input items into output items, produces only one output group for the empty list. Most processors take input and produce output. Many of them produce one output per input (for example, if input is a DSS, output is a DSS of same size). However, some processors reduce the size of the output relative to the size of the input. Effectively, they partition the input into groups, run some calculation on each of the groups that produce a single value per each group, and use that as output. Clearly, the size of the output set depends on the grouping scheme. We call such processors <b>grouping processors</b> and they all take the <b>Group by</b> configuration parameter.</p> <p>In the case of an empty list, the input is considered to be a single group; thus, the output is of size 1 and either N, DS, or TS. If a list of property names is specified, for example ["system_id", "iface_role"], or a single property is specified, for example ["system_id"], we divide the input into groups such that for each group, every item in the group has the same values for the given list of property names. See the <a href="#">"standard deviation processor" on page 1040</a> example for how this works.</p> <p>The output type of a processor depends on a value of the group_by parameter; for an empty list, a processor produces a single value result, such as N, DS, or T, and for grouping by one or more properties it returns a set result, such as NS, DSS, or TS.</p>
DDoF (ddof)	Delta Degrees of Freedom, standard deviation correction value, is used to correct divisor ( $N - \text{DDoF}$ ) in calculations, e.g. $\text{DDoF}=0$ - uncorrected sample standard deviation, $\text{DDoF}=1$ - corrected sample standard deviation.
Enable Streaming (enable_streaming)	Makes samples of output stages streamed if enabled. An optional boolean that defaults to False. If set to True, all output stages of this processor are streamed in the generic protobuf schema.

### Example: Standard Deviation

```
group_by: ["role", "system_id"]
ddof: 1
```

Also assume an NS input of

```
[role:fabric, system_id:spine1, if_name=eth0] :10
[role:fabric, system_id:spine1, if_name=eth1] :11
[role:server, system_id:spine1, if_name=eth3] :12
[role:server, system_id:spine1, if_name=eth4] :13
[role:fabric, system_id:spine2, if_name=eth0] :14
[role:fabric, system_id:spine2, if_name=eth1] :15
[role:server, system_id:spine2, if_name=eth3] :16
[role:server, system_id:spine2, if_name=eth4] :17
```

Given the above, the output would be a number-set of

```
[role:fabric, system_id:spine1] : stddev([10, 11])
[role:fabric, system_id:spine2] : stddev([14, 15])
[role:server, system_id:spine1] : stddev([12, 13])
[role:server, system_id:spine2] : stddev([16, 17])
```

Processor: State

IN THIS SECTION

Example: State | 1044

The State processor checks that a value is one of the specified anomalous states. It outputs DSS with anomaly values, such as 'true' if the value is in the specified states, and otherwise, it returns 'false'. (previously called 'state\_check' and 'in\_state'). The State processor supports multiple reference states and output is 'true' when input is in any of the specified states.

Parameter	Description
Input Types	Table( discrete state, accumulate=True or False)
Output Types	Table (discrete state)

*(Continued)*

Parameter	Description
Graph Query (graph_query)	<p>One or more queries on graph specified as strings, or a list of such queries. (String will be deprecated in a future release.) Multiple queries should provide all the named nodes referenced by the expression fields (including additional_properties). Graph query is executed on the "operation" graph. Results of the queries can be accessed using the "query_result" variable with the appropriate index. For example, if querying property set nodes under name "ps", the result will be available as "query_result[0]["ps"]".</p> <p>In collector processors (*_collector, if_counter) it is used to choose a set of nodes for further processing (for example, all leafs, or all interfaces between leaf and spines)</p> <p>In other processors it is used for general parameterization and it is only supported as a list of queries.</p> <pre>graph_query: "node("system", role="leaf", name="system").               out("hosted_interfaces").               node("interface", name="iface").out("link").               node("link", role="spine_leaf")"</pre> <pre>graph_query: ["node("system", role="leaf", name="system")",               "node("system", role="spine", name="system)"]</pre> <p>Non-collector processors containing the graph_query configuration parameter, can be parameterized to use data from arbitrary nodes in the graph, such as property set nodes. Property sets allow you to parameterize macro level SLAs for individual business units. In the example below, graph_query matches a node of type property_set with label probe_propset. It's accessed using the special query_result variable, where Index 0 means it's the first node in query results. If a query returned N nodes, they could be accessed using indices starting from 0 to N-1. ps is what the actual node is referred to in the query; the rest depends on the structure of the node. The int() casting is required because values of property_set nodes are strings. Here it's assumed that a property set node has the label probe_propset and that the value accumulate_duration was already created.</p> <pre>graph_query: [node("property_set", label="probe_propset", name="ps")] duration: int(query_result[0]["ps"].values["accumulate_duration"])</pre>

*(Continued)*

Parameter	Description
	Another example is a that probes can validate a compliance requirement; the compliance value may change over time and/or it can be used by more than one probe. Also, a probe can validate NOS versions on devices. In this case, property sets can be used to define the current NOS version requirement. If it changes tomorrow: change the property set value, instead of going under the probe stage.
Anomalous States	Expression that evaluates to DS value or list of DS values which is used for the check. For example, it can be: <code>"true"</code> (expression evaluating to a string) or <code>"['missing', 'unknown', 'down']"</code> (expression evaluating to a list of strings).
Anomaly MetricLog Retention Duration	Retain anomaly metric data in MetricDb for specified duration in seconds
Anomaly MetricLog Retention Size	Maximum allowed size, in bytes of anomaly metric data to store in MetricDB
Anomaly Metric Logging	Enable metric logging for anomalies
Enable Streaming (enable_streaming)	Makes samples of output stages streamed if enabled. An optional boolean that defaults to False. If set to True, all output stages of this processor are streamed in the generic protobuf schema.
Raise Anomaly (raise_anomaly)	Outputs <code>"true"</code> and <code>"false"</code> values, <code>"true"</code> meaning an appropriate item is anomalous, and <code>"false"</code> meaning the item is not anomalous. When Raise Anomaly is set to True, an actual anomaly is generated in addition to a sample in the output.

**Example: State**

```
state: '"up"'
```

**Sample Input (DS)**

```
[if_name=eth0] : "up"
[if_name=eth1] : "down"
[if_name=eth3] : "up"
```

### Sample Output (DSS)

```
[if_name=eth0] : "false"
[if_name=eth1] : "true"
[if_name=eth3] : "false"
```

If expression is used for the state field, then it's evaluated for each input item, and it results into item-specific state value. Properties of the respective output item are extended by the state property with value of the evaluated expression.

```
state: expected_if_state
```

### Sample Input (DS):

```
[if_name=eth0,expected_if_state=up] : "up"
[if_name=eth1,expected_if_state=down] : "down"
[if_name=eth3,expected_if_state=up] : "down"
```

### Sample Output (DSS)

```
[if_name=eth0,state=up] : "false"
[if_name=eth1,state=down] : "false"
[if_name=eth3,state=up] : "true"
```

## Processor: Subtract

One number is created on output for each number with the same properties in both inputs. For each input item the processor leaves only significant keys, drops the others and puts the result. If there is no common set of properties between both inputs, the output is the empty set.

Parameter	Description
Input Types	Table (number)
Output Types	Table (number)

(Continued)

Parameter	Description
Significant Keys (significant_keys)	List of keys to map items from the inputs for applying the specified operation. It is typically used by processors that take multiple inputs and perform operations on them. When inputs have the same sets of keys it does not need to be specified. When inputs have different sets of keys, it must be specified and it must allow only 1:1 items mapping from the given inputs, otherwise the probe will go into error state.
Enable Streaming (enable_streaming)	Makes samples of output stages streamed if enabled. An optional boolean that defaults to False. If set to True, all output stages of this processor are streamed in the generic protobuf schema.

Processor: Sum

IN THIS SECTION

[Example: Sum Output | 1047](#)

The Sum processor groups as described by **Group by** property, then calculates sum and outputs one for each group.

Parameter	Description
Input Types	Table (number), Table (number, accumulate=True)
Output Types	Table (number)

*(Continued)*

Parameter	Description
Group by (group_by)	<p>Accepts a list of property names to group input items into output items, produces only one output group for the empty list. Most processors take input and produce output. Many of them produce one output per input (for example, if input is a DSS, output is a DSS of same size). However, some processors reduce the size of the output relative to the size of the input. Effectively, they partition the input into groups, run some calculation on each of the groups that produce a single value per each group, and use that as output. Clearly, the size of the output set depends on the grouping scheme. We call such processors <b>grouping processors</b> and they all take the <b>Group by</b> configuration parameter.</p> <p>In the case of an empty list, the input is considered to be a single group; thus, the output is of size 1 and either N, DS, or TS. If a list of property names is specified, for example ["system_id", "iface_role"], or a single property is specified, for example ["system_id"], we divide the input into groups such that for each group, every item in the group has the same values for the given list of property names. See the <a href="#">"standard deviation processor" on page 1040</a> example for how this works.</p> <p>The output type of a processor depends on a value of the group_by parameter; for an empty list, a processor produces a single value result, such as N, DS, or T, and for grouping by one or more properties it returns a set result, such as NS, DSS, or TS.</p>
Enable Streaming (enable_streaming)	Makes samples of output stages streamed if enabled. An optional boolean that defaults to False. If set to True, all output stages of this processor are streamed in the generic protobuf schema.

**Example: Sum Output**

See ["standard deviation" on page 1040](#) example. It's the same except we calculate sum instead of std deviation.

**Processor: System Utilization**

Interface Counters Utilization Per System processor groups detailed interface counter data by system ID and then calculates aggregate TX and RX bits, their aggregate utilization and identifies the highest TX and RX utilizations among the interfaces.

Parameter	Description
Enable Streaming (enable_streaming)	Makes samples of output stages streamed if enabled. An optional boolean that defaults to False. If set to True, all output stages of this processor are streamed in the generic protobuf schema.

Processor: Time in State

IN THIS SECTION

[Example: Time in State | 1050](#)

The Time in State processor measures time when a value is in the range. For each input DS, monitor it over the last time\_window seconds. If at any moment, for the state in state\_range, the amount of time we have been in that state over the last time\_window seconds falls into a range specified in the corresponding state\_range entry, we set the corresponding output DS to 'true'. Otherwise, the output DS for a given input DS is nominally 'false'. (previously called 'time\_in\_state\_check')

Parameter	Description
Input Types	Discrete-State (DS)
Output Types	Discrete-State (DS)
Time Window (time_window)	How long to monitor state. (seconds or an expression that evaluates to integer)
State Range (state_range)	Map state value to its allowed time range in seconds. dict mapping from a single possible state to a single range of time during the most recent time_window seconds that the value from input state is allowed to be in that state. At least one of the range object's two fields must be specified. The omitted field is regarded as "infinity". The fields are numbers (integers or floats) or expressions evaluated into numbers. State is a string or an expression that evaluates to string.

*(Continued)*

Parameter	Description
Graph Query (graph_query)	<p>One or more queries on graph specified as strings, or a list of such queries. (String will be deprecated in a future release.) Multiple queries should provide all the named nodes referenced by the expression fields (including additional_properties). Graph query is executed on the "operation" graph. Results of the queries can be accessed using the "query_result" variable with the appropriate index. For example, if querying property set nodes under name "ps", the result will be available as "query_result[0]["ps"]".</p> <p>In collector processors (*_collector, if_counter) it is used to choose a set of nodes for further processing (for example, all leafs, or all interfaces between leaf and spines)</p> <p>In other processors it is used for general parameterization and it is only supported as a list of queries.</p> <pre>graph_query: "node("system", role="leaf", name="system").               out("hosted_interfaces").               node("interface", name="iface").out("link").               node("link", role="spine_leaf")"</pre> <pre>graph_query: ["node("system", role="leaf", name="system"),               "node("system", role="spine", name="system)"]</pre> <p>Non-collector processors containing the graph_query configuration parameter, can be parameterized to use data from arbitrary nodes in the graph, such as property set nodes. Property sets allow you to parameterize macro level SLAs for individual business units. In the example below, graph_query matches a node of type property_set with label probe_propset. It's accessed using the special query_result variable, where Index 0 means it's the first node in query results. If a query returned N nodes, they could be accessed using indices starting from 0 to N-1. ps is what the actual node is referred to in the query; the rest depends on the structure of the node. The int() casting is required because values of property_set nodes are strings. Here it's assumed that a property set node has the label probe_propset and that the value accumulate_duration was already created.</p> <pre>graph_query: [node("property_set", label="probe_propset", name="ps")] duration: int(query_result[0]["ps"].values["accumulate_duration"])</pre>

*(Continued)*

Parameter	Description
	Another example is a that probes can validate a compliance requirement; the compliance value may change over time and/or it can be used by more than one probe. Also, a probe can validate NOS versions on devices. In this case, property sets can be used to define the current NOS version requirement. If it changes tomorrow: change the property set value, instead of going under the probe stage.
Anomaly MetricLog Retention Duration	Retain anomaly metric data in MetricDb for specified time period
Anomaly MetricLog Retention Size	Maximum allowed size, in bytes of anomaly metric data to store in MetricDB
Anomaly Metric Logging	Enable metric logging for anomalies
Enable Streaming (enable_streaming)	Makes samples of output stages streamed if enabled. An optional boolean that defaults to False. If set to True, all output stages of this processor are streamed in the generic protobuf schema.
Raise Anomaly (raise_anomaly)	Outputs "true" and "false" values, "true" meaning an appropriate item is anomalous, and "false" meaning the item is not anomalous. When Raise Anomaly is set to True, an actual anomaly is generated in addition to a sample in the output.

### Example: Time in State

Config is set to:

```
time_window : 2 seconds
state_range: { "down" : [{"max": 1},] }
```

The above configuration means that for the input DS, we will set output to True and optionally raise an anomaly if the input is in the "down" state for more-than one second out of the last two seconds.

In the sample below, certain values are capitalized to indicate what has changed from the previous time.

Sample Input at time t=0

```
[if_name=eth0] : "up"  
[if_name=eth1] : "up"  
[if_name=eth3] : "up"
```

Sample Output at time t=0

```
[if_name=eth0] : "false"  
[if_name=eth1] : "false"  
[if_name=eth3] : "false"
```

Sample Input at time t=1:

```
[if_name=eth0] : "up"  
[if_name=eth1] : "down"  
[if_name=eth3] : "up"
```

Sample Output at time t=1

```
[if_name=eth0] : "false"  
[if_name=eth1] : "false"  
[if_name=eth3] : "false"
```

Sample Input at time t=2:

```
[if_name=eth0] : "up"  
[if_name=eth1] : "down"  
[if_name=eth3] : "up"
```

Sample Output at time t=2

```
[if_name=eth0] : "false"  
[if_name=eth1] : "true"  
[if_name=eth3] : "false"
```

Sample Input at time t=3:

```
[if_name=eth0] : "up"
[if_name=eth1] : "up"
[if_name=eth3] : "up"
```

Sample Output at time t=3

```
[if_name=eth0] : "false"
[if_name=eth1] : "True"
[if_name=eth3] : "false"
```

Sample Input at time t=4:

```
[if_name=eth0] : "up"
[if_name=eth1] : "up"
[if_name=eth3] : "up"
```

Sample Output at time t=4

```
[if_name=eth0] : "false"
[if_name=eth1] : "false"
[if_name=eth3] : "false"
```

If expressions are used for min or max fields for states specified in the state property, then they are evaluated for each input item which results into item-specific thresholds. Properties of the respective output items are extended by range\_min or range\_max keys with calculated values.

If state key is an expression, output items are extended with state key. The same applies for time\_window property.

Configuration:

```
time_window : int(100/context.severity)
state_range: { context.ref_state : [{"max": "int(20*(context.severity/5.0))"}] }
```

Sample Input at times t=0..6:

```
[if_name=eth0,severity=1,ref_state=down] : "down"
[if_name=eth1,severity=2,ref_state=down] : "down"
```

Sample Output at time t=6:

```
[if_name=eth0,range_max=4,time_window=100,state=down] : "true"
[if_name=eth1,range_max=8,time_window=50,state=down] : "false"
```

### Processor: Traffic Monitor

The Traffic Monitor processor selects interfaces according to the configuration and outputs all available interface-related counters (e.g tx\_bits, rx\_bits etc) and interface utilization.

Parameter	Description
Input Types	No inputs. This is a source processor
Graph Query (graph_query)	<p>One or more queries on graph specified as strings, or a list of such queries. (String will be deprecated in a future release.) Multiple queries should provide all the named nodes referenced by the expression fields (including additional_properties). Graph query is executed on the "operation" graph. Results of the queries can be accessed using the "query_result" variable with the appropriate index. For example, if querying property set nodes under name "ps", the result will be available as "query_result[0]["ps"]".</p> <p>In collector processors (*_collector, if_counter) it is used to choose a set of nodes for further processing (for example, all leafs, or all interfaces between leaf and spines)</p> <p>In other processors it is used for general parameterization and it is only supported as a list of queries.</p> <pre>graph_query: "node("system", role="leaf", name="system").               out("hosted_interfaces").               node("interface", name="iface").out("link").               node("link", role="spine_leaf")"</pre> <pre>graph_query: ["node("system", role="leaf", name="system")",               "node("system", role="spine", name="system)"]</pre>

*(Continued)*

Parameter	Description
Query Expansion	For every path, originally returned by graph queries, passed to each generator the latter one produces a set of items and for each item it produces a new path extended by a corresponding property name which value is set of a value of the produced item.
Query Group by (query_group_by)	<p>List (of strings) of node and relationship names used in the graph query to group query results by. Each element in this list represents a named node or relationship matcher in the graph_query field. It is not an expression to be consistent with existing group_by field in grouping processors. Non-expression is simple and more intuitive.</p> <p>When grouping is active (query_group_by is not null), query results are divided by the specified list of names, where one output item is created per each group. In this case, the expressions can only access matcher names specified in query_group_by and the query results for each group are accessed using a new group_items variable. The group_items variable is a list of query results, where each result has named nodes/relationships, not present in query_group_by.</p> <p>The following list describes the behavior for various values of this field:</p> <ul style="list-style-type: none"> <li>• Value of query_group_by field - Semantics</li> <li>• Omitted or provided as json null (ala None in Python) - No grouping is done. This is equivalent to current behavior of extensible_data_collector. Using 'group_items' in this case is not permitted and results in probe error state.</li> <li>• Empty list ([]) - Produces one group containing all the query results.</li> <li>• One or more matcher names - The query results are grouped by the specified nodes or relationships. If this list covers all available matchers in the query, the number of groups is equal to the number of query results.</li> </ul>
Query Tag Filter (query_tag_filter)	Filters named nodes in the graph queries by assigned tags.
Interface	Expression mapping from graph query to interface name, e.g. "iface.if_name" if "iface" is a name in the graph query.
Port Speed	Expression mapping from graph query to link speed in bits per second, e.g. "functions.speed_to_bits(link.speed)" if "link" is a name in the graph query.
System ID	Expression mapping from graph query to a system_id, e.g. "system.system_id" if "system" is a name in the graph query.

*(Continued)*

Parameter	Description
Period	Duration of the averaging period
Additional Keys	Each additional key/value pair is used to extend properties of output stages where value is considered as an expression executed in context of the graph query and its result is used as a property value with respective key. The value of this property is evaluated for each item to associate items with metrics provided by a corresponding collector service. The association is done by keys because each collector reports a set of metrics where each metric is identified by a key in a format that is specific for each collector.
Enable Streaming (enable_streaming)	Makes samples of output stages streamed if enabled. An optional boolean that defaults to False. If set to True, all output stages of this processor are streamed in the generic protobuf schema.

## Processor: Union

### IN THIS SECTION

- [Example: Union | 1056](#)

The Union processor merges all input items into one set of items. For each input item the processor leaves only signification keys, drops the others and puts the result.

Parameter	Description
Input Types	Table (number or text or discrete state)
Output Types	Table (number or text or discrete state)
Significant Keys (significant_keys)	List of keys to map items from the inputs for applying the specified operation. It is typically used by processors that take multiple inputs and perform operations on them. When inputs have the same sets of keys it does not need to be specified. When inputs have different sets of keys, it must be specified and it must allow only 1:1 items mapping from the given inputs, otherwise the probe will go into error state.

*(Continued)*

Parameter	Description
Enable Streaming (enable_streaming)	Makes samples of output stages streamed if enabled. An optional boolean that defaults to False. If set to True, all output stages of this processor are streamed in the generic protobuf schema.

### Example: Union

Config is set to:

```
significant_keys: ["system_id"]
```

Consider we have inputs with device temperature information.

Input "in\_1":

```
[system_id=leaf1,interface=eth1]: 45
[system_id=leaf2,interface=eth0]: 52
[system_id=leaf3,interface=eth0]: 61
```

Input "in\_2":

```
[system_id=leaf4,interface=eth2]: 52
[system_id=leaf5,interface=eth3]: 64
```

Input "in\_3":

```
[system_id=leaf6,interface=eth3]: 41
```

Output will be the following.

Output "out":

```
[system_id=leaf1]: 45
[system_id=leaf2]: 52
[system_id=leaf3]: 61
[system_id=leaf4]: 52
```

```
[system_id=leaf5]: 64
[system_id=leaf6]: 41
```

## Processor: VXLAN Floodlist

The VXLAN Floodlist processor generates a configuration containing expectations of vxlan floodlist routes.

Parameter	Description
Execution count	Number of times the data is collected
Service input (service_input)	Data to pass to telemetry collectors, if any. Can be an expression.
Service interval (service_interval)	Telemetry collection interval in seconds. Can be an expression.
Service name (service_name)	Name of the custom collector service.
Enable Streaming (enable_streaming)	Makes samples of output stages streamed if enabled. An optional boolean that defaults to False. If set to True, all output stages of this processor are streamed in the generic protobuf schema.

## Configlet Examples (Design)

### IN THIS SECTION

- [Juniper Junos Configlet Interface-Level Example on 4.0.2: gigheter-options](#) | **1058**
- [Juniper Junos Configlet Example on 4.0.2: MTU \(section Interface-Level: Delete\)](#) | **1059**
- [Juniper Junos Configlet Example on 4.0.2 Example: SNMP \(multiple sections\)](#) | **1059**
- [Juniper Junos Configlet Example on 4.0.1 and 4.0.0: NTP \(section SYSTEM\)](#) | **1060**
- [Cisco NX-OS Configlet Example: Syslog \(section SYSTEM\)](#) | **1061**
- [Arista EOS Configlet Example: NTP \(section SYSTEM\)](#) | **1061**
- [Arista EOS Configlet Example: Interface Speed \(section INTERFACE\)](#) | **1061**
- [Enterprise SONiC Configlet Example: NTP \(section SYSTEM\)](#) | **1062**
- [Enterprise SONiC Configlet Example: SNMP \(section SYSTEM\)](#) | **1062**

- [Enterprise SONiC Configlet Example: Syslog \(section SYSTEM\) | 1062](#)
- [Enterprise SONiC Configlet Example: Static Route \(section FRR\) | 1062](#)
- [Enterprise SONiC Configlet Example: sonic-cli Commands \(section SYSTEM\) | 1063](#)
- [Cumulus Linux Configlet Example: NTP \(section SYSTEM\) | 1063](#)
- [Cumulus Linux Configlet Example: SNMP \(section SYSTEM\) | 1063](#)
- [Cumulus Linux Configlet Example: Static Route \(section FRR\) | 1064](#)
- [Cumulus Linux Configlet Example: Syslog \(section FILE\) | 1065](#)

## Juniper Junos Configlet Interface-Level Example on 4.0.2: gigether-options

When you're creating an interface-level configlet during the design phase, you won't know interface names. It's not until you're working in the blueprint that you'll have that information. Interface-level configlets for Junos are designed for you to enter details without including the `set interface` command. For example, to change Junos interface "gigether-options", you can use a interface-level hierarchical or set configlet.

```
gigether-options no-auto-negotiation
gigether-options fec none
```

```
gigether-options {
    no-auto-negotiation;
    fec none;
}
```

When you import the configlet into your blueprint, you'll specify interfaces such as `xe-0/0/0`. For a Junos Interface-Level set configlet Apstra software will prepend the set commands:

```
set interfaces xe-0/0/0 gigether-options no-auto-negotiation
set interfaces xe-0/0/0 gigether-options fec none
```

For a Junos Interface-Level hierarchical configlet Apstra software will load Junos structured configuration:

```
interfaces {
  xe-0/0/0 {
    gigether-options {
      no-auto-negotiation;
      fec none;
    }
  }
}
```

### Juniper Junos Configlet Example on 4.0.2: MTU (section Interface-Level: Delete)

If you want to use a Junos interface-level configlet to remove an existing configuration, you can use an interface level delete configlet. Like the interface level set configlet, when you are creating the configlet during the design phase, you won't know interface names. It's not until you're working in the blueprint that you'll have that information. Interface-level delete configlets for Junos are designed for you to enter details without including the `delete interface` command. For example, to remove the Junos interface "mtu" configuration.

```
mtu
```

When you import the configlet into your blueprint, you'll specify interfaces such as `xe-0/0/0`. For a Junos Interface-Level delete configlet Apstra software will prepend the delete commands:

```
delete interfaces xe-0/0/0 mtu
```

### Juniper Junos Configlet Example on 4.0.2 Example: SNMP (multiple sections)

You can create a configlet with a generator at the Top-Level to enable SNMP. To avoid SNMP alarms on server-facing interfaces, for example, you can create a second generator at the Interface-Level to set up `no-traps`.

Top-Level template text is validated to begin with 'set' or 'delete'. See below for example text.

```
set snmp community public authorization read-only
set snmp description "this is configlet test" set snmp location "Apstra DC"
```

```

set snmp contact "june at juniper dot net"
set snmp trap-group authentication-traps targets 10.0.10.1
set snmp trap-group authentication-traps targets 192.168.15.27
set snmp trap-group authentication-traps categories authentication

```

Interface-Level template text is not validated because it's not a complete CLI command. See below for example text.

```
no-traps
```

When you import the configlet into your blueprint, you'll specify interfaces such as `ex-0/0/0` and Apstra software will prepend the set command as `.`

```
set interface xe-0/0/0 no-traps
```

### Juniper Junos Configlet Example on 4.0.1 and 4.0.0: NTP (section SYSTEM)

Sample text for configuring NTP servers on Junos devices. (On Apstra version 4.0.2 SYSTEM is called Top-Level/Hierarchical.)

```

system {
  ntp {
    boot-server 10.1.4.1;
    server 10.1.4.2;
  }
}

```

## Cisco NX-OS Configlet Example: Syslog (section SYSTEM)

Sample text for configuring Syslog on NX-OS devices.

```
logging server 192.168.0.30
logging facility local3
logging trap warning
```

```
no logging server 192.168.0.30
no logging facility local3
no logging trap warning
```

## Arista EOS Configlet Example: NTP (section SYSTEM)

Sample text for configuring NTP servers on EOS devices. This configlet uses property sets for the NTP server IP addresses.

```
ntp server {{NTP_SERVER_1}}
ntp server {{NTP_SERVER_2}}
```

```
no ntp server {{NTP_SERVER_1}}
no ntp server {{NTP_SERVER_2}}
```

## Arista EOS Configlet Example: Interface Speed (section INTERFACE)

Sample text for applying 'speed auto' to an interface. (You specify devices and interfaces when you import the configlet into a blueprint.)

```
speed auto
```

```
no speed auto
```

## Enterprise SONiC Configlet Example: NTP (section SYSTEM)

Sample text for using the `config` command to set up an NTP server to use mgmt VRF on SONiC devices.

```
sonic-db-cli CONFIG_DB hset 'NTP |global' vrf mgmt  
config ntp add {{ntp_server}}
```

```
config ntp del {{ntp_server}}
```

## Enterprise SONiC Configlet Example: SNMP (section SYSTEM)

Sample text for using the `config` command to set up an SNMP snmptrap to use mgmt VRF on SONiC devices.

```
config snmptrap modify 2 {{SNMP_SERVER}} -v mgmt -c mypass
```

```
config snmptrap del 2
```

## Enterprise SONiC Configlet Example: Syslog (section SYSTEM)

Sample text for using the `config` command to set the Syslog server for SONiC devices.

```
config syslog add {{syslog_host}}
```

```
config syslog del {{syslog_host}}
```

## Enterprise SONiC Configlet Example: Static Route (section FRR)

Sample text for adding a static route

```
ip route 4.2.2.2/32 {{static_route_next_hop}}  
ip route 4.2.2.3/32 {{static_route_next_hop}}
```

## Enterprise SONiC Configlet Example: sonic-cli Commands (section SYSTEM)

Sample text for using the sonic-cli command to set up the delay-restore option for SONiC mclag. Put sudo -u admin at the beginning, and single quotes around phrases with spaces in each sonic-cli command, and < /dev/console at the end.

```
sudo -u admin sonic-cli -c config -c 'mclag domain 1' -c 'delay-restore 600' < /dev/console
```

```
sudo -u admin sonic-cli -c config -c 'mclag domain 1' -c 'no delay-restore' < /dev/console
```

## Cumulus Linux Configlet Example: NTP (section SYSTEM)

By default, NTP in Cumulus runs in the default VRF. Certain scenarios require NTP to run in the mgmt VRF. Services in the default VRF must be stopped, then services in the mgmt VRF must be started. You can apply NTP configuration on the management VRF with the following CLI commands:

```
systemctl stop ntp.service
systemctl disable ntp.service
systemctl daemon-reload
systemctl start ntp@mgmt.service
systemctl enable ntp@mgmt.service
```

```
systemctl stop ntp@mgmt.service
systemctl disable ntp@mgmt.service
systemctl daemon-reload
systemctl start ntp.service
systemctl enable ntp.service
```

## Cumulus Linux Configlet Example: SNMP (section SYSTEM)

Sample text for setting up an SNMP server on Cumulus Linux that uses NCLU. Note the net commit command:

```
net add snmp-server listening-address 172.20.40.10
net add snmp-server readonly-community MyCommunity access
```

```
10.0.0.1  
net commit
```

```
net del snmp-server listening-address 172.20.40.10  
net del snmp-server readonly-community MyCommunity access  
10.0.0.1  
net commit
```

## Cumulus Linux Configlet Example: Static Route (section FRR)

Sample text for adding a static route

```
vrf SZ_ART_DR  
ip route 10.90.0.0/16 10.136.118.24
```

When you look at the rendered config for the device, the configlet content is shown at the end of the

### I2\_virtual\_ext\_001\_leaf1 Rendered Config Preview

```

368 " neighbor l3rtr timers connect 5",
369 " neighbor l3rtr next-hop-self",
370 " neighbor l3rtr soft-reconfiguration inbound",
371 " address-family l2vpn evpn",
372 "   rd 172.16.0.2:2",
373 "   route-target import 10000:1",
374 "   route-target export 10000:1",
375 "   advertise ipv4 unicast",
376 " exit-address-family",
377 " address-family ipv4 unicast",
378 "   redistribute connected route-map AllPodNetworks",
379 "   no neighbor l3rtr send-community extended",
380 " exit-address-family",
381 "",
382 "!",
383 "vrf SZ_ART_DR",
384 " ip route 10.90.0.0/16 10.136.118.24",
385 ""
386 ]
387 },
388 {
389   "command": "systemctl reset-failed frr.service"
390 },
391 {
392   "command": "/usr/sbin/aos_reset_frr_service"
393 }
394 ], "dhcp": [
395 /

```

routing section.

## Cumulus Linux Configlet Example: Syslog (section FILE)

A text file replaces the contents of a targeted file with the contents in the file referenced by the configlet. For example, to configure a Syslog server with enabled management VRF, add the configuration below (it can require double quotation marks as below):

```

cumulus@switch:~$ cat /etc/rsyslog.d/11-remotesyslog.conf
## Copy all messages to the remote syslog server at 192.168.0.254 port 514
action(type="omfwd" Target="192.168.0.254" Device="mgmt" Port="514"
Protocol="udp")

```

Each double quotation mark must be preceded by three (3) backslashes. Three backslashes are needed because double quotes must be escaped and backslashes need escaping as well. The following template text shows how to apply a File configlet that contains double quotation marks.

```

*. * @192.168.0.253:514 #UDP
*. * @192.168.0.254:514 #UDP
action(type=\\\\"omfwd\\\\" Target=\\\\"192.168.0.254\\\\"
Device=\\\\"mgmt\\\\" Port=\\\\"514\\\\" Protocol=\\\\"udp\\\\"")

```

```
/etc/rsyslog.d/11-remotesyslog.conf
```

## Graph

### IN THIS SECTION

- [Graph Overview | 1066](#)
- [Query Specification | 1067](#)
- [Change Notification | 1069](#)
- [Notification Processing | 1070](#)
- [Putting It All Together | 1071](#)
- [Convenience Functions | 1072](#)
- [Apstra Graph Datastore | 1081](#)

## Graph Overview

Apstra uses the Graph model to represent a single source of truth regarding infrastructure, policies, constraints etc. This Graph model is subject to constant change and we can query it for various reasons. It is represented as a graph. All information about the network is modeled as nodes and relationships between them.

Every object in a graph has a unique ID. Nodes have a type (a string) and a set of additional properties based on a particular type. For example, all switches in our system are represented by nodes of type system and can have a property role which determines which role in the network it is assigned (spine/

leaf/server). Physical and logical switch ports are represented by an interface node, which also has a property called `if_type`.

Relationships between different nodes are represented as graph edges which we call relationships. Relationships are directed, meaning each relationship has a source node and a target node. Relationships also have a type which determines which additional properties particular relationship can have. E.g. system nodes have relationships of type `hosted_interfaces` towards interface nodes.

A set of possible node and relationship types is determined by a graph schema. The schema defines which properties nodes and relationships of particular type can have along with types of those properties (string/integer/boolean/etc) and constraints. We use and maintain an open source schema library, Lollipop, that allows flexible customization of value types.

Going back to the graph representing a single source of truth, one of the most challenging aspects was how to reason about it in the presence of change, coming from both the operator and the managed system. In order to support this we developed what we call Live Query mechanism which has three essential components:

- Query Specification
- Change Notification
- Notification Processing

Having modeled our domain model as a graph, you can find particular patterns (subgraphs) in a graph by running searches on the graph specified by graph queries. The language to express the query is conceptually based on Gremlin, an open source graph traversal language. We also have parsers for queries expressed in another language - Cypher, which is a query language used by popular graph database neo4j.

## Query Specification

You start with a `node()` and then keep chaining method calls, alternating between matching relationships and nodes:

```
node('system', name='system').out().node('interface', name='interface').out().node('link',
name='link')
```

The query above translated in english reads something like: starting from a node of type system, traverse any outgoing relationship that reaches node of type interface, and from that node traverse all outgoing relationship that lead to node of type `link`.

At any point you can add extra constraints:

```
node('system', role='spine', name='system').out().node('interface', if_type='ip',
name='interface')
```

Notice `role='spine'` argument, it will select only system nodes that have `role` property set to `spine`.

Same with `if_type` property for interface nodes.

```
node('system', role=is_in(['spine', 'leaf']), name='system')
.out()
.node('interface', if_type=ne('ip'), name='interface')
```

That query will select all system nodes that have `role` either `spine` or `leaf` and interface nodes that have `if_type` anything but `ip` (`ne` means not equal).

You can also add cross-object conditions which can be arbitrary Python functions:

```
node('system', name='system')
.out().node('interface', name='if1')
.out().node('link')
.in().node('interface', name='if2')
.in().node('system', name='remote_system')
.where(lambda if1, if2: if1.if_type != if2.if_type)
```

You refer to objects by giving them names and using those names as argument names for your constraint function (of course you can override that but it makes a convenient default behavior). So, in example above it will take two interface nodes named `if1` and `if2`, pass them into given where function and filter out those paths, for which function returns `False`. Don't worry about where you place your constraint: it will be applied during search as soon as all objects referenced by constraint are available.

Now, you have a single path, you can use it to do searches. However, sometimes you might want to have a query more complex than a single path. To support that, query DSL allows you to define multiple paths in the same query, separated by comma(s):

```
match(
    node('a').out().node('b', name='b').out().node('c'),
    node(name='b').out().node('d'),
)
```

This `match()` function creates a grouping of paths. All objects that share the same name in different paths will actually be referring to the same object. Also, `match()` allows adding more constraints on objects with `where()`. You can do a distinct search on particular objects and it will ensure that each combination of values is seen only once in results:

```
match(
  node('a', name='a').out().node('b').out().node('c', name='c')
).distinct(['a', 'c'])
```

This matches a chain of `a -> b -> c` nodes. If two nodes `a` and `c` are connected through more than one node of type `b`, the result will still contain only one `(a, c)` pair.

There is another convenient pattern to use when writing queries: you separate your structure from your criteria:

```
match(
  node('a', name='a').out().node('b').out().node('c', name='c'),
  node('a', foo='bar'),
  node('c', bar=123),
)
```

Query engine will optimize that query into:

```
match(
  node('a', name='a', foo='bar')
  .out().node('b')
  .out().node('c', name='c', bar=123)
)
```

No cartesian product, no unnecessary steps.

## Change Notification

Ok, now you have a graph query defined. What does a notification result look like? Each result will be a dictionary mapping a name that you have defined for a query object to object found. E.g. for following query

```
node('a', name='a').out().node('b').out().node('c', name='c')
```

results will look like `{'a': <node type='a'>, 'c': <node type='c'>}`. Notice, only named objects are present (there is no `<node type='b'>` in results, although that node is present in query because it does not have a name).

You register a query to be monitored and a callback to execute if something will change. Later, if someone will modify the graph being monitored, it will detect that new graph updates caused new query results to appear, or old results to disappear or update. The response executes the callback that is associated with the query. The callback receives the whole path from the query as a response, and a specific action (added/updated/removed) to execute.

## Notification Processing

When the result is passed to the processing (callback) function, from there you can specify reasoning logic. This could really be anything, from generating logs, errors, to rendering configurations, or running semantic validations. You could also modify the graph itself, using graph APIs and some other piece of logic may react to changes you made. This way, you can enforce the graph as a single source of truth while it also serves as a logical communication channel between pieces of your application logic. The Graph API consists of three parts:

Graph management - methods to add/update/remove stuff in a graph. `add_node()`, `set_node()`, `del_node()`, `get_node()`, `add_relationship()`, `set_relationship()`, `del_relationship()`, `get_relationship()`, `commit()` Query `get_nodes()`, `get_relationships()` Observable interface `add_observer()`, `remove_observer()`

Graph management APIs are self-explanatory. `add_node()` creates new node, `set_node()` updates properties of existing node, and `del_node()` deletes a node.

`commit()` is used to signal that all updates to the graph are complete and they can be propagated to all listeners.

Relationships have similar API.

The observable interface allows you to add/remove observers - objects that implement notification a callback interface. Notification callback consists of three methods:

- `on_node()` - called when any node/relationship is added, removed or updated
- `on_relationship()` - called when any node/relationship is added, removed or updated
- `on_graph()` - called when the graph is committed

The Query API is the heart of our graph API and is what powers all searching. Both `get_nodes()` and `get_relationships()` allow you to search for corresponding objects in a graph. Arguments to those functions are constraints on searched objects.

E.g. `get_nodes()` returns you all nodes in a graph, `get_nodes(type='system')` returns you all system nodes, `get_nodes(type='system', role='spine')` allows you to constrain returned nodes to those having particular property values. Values for each argument could be either a plain value or a special property matcher

object. If the value is a plain value, the corresponding result object should have its property equal to the given plain value. Property matchers allow you to express a more complex criterias, e.g. not equal, less than, one of given values and so on:

**NOTE:** The example below is for directly using Graph python. For demonstration purposes, you can replace `graph.get_nodes` with `node` in the Graph explorer. This specific example will not work on the Apstra web interface.

```
graph.get_nodes(
    type='system',
    role=is_in(['spine', 'leaf']),
    system_id=not_none(),
)
```

In your graph schema you can define custom indexes for particular node/relationship types and the methods `get_nodes()` and `get_relationships()` pick the best index for each particular combination of constraints passed to minimize search time.

Results of `get_nodes()/get_relationships()` are special iterator objects. You can iterate over them and they will yield all found graph objects. You can also use APIs that those iterators provide to navigate those result sets. E.g. `get_nodes()` returns you a `Nodelerator` object which has methods `out()` and `in_()`. You can use those to get an iterator over all outgoing or incoming relationship from each node in the original result set. Then, you can use those to get nodes on the other end of those relationships and continue from them. You can also pass property constraints to those methods the same way you can do for `get_nodes()` and `get_relationships()`.

```
graph.get_nodes('system', role='spine') \
    .out('interface').node('interface', if_type='loopback')
```

The code in the example above finds all nodes with type system and role spine and then finds all their loopback interfaces.

## Putting It All Together

Thequery below is an example of an internal rule that Apstra can use to derive telemetry expectations -- for example, link and interface status. The `@rule` will insert a callback to `process_spine_leaf_link`, in which case we write to telemetry expectations.

```
@rule(match(
    node('system', name='spine_device', role='spine')
```

```

        .out('hosted_interfaces')
        .node('interface', name='spine_if')
        .out('link')
        .node('link', name='link')
        .in_('link')
        .node('interface', name='leaf_if')
        .in_('hosted_interfaces')
        .node('system', name='leaf_device', role='leaf')
    ))
def process_spine_leaf_link(self, path, action):
    """
    Process link between spine and leaf

    """
    spine = path['spine_device']
    leaf = path['leaf_device']
    if action in ['added', 'updated']:
        # do something with added/updated link
        pass
    else:
        # do something about removed link
        pass

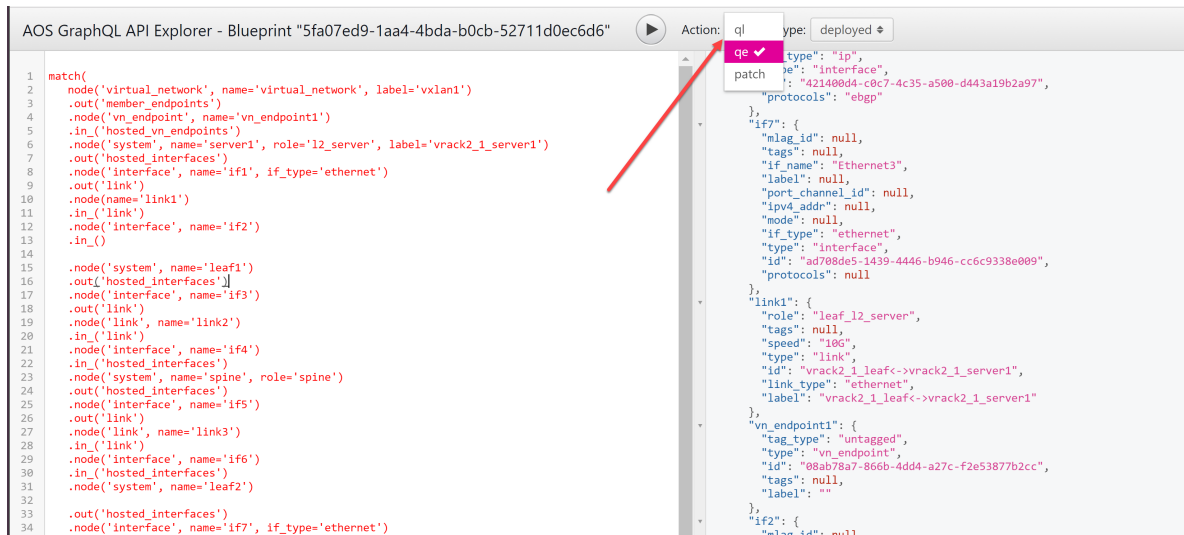
```

## Convenience Functions

To avoid creating complex `where()` clauses when building a graph query, use convenience functions, available from the Apstra web interface.

1. From the blueprint navigate to the **Staged** view or **Active** view, then click the **GraphQL API Explorer** button (top-right >\_). The graph explorer opens in a new tab.
2. Type a graph query on the left. See function descriptions below.
3. From the **Action** drop-down list, select **qe**.

4. Click the **Execute Query** button (looks like a play button) to see results.



## Functions

The Query Engine describes a number of helpful functions:

### `match(*path_queries)`

This function returns a `QueryBuilder` object containing each result of a matched query. This is generally a useful shortcut for grouping multiple match queries together.

These two queries are not a 'path' together (no intended relationship). Notice the comma to separate out arguments. This query will return all of the leafs and spines together.

```

match(
  node('system', name='leaf', role='leaf'),
  node('system', name='spine', role='spine'),
)
  
```

### `node(self, type=None, name=None, id=None, **properties)`

- Parameters

- type** (str or None) - Type of node to search for
- name** (str or None) - Sets the name of the property matcher in the results
- id** (str or None) - Matches a specific node by node ID in the graph
- properties** (dict or None) - Any additional keyword arguments or additional property matcher convenience functions to be used

- **Returns** - Query builder object for chaining queries
- **Return type** - QueryBuilder

While both a function, this is an alias for the PathQueryBuilder nodes -- see below.

### iterate()

- Returns - generator
- Return type: generator

Iterate gives you a generator function that you can use to iterate on individual path queries as if it were a list. For example:

```
def find_router_facing_systems_and_intfes(graph):
    return q.iterate(graph, q.match(
        q.node('link', role='to_external_router')
        .in_('link')
        .node('interface', name='interface')
        .in_('hosted_interfaces')
        .node('system', name='system')
    ))
```

## PathQueryBuilder Nodes

### node(self, type=None, name=None, id=None, \*\*properties)

This function describes specific graph node, but is also a shortcut for beginning a path query from a specific node. The result of a `node()` call returns a path query object. When querying a path, you usually want to specify a node `type`: for example `node('system')` would return a system node.

- **Parameters**
  - **type** (str or None) - Type of node to search for
  - **name** (str or None) - Sets the name of the property matcher in the results
  - **id** (str or None) - Matches a specific node by node ID in the graph
  - **properties** (dict or None) - Any additional keyword arguments or additional property matcher convenience functions to be used
- **Returns** - Query builder object for chaining queries
- **Return type** - QueryBuilder

If you want to use the node in your query results, you need to name it `--node('system', name='device')`. Furthermore, if you want to match specific kwarg properties, you can directly specify the match requirements -

```
node('system', name='device', role='leaf')
```

```
node('system', name='device', role='leaf')
```

### **out(type=None, id=None, name=None, \*\*properties)**

Traverses a relationship in the 'out' direction according to a graph schema. Acceptable parameters are the type of relationship (for example, interfaces), the specific name of a relationship, the id of a relationship, or other property matches that must match exactly given as keyword arguments.

- **Parameters**

- **type** (str or None) - Type of node relationship to search for
- **id** (str or None) - Matches a specific relationship by relationship ID in the graph
- **name** (str or None) - Matches a specific relationship by named relationship

For example:

```
node('system', name='system') \
    .out('hosted_interfaces')
```

### **in\_(type=None, id=None, name=None, \*\*properties)**

Traverses a relationship in the 'in' direction. Sets current node to relationship source node. Acceptable parameters are the type of relationship (for example, interfaces), the specific name of a relationship, the id of a relationship, or other property matches that must match exactly given as keyword arguments.

- **Parameters**

- **type** (str or None) - Type of node relationship to search for
- **id** (str or None) - Matches a specific relationship by relationship ID in the graph
- **name** (str or None) - Matches a specific relationship by named relationship
- **properties** (dict or None) - Matches relationships by any further kwargs or functions

```
node('interface', name='interface') \
    .in_('hosted_interfaces')
```

**where(predicate, names=None)**

Allows you to specify a callback function against the graph results as a filter or constraint. The predicate is a callback (usually lambda function) run against the entire query result. `where()` can be used directly on an a path query result.

- Parameters
  - predicate (callback) - Callback function to run against all nodes in graph
  - names (str or None) - If names are given they are passed to callback function for match

```
node('system', name='system') \
    .where(lambda system: system.role in ('leaf', 'spine'))
```

**ensure\_different(\*names)**

Allows a user to ensure two different named nodes in the graph are not the same. This is helpful for relationships that may be bidirectional and could match on their own source nodes. Consider the query:

- Parameters
  - names (tuple or list) - A list of names to ensure return different nodes or relationships from the graph

```
match(node('system', name='system', role='leaf') \
    .out('hosted_interfaces') \
    .node('interface', name='interface', ipv4_addr=not_none()) \
    .out('link') \
    .node('link', name='link') \
    .in_('link') \
    .node('interface', name='remote_interface', ipv4_addr=not_none())) \
    .ensure_different('interface', 'remote_interface')
```

The last line could be functionally equivalent to the `where()` function with a lambda callback function

```
match(node('system', name='system', role='leaf') \
    .out('hosted_interfaces') \
    .node('interface', name='interface', ipv4_addr=not_none()) \
    .out('link') \
    .node('link', name='link') \
    .in_('link') \
```

```
.node('interface', name='remote_interface', ipv4_addr=not_none())) \
.where(lambda interface, remote_interface: interface != remote_interface)
```

## Property matchers

Property matches can be run on graph query objects directly - usually used within a `node()` function. Property matches allow for a few functions.

### `eq(value)`

Ensures the property value of the node matches exactly the results of the `eq(value)` function.

- Parameters
  - `value` - Property to match for equality

```
node('system', name='system', role=eq('leaf'))
```

Which is similar to simply setting a value as a kwarg on a node object:

```
node('system', name='system', role='leaf')
```

```
node('system', name='system').where(lambda system: system.role == 'leaf')
```

Returns:

```
{
  "count": 4,
  "items": [
    {
      "system": {
        "tags": null,
        "hostname": "l2-virtual-mlag-2-leaf1",
        "label": "l2_virtual_mlag_2_leaf1",
        "system_id": "000C29EE8EBE",
        "system_type": "switch",
        "deploy_mode": "deploy",
        "position": null,
        "role": "leaf",
        "type": "system",
```

```

    "id": "391598de-c2c7-4cd7-acdd-7611cb097b5e"
  },
  {
    "system": {
      "tags": null,
      "hostname": "l2-virtual-mlag-2-leaf2",
      "label": "l2_virtual_mlag_2_leaf2",
      "system_id": "000C29D62A69",
      "system_type": "switch",
      "deploy_mode": "deploy",
      "position": null,
      "role": "leaf",
      "type": "system",
      "id": "7f286634-fbd1-43b3-9aed-159f1e0e6abb"
    }
  },
  {
    "system": {
      "tags": null,
      "hostname": "l2-virtual-mlag-1-leaf2",
      "label": "l2_virtual_mlag_1_leaf2",
      "system_id": "000C29CFDEAF",
      "system_type": "switch",
      "deploy_mode": "deploy",
      "position": null,
      "role": "leaf",
      "type": "system",
      "id": "b9ad6921-6ce3-4d05-a5c7-c31d96785045"
    }
  },
  {
    "system": {
      "tags": null,
      "hostname": "l2-virtual-mlag-1-leaf1",
      "label": "l2_virtual_mlag_1_leaf1",
      "system_id": "000C297823FD",
      "system_type": "switch",
      "deploy_mode": "deploy",
      "position": null,
      "role": "leaf",
      "type": "system",
      "id": "71bbd11c-ed0f-4a38-842f-341781c01c24"
    }
  }
}

```

```

    }
  }
]
}

```

### **ne(value)**

Not-equals. Ensures the property value of the node does NOT match results of ne(value) function

- Parameters
  - value - Value to ensure for inequality condition

```
node('system', name='system', role=ne('spine'))
```

Similar to:

```
node('system', name='system').where(lambda system: system != 'spine')
```

### **gt(value)**

Greater-than. Ensures the property of the node is greater than the results of gt(value) function.

- Parameters
  - value - Ensure property function is greater than this value

```
node('vn_instance', name='vlan', vlan_id=gt(200))
```

### **ge(value)**

Greater-than or Equal To. Ensures the property of the node is greater than or equal to results of ge().

- Parameters: value - Ensure property function is greater than or equal to this value

```
node('vn_instance', name='vlan', vlan_id=ge(200))
```

### **lt(value)**

Less-than. Ensures the property of the node is less than the results of lt(value).

- Parameters

- value - Ensure property function is less than this value

```
node('vn_instance', name='vlan', vlan_id=lt(200))
```

Similar to:

```
node('vn_instance', name='vlan').where(lambda vlan: vlan.vlan_id <= 200)
```

### **le(value)**

Less-than or Equal to. Ensures the property is less than, or equal to the results of le(value) function.

- Parameters
  - value - Ensures given value is less than or equal to property function

```
node('vn_instance', name='vlan', vlan_id=le(200))
```

Similar to:

```
node('vn_instance', name='vlan').where(lambda vlan: vlan.vlan_id < 200)
```

### **is\_in(value)**

Is in (list). Check if the property is in a given list or set containing items is\_in(value).

- Parameters
  - value (list) - Ensure given property is in this list

```
node('system', name='system', role=is_in(['leaf', 'spine']))
```

Similar to:

```
node('system', name='system').where(lambda system: system.role in ['leaf', 'spine'])
```

### **not\_in(value)**

Is not in (list). Check if the property is NOT in a given list or set containing items not\_in(value).

- Parameters

- value (list) - List Value to ensure property matcher is not in

```
node('system', name='system', role=not_in(['leaf', 'spine']))
```

Similar to:

```
node('system', name='system').where(lambda system: system.role not in ['leaf', 'spine'])
```

### **is\_none()**

A query that expects is\_none expects this particular attribute to be specifically None.

```
node('interface', name='interface', ipv4_addr=is_none())
```

Similar to:

```
node('interface', name='interface').where(lambda interface: interface.ipv4_addr is None)
```

### **not\_none()**

A matcher that expects this attribute to have a value.

```
node('interface', name='interface', ipv4_addr=not_none())
```

Similar to:

```
node('interface', name='interface').where(lambda interface: interface.ipv4_addr is not None)
```

## **Apstra Graph Datastore**

The Apstra graph datastore is an in-memory graph database. The log file size is checked periodically, and when a blueprint change is committed. If the graph datastore reaches 100MB or more, a new graph datastore checkpoint file is generated. The database itself does not remove any graph datastore persistence logs or checkpoint files. Apstra provides clean-up tools for the main graph datastore.

Valid graph datastore persistence file groups contain four files: log, log-valid, checkpoint, and checkpoint-valid. Valid files are the effective indicators for log and checkpoint files. The name of each persistence file has three parts: basename, id, and extension.

```
# regex for sysdb persistence files.
# e.g.
#      _Main-0000000059ba612e-00017938-checkpoint-valid
#      \--/ \-----/ \-----/
#      basename      id      extension
```

- **basename** - derived from the main graph datastore partition name.
- **id** - a unix timestamp obtained from `gettimeofday`. Seconds and microseconds in the timestamp are separated by a "-". A persistence file group can be identified by id. The timestamp can also help to determine the generated time sequence of persistence file groups.
- **extension** - log, log-valid, checkpoint, or checkpoint-valid.

## Juniper Apstra Technology Previews (Tech Previews)

Tech Previews give you the ability to test functionality and provide feedback during the development process of innovations that are not final production features. The goal of a Tech Preview is for the feature to gain wider exposure and potential full support in a future release. Customers are encouraged to provide feedback and functionality suggestions for a Technology Preview feature before it becomes fully supported.

Tech Previews may not be functionally complete, may have functional alterations in future releases, or may get dropped under changing markets or unexpected conditions, at Juniper's sole discretion. Juniper recommends that you use Tech Preview features in non-production environments only.

Juniper considers feedback to add and improve future iterations of the general availability of the innovations. Your feedback does not assert any intellectual property claim, and Juniper may implement your feedback without violating your or any other party's rights.

These features are "as is" and voluntary use. Juniper Support will attempt to resolve any issues that customers experience when using these features and create bug reports on behalf of support cases. However, Juniper may not provide comprehensive support services to Tech Preview features. Certain features may have reduced or modified security, accessibility, availability, and reliability standards relative to General Availability software. Tech Preview is not supported under existing service agreements, SLAs, or support service.

For additional details, please contact ["Juniper Support " on page 777](#) or your local account team.

---

Juniper Networks, the Juniper Networks logo, Juniper, and Junos are registered trademarks of Juniper Networks, Inc. in the United States and other countries. All other trademarks, service marks, registered marks, or registered service marks are the property of their respective owners. Juniper Networks assumes no responsibility for any inaccuracies in this document. Juniper Networks reserves the right to change, modify, transfer, or otherwise revise this publication without notice. Copyright © 2023 Juniper Networks, Inc. All rights reserved.