

Juniper Apstra 4.1.2 / 4.1.1 / 4.1.0 User Guide

Published
2025-03-06

RELEASE

Table of Contents

General Workflow

Install Apstra Software | 1

Devices | 2

Design | 2

Resources | 2

Blueprints | 2

Next Steps | 3

Apstra GUI

Access Apstra GUI | 3

Reset Apstra GUI Admin Password | 4

Check Apstra GUI Version | 5

Update Apstra GUI Version | 5

Restore Apstra GUI Version | 6

Design

Logical Devices (Datacenter Design) | 7

Logical Device Overview | 7

Create Logical Device | 10

Edit Logical Device | 12

Delete Logical Device | 13

Interface Maps (Datacenter Design) | 13

Interface Map Overview | 13

Create Interface Map | 15

Example: Create Interface Map with Breakout Ports | 16

Example: Inter Port Constraints - Disabled Ports | 19

Edit Interface Map | 22

Delete Interface Map (Design) | 23

Rack Types (Datacenter Design) | 23

Rack Type Overview | 23

Create Rack Type | 30

Example: Create Rack Type | 31

Edit Rack Type in Global Catalog | 34

Edit Rack Type in Template | 34

Edit Rack Type in Blueprint | 35

Delete Rack Type | 35

Templates (Datacenter Design) | 35

Template Overview | 35

Create Rack Based Template | 41

Create Pod Based Template | 41

Create Collapsed Template | 42

Edit Template | 43

Update Rack Type in Rack Based Template | 43

Delete Template | 43

Config Templates (Freeform Design) | 44

Create Config Template | 44

Edit Config Template | 45

Delete Config Template | 45

Configlets (Datacenter Design) | 45

Configlet Overview | 46

Create Configlet | 50

Export / Edit / Delete Configlet (Design) | 51

Export Configlet | 51

Edit Configlet | 52

Delete Configlet | 52

Property Sets (Datacenter Design) | 52

Create Property Set (Datacenter Design) | 54

Edit / Delete Property Set (Datacenter Design) | 55

Edit Property Set | 56

Delete Property Set (Datacenter Design) | 56

TCP/UDP Port Aliases (Datacenter Design) | 56

TCP/UDP Port Alias Overview | 56

Create TCP/UDP Port Alias | 57

Edit TCP/UDP Port Alias | 57

Delete TCP/UDP Port Alias | 57

Tags (Design) | 58

Tags Overview | 58

Create Tag (Design) | 59

Edit Tag (Design) | 59

Delete Tag (Design) | 60

Devices

Device Configuration Lifecycle | 61

Terminology | 61

Configuration Stages: Overview | 62

Configuration Stages: Detail | 64

View Device Config from Blueprint | 68

Configuration Deviations | 71

Device Offline (Unavailable) | 71

Manually Apply Full Config | 71

| [Deploy Modes | 72](#)

Managed Devices | 74

[Managed Devices Overview | 75](#)

[Add Device to Managed Devices | 79](#)

[Remove \(Decommission\) Device from Managed Devices | 80](#)

[Drain Device Traffic | 82](#)

[Edit Device | 84](#)

[Delete Device | 85](#)

[Upgrade Device NOS | 86](#)

| [NOS Upgrade Overview | 86](#)

| [Update User-defined Device Profiles | 87](#)

| [Register / Upload OS Image | 89](#)

| [Upgrade OS Image | 92](#)

[Device AAA | 93](#)

[Set Device Admin State | 95](#)

[Device Profiles | 95](#)

| [Device Profiles | 95](#)

| [Create Device Profile | 103](#)

| [Edit / Delete Device Profile | 104](#)

| [Juniper Device Profiles | 104](#)

| [SONiC Device Profile | 106](#)

System Agents | 151

[Create Onbox Agent | 151](#)

[Create Offbox Agent | 154](#)

[Edit Agent | 160](#)

| [Edit One Agent | 160](#)

| [Edit Multiple Agents | 161](#)

[Uninstall and Delete Agent | 162](#)

[Juniper Device Agent | 164](#)

- Juniper ZTP | 164
- Disable ZTP | 164
- Apply Initial Juniper Junos Configuration | 165
- Configure super-user User | 167
- Configure IP address and Management VRF | 168
- Configure SSH and NETCONF | 168
- Add Junos License Configuration | 169

SONiC Device Agent | 169

- SONiC Device Agent Overview | 169
- Configure Management IP Manually (SONiC) | 170
- Install Agent Manually (SONiC) | 172
- Uninstall Agent Manually (SONiC) | 176

Cisco Device Agent | 177

- Cisco NX-OS Device Agent Overview | 177
- Device Configuration Requirements | 178
- Resize and Enable Guestshell | 179
- Download Agent Installer | 179
- Install Cisco Device Agent | 180
- Update Agent Config File and Start Service | 180
- Activate Apstra Devices on Apstra Server | 181
- Deploy Device | 181
- Reset Apstra Device Agent | 181
- Uninstall Apstra Device Agent | 181
- Remove Apstra EEM Scripts | 182
- Cisco Agent Troubleshooting | 182

Arista Device Agent | 189

- Initial Arista EOS Configuration | 190
- Decommission Device | 193
- Remove Apstra Package from Device | 193
- Restart System | 194
- Manually Install Arista Device Agent | 195
- Device Agent Configuration File | 197
- Arista Agent Troubleshooting | 198

Packages (Devices) | 210

- Packages Overview | 210

- Upload Packages | 210

Agent Profiles | 211

- Agent Profiles (Devices) | 211

- Create Agent Profile | 212

- Edit / Delete Agent Profile | 212

Pristine Configuration | 213

- Edit Pristine Config | 213

- Update Pristine Config from Device | 214

Telemetry | 215

- Services | 215

- Service Registry | 218

- Service Registry Overview | 218

- Import Service Schemas | 220

- Delete Service Registry | 220

- Telemetry Collection Statistics | 220

- Telemetry Streaming | 222

- Route Anomalies for a Host - Example | 223

- Juniper Telemetry Commands | 225

- Cisco Telemetry Commands | 226

- Arista Telemetry Commands | 227

- Linux Servers | 228

- Debugging Telemetry | 229

Apstra ZTP | 230

- Apstra ZTP (Devices) | 230

- Apstra ZTP Overview | 230

- Download and Deploy Apstra ZTP VM | 235

- Configure Static Management IP Address (Apstra ZTP) | 236

- Configure ZTP User | 237

- Configure DHCP Server | 238
- Configure Controller IP Address for ZTP | 241
- Edit Apstra ZTP Configuration File | 241

Apstra ZTP - Juniper | 247

- Juniper and ZTP Disk Space | 247
- Example: Juniper Junos - ztp.json | 248
- Example: Juniper Junos OS Evolved - ztp.json | 248
- Juniper Junos Bootstrap File | 249
- Juniper Junos Custom Config File | 249
- Restart Juniper Junos ZTP | 251
- Troubleshoot Juniper Junos ZTP | 251

Apstra ZTP - SONiC | 251

- Enterprise SONiC and ZTP Overview | 252
- Example: Enterprise SONiC -ztp.json | 252
- Enterprise SONiC Custom Config File | 253
- Restart Enterprise SONiC ZTP | 253

Apstra ZTP - Cisco | 253

- Cisco NX-OS and ZTP Disk Space | 254
- Example: Cisco NX-OS - ztp.json | 254
- Cisco NX-OS Custom Config File | 255
- Cisco NX-OS Offbox Agent Custom Config File | 255
- Restart Cisco NX-OS ZTP | 256

Apstra ZTP - Arista | 257

- Arista EOS | 257
- Example: Arista EOS - ztp.json | 258
- Arista EOS Custom Config File | 258
- Restart Arista EOS ZTP | 259

Upgrade Apstra ZTP | 260

Resources Introduction

ASN Pools (Resources) | 262

- ASN Pool Overview | 262
- Create ASN Pool | 263

Edit ASN Pool | 263

Delete ASN Pool | 264

VNI Pools (Resources) | 264

VNI Pool Overview | 264

Create VNI Pool | 265

Edit VNI Pool | 265

Delete VNI Pool | 266

Integers | 266

Integers in Apstra | 266

Create Integer Pool | 267

Update Integer Pool | 267

Delete Integer Pool | 267

IPv4 Addresses | 267

IP Pool Overview | 268

Create IPv4 Pool | 269

Edit IPv4 Pool | 269

Delete IPv4 Pool | 269

IPv6 Addresses | 270

IPv6 Pool Overview | 270

Create IPv6 Pool | 271

Edit IPv6 Pool | 271

Delete IPv6 Pool | 272

Datacenter Reference Design

Create / Delete Datacenter Blueprint | 273

Create Datacenter Blueprint | 273

Delete Datacenter Blueprint | 274

Datacenter Blueprint Summary and Dashboard | 274

- Blueprints Summary | 274

- Blueprint Dashboard | 276

Assign Physical Resources (Datacenter) | 277

- Update Physical Resource Assignments | 278

- Reset Physical Resource Group Overrides | 279

Assign Device Profiles | 280

Configlets (Datacenter Blueprint) | 281

- Import Configlet | 283

- Edit / Delete Configlet (Blueprint) | 285

 - Edit Configlet Scope | 285

 - Edit Configlet Generators | 286

 - Delete Configlet | 286

- Assign Configlet | 286

Topology (Datacenter) | 287

- 2D Topology View | 288

- 3D Topology View | 289

- Neighbors Selection View | 291

- Links Selection View | 292

- Virtual Network Endpoints | 292

Nodes (Datacenter) | 293

- Assign Device (Datacenter) | 295

 - Device Assignment Overview | 295

 - Assign Device(s) (from Devices Build Panel) | 295

 - Assign One Device (from Devices Build Panel) | 298

 - Assign One System ID (from Selection Panel) | 299

- Unassign Device (Datacenter) | 300

 - Unassign Device (from Device Selection Panel) | 300

 - Unassign Device(s) (from Devices Build Panel) | 303

Set Deploy Mode (Datacenter) | 304

- Set Deploy Mode (from Build Panel) | 304
- Set Deploy Mode (from Selection Panel) | 305
- Set Deploy Mode (from Nodes View) | 305

Generic Systems vs. External Generic Systems | 305**Add Generic System | 306**

- Add Generic System (from Topology View) | 307
- Copy Existing Generic System (from Topology View) | 311

Add External Generic System | 314

- Add External Generic System (from Topology View) | 314
- Add External Generic System (from Nodes View) | 319

Add Access Switch | 319**Update Node Tags | 324**

- Update Node Tags (One Node) | 324
- Update Node Tags (Multiple Nodes) | 325

Update Port Channel ID Range | 327

- Update Port Channel ID Range (from Topology View) | 327
- Update Port Channel ID Range (from Nodes view) | 329

Edit Hostname (Datacenter) | 330

- Edit Hostname (from Build Panel) | 330
- Edit Hostname (from Selection Panel) | 331
- Edit Hostname (from Nodes View) | 331

Edit Generic System Name | 332

- Edit Generic System Name (from Nodes View) | 333

Edit Device Properties (Datacenter) | 333**View Node's Static Routes | 334****Delete Node | 335****Links (Datacenter) | 338****Add Links to Leaf | 340**

Add Links to Spine | 343

Add Links to Generic System | 347

Add Links to External Generic System | 352

Add Leaf Peer Links | 357

Form LAG | 361

Break LAG | 364

Update LAG Mode | 366

Update Link Tags | 369

Update Link Tags (One Link - Topology View) | 369

Update Link Tags (One Link - Links View) | 372

Update Link Tags (Multiple Link - Links View) | 372

Update Link Speed | 374

Update Link Speed (Topology View) | 374

Update Link Speed (Links View) | 376

Update Link Properties | 377

Delete Link (Datacenter) | 379

Delete Link (Neighbors View) | 379

Delete Link (Links View) | 381

Import / Export Cabling Map (Datacenter) | 384

Import Cabling Map | 384

Export Cabling Map | 384

Edit Cabling Map (Datacenter) | 385

Edit Cabling Map (GUI) | 385

Edit Cabling Map (JSON) | 386

Fetch Discovered LLDP Data (Datacenter) | 386

Racks (Datacenter) | 387

Change Rack Name | 388

Add Rack | 389

Export Rack Type | 389

Edit Rack | 390

Delete Rack | 391

Pods (Datacenter) | 391

Add Pod (5-Stage Only) | 392

Change Pod Name | 393

Add Spine per Pod | 394

Add Link per Superspine (5-Stage) | 397

Change Link Speed per Superspine (5-Stage) | 399

Change Spine Logical Device (Pod) | 401

Delete Pod | 404

Planes (Datacenter) | 405

Add Superspine per Plane | 406

Change Superspine Logical Device (Plane) | 408

Virtual Networks | 409

Assign Virtual Resources | 414

Update Virtual Resources Assignments | 414

Reset Virtual Resource Group Overrides | 415

Create Virtual Networks | 415

Create Virtual Networks (using GUI) | 416

Create Virtual Networks (using CSV File) | 417

Assign / Unassign Virtual Networks | 419

Assign / Unassign One Virtual Network | 419

Assign / Unassign Multiple Virtual Networks | 420

Import / Export Virtual Networks | 422

Import Virtual Networks | 422

Export Virtual Networks | 423

Edit Virtual Networks | 424

- Edit One Virtual Network | 424
- Edit Multiple Virtual Networks | 425

Delete Virtual Networks | 426

- Delete One Virtual Network | 426
- Delete Multiple Virtual Networks | 427

Routing Zones | 429

Create Routing Zones | 431

- Create Routing Zones (using GUI) | 431
- Create Routing Zones (using CSV File) | 432

Assign DHCP Server to Routing Zone | 433

Assign Resources to Routing Zone | 434

Import / Export Routing Zones | 435

- Import Routing Zones | 435
- Export Routing Zones | 436

Edit Routing Zones | 437

- Edit One Routing Zone | 437
- Edit Multiple Routing Zones | 437

Delete Routing Zones | 439

- Delete One Routing Zone | 439
- Delete Multiple Routing Zones | 439

Static Routes (Virtual) | 439

Protocol Sessions (Virtual) | 440

Data Center Interconnect (DCI) / Remote EVPN Gateways (Virtual) | 441

- DCI / EVPN Gateway Overview | 442
- DCI Deployment Options | 443
- Implementation | 445
- Apstra Workflow | 449

Virtual Infra (Virtual) | 454

- vCenter Virtual Infra | 455

VMware vSphere Integration Overview | 455

Enable vCenter Integration | 456

VM Visibility | 457

Validate Virtual Infra Integration | 458

Auto-Remediation Overview | 460

Enable Auto-Remediation | 460

Remediate Probe Anomalies | 461

Disable Virtual Infra Integration | 461

NSX-T Integration | 462

VMware NSX-T Integration Overview | 462

Enable NSX-T Integration | 463

Virtual Infrastructure Visibility | 468

Validate Virtual Infra Integration | 472

Disable Virtual Infra Integration | 474

NSX-T Edge and Connectivity Templates | 474

Overview | 474

Set Up NSX-T Tier-0 Router BGP peering | 474

Set Up NSX-T VRF Lite | 480

Set Up Default Static Route towards NSX-T Edge | 483

Set Up BGP IPv6 towards NSX-T Edge | 484

Un-assign BGP on VXLAN VN towards NSX-T Edge | 485

NSX-T Inventory Mapping to Apstra Virtual Infrastructure | 485

Overview | 485

NSX-T Networking Terminology and correlation | 486

NSX Inventory Model | 494

Model Details and Relationship | 495

Endpoints Overview (Virtual) | 521

Internal Endpoints (Virtual) | 522

Create Internal Endpoint | 523

Edit Internal Endpoint | 523

Delete Internal Endpoint | 523

External Endpoints (Virtual) | 524

Create External Endpoint | 524

- Edit External Endpoint | 524
- Delete External Endpoint | 525

Enforcement Points (Virtual) | 525

Endpoint Groups (Virtual) | 525

- Create Endpoint Group | 525
- Edit Endpoint Group | 526
- Delete Endpoint Group | 526

Policies (Datacenter) Staged | 527

Security Policies | 527

- Security Policy Overview | 527
- Security Policy Parameters | 529
- Create Security Policy | 531
- Policy Errors | 532
- Edit Security Policy | 533
- Delete Security Policy | 533
- Security Policy Search | 533
- Security Policy Conflicts | 534
- Security Policy Settings | 535

Interface Policies | 535

Routing Policies | 543

- Routing Policy Overview | 543
- Create Routing Policy | 548
- Edit Routing Policy | 548
- Delete Routing Policy | 548

Routing Zone (VRF) Constraints | 549

- Create Routing Zone Groups (Optional) | 549
- Create Routing Zone Constraint Policy | 549
- Edit / Delete Routing Zone Constraint Policy | 551
- Apply Routing Zone Constraint | 551

Fabric Addressing Policy | 551

- Enable IPv6 Applications | 552
- ESI MAC Most Significant Byte | 552

Virtual Network Policy | 553

- Virtual Network Policy Overview | 553

- Modify Virtual Network Policy | 556

Anti-Affinity Policy | 556

- Anti-Affinity Policy Overview | 556

- Enable/Disable Anti-Affinity Policy | 557

Validation Policy | 558**Logical Devices (Datacenter Blueprint) | 559**

- Logical Devices Overview (Blueprint Catalog) | 559

- Export Logical Device | 560

Interface Maps (Datacenter Blueprint) | 561

- Interface Maps Overview (Blueprint) | 561

- Import Interface Map | 561

- Delete Interface Map (Blueprint) | 562

Property Sets (Datacenter Blueprint) | 562

- Import / Re-import Property Set (Datacenter Blueprint) | 563

- Import Property Set | 563

- Re-import Property Set | 563

- Delete Property Set (Datacenter Blueprint) | 564

AAA Servers (Datacenter Blueprint) | 564

- AAA Servers Overview | 564

- Create AAA Server | 566

- Edit AAA Server | 566

- Delete AAA Server | 566

- Configure AAA RADIUS Server | 566

- Configure Client Supplicant | 567

Tags (Datacenter Blueprint) | 568

- Tags Overview (Blueprint) | 568

Search Tags (Blueprint) | 569

Find by Tags | 569

Create Tag (Blueprint) | 570

Import Tag | 570

Export Tag | 570

Edit Tag (Blueprint) | 570

Delete Tag (Blueprint) | 570

Tasks (Datacenter) Staged | 571

Connectivity Templates | 571

Primitives | 574

Virtual Network (Single) Primitive | 575

Virtual Network (Multiple) Primitive | 576

IP Link Primitive | 576

Static Route Primitive | 577

Custom Static Route Primitive | 578

BGP Peering (IP Endpoint) Primitive | 579

BGP Peering (Generic System) Primitive | 580

Dynamic BGP Peering Primitive | 582

Routing Policy Primitive | 583

Routing Zone Constraint Primitive | 584

User-defined | 585

Pre-defined | 586

Create Connectivity Template for Multiple VNs on Same Interface (Example) | 586

Create Connectivity Template for Layer 2 Connected External Router (Example) | 589

Assign Connectivity Template | 592

Assign Connectivity Template Overview | 592

Method 1 | 593

Method 2 | 594

Force Assign VN Templates | 595

Edit Connectivity Template | 596

Delete Connectivity Template | 596

Active (Datacenter Blueprint) | 596

Active Blueprint Overview | 597

Selection Panel | 597

Status Panel | 598

Topology (Active) | 598

2D Topology View (Active) | 599

3D Topology View (Active) | 600

Neighbors View (Active) | 601

Links View (Active Topology) | 604

Virtual Networks Endpoints (Active) | 605

Headroom (Topology) | 605

Nodes (Active) | 607

Active Nodes Overview | 608

Apply Full Config | 608

Links (Active) | 609

Active Links Overview | 609

Export Cabling Map | 609

Racks (Active) | 610

Change Rack Name | 610

Pods (Active) | 611

Query | 612

Anomalies (Service) | 613

Discovery Anomalies | 613

Configuration Deviation | 617

Root Causes | 620

Root Cause Overview | 621

Enable Root Cause Analysis | 621

View Root Cause Analysis | 622

BGP Route Tagging | 623

Freeform Reference Design

Freeform Overview | 626

Freeform Workflow | 628

Create / Delete Freeform Blueprint | 629

Create Freeform Blueprint | 629

Delete Freeform Blueprint | 629

Freeform Blueprint Summary and Dashboard | 630

Blueprints Summary | 630

Blueprint Dashboard | 630

Topology (Freeform) | 631

Systems (Freeform) | 632

Create Internal System (Freeform) | 633

Create Internal System (from Topology Editor) | 634

Create Internal System (from Systems View) | 636

Clone Internal System (from Topology Editor) | 637

Clone Internal System (from Systems View) | 637

Create External System (Freeform) | 638

Create External System (from Topology Editor) | 638

Create External System (from Systems View) | 640

Clone External System (from Topology Editor) | 640

Clone External System (from Systems View) | 641

Edit System Properties (Freeform) | 642

Delete System (Freeform) | 643

Delete System (from Topology Editor) | 643

Delete System (from Systems View) | 644

Assign Config Template | 644

Remove Config Template Assignment | 645

Set Deploy Mode (Freeform) | 646

Set Deploy Mode on One System | 646

Set Deploy Modes on Multiple Systems | 647

Assign System (Freeform) | 649

Assign System(s) (from Systems View) | 649

Assign System (from Topology Editor) | 650

Assign System (from Device Panel) | 650

Unassign System (Freeform) | 651

Unassign System(s) (from Systems View) | 651

Unassign System (from Topology Editor) | 653

Unassign System (from Device Panel) | 655

Device Context (Freeform) | 655

Links (Freeform) | 657

Add Link (Freeform) | 657

Edit Cabling Map (Freeform) | 659

Fetch Discovered LLDP Data (Freeform) | 660

Manage Link Tags (Freeform) | 661

Delete Link (Freeform) | 661

Resource Management | 663

Resource Management Introduction (Freeform) | 664

Create Allocation Group (Freeform) | 669

Create Allocation Group (from Resource Management Tab) | 669

Create Allocation Group (from Topology View) | 670

Create Group (Freeform) | 671

Create Group Generator (Freeform) | 672

Create Resource (Freeform) | 675

Create Resource Generator (Freeform) | 676

Create Local Pool (Freeform) | 677

Create Local Pool Generator (Freeform) | 678

Config Templates (Freeform Blueprint) | 680

A Simple Config Template | 680

Config Template With Variable | 681

Config Template and Property Sets | 681

Create Config Template (Freeform Blueprint) | 681

Edit / Delete Config Template (Freeform) | 683

 Edit Config Template | 683

 Delete Config Template | 684

Import / Export Config Template (Freeform) | 684

 Import Config Template | 684

 Export Config Template | 684

Import Device Profile (Freeform) | 685

Property Sets (Freeform Blueprints) | 686

Create Property Set (Freeform Blueprint) | 686

 Create Property Set with Builder | 686

 Create Property Set with Editor | 687

Edit / Delete Property Set (Freeform Blueprint) | 687

 Edit Property Sets | 687

 Delete Property Sets | 688

Tags (Freeform Blueprint) | 688

Create Tag (Freeform Blueprint) | 689

Edit / Delete Tag (Freeform Blueprint) | 689

 Edit Tags | 689

 Delete Tag | 690

Tasks - Staged (Freeform) | 690

Active | 691

 Traffic Heat (Freeform) | 691

Commit Blueprint

Uncommitted Overview | 692

Review Staged Changes | 694

[Commit Staged Changes | 696](#)

[Revert Staged Changes | 697](#)

Time Voyager

[Time Voyager Overview | 698](#)

[Jump to Previous Blueprint Revision | 700](#)

[Keep Saved Blueprint Revision | 701](#)

[Update Blueprint Revision Description | 701](#)

[Delete Kept Blueprint Revision | 701](#)

Analytics

[Analytics Overview | 702](#)

[Analytics Dashboard | 703](#)

[Configure Auto-Enabled Dashboards | 704](#)

[Instantiate Predefined Dashboard | 704](#)

[Create Analytics Dashboard | 705](#)

[Edit / Delete Dashboard | 705](#)

[Edit Dashboard | 705](#)

[Delete Dashboard | 706](#)

[Anomalies \(Analytics\) | 706](#)

[Widgets Overview | 706](#)

[Create Anomaly Heat Map Widget | 707](#)

[Create Stage Widget | 707](#)

[Create Stage Widget from Widgets View | 708](#)

[Create Stage Widget from Probes View | 708](#)

[Edit / Delete Widget | 708](#)

[Edit Widget | 708](#)

[Delete Widget | 709](#)

Probes | 709

- IBA Probes Overview | 709

Instantiate Predefined Probe | 715

Create Probe | 716

Import / Export Probe | 716

- Import Probe | 716

- Export Probe | 716

Edit / Delete Probe | 717

- Edit Probe | 717

- Delete Probe | 717

Providers (External Systems)

LDAP Provider | 719

- Create LDAP Provider | 719

- Configure LDAP Provider | 721

Active Directory Provider | 722

- Create Active Directory Provider | 722

TACACS+ Provider | 723

- Create TACACS+ Provider | 724

- Configure TACACS+ Provider | 724

RADIUS Provider | 725

- RADIUS Limitations | 725

- Create RADIUS Provider | 726

Edit / Delete Provider | 727

- Edit Provider | 727

- Delete Provider | 728

Provider Role Map Overview | 728

Create Provider Role Map | 729

Edit / Delete Role Map | 729

- Edit Role Map | 730**

- Delete Role Map | 730**

Platform

User/Role Management (Platform) | 731

- User Profile Management | 731**

- User Role Management | 732**

- User Profile Use Cases | 735**

- Use Case Overview | 735**

- Create User Profile | 739**

- Change Apstra GUI User Password | 739**

- Log Out User | 739**

- Edit / Delete User Profile | 739**

- Edit User Profile | 739**

- Delete User Profile | 740**

- User Role Use Cases | 740**

- Use Cases Overview | 740**

- Create User Role | 745**

- Edit / Delete User Role | 745**

- Edit User Role | 746**

- Delete User Role | 746**

Security (Platform) | 746

- Allowed List | 747**

- Allowed List Overview | 747**

- Add IP/Subnet to Allowed List | 747**

- Edit IP/Subnet to Allowed List | 748**

- Delete IP/Subnet from Allowed List | 748**

- Banned List | 748**

- Banned List Overview | 748**

Delete IP/Subnet from Banned List | 749

ACL Rules | 749

Overview | 749

Enable / Disable ACL Rules | 750

Add ACL Rule | 750

Edit ACL Rule | 750

Delete ACL Rule | 751

Rate Limit Configuration | 751

Rate Limit Configuration Overview | 751

Edit Rate Limit Configuration | 751

Edit Password Complexity Requirements | 752

Syslog Configuration (Platform) | 754

Syslog Overview | 754

Create Syslog Config | 760

Edit Syslog Config | 760

Delete Syslog Config | 760

Receivers (Platform) | 761

Streaming Receivers Overview | 761

Create Receiver | 762

Delete Receiver | 762

Configure Receivers Using Telegraf Plugin | 762

Global Statistics (Platform) | 764

Event Log (Platform) | 765

Event Log Overview | 765

Export Event Log to CSV File | 767

Send Event Log to External System | 767

Apstra VM Clusters | 767

Apstra Cluster Nodes | 768

- Nodes Overview | 768
- Create Apstra Node | 773
- Edit Apstra Node | 774
- Delete Apstra Node | 774

Apstra Cluster Management | 775

Change Cluster Application Memory Usage (API) | 777

Developers (Platform) | 778

- Authenticate User (API) | 779
- Resource Pools (API) | 782
- Configlets (API) | 793
- Property Sets (API) | 796
- Interface Descriptions (API) | 798
- Probes (API) | 802
- RCI Fault Model (API) | 816
- Health Check Apstra VMs (API) | 820
- API From Python | 820
- REST API Explorer | 823

Juniper Technical Support | 824

- Show Tech: Apstra Controller and Device Agents (GUI) | 825
- Show Tech: Offbox Agents (CLI) | 827
- Show Tech: Infra Offbox Agents (CLI) | 829
- Show Tech: Apstra Controller (CLI) | 829
- Show Tech: Onbox Agents (CLI) | 830

Favorites & User

Manage Favorites | 832

Change Your User Password | 833

Change Your User Name/Email | 833

[Log Out | 834](#)

Apstra Server Management

[Monitor Apstra Server via CLI | 835](#)

[Restart Apstra Server | 836](#)

[Reset Apstra Server VM Password | 836](#)

[Reinstall Apstra Server | 841](#)

[Apstra Database Overview | 842](#)

[Back up Apstra Database | 843](#)

[Restore Apstra Database | 844](#)

[Reset Apstra Database | 849](#)

[Migrate Apstra Database | 850](#)

[Replace SSL Certificate on Apstra Server with Signed One | 854](#)

[Replace SSL Certificate on Apstra Server with Self-Signed One | 857](#)

[Change Apstra Server Hostname | 858](#)

Apstra CLI Utility

[Install Apstra-CLI | 859](#)

[Access Apstra-cli | 860](#)

Guides

Extensible Telemetry Guide | 861

[Extensible Telemetry Overview | 862](#)

[Set Up Development Environment | 862](#)

[Develop Collector | 863](#)

[Write Collector | 866](#)

[Unit Test Collector | 873](#)

[Package Collector | 874](#)

Upload Packages | 875

Use Telemetry Collector | 875

5-Stage Clos Architecture | 877

5-Stage Clos Overview | 877

Create 5-Stage Clos Network | 879

Modify 5-stage Clos Network | 880

Juniper EVPN Support | 880

Overview | 881

EVPN multi-homing Terminology and Concepts | 881

Topology Specification | 883

EVPN Services | 884

Configuration Rendering | 886

Intent-Based Analytics with apstra-cli Utility | 889

IBA with apstra-cli Overview | 889

Install apstra-cli | 890

Install Packages | 890

Create Agent Profiles | 892

Create Agents | 893

Update Agents from apstra-cli | 895

Install IBA Probes | 896

Apstra IBA Probes Examples | 898

AOSOM-Streaming Guide | 902

AOSOM-Streaming Overview | 902

Configure Aosom-Streaming | 907

Reconfigure Aosom-streaming after Apstra Server Upgrade | 909

Build Aosom-Streaming VM (Optional) | 910

| Troubleshooting | 914

Mixed Uplink Speeds between Leaf Devices and Spine Devices | 915

References

Apstra Feature Matrix | 918

| Apstra 4.1.2 Feature Matrix | 919

| Apstra 4.1.1 Feature Matrix | 939

| Apstra 4.1.0 Feature Matrix | 959

Qualified Devices and NOS Versions | 979

| Apstra Release 4.1.2 | 981

| Apstra Release 4.1.1 | 984

| Apstra Release 4.1.0 | 986

NOS Upgrade Paths (Devices) | 988

Predefined Dashboards (Analytics) | 995

| Device Environmental Health Summary Dashboard (New in 4.1.2) | 996

| Device Health Summary Dashboard | 996

| Device Telemetry Health Summary Dashboard (New in 4.1.2) | 997

| Drain Validation Dashboard | 997

| Throughput Health MLAG Dashboard | 997

| Traffic Trends Dashboard | 998

| Virtual Infra Fabric Health Check Dashboard | 998

| Virtual Infra Redundancy Check Dashboard | 998

Predefined Probes (Analytics) | 999

| BGP Session Flapping Probe | 1000

| Bandwidth Utilization Probe | 1002

| Critical Services: Utilization, Trending, Alerting Probe | 1005

| Device Environmental Checks Probe (New in 4.1.2) | 1006

| | |
|--|------|
| Device System Health Probe | 1007 |
| Device Telemetry Health Probe | 1009 |
| Device Traffic Probe | 1010 |
| Drain Traffic Anomaly Probe | 1014 |
| ECMP Imbalance (External Interfaces) Probe | 1015 |
| ECMP Imbalance (Fabric Interfaces) Probe | 1017 |
| ECMP Imbalance (Spine to Superspine Interfaces) Probe | 1020 |
| ESI Imbalance Probe | 1022 |
| EVPN Host Flapping Probe | 1024 |
| EVPN VXLAN Type-3 Route Validation Probe | 1025 |
| EVPN VXLAN Type-5 Route Validation Probe | 1027 |
| External Routes Probe | 1029 |
| Hot/Cold Interface Counters (Fabric Interfaces) Probe | 1030 |
| Hot/Cold Interface Counters (Specific Interfaces) Probe | 1034 |
| Hot/Cold Interface Counters (Spine to Superspine Interfaces) Probe | 1036 |
| Hypervisor and Fabric LAG Config Mismatch Probe (Virtual Infra) | 1038 |
| Hypervisor and Fabric VLAN Config Mismatch Probe (Virtual Infra) | 1039 |
| Hypervisor & Fabric VLAN Config Mismatch Probe Overview | 1040 |
| Usage with NSX-T Integration | 1041 |
| Usage with VCenter Integration | 1045 |
| Hypervisor MTU Mismatch Probe (Virtual Infra NSX-T Only) | 1046 |
| Hypervisor MTU Threshold Check Probe (Virtual Infra) | 1046 |
| Hypervisor Missing LLDP Config Probe (Virtual Infra) | 1047 |
| Hypervisor Redundancy Checks Probe (Virtual Infra) | 1048 |
| Interface Flapping (Fabric Interfaces) Probe | 1049 |
| Interface Flapping (Specific Interfaces) Probe | 1051 |
| Interface Flapping (Specific Interfaces) Probe | 1052 |

Interface Policy 802.1x Probe | 1054

LAG Imbalance Probe | 1055

Leafs Hosting Critical Services: Utilization, Trending, Alerting Probe | 1057

Link Fault Tolerance in Leaf and Access LAGs Probe | 1058

MLAG Imbalance Probe | 1060

Multiagent Detector Probe | 1064

Optical Transceivers Probe | 1065

Packet Discard Percentage Probe | 1067

Spine Fault Tolerance Probe | 1069

Total East/West Traffic Probe | 1070

VMs without Fabric Configured VLANs Probe (Virtual Infra) | 1072

VXLAN Flood List Validation Probe | 1075

Probe Processors (Analytics) | 1077

Processor: Accumulate | 1078

Processor: Average | 1082

Processor: Comparison | 1083

Processor: EVPN Type 3 | 1085

Processor: EVPN Type 5 | 1085

Processor: Extensible Service Data Collector | 1086

Processor: Generic Graph Collector | 1090

Processor: Generic Service Data Collector | 1093

Processor: Interface Counters | 1096

Processor: Logical Operator | 1099

Processor: Match Count | 1100

Processor: Match Percentage | 1102

Processor: Match String | 1104

Processor: Max | 1107

Processor: Min | 1109

Processor: Periodic Average | 1111

Processor: Range | 1114

Processor: Ratio | 1117

Processor: Service Data Collector | 1119

Processor: Set Comparison | 1122

Processor: Set Count | 1124

Processor: Standard Deviation | 1125

Processor: State | 1127

Processor: Subtract | 1130

Processor: Sum | 1131

Processor: System Utilization | 1132

Processor: Time in State | 1133

Processor: Traffic Monitor | 1138

Processor: Union | 1141

Processor: VXLAN Floodlist | 1143

Configlet Examples (Design) | 1143

Apstra-CLI Commands | 1149

Apstra EVPN Support Addendum | 1151

Qualified Vendor and NOS | 1151

Limitations | 1152

TCAM Carving in NX-OS | 1153

Arista EOS VxLAN Routing | 1154

Graph Node VTEP Types | 1156

Apstra Server Configuration File | 1159

Agent Configuration File (Devices) | 1170

Controller Section | 1171

Service Section | 1172

Logrotate Section | 1173

Device Info Section | 1174

Device Profile Section | 1174

Graph | 1175

Graph Overview | 1175

Query Specification | 1176

Change Notification | 1178

Notification Processing | 1178

Putting It All Together | 1180

Convenience Functions | 1181

Apstra Graph Datastore | 1190

Juniper Apstra Technology Preview | 1191

Introduction

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.

You can familiarize yourself with Apstra by looking at the general workflow, then get started by referring to the [Apstra Quick Start](#) guide.

General Workflow

IN THIS SECTION

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

The general workflow starts with installing and configuring Apstra software on one of the supported hypervisors. You'll replace the SSL certificate and default passwords to increase security. You can 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 listed here.

Install Apstra Software

[Install and configure Apstra software](#) on one of the supported hypervisors.

Devices

Access the ["Apstra GUI" on page 3](#) and get your devices ready.

1. ["Device profiles" on page 95](#) (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. ["Add devices" on page 79](#) to be managed by the Apstra environment.

Design

1. ["Logical devices" on page 7](#) (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 13](#) (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 23](#) (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 35](#) (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 262](#), ["IPv4 addresses" on page 267](#), and ["IPv6 addresses" on page 270](#) 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 274](#) from one of the templates in the design section.
2. Assign ["resources" on page 277](#), ["device profiles" on page 280](#), and ["devices" on page 295](#) (S/Ns) to build the network (Blueprints > <your_blueprint_name> > Staged > Physical > Build)
3. Review the calculated cabling map (Blueprints > <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've finished building your network, "[commit](#)" [on page 692](#) the blueprint (Blueprints > <your_blueprint_name> > Uncommitted). Committing a blueprint initiates work on the intent and pushes configuration changes on assigned devices to realize it on the network.
5. Review the "[blueprint dashboard](#)" [on page 274](#) (Blueprints > Dashboard) for "[anomalies](#)" [on page 613](#). 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 the "[Apstra-CLI](#)" [on page 859](#) utility to override the cabling in the blueprint with discovered cabling.

Next Steps

After your deployment is running, you can "[build](#)" [on page 414](#) the virtual environment with "[virtual networks](#)" [on page 409](#) and "[routing zones](#)" [on page 429](#), as needed.

Apstra GUI

IN THIS SECTION

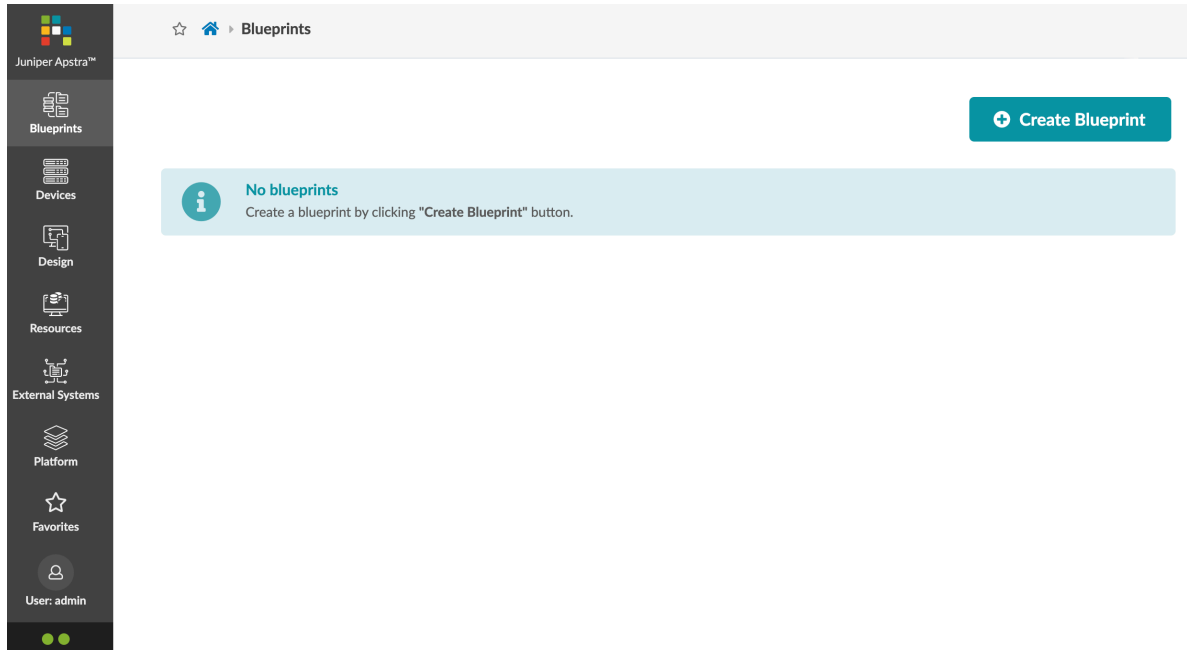
- [Access Apstra GUI | 3](#)
- [Reset Apstra GUI Admin Password | 4](#)
- [Check Apstra GUI Version | 5](#)
- [Update Apstra GUI Version | 5](#)
- [Restore Apstra GUI Version | 6](#)

Access Apstra GUI

You can design, build, deploy, operate and validate your network from the Apstra 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, and you didn't replace it with a signed one when you installed the software. We recommend, for security reasons, that you replace the SSL certificate.
3. From the login page, enter username **admin** and the secure password that you set when you configured the Apstra server. (Entering the password incorrectly too many times locks you out for a few minutes depending on how password requirements have been configured.) The main screen appears.



RELATED DOCUMENTATION

[Edit Password Complexity Requirements](#) | 752

Reset Apstra GUI Admin Password

If you reset (a lost) Apstra GUI admin password to the default, we highly recommend that you immediately change it to a secure one. User **admin** has full root access. Juniper is not responsible for security-related incidents because of not changing default passwords.

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

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:~$
```

3. Log in to the Apstra GUI (default password: **admin**), then navigate to **Platform > User Management > Users**.
4. Click username **admin**, then click the **Change Password** button (top-right)
5. Enter a secure password that meets complexity requirements, then re-enter the new password.
6. Click **Change Password** to update the password.

RELATED DOCUMENTATION

[Edit Password Complexity Requirements](#) | 752

Check Apstra GUI Version

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

Update Apstra GUI Version

When it's available, you can install an optional Apstra server GUI 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 GUI 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 Apstra GUI Version

You can restore a previous Apstra 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.

Design

IN THIS SECTION

- [Logical Devices \(Datacenter Design\) | 7](#)
- [Interface Maps \(Datacenter Design\) | 13](#)
- [Rack Types \(Datacenter Design\) | 23](#)
- [Templates \(Datacenter Design\) | 35](#)
- [Config Templates \(Freeform Design\) | 44](#)
- [Configlets \(Datacenter Design\) | 45](#)
- [Property Sets \(Datacenter Design\) | 52](#)
- [TCP/UDP Port Aliases \(Datacenter Design\) | 56](#)
- [Tags \(Design\) | 58](#)

Logical Devices (Datacenter Design)

IN THIS SECTION

- [Logical Device Overview | 7](#)
- [Create Logical Device | 10](#)
- [Edit Logical Device | 12](#)
- [Delete Logical Device | 13](#)

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 1: Logical Device Parameters

| Name | Description |
|---------------------|---|
| Logical device name | A unique name to identify the logical device, 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 1: 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"> Superspines (5-stage DC fabric only) Spine Leaf Access (limited support) - Port is configured to face an access device. To learn more about this feature and its limitations, contact "Juniper Support" on page 824. Peer (link between two leaf devices) Unused - not rendered (for example, a dead port) Generic - Certain roles are not specified in logical devices (for example, a firewall, external router, bare metal server, or load balancer). |

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™

☆ Home > Design > Logical Devices

Blueprints

Devices

Design

Resources

External Systems

Platform

Favorites

User: admin

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 x 10 Gbps Generic</p> <p>8 x 40 Gbps Superspine • Spine • Generic</p> | <p>Edit Clone Delete</p> |
| 1 × 1 Gbps | 1 | 1 | <p>AOS-1x1-1</p> <p>1 x 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](#) | 10

1. From the left navigation menu, navigate to **Design > Logical Devices** and click **Create Logical Device**.
2. Enter a unique 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 table 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 *
24

Speed *
10 Gbps ▾

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 PORT GROUPS Connected to ▾

96 ports
0 assigned • 96 available

No port groups created

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 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 table view. (In the overview above, it's the first one in the table.)

Edit Logical Device

If a logical device is linked to an "interface map" on page 13, 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 35.

1. Either from the table 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 table view.

Delete Logical Device

If a logical device is linked to an ["interface map" on page 13](#), it cannot be deleted.

1. Either from the table 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 table view.

Interface Maps (Datacenter Design)

IN THIS SECTION

- [Interface Map Overview | 13](#)
- [Create Interface Map | 15](#)
- [Example: Create Interface Map with Breakout Ports | 16](#)
- [Example: Inter Port Constraints - Disabled Ports | 19](#)
- [Edit Interface Map | 22](#)
- [Delete Interface Map \(Design\) | 23](#)

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 |

2. Enter a unique name (64 characters or fewer). This field can be left blank for the name to be created for you that consists of the concatenation of the names of the selected logical device and device profile.
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 7](#) 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 95](#) 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 table 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 won't 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 7](#) 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. This displays 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 | 1. Click to see ports ▼ Select interfaces |
| 2. Drag from here <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 | To here ▶ 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.

6. Under **Device profile interfaces** (middle-right) click **Select Interfaces** for the 40 GbE ports. This displays 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 | 96 / 96 | Select interfaces |
| 40 Gbps | Superspine • Spine • External Router | 0 / 8 | Select interfaces |

1. Click to see ports

2. Drag from here

To here

Interface map preview Click on interface to toggle the details

Checkmarks indicate mapped interfaces

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]

8. Click **Create** to create the interface map and return to the table view. The new interface map is shown in the overview screenshot above.

Example: Inter Port Constraints - Disabled Ports

IN THIS SECTION

- Inter Port Constraint Overview | 20
- Disable Unused Ports | 21

Inter Port Constraint Overview

(Cumulus is no longer supported as of Apstra version 4.1.0, although Cumulus examples remain for illustrative purposes.) Inter port constraints for Cumulus devices are handled in both the device profile 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

1. Either from the table 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 table view.

Delete Interface Map (Design)

1. Either from the table 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 table view.

Rack Types (Datacenter Design)

IN THIS SECTION

- [Rack Type Overview | 23](#)
- [Create Rack Type | 30](#)
- [Example: Create Rack Type | 31](#)
- [Edit Rack Type in Global Catalog | 34](#)
- [Edit Rack Type in Template | 34](#)
- [Edit Rack Type in Blueprint | 35](#)
- [Delete Rack Type | 35](#)

Rack Type Overview

IN THIS SECTION

- [Summary | 24](#)
- [Leaf Devices | 24](#)
- [Access Switches | 26](#)
- [Generic Systems | 28](#)
- [Access Rack Types | 30](#)

Rack types define the type and number of leaf devices, access switches and/or generic systems 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 35](#), you'll use rack types to build the structure of your network. Rack types include the details in the following sections:

Summary

| Summary | Description |
|---------------------------------|--|
| Name (and optional description) | A unique name to identify the rack type, 17 characters or fewer |
| Fabric connectivity design | <ul style="list-style-type: none"> • L3 Clos - used in 3-stage and 5-stage fabric templates with spine devices. The spine level connects leaf devices to each other. • L3 Collapsed - (Junos only) - used in collapsed (spineless) templates. Leaf devices are connected directly to each other via full mesh. |

Leaf Devices

| Leaf Devices | 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)

| Leaf Devices | Description |
|---------------------|--|
| Redundancy Protocol | <p>CAUTION: Make sure that the intended platform supports the chosen redundancy protocol. For example, L3 MLAG peers are not supported on SONiC, and ESI is supported on Junos only.</p> <ul style="list-style-type: none"> • None - For single-homed connections • MLAG - For dual-homed connections. Both switches use the same logical device. <ul style="list-style-type: none"> • MLAG Keepalive VLAN ID - 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. <p>NOTE: Network device vendors have varying requirements for "reserved" VLAN ID ranges. For example, Cisco NX-OS reserves the 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> • Peer Links, and Link speed - Number of links between the MLAG devices, and their speed • Peer Link Port Channel ID • L3 peer links, and Link speed -Used mainly for BGP peering between border MLAG leaf devices 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 leaf devices as well as for avoiding blackholing traffic for a VRF. • L3 Peer Link Port Channel ID • ESI (Junos only) - Ethernet Segment ID assigned to the bundled links. Specifying device platforms other than Juniper Junos (such as Cisco, Arista) results in blueprint build errors. See "Juniper EVPN Support" on page 880 for information about Juniper ESI support and "ESI MAC MSB settings" on page 551 for more information about ESI. |
| Tags | <p>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

ESI support at the access layer is supported as a technology preview in Apstra version 4.1.0 and as GA in Apstra version 4.1.1. Before 4.1.0, you couldn't dual-home generic systems (servers) to access switches. If you wanted to dual-home a generic system, it had to be up to the leaf layer. In Apstra version 4.1.0, we extend the concept of switch pairs to the access layer. We're leveraging EVPN at the access layer to enable ESI-LAG towards the generic system while keeping the L2 only nature of the access switch role.

Supported/Unsupported Topologies for ESI Access:

- Each member of an access switch pair dual-attached to the leaf pair is supported.
- Each member of an access switch pair single-attached to the leaf pair is supported.
- One member of an access switch pair dual-attached to the leaf pair and the other member of an access switch pair single-attached to the leaf pair is not supported.

This is supported on 3-Stage, 5-Stage, and collapsed fabric blueprints. Day 2 topology changes are available through Add/Edit/Remove Racks.

Requirements for the switch model acting as Access Switch are:

- EVPN-VxLAN with VTEP support is required on the Access Switches.
- L2 VxLAN only is required, L3 VxLAN (RIOT) is not required, and will continue to be available only at the leaf layer.

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 7](#), 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 13](#) per standard procedure.
3. ["Create a rack type" on page 30](#) with configured access switches.
4. Create a ["template" on page 35](#) that uses rack types with access switches.
5. Create a blueprint and build it following the general ["workflow" on page 2](#). You can perform the same tasks as for other blueprints.

| Access Switches | Description |
|-----------------|------------------------|
| Name | 64 characters or fewer |

(Continued)

| Access Switches | Description |
|---------------------|--|
| Access Switch count | Number of access switches. These switches share the same logical link group. |
| Logical Device | Logical device is applied to this access switch. |
| Redundancy Protocol | <ul style="list-style-type: none"> • None - For single-homed connections • ESI (Junos only) - Ethernet Segment ID assigned to the bundled links. Specifying device platforms other than Juniper Junos (such as Cisco, Arista) results in blueprint build errors. For information about Juniper ESI support, see "Juniper EVPN Support" on page 880 and for information about ESI, see "ESI MAC MSB settings" on page 551. • L3 Peer Links - Number of L3 peer links between both access switches. • Link Speed - Link speed on the peer link interfaces. |
| Tags | 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. |

(Continued)

| Access Switches | Description |
|-----------------|--|
| Logical Link | <ul style="list-style-type: none"> • Name - 64 characters or fewer • Leaf - Leaf configured in Leafs section • Physical link count per individual switch • Link speed • Tags - 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. |

Access Layers on Apstra Version 4.1.1

This feature is classified as General Availability as of Apstra version 4.1.1.

Access Layers on Apstra Version 4.1.0



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 1191](#) page or contact ["Juniper Support" on page 824](#).

Generic Systems

| Generic Systems | Description |
|-----------------|------------------------|
| Name | 64 characters or fewer |

(Continued)

| Generic Systems | Description |
|------------------------------|--|
| 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. |
| Logical Device | The generic system network device |
| Tags | 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. |
| Logical Link | <ul style="list-style-type: none"> • Name - 64 characters or fewer • Switch - Leaf configured in Leafs section • LAG Mode <ul style="list-style-type: none"> • LACP (Active) - Link Aggregation Control Group (LACP) in active mode - This mode actively advertises LACP BPDU even when the neighbor does not. • LACP (Passive) - Link Aggregation Control Group (LACP) in passive mode - This mode doesn't generate LACP BPDU until it sees one from a neighbor. • Static LAG (no LACP) - Static LAGs don't participate in LACP and will unconditionally operate in forwarding mode. • No LAG - This link is not part of a LAG. • Physical link count per individual leaf, and Link speed) - 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. • Tags - 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. |



NOTE: You can also add generic systems to blueprints as a Day 2 operation. For more information, see ["Add Generic System" on page 306](#).

Access Rack Types

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

| Name | Leaf Count | Generic System Count | Actions |
|-------------------|---------------|----------------------|-------------------|
| leaf | 1 single leaf | 2 | Edit Clone Delete |
| leafs | 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 |
| L2 MLAG 1x access | 1 MLAG pair | 2 | Edit Clone Delete |

Create Rack Type

1. From the left navigation menu, navigate to **Design > Rack Type** and click **Create Rack Type**.
2. Enter a unique rack type name (17 characters or fewer).
3. Enter a description (optional), then select a fabric connectivity design (L3 Clos, L3 Collapsed).
4. 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 unique 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

Preview

Leafs Access Switches Generic Systems

Topology Logical Devices

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

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 leaf devices.
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 device. 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 a 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 labeled 'MyLeaf1_1' and 'MyLeaf2_1'. Below these, there 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 table 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 table 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 table 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 35](#) page (Design > Templates > Edit Template).

Edit Rack Type in Blueprint

You can edit rack types in active running blueprints. For more information, see ["Edit Rack" on page 390](#) (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 391](#) (Blueprints > Staged > Physical > Racks).

1. To delete a rack type in the global catalog, either from the table 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 table view.

Templates (Datacenter Design)

IN THIS SECTION

- [Template Overview | 35](#)
- [Create Rack Based Template | 41](#)
- [Create Pod Based Template | 41](#)
- [Create Collapsed Template | 42](#)
- [Edit Template | 43](#)
- [Update Rack Type in Rack Based Template | 43](#)
- [Delete Template | 43](#)

Template Overview

IN THIS SECTION

- [Rack-based Template | 36](#)
- [Pod-based Template | 38](#)
- [Collapsed Template | 40](#)

Templates are used to create blueprints. They define a network's policy intent and structure. 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 table view. Many predefined templates are provided for you. Click a template name to see its details. You can create, clone, edit, and delete templates.

The screenshot shows the Juniper Apstra web interface. On the left, the navigation menu is open, with 'Design' selected and 'Templates' highlighted. A red arrow labeled '1.' points to the 'Design' menu item, and another red arrow labeled '2.' points to the 'Templates' sub-item. The main content area displays a table of templates. A red arrow points to the template name 'L2 superspine single plane' with the text 'Click template name for details'.

| | Type | Overlay Control Protocol | Actions |
|--------------|------------|--------------------------|-------------------------|
| MP-EBGP EVPN | COLLAPSED | MP-EBGP EVPN | [Edit] [Clone] [Delete] |
| Static VXLAN | RACK BASED | Static VXLAN | [Edit] [Clone] [Delete] |
| Static VXLAN | RACK BASED | Static VXLAN | [Edit] [Clone] [Delete] |
| Static VXLAN | RACK BASED | Static VXLAN | [Edit] [Clone] [Delete] |
| Static VXLAN | RACK BASED | Static VXLAN | [Edit] [Clone] [Delete] |
| Static VXLAN | POD BASED | Static VXLAN | [Edit] [Clone] [Delete] |
| Static VXLAN | POD BASED | Static VXLAN | [Edit] [Clone] [Delete] |

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 2: Rack-based Template Policies

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

Table 2: Rack-based Template Policies (*Continued*)

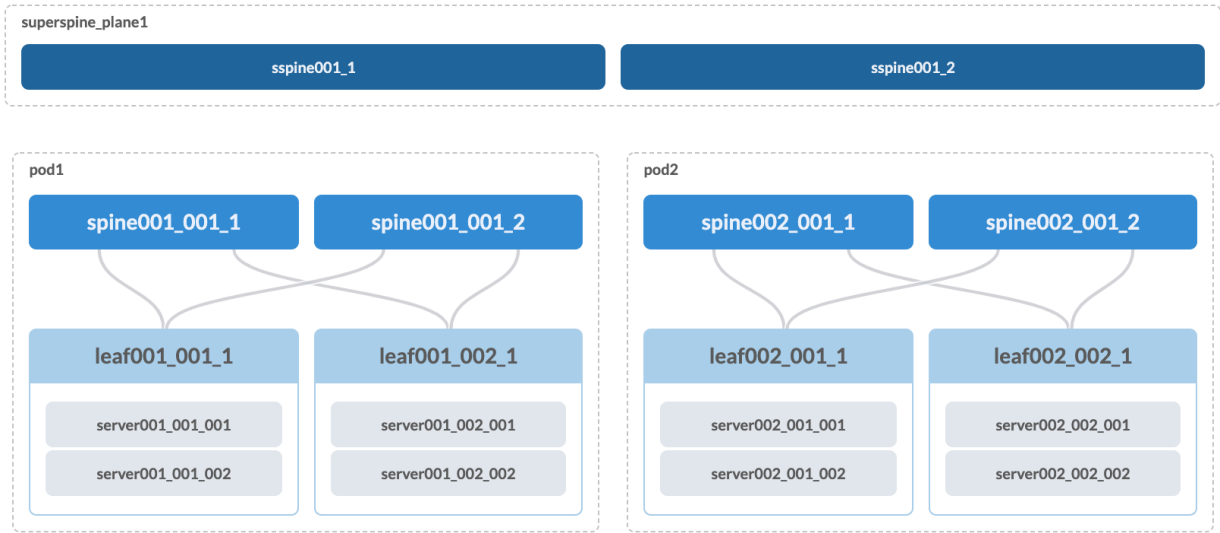
| Policy | Options |
|-----------------------------------|---|
| Overlay Control Protocol | <ul style="list-style-type: none"> Defines the inter-rack virtual network overlay protocol in the fabric. Overlay control protocol on <i>deployed</i> blueprints can't be changed. Static VXLAN - uses static VXLAN routing the Head End Replication (HER) flooding to distribute Layer 2 virtual network traffic between racks. MP-EBGP EVPN - 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 "Virtual Networks" on page 409 for more information. External systems must be connected to racks (not spine devices). |
| Spine to Leaf Links Underlay Type | <ul style="list-style-type: none"> IPv4 - uses addresses from "IPv4 resource pools" on page 267. IPv6 RFC-5549 - uses addresses from "IPv6 resource pools" on page 270. Not supported when overlay control protocol is MP-EBGP EVPN. IPv6-IPv6 Dual Stack |

Table 3: Rack-based Template Structure

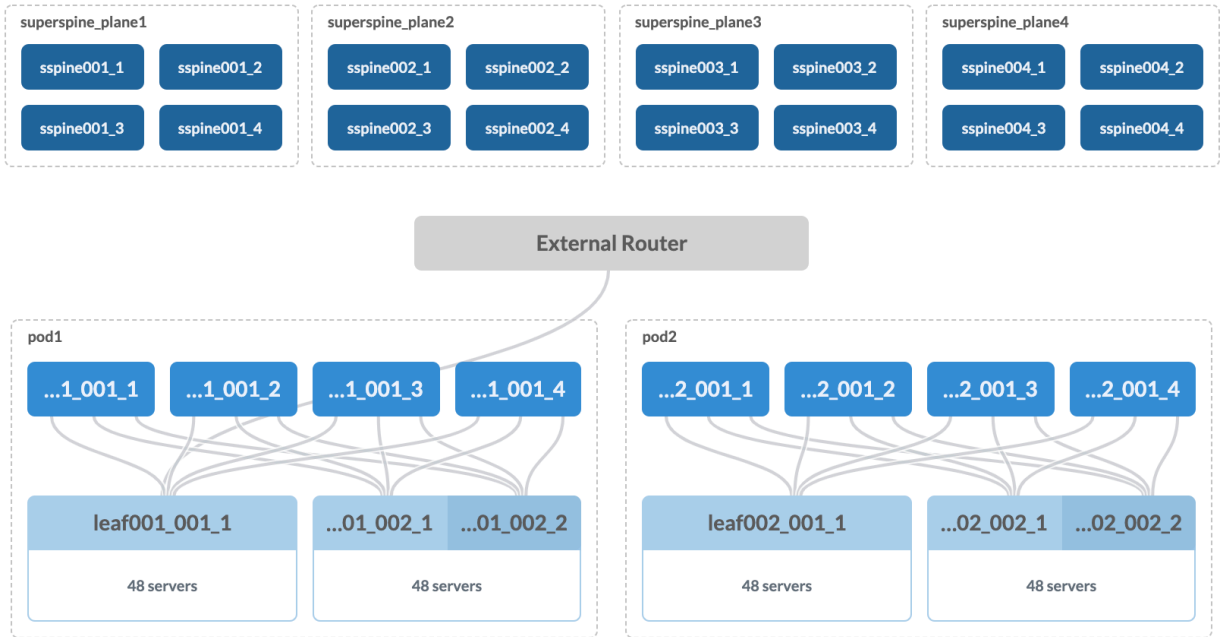
| Structure | Options |
|------------|---|
| Rack Types | Type of rack and number of each selected " rack type " on page 23. ESI-based rack types in rack-based templates without EVPN are invalid. |
| Spines | <ul style="list-style-type: none"> Spine Logical Device and Count - Type and number of spine "logical devices" on page 7 Links per Superspine Count and Speed - Number and speed of links to any superspine devices Tags - User-specified. Select "tags" on page 58 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. |

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 superspine devices. 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 877](#) for more information.



Single plane, dual superspine



4 x plane, 4 x superspine

Pod-based templates include the following details:

Table 4: Pod-based Template Policies

| Policy | Option |
|---------------------------|--|
| Spine to Superspine Links | <ul style="list-style-type: none"> • IPv4 - uses addresses from "IPv4 resource pools" on page 267. • IPv6 RFC-5549 - uses addresses from "IPv6 resource pools" on page 270. Not supported when overlay control protocol is MP-EBGP EVPN. • IPv4-IPv6 Dual Stack |
| Overlay Control Protocol | <ul style="list-style-type: none"> • Defines inter-rack virtual network overlay protocol used in the fabric. Overlay control protocol on <i>deployed</i> blueprints can't be changed. • Static VXLAN - uses static VXLAN routing the Head End Replication (HER) flooding to distribute Layer 2 virtual network traffic between racks. • MP-EBGP EVPN - 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 "Virtual Networks" on page 409 for more information. External systems must be connected to racks (not spine devices). |

Table 5: 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"> • Superspine Logical Device and Count • Plane Count and Per Plane Count - Number of planes and number of superspine devices per plane • Tags - User-specified. Select "tags" on page 58 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. |

Collapsed Template

Collapsed templates 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. Collapsed templates have the following limitations:

- No support for upgrading collapsed L3 templates to L3 templates with spine devices (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 leaf devices - the two leaf devices 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.

Collapsed templates include the following details:

Table 6: Collapsed Template Policies

| Policy | Options |
|--------------------------|---|
| Overlay Control Protocol | <ul style="list-style-type: none"> • Defines the inter-rack virtual network overlay protocol used in the fabric. Overlay control protocol on deployed blueprints can't be changed. • Static VXLAN - uses static VXLAN routing the Head End Replication (HER) flooding to distribute Layer 2 virtual network traffic between racks. • MP-EBGP EVPN - 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 "Virtual Networks" on page 409 for more information. External systems must be connected to racks (not spine devices). |

Table 7: Collapse Template Structure

| Structure | Options |
|------------|--|
| Rack Types | Type of L3 collapsed rack and number of each selected "rack type" on page 23 . |

Table 7: Collapse Template Structure *(Continued)*

| Structure | Options |
|----------------------------|--|
| 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 at the mesh level. |

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 23](#) and/or ["logical devices" on page 7](#) 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 unique name (64 characters or fewer).
4. Select **RACK BASED**.
5. Select applicable policies.
6. 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**.
7. 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.
8. For 5-stage designs, enter the number and connection speed of links for **Superspine Connectivity**.
9. Select tags, as applicable (to specify external routers for example), from the drop-down list or create them on-the-fly.
10. Click **Create** to create the template and return to the table view.

Next Steps: Create a ["blueprint" on page 274](#) 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" on page 877](#).

1. If your design requires ["templates" on page 35](#), ["rack types" on page 23](#) and/or ["logical devices" on page 7](#) 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 unique name (64 characters or fewer).

4. Select **POD BASED**.
5. Select applicable policies.
6. 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.
7. Select a **Superspine Logical Device** from the drop-down list.
8. Select the number of planes and the number of superspine devices per plane.
9. Select tags, as applicable (to specify external routers for example), from the drop-down list or create them on-the-fly.
10. Click **Create** to create the template.

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

Create Template

Structure

Pods

L2 Virtual Superspine x 3

[Add pods](#)

Superspines

Superspine Logical Device *
AOS-7x10-superspine x

Ports Summary
AOS-7x10-superspine
7 x 10 Gbps Spine

Plane Count * 2 Per Plane Count * 2

Preview

Topology Pods Superspine Logical Device

⚠ Superspine links are hidden

Next Steps: Create a ["blueprint" on page 274](#) from the template.

Create Collapsed Template

1. From the left navigation menu, navigate to **Design > Templates** and click **Create Template**.
2. Enter a unique name (64 characters or fewer).
3. Select **COLLAPSED**.
4. Select applicable policies.
5. Select a ["rack type" on page 23](#) from the drop-down list (only L3 collapsed rack types are available for selecting), then select the number of mesh links and their speeds. Notice that as you enter information, the topology preview on the right changes accordingly.
6. Click **Create** to create the template and return to the table view.

Next Steps: Create a ["blueprint" on page 274](#) 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 and return to the table view.

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 and return to the table view.

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 and return to the table view.

Config Templates (Freeform Design)

IN THIS SECTION

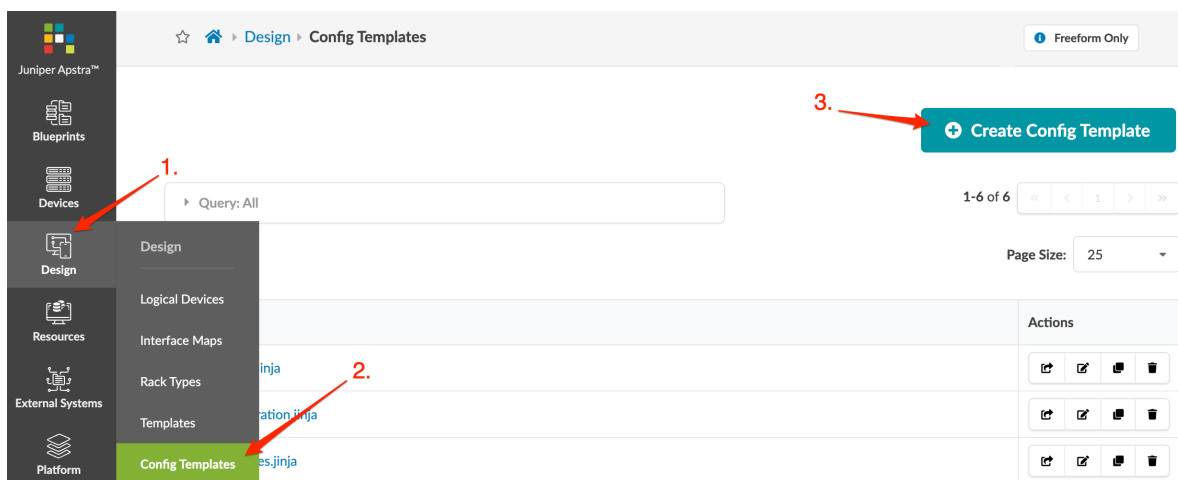
- [Create Config Template | 44](#)
- [Edit Config Template | 45](#)
- [Delete Config Template | 45](#)

Config templates are text files used to configure internal systems in Freeform. You'll assign a config template to every internal system. You could paste configuration directly from your devices into a config template to create a static config template, but then you wouldn't be using the potential of config templates. With some Jinja2 knowledge (and maybe some Python), you can parametrize config templates to do powerful things.

For more information about config templates, see "[Config Templates \(Freeform Blueprint\)](#)" on page 680.

Create Config Template

1. From the left navigation menu of the Apstra GUI, navigate to **Design > Config Templates** and click **Create Config Template**.



2. Enter a unique name for the config template including the `.jinja` extension. (The `.jinja` extension is required even if you're not using Jinja.)
3. Enter or paste your content into the **Template Text** field.
4. Click **Create** to create the config template and return to the config template table view. Your newly created config template is available to be imported into any blueprint catalog.



NOTE: You can also create config templates directly in the blueprint catalog. If you've already created your internal systems in your blueprint, you'll have access to its Device Context all in one place which makes it easier to get device information that you need for config templates.

Edit Config Template

1. From the left navigation menu of the Apstra GUI, navigate to **Design > Config Templates** to go to the table view.
2. Either from the table view or the details view, click the **Edit** button for the config template to edit.
3. Make your changes.
4. Click **Update** to update the config template and return to the table view.

Delete Config Template

1. From the left navigation menu of the Apstra GUI, navigate to **Design > Config Templates** to go to the table view.
2. Either from the table view or the details view, click the **Delete** button for the config template to delete.
3. Click **Delete** to stage the deletion and return to the table view.

Configlets (Datacenter Design)

IN THIS SECTION

- [Configlet Overview](#) | 46
- [Create Configlet](#) | 50
- [Export / Edit / Delete Configlet \(Design\)](#) | 51

Configlet Overview

IN THIS SECTION

- [Configlet Applications | 46](#)
- [When Not to Use Configlets | 47](#)
- [Configlet Parameters | 47](#)
- [Configuration Rendering Order | 49](#)
- [View Configlets \(Design\) | 50](#)

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 sets" on page 52](#) instead.



NOTE: Improperly configured configlets may 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 config. For more information, see ["Configuration Deviation" on page 613](#).

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 8: Configlet Parameters

| Name | Description |
|-------------------------|---|
| Configlet Name | A unique name to identify the configlet, 64 characters or fewer |
| Config Style (NOS Type) | Junos, NX-OS, EOS, SONiC |

Table 8: Configlet Parameters *(Continued)*

| Name | Description |
|---|--|
| <ul style="list-style-type: none"> Section: System (NX-OS, EOS, SONiC) | <ul style="list-style-type: none"> Runs commands as root user. Improper changes could break the functionality of the reference design and take down a network. |
| <ul style="list-style-type: none"> Section: Top-Level: Hierarchical (previously called System) (Junos) | <ul style="list-style-type: none"> 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> Can use in conjunction with File configlets to restart processes or perform administrative tasks after File configlets render. System configlets can nest other configuration. For NX-OS and EOS, the appropriate configure terminal context is applied. It doesn't need to be part of the configlet. |
| Section: Top-Level: Set / Delete (Junos) | Author configlets using Juniper "Set" style rather than structured JSON |
| <ul style="list-style-type: none"> Section: Interface (NX-OS, EOS) Section: Interface-Level: Hierarchical (Junos) | <ul style="list-style-type: none"> For physical devices only. You specify the interface when you "import" on page 283 the configlet into a blueprint (scope). |
| Section: Interface-Level: Set (Junos) | Author configlets using Juniper "Set" command rather than structured JSON. Text is validated to begin with 'set'. |
| Section: Interface-Level: Delete (Junos) | Author configlets using Juniper "Delete" command rather than structured JSON. Text is validated to begin with 'delete'. |

Table 8: Configlet Parameters (*Continued*)

| Name | Description |
|----------------------------------|--|
| Section: File (SONiC) | <ul style="list-style-type: none"> 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 you can't restore it to its original content. Improper use can take down a network. Do not use on config files of critical processes (such as <code>/etc/frr/frr.conf</code> or <code>/etc/network/interfaces/</code>). Contents are written, as root user, to the <code>/etc</code> directory file (because of Apstra's Docker container host mount). To write to a file outside of <code>/etc</code> (<code>/usr</code> for example) build the File configlet, then use a System configlet to move the file afterwards. |
| Section: System Top (NX-OS, EOS) | Ensures that you can overwrite a setting 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) | <ul style="list-style-type: none"> Configlet configuration is appended to the end of the Apstra-generated <code>/etc/frr/frr.conf</code> file and becomes part of FRR intent. Configuration is incrementally included in <code>frr-reload</code>. Template text is not validated. Errors are likely to cause deployment errors, unintended configuration and device impact. |
| Template Text | CLI commands to add configuration to devices. Issued directly to devices without validation. |
| 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:

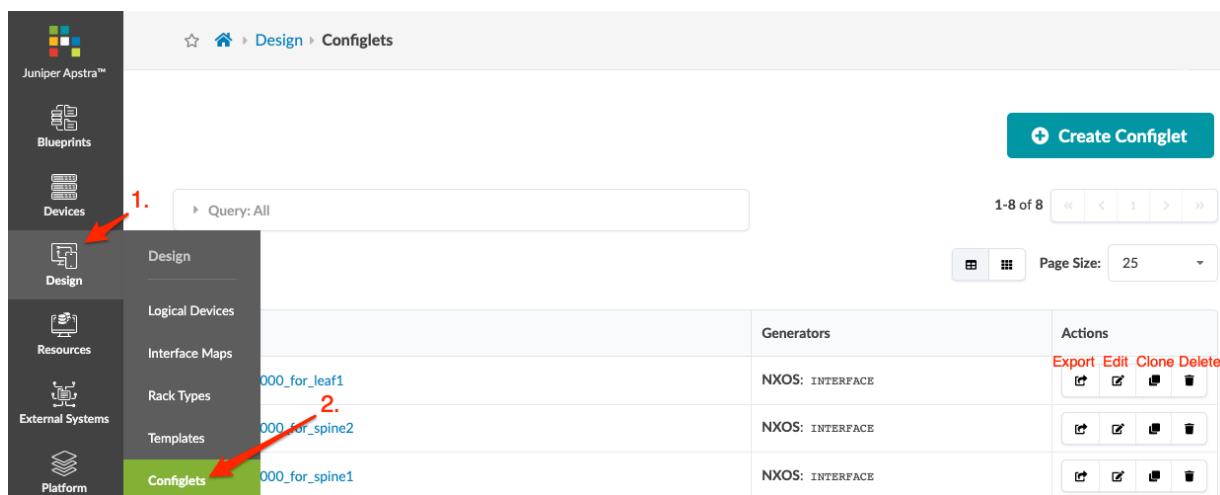
1. System Top: negation template text (NX-OS, EOS)
2. System Top: template text (NX-OS, EOS)
3. Apstra reference design

4. Interface: negation template text (NX-OS, EOS)
5. System: negation template text (Junos, NX-OS, EOS, SONiC)
6. File (SONiC)
7. System: template text (Junos, NX-OS, EOS, SONiC)
8. Interface: template text (NX-OS, EOS)

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, import, export, edit and delete configlets.



Create Configlet

To learn how you can access a dictionary of variables (device model) that you can use when you create configlets, see the ["Device Configuration Lifecycle" on page 61](#).

1. From the left navigation menu, navigate to **Design > Configlets** and click **Create Configlet**.
2. If you've created a JSON payload, click **Import Configlet** and select the file to import it. Otherwise, continue to the next step.
3. Enter a unique configlet name.
4. Select a NOS type (config style). (Cumulus is no longer supported as of Apstra 4.1.0.)
5. Select the section where you want to render the configlet. Available choices depend on the selected config style. (OSPF for external routers is no longer supported. While OSPF configlets still appear in the Apstra GUI, they should not be used.)

6. In the **Template Text** and **Negation Template Text** fields (as applicable), enter CLI commands. For Interface-Level Set or Delete configlets, do not include set or delete in the text. See Configlet examples in the Reference section. Avoid using shortened versions of commands. Jinja syntax is highlighted with color coding to improve readability, especially for complex configlets with multiple property set variables or when Jinja control structures (such as loops and conditionals) are used. Jinja syntax is validated. If Jinja syntax is incorrect, a validation error is raised.



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 52](#) (key-value pairs). For an example, see the ["Arista NTP example" on page 1143](#) in the References section.

7. If **Negation Template Text** is required, enter the CLI commands to remove the configuration.
8. For File configlets, enter the filename in the **Filename** field.
9. 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.)
10. Click **Create** to add the configlet to the global catalog.

When you're ready to use the configlet in a blueprint, ["import" on page 283](#) it into the blueprint's catalog.

Export / Edit / Delete Configlet (Design)

IN THIS SECTION

- [Export Configlet | 51](#)
- [Edit Configlet | 52](#)
- [Delete Configlet | 52](#)

Export Configlet

Exporting configlets makes it easier for SEs to deliver predefined configlets to customers and makes it easier to copy configlets across Apstra instances. (New in Apstra version 4.1.0.)

1. From the table view (Design > Configlets) or the details view, click the **Export configlet** button for the configlet to export. Configlet details are displayed.
2. To copy the contents, click **Copy**, then paste it.
3. To download the JSON file to your local computer, click **Save as File**.
4. When you've copied and/or downloaded the file, click the **X** to close the dialog.

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 / Delete Configlet \(Blueprint\)" on page 285](#) for the workflow.

1. From the table view (Design > Configlets) or the details view, click the **Edit** button for 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 table view.

Delete Configlet

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

1. Either from the table 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 table view.

Property Sets (Datacenter Design)

IN THIS SECTION

- [Create Property Set \(Datacenter Design\) | 54](#)
- [Edit / Delete Property Set \(Datacenter Design\) | 55](#)

Property sets are data sets that define device properties. They work in conjunction with configlets and Analytics probes. (Config templates in Freeform blueprints also use property sets, but they're not related to property sets in the Design catalog, as discussed here.) Instead of embedding data directly into

configlets or probes, you can store variable values in a property set, then refer to the property set from the configlet or probe. This gives you flexibility to change values later. After you create your blueprint, you'll import your configlets and property sets from the Design (global) catalog into the blueprint catalog.

First, you need to write the property set (and the configlet or probe that'll use it). You can write it in JSON, or starting in Apstra version 4.1.2, you can also write it in YAML. Apstra versions 4.1.1 and earlier, support key-value pairs only. Starting with Apstra version 4.1.2, you can also use lists, dictionaries, and nested combination of these data structures.

Below is an example of a property set and configlet that uses it to change the SNMP location field based on a provided list of system_name to location mapping.

Property Set

```
{
  "created_at": "2022-08-26T13:20:04.488463+0000",
  "updated_at": "2022-08-28t18:57:41.169692+0000",
  "values_yaml": "PS_SNMP_Locations:\n leaf1: DC-Room1-Rack32\n leaf2: DC1-room1-Rack34\n
leaf3: DC1-Room1-Rack33\n spine1: DC1-Room1-Rack30\n spine2: DC1-Room1-Rack31\n",
  "values": {
    "PS_SNMP_Locations": {
      "spine1": "DC1-Room1-Rack30",
      "spine2": "DC1-Room1-Rack31",
      "leaf1": "DC1-Room1-Rack32",
      "leaf3": "DC1-Room1-Rack33",
      "leaf2": "DC1-Room1-Rack34"
    }
  },
  "label": "PS_SNMP_Locations",
  "id": "c4006bb8-f8f4-4aa7-82c3-8da5dfc03c43"
}
```

Configlet

```
{
  "ref_archs": [
    "two_stage_l3clos"
  ],
  "generators": [
    {
      "config_style": "junos",
      "section": "system",

```


Create Property Set

Name *

Properties *

Key

Value

➕ Add a property

☐ Create Another?

Create

4. For Apstra version 4.1.2, you can use key:value pairs, lists (arrays), dictionaries and nesting of these data structures. 4.1.2 also adds YAML as an option. Select JSON or YAML, then select whether to use the editor or the builder. The builder is for JSON only. Enter property set content. To add another property, click **Add a Property**.

Create Property Set

Name *

Input Type

☒ Editor

☐ Builder

Values *

JSON

YAML

1 {}

2

☐ Create Another?

Create

5. Click **Create** to create the property set and return to the table view.

Edit / Delete Property Set (Datacenter Design)

IN THIS SECTION

Edit Property Set | 56

Delete Property Set (Datacenter Design) | 56

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 (Datacenter Design)

If a property set is assigned to a ["configlet" on page 45](#), it cannot be deleted.

1. Either from the table 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 table view.

TCP/UDP Port Aliases (Datacenter Design)

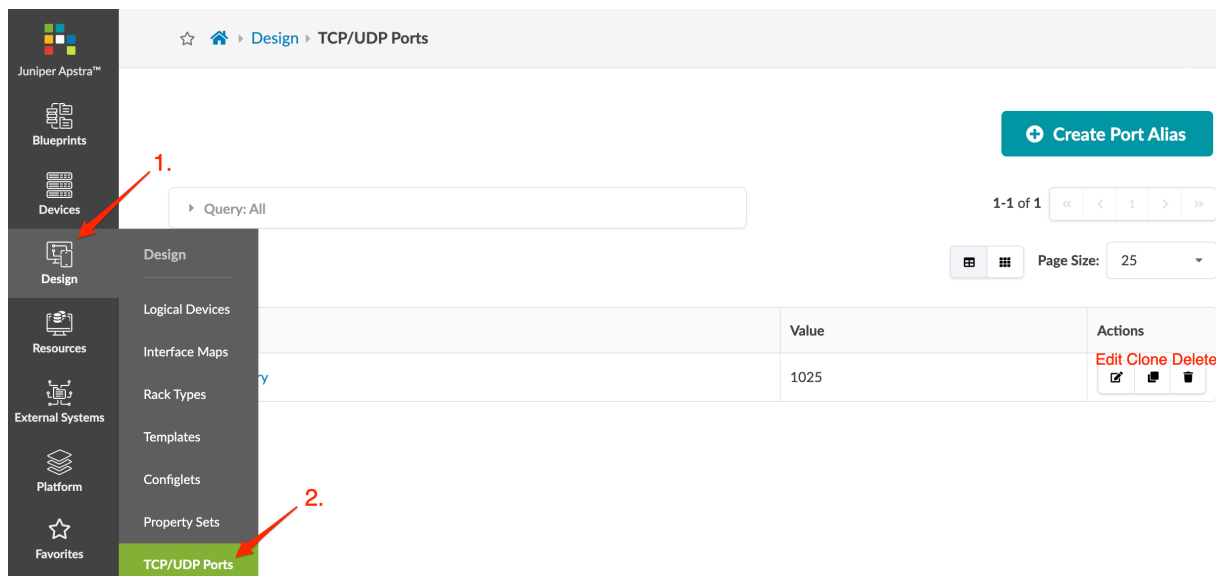
IN THIS SECTION

- [TCP/UDP Port Alias Overview | 56](#)
- [Create TCP/UDP Port Alias | 57](#)
- [Edit TCP/UDP Port Alias | 57](#)
- [Delete TCP/UDP Port Alias | 57](#)

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.



Create TCP/UDP Port Alias

1. From the left navigation menu, navigate to **Design > TCP/UDP Ports** and click **Create Port Aliases**.
2. Enter a unique alias name.
3. Enter one or more values.
4. Click **Create** to create the alias and return to the table view. 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 table 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 and return to the table view.

Tags (Design)

IN THIS SECTION

- [Tags Overview | 58](#)
- [Create Tag \(Design\) | 59](#)
- [Edit Tag \(Design\) | 59](#)
- [Delete Tag \(Design\) | 60](#)

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
 - Leafs Hosting Critical Services: Utilization, Trending, Alerting probe

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 568](#) them.

You can ["export" on page 568](#) 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 has 'Design' selected, and the 'Tags' sub-menu is highlighted. A red arrow labeled '1.' points to the 'Design' menu item. Another red arrow labeled '2.' points to the 'Tags' sub-menu item. The main content area shows a table of predefined tags:

| | Description | Actions |
|--|---|-------------------|
| | Bare Metal Servers. | Edit Clone Delete |
| | L2 and L3 Firewalls. | Edit Clone Delete |
| | Hypervisor/Compute Nodes. | Edit Clone Delete |
| | External Routers, Virtual Routers, etc. | Edit Clone Delete |

Create Tag (Design)

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

Devices

IN THIS SECTION

- [Device Configuration Lifecycle | 61](#)
- [Managed Devices | 74](#)
- [System Agents | 151](#)
- [Pristine Configuration | 213](#)
- [Telemetry | 215](#)
- [Apstra ZTP | 230](#)

Device Configuration Lifecycle

IN THIS SECTION

- [Terminology | 61](#)
- [Configuration Stages: Overview | 62](#)
- [Configuration Stages: Detail | 64](#)
- [View Device Config from Blueprint | 68](#)
- [Configuration Deviations | 71](#)
- [Device Offline \(Unavailable\) | 71](#)
- [Manually Apply Full Config | 71](#)
- [Deploy Modes | 72](#)



CAUTION: A good understanding of the Apstra device configuration lifecycle is essential. Before working with devices in the Apstra environment, we strongly recommend that you fully understand how devices are configured from the moment they are on-boarded to the moment they are decommissioned.

Terminology

The following terminology is used to identify configuration stages:

| Config | Description |
|---|--|
| Pristine Config | When you install a device agent, configuration is added to the pre-existing config on the device. Normally, the pristine config doesn't change throughout the device's lifecycle. |
| Discovery 1 Config | When you <i>acknowledge</i> a device, Apstra adds basic configuration, including enabling LLDP on all interfaces. |
| Ready Config (previously known as Discovery 2 Config) | When you assign a device to a blueprint without deploying it (deploy mode: ready), Apstra adds basic configuration, including device hostnames, interface descriptions and port speed / breakout config. |

(Continued)

| Config | Description |
|--------------------|--|
| Service Config | When you deploy a device (deploy mode: deploy), Apstra adds configuration that's required in the Apstra environment. <i>Service Config</i> consists of Discovery 1 config, Ready (Discovery 2) config and this additional config. |
| Rendered Config | Complete Apstra-rendered configuration for the device, per the Apstra Reference Design. |
| Incremental Config | The configuration that will be applied when you commit changes that you've made. |
| Golden Config | When you commit config changes, Apstra collects a new running configuration called <i>Golden Config</i> . Golden config serves as Intent: Apstra continuously compares the running config against the Golden config. When a deployment fails, Apstra unsets the Golden Config. |

Configuration Stages: Overview

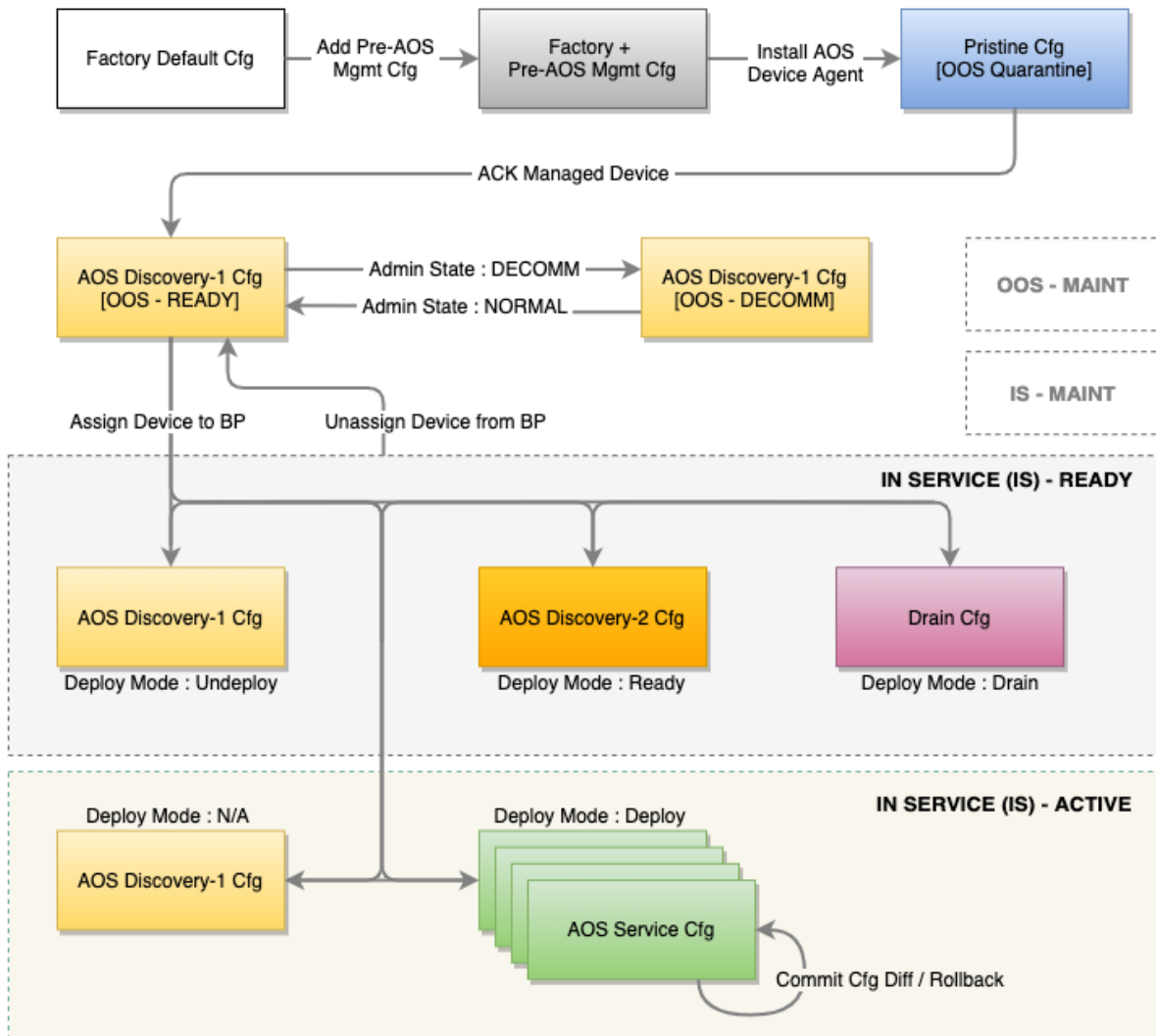
The following table describes the various config events and their resulting device config, Apstra-managed device state, and blueprint deployment mode:

| 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 | Pristine Config: Factory + Pre-Apstra + Agent Install config | OOS-QUARANTINED | Not Assigned |
| Acknowledge device | Discovery 1: Pristine, plus Interfaces Enabled | OOS-READY | Not Assigned |
| Assign device to blueprint (no deploy) | Ready (Discovery 2): Discovery 1, plus various basic config | IS-READY | Ready |

(Continued)

| Event | Resulting Device Configuration | Resulting Apstra Managed Device State | Apstra Blueprint Deployment Mode |
|--------------------------------------|--|---------------------------------------|----------------------------------|
| Deploy device | Service Config: Ready (Discovery 2) config 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: When you install an agent on a device, any configuration that was already there becomes part of the Pristine Config, which means it's included in the device's entire configuration lifecycle. Any corrections that you make will be service-impacting.

Configuration Stages: Detail

IN THIS SECTION

● [New Device \(Factory Default\) | 65](#)

- Add Pre-Apstra Config (User-required) | 65
- Install Agent (Pristine) | 65
- Acknowledge Device (Discovery 1 / Ready) | 66
- Assign Device (Ready / Ready) | 66
- Deploy Device (Rendered / Active) | 67
- Stage Device Update (Incremental / Active) | 68
- Commit Device Again (Rendered-Updated / Active) | 68

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 is required for the entire configuration lifecycle. This includes configuration for agent installation and 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.

You can bootstrap this **User-required** config with "[Apstra ZTP](#)" on page 230, or add it with scripts (or other methods).



CAUTION: Only add configuration that's required for connectivity, for installing the device agent, or that's known to be required **throughout the device lifecycle** (for example Banners or NTP / SNMP / syslog server IP addresses). You can add required configuration that's not rendered by Apstra with "[configlets](#)" on page 45.

Install Agent (Pristine)

When you install an onbox agent on a device (or an offbox agent on the server) the device connects and registers with Apstra in the **Quarantined** state. Apstra applies partial configuration to the pre-Apstra configuration. This configuration is called **Pristine configuration**. Pristine configuration is the basis for all subsequent device configuration.

Acknowledge Device (Discovery 1 / Ready)

When you acknowledge a device, you're putting it in the **Ready** state. This acknowledgment signals your intent to have Apstra manage the device. To the pristine config, Apstra adds minimal base configuration that's essential to Apstra agent operation. This configuration is called **Discovery 1 config**. Discovery 1 applies a *complete* configuration (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.



NOTE: Devices that have been acknowledged cannot simply be deleted. Since the device would still have an active agent installed, the devices would re-appear within seconds. To remove a device from Apstra management, see ["Remove \(Decommission\) Device from Managed Devices" on page 80](#) for the complete workflow.

Assign Device (Ready / Ready)

When you assign a device to a blueprint and set its Deploy Mode to **Ready**, you're putting it in the **Ready (Discovery 2)** state. The device has been staged, but not yet committed (deployed) to the active blueprint. Ready config applies a *complete* configuration (Full config push) to ensure config integrity. Ready configuration brings up network interfaces and configures interface descriptions and validates telemetry, such as LLDP, to ensure it's 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 spine devices to 9050 bytes.

Deploy Device (Rendered / Active)



CAUTION: The first time you assign a device and deploy it (set deploy mode to Deploy and commit the blueprint), you're triggering a full configuration push on the device. This action overwrites the complete running configuration with the pristine configuration, then adds the full rendered Apstra configuration. Apstra discards any configuration that's not part of the Apstra-rendered configuration.

When you commit a device, 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, and so on, 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.
- 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 (for example show running-config) and stores it as the **Golden Configuration**.



CAUTION: If you add configuration at this point, you'll raise configuration deviation anomalies. The deviation is the difference between the current configuration and the stored Golden configuration. Before you can proceed with deployment tasks, you must correct any anomalies.

To see the rendered config file after committing the blueprint, select the device in the **Active** blueprint and click **Config** (right-side).

You can modify a running configuration multiple ways. To modify a config that's not part of the reference design, use ["configlets" on page 45](#).

Stage Device Update (Incremental / Active)

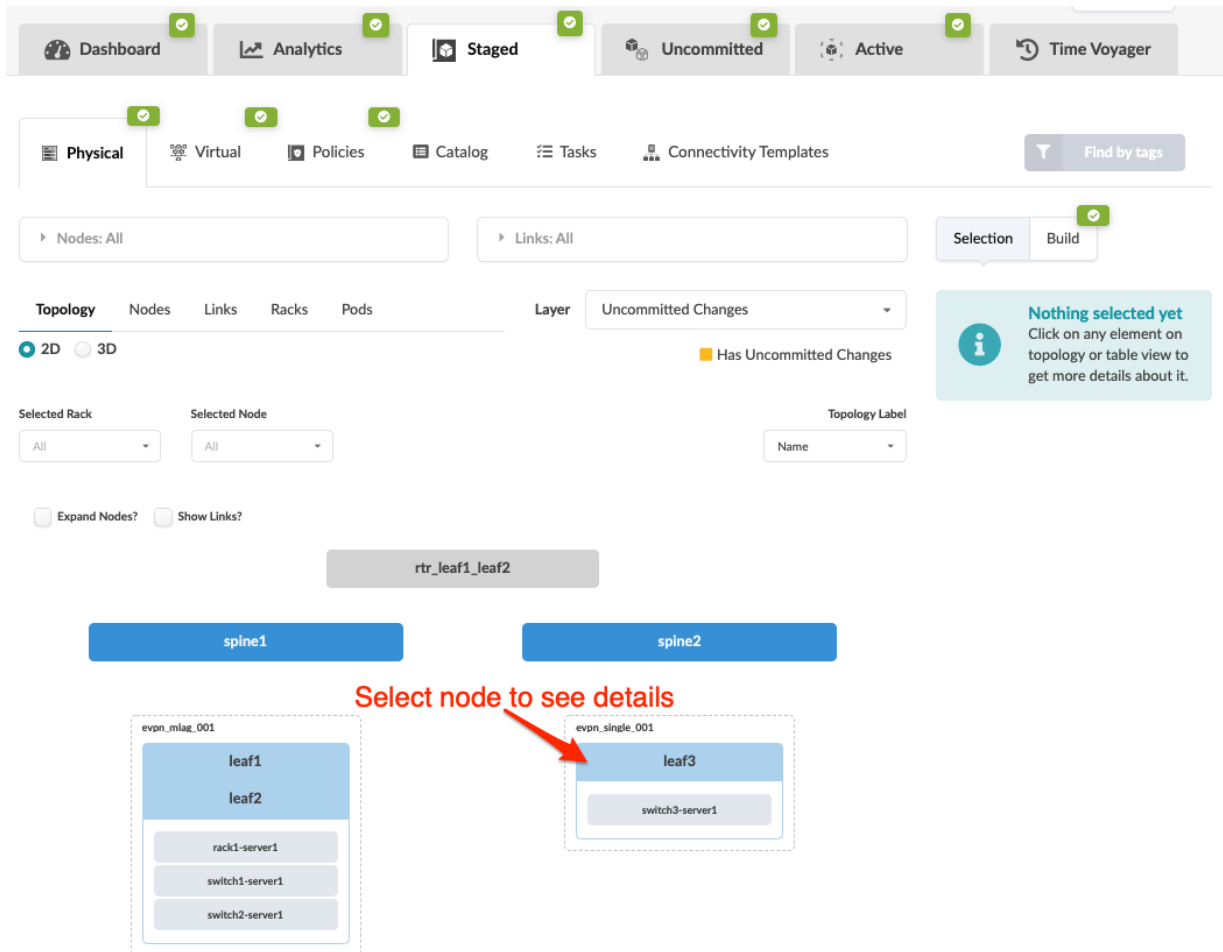
When you stage changes to a running blueprint, you're creating an **Incremental** configuration.

Commit Device Again (Rendered-Updated / Active)

When you commit a change to a blueprint that affects the device's configuration, a partial config updates the rendered config.

View Device Config from Blueprint

From the blueprint, navigate to **Staged > Physical** to go to the **Topology** view of the physical blueprint.



Click a node in the topology, then from the **Device** tab in the panel on the right, you can click links for rendered, incremental, pristine, or (new in Apstra version 4.1.0) device context in the **Config** section.

The screenshot displays the Apstra web interface. At the top, there are tabs for Dashboard, Analytics, Staged, Uncommitted, Active, and Time Voyager. Below these are tabs 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 a 'Nodes: All' filter. The 'Topology' tab is selected, and the '2D' view is active. The 'Selected Rack' is 'evpn_mlag_001' and the 'Selected Node' is 'leaf1 (Leaf)'. The 'Neighbors' and 'Links' tabs are visible. The topology diagram shows 'leaf1' connected to various other nodes: rack1-server1, leaf2, spine1, spine2, switch1-server1, and rtr_leaf1_leaf2. The 'Device' tab on the right panel is selected, showing the 'leaf1' device. The 'Deploy Mode' is 'deploy'. The 'S/N' is '525400D8ACBF'. The 'Device Info' section shows 'Management IP' as '10.29.36.13', 'OS' as 'NXOS 9.3(8)', and 'Operation Mode' as 'FULL CONTROL'. The 'Hostname' is 'leaf1'. The 'Config' section has a red arrow pointing to the 'Device Context' link.

The device model is a nested dictionary of variables that you can leverage when creating configlets.

Device Context

- bgpService
- bgp_sessions
- configlets
- device_capabilities
- dhcp_servers
- interface
- ip
- loopbacks
- portSetting
- routing
- security_zones
- vlan
- vn_policy
- vxlan
- aaa_servers: {}
- access_lists: {}
- aos_version: "4.1.0"
- configured_role: "leaf"
- deploy_mode: "deploy"
- dot1x_config: {}
- ecmp_limit: 64
- hcl: "Cisco_NXOSv"
- hostname: "leaf3"
- ipv6_support: true
- lo0_ipv4_address: "10.0.0.4/32"
- lo0_ipv6_address: "fc01:a05:fab::4/128"
- logical_vtep_ipv4_address: "10.0.0.14/32"
- mac_msb: 2
- model: "Cisco_(NX-OSv|C9300v)"
- name: "leaf3"
- os: "NXOS"
- os_version: "9.3(8)"
- ospf_services: {}
- port_count: 9
- reference_architecture: "two_stage_l3clos"
- role: "leaf"
- slots: [
 - 0

The query tab provides dynamic search capabilities to quickly search through keys or values and identify the variables of interest. Syntax is case-sensitive. For example, a search of the keyword **bgp** provides information on the BGP configuration of the switch as well as the BGP sessions (protocol sessions), while a search on the key word **BGP** provides the list of BGP route maps such as "BGP-AOS-Policy". The use of these variables as built-in property-sets inside a configlet must also respect the case-sensitive attribute of the device model.



CAUTION: Device models are an internal data model used in the Apstra environment. They are subject to change without notice or documentation of schema changes.

Configuration Deviations

After each **successful** config deploy the running config is collected and stored internally as the **Golden** configuration. Intent is the cornerstone of the Apstra product. 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:

- Each *successful* configuration deployment results in an updated Golden Config.
- If configuration deployment fails, Golden Config is not set. This means both a config deviation and deployment failure anomaly are raised.
- Running configuration telemetry is continuously collected and matched against the Golden Config. Any difference result 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 613](#) 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

IN THIS SECTION

- Not Set | 72
- Deploy | 72
- Ready | 72
- Drain | 72
- Undeploy | 74

Managed devices in blueprints can be in one of several ["modes" on page 304](#):

Not Set

Initial device state. The device is not active in the fabric.

Deploy

The device is active in the fabric.

Ready

When a device is assigned to a blueprint, its deploy mode changes to **Ready**; Apstra renders Ready (Discovery 2) configuration (hostnames, interface descriptions, port speed / breakout configuration). The device isn't active in the fabric. Changing from **Deploy** to **Ready** removes Apstra-rendered configuration.

Drain

["Draining a device" on page 82](#) for physical maintenance enables it to be taken out of service without impacting existing TCP flows. Depending on the device being drained, Apstra uses one of two methods:

For L2 Servers

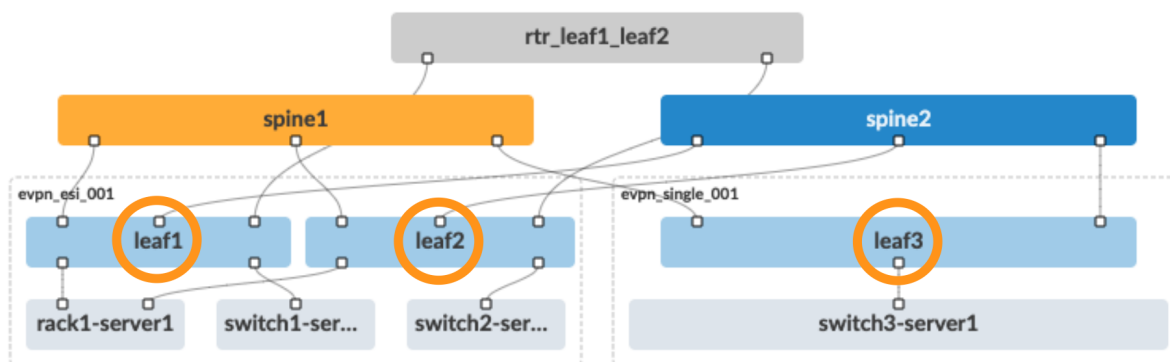
- MLAG peer-links port channels and bond interfaces on any NOS are not changed.
- For Arista EOS, Cisco NX-OS, all interfaces towards L2 servers in the blueprint are shutdown.

For Network L3 Switches

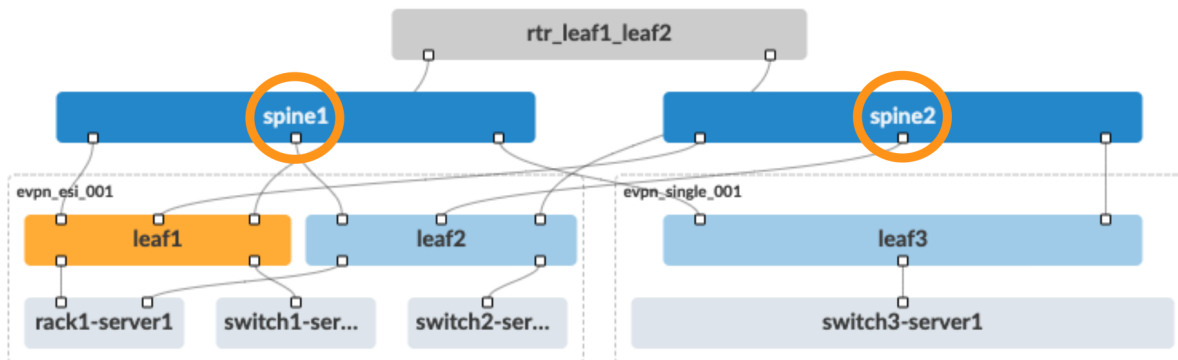
The device uses Inbound/Outbound route-maps 'deny' statements to block any advertisements to 0.0.0.0/0 le 32. This allows existing L3 TCP flows to continue without interruption. After a second or two, the TCP sessions should be re-established by the src/dst devices, or they should negotiate a new TCP 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 **Drain** mode. The device is effectively drained of traffic and can be removed from the fabric (by changing Deploy mode to **Undeploy**).

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.

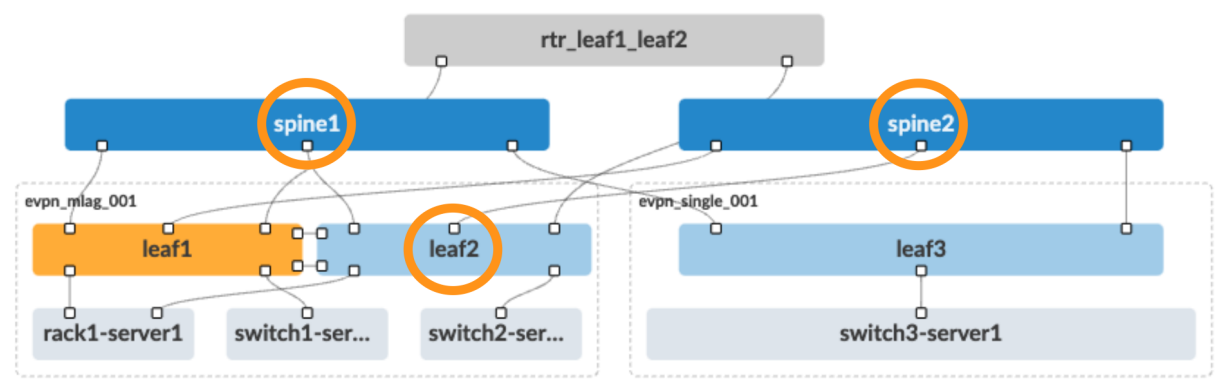
When you change the deploy mode to **Drain** on a device, neighboring device configuration may also be affected, not just the device you're draining. For example, when you drain a spine device, configuration on all connected leaf devices change. Neighboring leaf devices use Inbound/Outbound route filters (route-maps) 'reject (deny)' statements to block any advertisements to 0.0.0.0/0 le 32, for both EVPN (overlay) and FABRIC (underlay).



Similarly, when you drain a leaf device, the configuration on connected spine devices changes. Neighboring spine devices use Inbound/Outbound route filters (route-maps) 'reject (deny)' statements to block any advertisements to 0.0.0.0/0 le 32, for both EVPN (overlay) and FABRIC (underlay).



In the case of an MLAG-based topology, in addition to the configuration on connected spine devices changing, the configuration on the paired leaf device also changes.



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.

Managed Devices

IN THIS SECTION

- [Managed Devices Overview | 75](#)
- [Add Device to Managed Devices | 79](#)
- [Remove \(Decommission\) Device from Managed Devices | 80](#)
- [Drain Device Traffic | 82](#)
- [Edit Device | 84](#)
- [Delete Device | 85](#)
- [Upgrade Device NOS | 86](#)
- [Device AAA | 93](#)
- [Set Device Admin State | 95](#)
- [Device Profiles | 95](#)

Managed Devices Overview

IN THIS SECTION

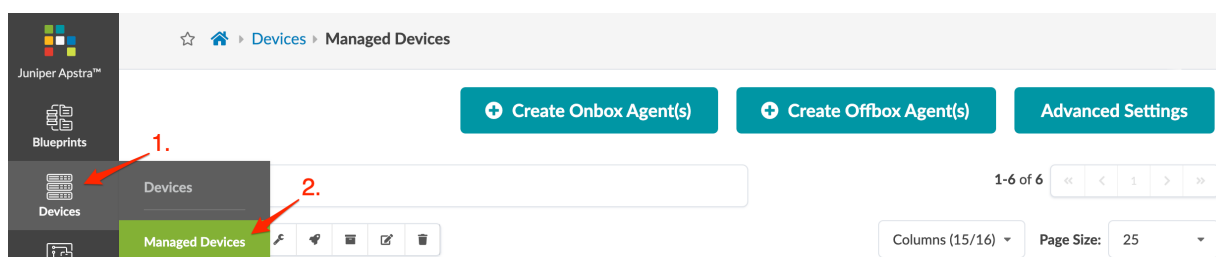
- [Device | 76](#)
- [Agent | 77](#)
- [Pristine Config | 78](#)
- [Telemetry | 79](#)

Apstra software uses device system agents to manage devices. These agents manage configuration, device-to-device communication and telemetry collection. You can use "[Apstra Zero Touch Provisioning on page 230](#) (ZTP) to install agents and bring devices under Apstra management or you can use the device installer.



CAUTION: A good understanding of the "[Apstra device configuration lifecycle on page 61](#)" is essential. Before working with devices in the Apstra environment, we strongly recommend that you fully understand how devices are configured from the moment they are on-boarded to the moment they are decommissioned.

From the left navigation menu in the Apstra GUI, navigate to **Devices > Managed Devices** to go to managed devices.



Devices with installed agents appear in the table. The **Managed Devices** page is the hub for many device-related tasks, which are described in later sections.

The screenshot shows the 'Managed Devices' page interface. At the top, there are buttons for 'Create Onbox Agent(s)', 'Create Offbox Agent(s)', and 'Advanced Settings'. Below these is a search bar and a table of devices. Red arrows point to various action buttons above the table: 'Acknowledge', 'Set to MAINT', 'Set to NORMAL', 'Set to DECOMM', 'Update User Config', 'Delete', 'Assign Profile', 'OS Upgrade', 'Install', 'Uninstall', and 'Check'. The table has two main sections: 'Device Information' and 'System Information'. The 'Device Information' section includes columns for Management IP, Device Key, Device Profile, Hostname, OS, State, Comms, Acknowledged?, and Blueprint. The 'System Information' section includes columns for Type, Agent Profile, Apstra Version, Last Job Type, Job State, and Actions. A single device is listed in the table with a 'SUCCESS' status.

| Device Information | | | | | | | | | System Information | | | | | |
|--------------------|--------------|----------------|-----------------|-------------|-----------|-------|---------------|-------------------------------|--------------------|---------------|------------------|---------------|-----------|---------|
| Management IP | Device Key | Device Profile | Hostname | OS | State | Comms | Acknowledged? | Blueprint | Type | Agent Profile | Apstra Version | Last Job Type | Job State | Actions |
| 10.28.93.13 | 505400CF501A | Arista vEOS | rack2-001-leaf1 | EOS 4.24.5M | IS-ACTIVE | 🟢 | ✅ | rack-based-blueprint-845e75ac | ONBOX | UNASSIGNED | AOS_4.1.0_OB.100 | INSTALL | SUCCESS | ⋮ |

Click a management IP to go to details for its device, agent, pristine config and telemetry as shown below.

Device

The device detail view shows the user config, the device status and other facts about the device. From the device detail page you can edit and delete the device. You can also edit or delete a device from the

table view or any of the other detail views (Agent, Pristine Config, Telemetry).

☆

🏠

Devices

Managed Devices

10.28.52.15

Device

Device

Agent

Pristine Config

Telemetry

Expanded View

Compact View

✎

🗑

Edit System

Delete System

User Config

| | |
|----------------|-------------|
| Device Profile | Cisco NXOSv |
| Admin State | normal |
| Location | leaf2 |

Status

| | |
|----------------|--------------|
| State | IS-ACTIVE |
| Acknowledged? | ✓ |
| Operation Mode | FULL CONTROL |
| Error Message | N/A |

Agent

Apstra device system agents handle configuration management, device-to-server communication, and telemetry collection. If you're not using ["Apstra ZTP" on page 230](#) to bootstrap your devices (or if you have a one-off installation) you can use this device installer to automatically install and verify devices. Depending on the device NOS, you can install device agents onbox (agent is installed on the device) or offbox (agent is installed on the Apstra server and communicates with devices via API). For support information, see the **Device Management** section of the ["feature matrix" on page 918](#).

The device agent view shows the agent config, agent status, last job status, jobs history and telemetry status. From the agent detail page you can perform various tasks similar to tasks in the table view. For example, you can restore a device's pristine configuration by clicking the **Revert to Pristine Config** button (as of Apstra version 4.0.1) as long as the device is not assigned to a blueprint.

☆ 🏠 > Devices > Managed Devices > 10.28.52.15 > Agent

Device Agent **Pristine Config** Telemetry

Check Install Uninstall OS Upgrade Revert Collect pristine Show Log Cancel job Edit Delete

Expanded View Compact View

Config

Status

Last Job Status

Jobs History

Telemetry Status

| | |
|-----------------------------------|--------------|
| Device Address | 10.28.52.15 |
| Operation Mode | FULL CONTROL |
| Profile | Not Selected |
| Install Requirements [®] | yes |
| Packages | Not provided |

Pristine Config

The pristine config view shows the pre-Apstra configuration on the device. You can edit the pristine config manually or update it directly from the device. You can edit and delete the device. You can also edit or delete the device from the table view or any of the other detail views (Device, Agent, Telemetry).

☆ 🏠 > Devices > Managed Devices > 10.28.52.15 > Pristine Config

Device Agent **Pristine Config** Telemetry

✎ 🗑

i This is the pre-Apstra config on the device Update From Device

Edit pristine config

checkpoint

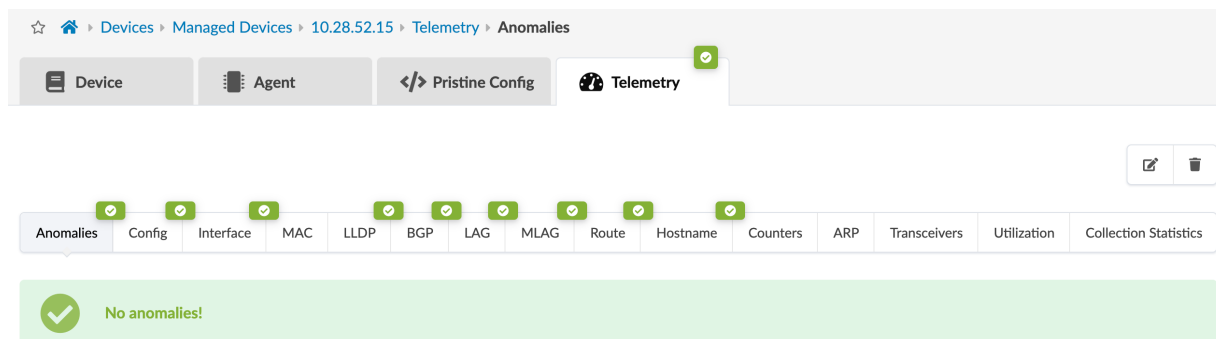
```

1
2 !Command: Checkpoint cmd vdc 1
3
4 version 9.3(8) Bios:version
5 class-map type network-qos c-nql

```

Telemetry

The telemetry view shows telemetry for the device. For more information, see ["Telemetry Services" on page 215](#).



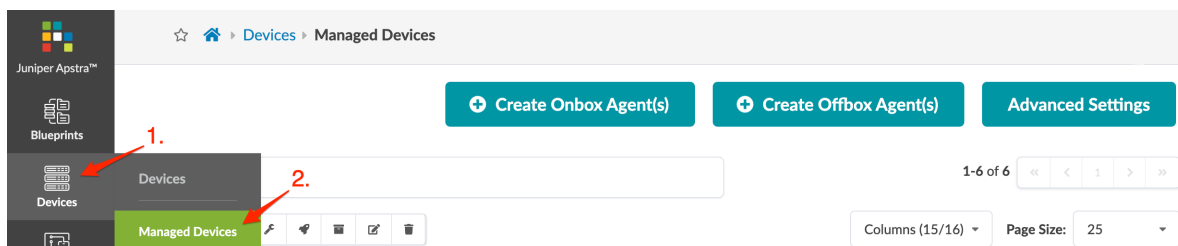
Add Device to Managed Devices

Before working with devices, it's important to have a good understanding of the ["device configuration lifecycle" on page 61](#).

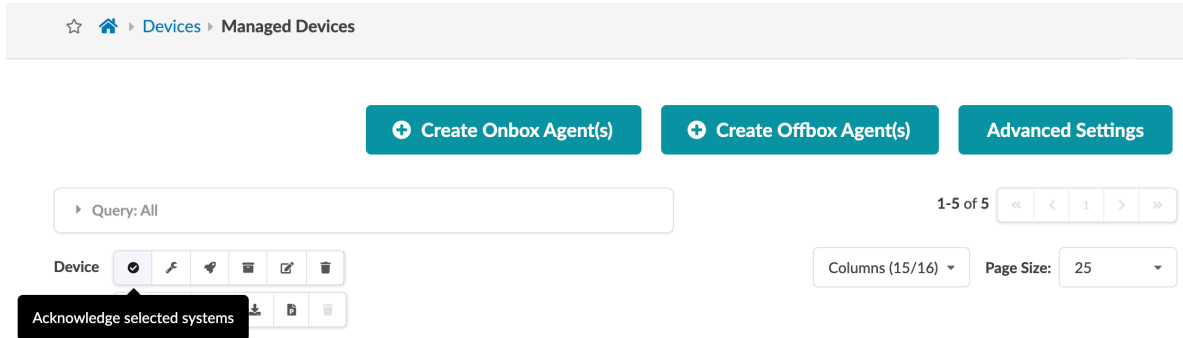


NOTE: Each device is expected to have a unique management IP address. If you're replacing a device (decommissioning for an RMA for example) and you want to use the same management IP address on the replacement device, you must ["remove \(decommission\) the device from Managed Devices" on page 80](#) before adding the new device.

1. If you're using Juniper offbox agents, ["increase the application memory usage" on page 777](#).
2. Create and install your ["onbox" on page 151](#) device agent(s) or ["offbox" on page 154](#) device agent(s) 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 211](#) (Device > Agent Profiles), which can streamline the task of creating many agents.
3. If you're deploying modular devices, you may need to ["change the default device profile" on page 84](#) that's assigned to your device.
4. Navigate to **Devices > Managed Devices** to see that the device state is **Out of Service Quarantine**. Configuration at this point is called **Pristine Config**.



5. In the left column of the table, select the check box(es) for the device(es) to manage in the Apstra environment.
6. Above where you just clicked, click the **Acknowledge selected systems** button (check mark) in the **Device** action bar.



7. Click **Confirm** to acknowledge the device(s) and return to the table view. The device state changes to **Out of Service Ready**. Configuration at this point is called **Discovery 1 Config** and you can now manage the device(s) from the Apstra environment.

Next Steps:

If you'll be using a Datacenter blueprint, before creating the blueprint make sure you have all your design elements ready, starting with ["logical devices" on page 7](#).

If you'll be using a Freeform blueprint, you can ["create the blueprint" on page 629](#) right away.

You'll assign your devices to a blueprint during the build phase. For details, see ["Assign Device \(Datacenter\)" on page 295](#) or ["Assign System \(Freeform\)" on page 649](#), as applicable.

Remove (Decommission) Device from Managed Devices

For successful device removal, it's important to follow these steps in the order specified.

1. If the device is assigned to a blueprint, unassign it from your ["datacenter blueprint" on page 300](#) or ["freeform blueprint" on page 651](#), as applicable.
2. From the left navigation menu, navigate to **Devices > Managed Devices** and check the box for the device to remove from Apstra management.
3. In the **Device** Actions panel that appears above the table, click the **Set admin state to DECOMM for selected systems** button, then click **Confirm** to set the admin state and return to the table. (If the device is assigned to a blueprint, you can't decommission the device.)

The screenshot shows the 'Managed Devices' page in Apstra. At the top, there are buttons for 'Create Onbox Agent(s)' and 'Create Offbox Agent(s)'. Below these is a search bar with 'Query: All'. A table header shows 'Device' and 'Agent' columns. The 'Agent' column has a dropdown menu open, showing options like 'Uninstall'. A tooltip points to the 'Uninstall' button, stating: 'Set admin state to DECOMM for selected systems'. The 'Filter selected by' section shows 'all' selected.

4. Check the box for the device again, then in the **Agent** Actions panel that appears above the table, click the **Uninstall** button, click **Uninstall selected elements**, then click **Close**.

This screenshot is similar to the previous one, but the 'Uninstall' button in the 'Agent' Actions panel is now highlighted with a black box. The tooltip is no longer visible.



NOTE: If the device is unreachable, the job will fail. You can force delete the agent (in the next step), as of Apstra version 4.1.2.

Prior to Apstra version 4.1.2, if the device is unreachable you could ["change the agent operation mode" on page 160](#) to **telemetry only**, then uninstall its agent.

5. Check the box for the device again, then in the **Agent** Actions panel that appears above the table, click the **Delete** button, click **Delete selected elements**, then click **Close**.

This screenshot shows the 'Managed Devices' page with the 'Delete' button in the 'Agent' Actions panel highlighted by a black box. The 'Filter selected by' section shows 'all' selected.

If you weren't able to uninstall the agent in the previous step because the device is unreachable, a dialog opens that gives you the option to force delete the agent. With the **Force Delete** box checked,

click **Delete** to force delete the agent and return to the table view.

Delete System Agent

Apstra cannot communicate with this device. It will not be possible to uninstall any agents or revert the device to its Pristine config before deleting. Only select this box if you understand the consequences and cannot reestablish communication with the device.

☒ Force delete?

Delete

6. Check the box for the device again, then in the **Device Actions** panel click the **Delete system(s)** button, then in the dialog that opens click **Confirm** to remove the device(s) from Apstra management and return to the table view. (If the device is not in STOCKED or DECOMM stage, you can't delete the device.) Device(s) are disconnected from the Apstra server and removed from the Apstra database.

☆ [Home](#) > [Devices](#) > Managed Devices

+ Create Onbox Agent(s)
+ Create Offbox Agent(s)
Advanced Settings

Query: All
1-1 of 1

Device

🔍
🔧
🔗
🔒
🔑
🔓
🔗
🔒
🔑
🔓

Agent

☒
☐
☐
☐
☐
☐
☐
☐
☐
☐

Columns (15/17)
Page Size: 25

Filter selected by

all
selected only

Delete system(s)

selected only

| Device Information | | | | | | | | | | Agent Information | | | | | |
|--------------------|------------|----------------|---------------------|-----------------|-----------------|-------|---------------|--------------|--------|-------------------|------------------|---------------|-----------|---------|--|
| Management IP | Device Key | Device Profile | Hostname | OS | State | Comms | Acknowledged? | Blueprint | Type | Agent Profile | Apstra Version | Last Job Type | Job State | Actions | |
| 1 selected | | | Juniper_QFX5100-48S | Junos 22.2R2.10 | OOS-QUARANTINED | 🟢 | 🟢 | Not assigned | OFFBOX | UNASSIGNED | AOS_4.1.2_OB.269 | INSTALL | SUCCESS | ⋮ | |

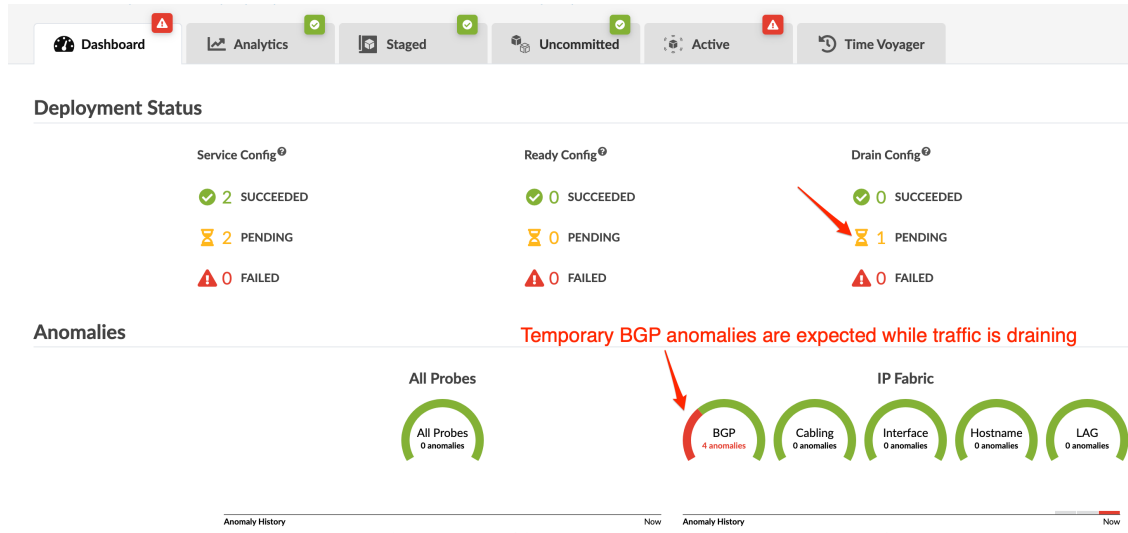
If you're replacing the device you just removed, follow the steps to ["add"](#) on page 79 the replacement device to Managed Devices.

Drain Device Traffic

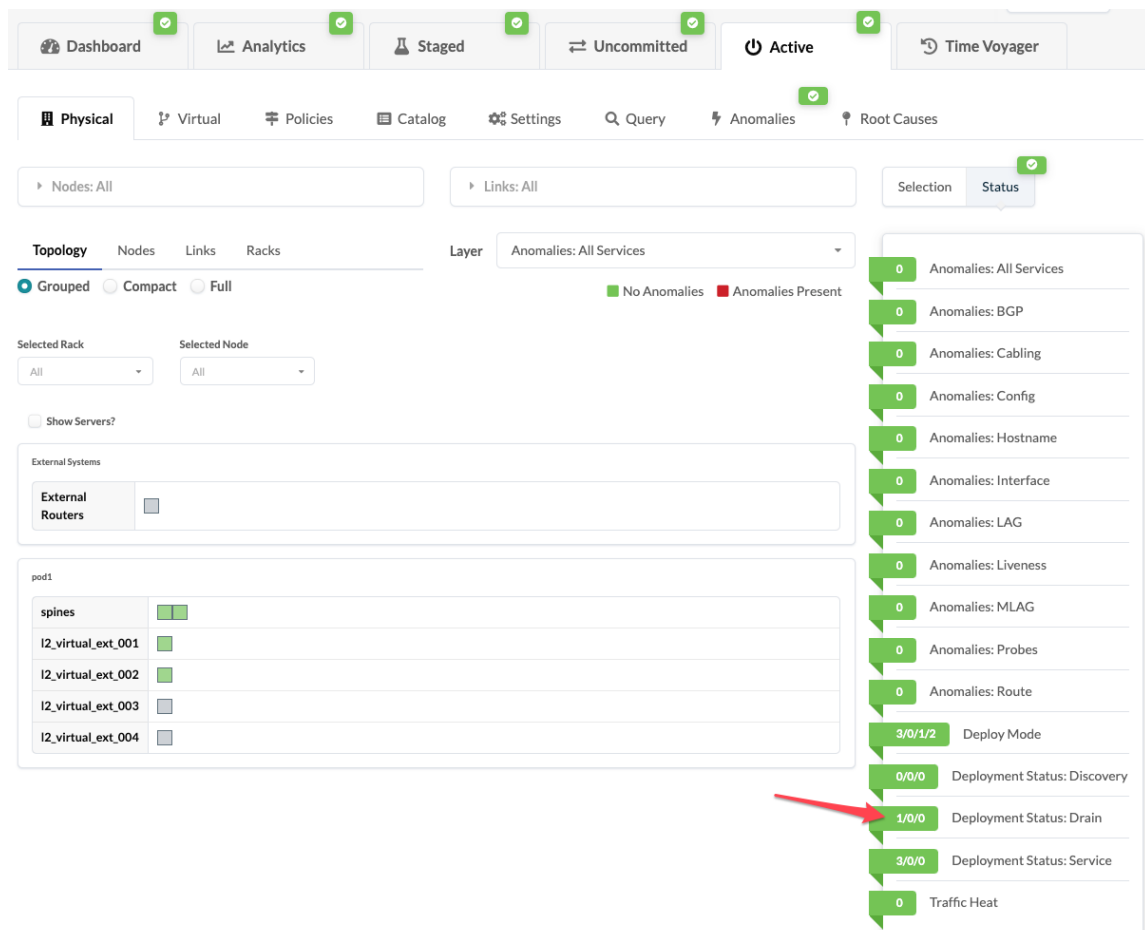
To take a device out-of-service for maintenance (or decommissioning), set its deploy mode to **Drain**. Draining a device may impact neighboring devices. For details, see ["Device Configuration Lifecycle"](#) on page 72.

1. From the blueprint, navigate to **Staged > Physical > Build > Devices** and change the ["deploy mode"](#) on page 304 on the device to **Drain**. The drain command applies changes to BGP policies and disables/shuts down all L2 ports.
2. Click **Uncommitted** to ["review staged changes"](#) on page 694. The **Logical Diff** tab shows the changes that will be made to the device, and possibly to its neighbors.
3. Commit staged changes to activate them. While draining is in progress (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.

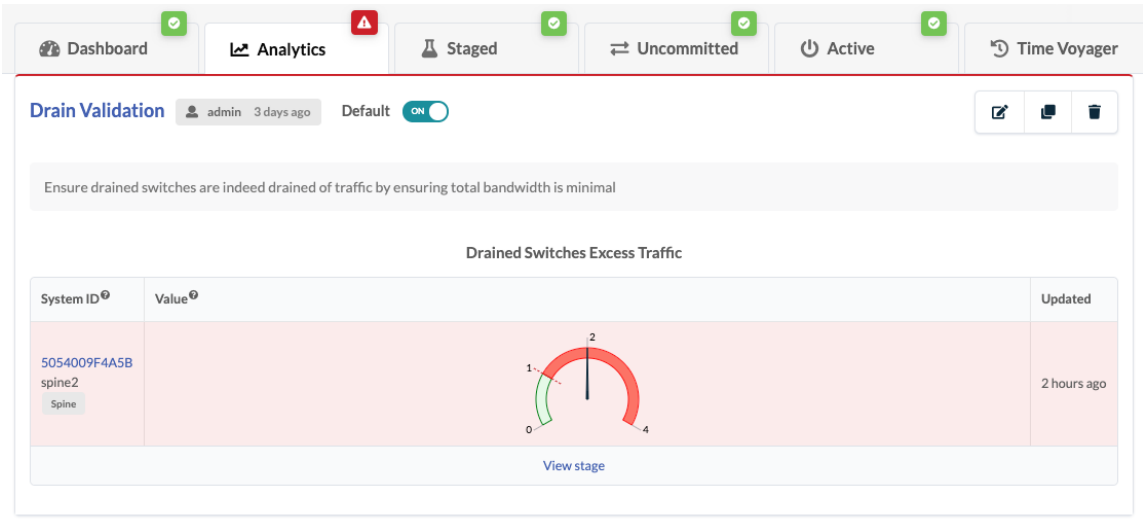
- You can monitor drain status from the **Deployment Status** section of the blueprint dashboard (Drain Config).



- You can monitor drain status from **Active > Physical** in the **Status** panel (Deployment Status: Drain).



- If you ["instantiate" on page 702](#) 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 692](#) the change to bring the device back into active service.

Edit Device

NOTE: You can also edit a device from any of the detail views (Device, Agent, Pristine Config, Telemetry.)

1. From the left navigation menu, navigate to **Devices > Managed Devices** and select the check box(es) for the device(s) to edit.
2. Click the **Update user config** button in the **Device** action bar (above the table), then change the device profile, admin state, and/or location, as applicable.

☆ 🏠 > [Devices](#) > Managed Devices

[+ Create Onbox Agent\(s\)](#) [+ Create Offbox Agent\(s\)](#) [Advanced Settings](#)

Query: All 1-5 of 5

Device [🔍](#) [🔧](#) [🔗](#) [📄](#) [📝](#) [🗑️](#)

Agent [✓](#) [📄](#) [📝](#) [🗑️](#) [Update user config](#)

Columns (15/16) Page Size: 25

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

2. In the **Device** Actions panel (above the table) click the **Delete system(s)** button, then in the dialog that opens click **Confirm** to remove the device(s) from Apstra management and return to the table view. (If the device is not in STOCKED or DECOMM stage, you can't delete the device.) Device(s) are disconnected from the Apstra server and removed from the Apstra database.



NOTE: You can also delete a single device from the **Device** detail view by clicking on the management IP address in the table.

Upgrade Device NOS

SUMMARY

Upgrade the network operating systems (NOS) of your Apstra-managed network devices from within the Apstra environment.

IN THIS SECTION

- [NOS Upgrade Overview | 86](#)
- [Update User-defined Device Profiles | 87](#)
- [Register / Upload OS Image | 89](#)
- [Upgrade OS Image | 92](#)

We highly recommend that you become familiar with this procedure before upgrading a device NOS.

NOS Upgrade Overview

You can upgrade a device NOS within the Apstra environment with a few simple steps. If you've defined your own device profiles, you may need to update them. Then you'll register the new OS image that you obtained from the vendor, and click a button to start the upgrade. Apstra takes care of upgrade tasks and other requirements and ensures that pristine config is updated.



NOTE: Though we don't recommend it, you *can*

upgrade a device NOS outside of the Apstra environment. This requires that you perform manual steps as follows: unassign and undeploy the device from the blueprint, commit the changes, uninstall the agent, upgrade the device NOS outside of the Apstra environment, install the agent, assign and deploy the device in the blueprint, then finally commit the changes.

For information about supported upgrade paths, see ["NOS Upgrade Paths" on page 988](#) in the References section.

Apstra software ships with built-in device profiles that support specific OS versions. When you upgrade the Apstra server, device profiles with the OS versions that are supported in the new Apstra version are also updated. 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 4.0.2 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 when you upgrade the Apstra server those device profiles aren't automatically updated with newly supported versions. You'll need to follow a few extra steps to add them 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 61](#) and that you're comfortable with managing deploy modes.
- Make sure that Apstra software is managing the device you're upgrading. Navigate to **Devices > Managed Devices** and confirm that your device is in the table 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 **Management IP** 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 can't 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.

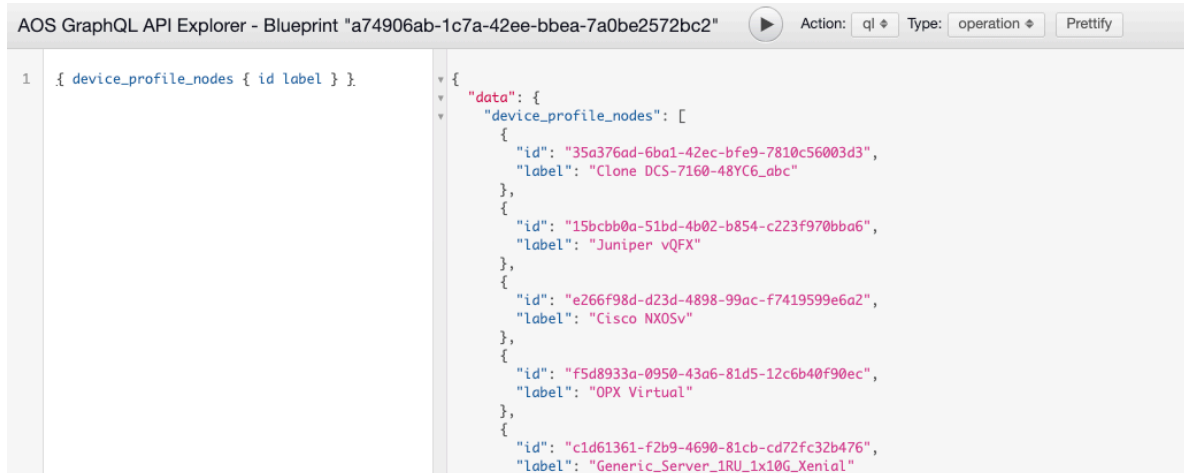
Update User-defined Device Profiles

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 register the new OS image.)

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.

- 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".



- Use `apstra-cli` to update the device profile.

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.

`apstra-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:

```

apstra-cli> 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.
- Proceed to the next section to upgrade the OS in the same manner as for devices using predefined device profiles.

Register / Upload OS Image

IN THIS SECTION

- Method One: Upload Image | 90
- Method Two: Provide Image URL | 90
- Add Checksum (Optional) | 91

1. Obtain the OS image from the device vendor.



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 is not 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 824](#).

2. From the left navigation menu, navigate to **Devices > System Agents > OS Images** and click **Register OS Image** (top-right). (Starting with Apstra release 4.1.0 you can see how much space is left for uploading new NOS images, and if the partition has under 5GB of free space a warning appears when you register.)

The screenshot shows the Juniper Apstra web interface. On the left, a navigation menu is open, highlighting 'OS Images' under the 'System Agents' section. A red arrow labeled '1.' points to the 'Devices' menu item, and another red arrow labeled '2.' points to the 'OS Images' menu item. In the top right corner, a green button labeled 'Register OS Image' with an upload icon is highlighted by a red arrow labeled '3.'. The main content area shows a breadcrumb 'Devices > OS Images', a warning message about partition space, and a table with columns: Platform, Type, Size, Description, Checksum, and Actions. The table currently contains no items.

3. Select the platform from the drop-down list (EOS, NXOS, SONIC, JUNOS) and enter a description.

4. Either upload the image directly to the Apstra server or provide 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. The image appears in the table view. (As of Apstra version 4.1.1 the size of the image is included.)
4. If the (optional) checksum is not verified, the upgrade process stops, before the device reboots.

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 supported.)

1. Select **Provide Image URL**.

Register Device OS Image

Platform *

NXOS

Description *

EOS-4.22.5M

☐ Upload Image ☒ Provide Image URL

Image URL *

http://192.168.59.254/EOS-4.22.5M.swi

Checksum

dbfd28d3597777a6ee5946b52277205fc714e11ab992574b7ef1156ffcd6e379979979f8c009f665fc212

SHA512 checksum (128 characters)

Register

2. Enter the URL that points to the image on the other server.
3. Add a checksum (optional) (described in the section below).
4. Click **Register** to register the image with the Apstra software. The image appears in the table view.
(As of Apstra version 4.1.1 the size of the image is included.)
5. If the (optional) checksum is not verified, the upgrade process stops, before the device reboots.

Add Checksum (Optional)

The platform determines the type of checksum that's used:

- Juniper Junos - MD5 (32 characters) or SHA256 (64 characters)
- Enterprise SONiC - MD5 (32 characters)
- Cisco NX-OS - SHA512 (128 characters)
- Arista EOS - SHA512 (128 characters)

If the device vendor provides a checksum file, we recommend that you download the file 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
$
```

Upgrade OS Image

Make sure that your devices are in the appropriate states for upgrading as described in the overview above, and that if you're device profiles are user-defined that you've updated them accordingly.

1. From the left navigation menu, navigate to **Devices > Managed Devices**, and select the check box(es) for the device(s) to upgrade. (If you have many devices, use the query function to filter selections.) All selected devices must be of the same type, and they must be upgraded to the same image and version. To search for specific devices (such as for all EOS devices) enter a query.
2. Click the **Upgrade OS Image** button (above table in **Agent** section). The dialog lists the available OS images that match the selected devices.
3. Select the appropriate image and click **Upgrade OS Image**. You can monitor the upgrade status from the **Active Jobs** section at the bottom of the page.
4. 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, incorrect remote URL, or as of Apstra version 4.1.1, when device NOS version is not changed post upgrade), 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. You can then resolve the issue and restart the upgrade process.

5. 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.
6. If the checksum is correct and no other failures occur, the job state changes to **SUCCESS** and the device reboots.
7. When the device has rebooted with the new image and has reestablished its agent connection with the controller, the upgrade is complete. The **Managed Devices** page displays the new OS version.

Device AAA

IN THIS SECTION

- Overview | 93
- Juniper Junos | 94
- Cisco NX-OS | 94
- Arista EOS | 94

Overview

RADIUS and TACACS+ device AAA (authentication, authorization and accounting) frameworks are supported on Juniper, Cisco and Arista devices. 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: When using AAA framework we recommend adding a local Apstra user to devices. 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 as described below:

Configlets (Recommended)

You add configuration to a configlet, then you import it into a blueprint. Local credentials must be available from the Apstra environment so the device can be added and the configlet can be applied. For details, see ["Configlets" on page 45](#).



CAUTION: Before you upgrade the Apstra server, device agent, or NOS, you **must** delete device AAA/TACACS configlets from blueprints. After the upgrade is complete, you can re-apply them.

User-required

Instead of using configlets, you can add configuration before acknowledging a device, so it becomes part of the Pristine Config. For more information, see ["Device Configuration Lifecycle" on page 61](#).

Juniper Junos



CAUTION: Credentials for the Junos offbox 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 offbox 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.

Set Device Admin State

1. From the left navigation menu, navigate to **Devices > Managed Devices** and check the box(es) for the device(s) to update.
2. In the **Device** action panel that appears above the table, click the button for the state to change the selection(s) to.
 - **Set admin state to NORMAL for selected systems** - If you're ["upgrading a device network operating system" on page 86](#), make sure the admin state is set to **NORMAL** before beginning the process.
 - **Set admin state to DECOMM for selected systems** - If you are decommissioning a device, setting the admin state to **DECOMM** is part of a larger process. See ["Remove Device from Managed Devices" on page 80](#) for the workflow and more details.
 - **Set admin state to MAINT for selected items** - this state is no longer used.
3. Click **Confirm** to set the admin state and return to the table view.

Device Profiles

IN THIS SECTION

- [Device Profiles | 95](#)
- [Create Device Profile | 103](#)
- [Edit / Delete Device Profile | 104](#)
- [Juniper Device Profiles | 104](#)
- [SONiC Device Profile | 106](#)

Device Profiles

IN THIS SECTION

- [Device Profile Overview | 96](#)

Device Profile Overview

IN THIS SECTION

●

Summary | 96

●

Selector | 96

●

Capabilities | 97

●

Supported Features (Cisco only) | 98

●

Ports | 100

●

View Device Profiles | 103

Device profiles define capabilities of supported hardware devices. Some feature capabilities have different behaviors across NOS versions and thus, capabilities are 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 979](#).

Device profiles are associated with ["logical devices" on page 7](#) (abstractions of physical devices) to create ["interface maps" on page 13](#).

The following sections describe device profile parameters.

Summary

Table 9: 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 10: Device Profile Selector

| Selector Section | Description |
|------------------|---|
| Manufacturer | Selected from drop-down list |
| 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 OS versions. Selected from drop-down list or entered as regex. If the OS version of a device doesn't match the OS version of the device profile, then the device will be in the ERROR state (Devices > Managed Devices). |

Capabilities

You can leverage the hardware and software capabilities defined in this section 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 11: 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" |

Table 11: Device Profile Capabilities (*Continued*)

| Capabilities Section | Description |
|----------------------------|---|
| 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. |
| 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. You can specify multiple versions.

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 12: 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". |

Table 12: Device Profile Ports *(Continued)*

| Ports Section | Description |
|--|--|
| Transformations (transformations: list) | Possible breakouts for the port. Every entry is a specific supported speed. Each transformation has a collection of interfaces. |
| 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())
    })
})

```

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 table view. You can create, clone, edit, and delete 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 | [Clone] [Edit] [Delete] |
| 2X-O | Accton | 6712-32X-O.* | no | Cumulus | ((3\[5-7\]\.d+)((4\2\.\d+))\.\d+)? | T2 | [Clone] [Edit] [Delete] |
| 2-54X_SONIC_BRCM_BUZZNIK_PLUS | Edgecore Accton | 5712-54X-O.* | no | SONIC | .*3\[34\].* | T2 | [Clone] [Edit] [Delete] |
| 5T_SONIC_BRCM_BUZZNIK_PLUS | Edgecore Accton | 5835-54T-O.* | no | SONIC | .*3\[34\].* | T3 | [Clone] [Edit] [Delete] |
| 5X_SONIC_BRCM_BUZZNIK_PLUS | Edgecore Accton | 5835-54X-O.* | no | SONIC | .*3\[34\].* | T3 | [Clone] [Edit] [Delete] |
| 636X_SONIC_BRCM_BUZZNIK_PLUS | Edgecore Accton | 7326-56X-O.* | no | SONIC | .*3\[34\].* | T3 | [Clone] [Edit] [Delete] |
| 2-32X_SONIC_BUZZNIK_PLUS | Edgecore Accton | 7712-32X-O.* | no | SONIC | .*3\[34\].* | TH | [Clone] [Edit] [Delete] |
| Accton-AS7726-32X_SONIC_BUZZNIK_PLUS | Edgecore Accton | 7726-32X-O.* | no | SONIC | .*3\[34\].* | T3 | [Clone] [Edit] [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 824](#).

1. From the left navigation menu, navigate to **Devices > Device Profiles** and click **Create Device Profile**.
2. If you've created a JSON payload, click **Import Device Profile** and select the file to import it. Otherwise, continue to the next step.
3. Enter a unique device profile name.
4. Configure the device profile to match the characteristics of the physical device. See ["Device Profile Overview" on page 95](#) for details.
5. Click **Create** to create the device profile and return to the table view.

Edit / Delete Device Profile

IN THIS SECTION

- [Edit Device Profile | 104](#)
- [Delete Device Profile | 104](#)

Edit Device Profile

If a device profile is used in an ["interface map" on page 13](#), you may not be able to change it if it would adversely affect that interface map. As of Apstra version 4.1.1, you can't change predefined profiles, since your changes would be discarded when you upgrade the Apstra server. You could clone and edit a predefined device profile instead.)



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.

1. Either from the table 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 table view.

Delete Device Profile

Predefined device profiles can't be deleted. Device profiles used in ["interface maps" on page 13](#) can't be deleted.

1. Either from the table 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 table 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 | 105](#)

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 979](#). Juniper device profile constraints are specified below.

Juniper QFX10002

The 36-port Juniper QFX10002-36Q and 72-port QFX10002-72Q are qualified devices. Both of these models have a port constraint where only certain ports can be used with QSFP28 100G transceivers.

The screenshot shows the Apstra software interface for configuring a Juniper_QFX10002-36Q device. The interface is divided into several sections:

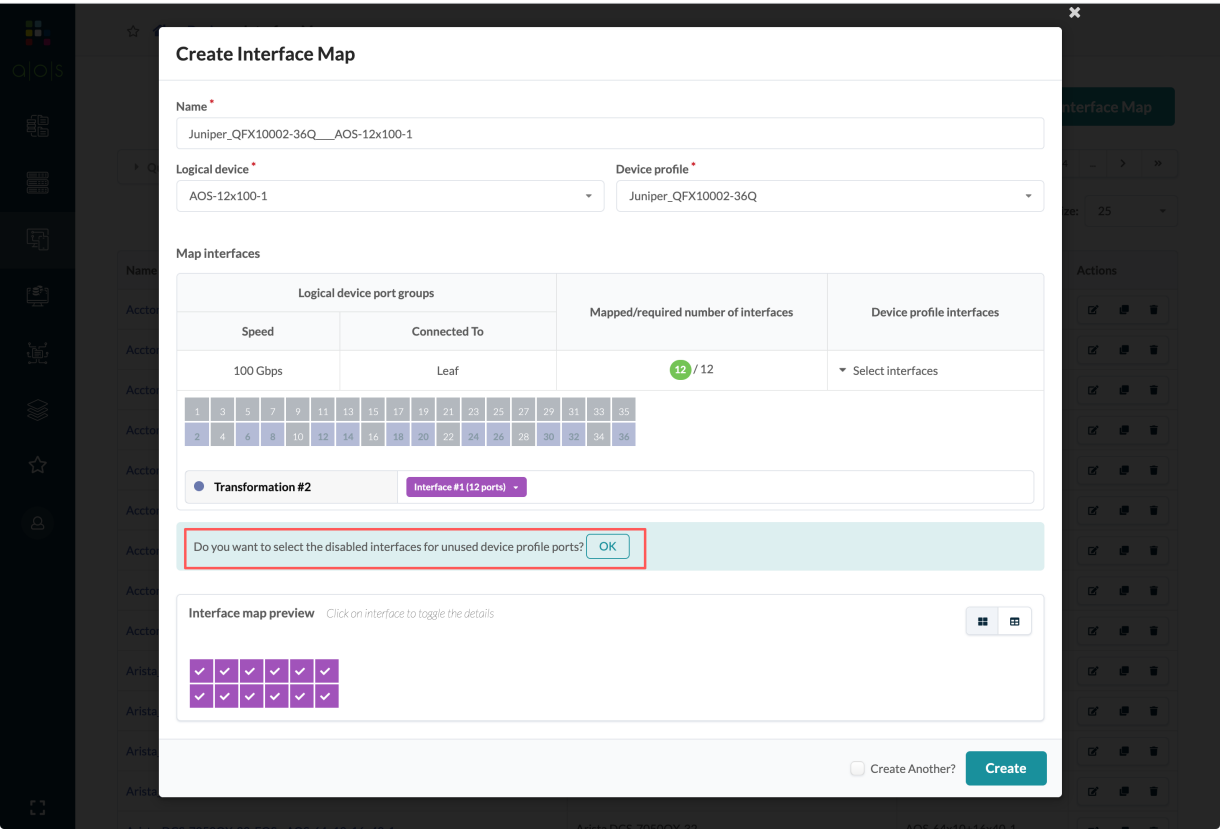
- Navigation Bar:** Located on the left, it contains icons for various functions like home, search, and device management.
- Header:** At the top, it shows the breadcrumb path: [Devices](#) > [Device Profiles](#) > Juniper_QFX10002-36Q.
- Device Information:** A table showing the device's specifications:

| |
|-------|
| 2RU |
| ASIC |
| Q5(3) |
- Ports:** A section titled "Panel #1" showing the "INTERFACES CAPACITY" and a grid of 36 ports. The ports are arranged in a 6x6 grid. Ports 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, and 36 are highlighted in red, indicating they are 100G ports. The other ports are highlighted in blue, indicating they are 40G ports.
- PORT DETAILS:** A section showing the details for the selected port (ID 2). It includes the "Connector type" (qsfp28) and the "Transformations" section, which shows the port is configured as "Port #2 Tr. #2 (100 Gbps)" with the transformation "et-0/0/1".

If these ports are used as 100G, then the adjacent QSFP 40G ports can't be used. The device profile can't automatically disable the adjacent QSFP 40G ports. You must create an interface map with these ports unused and disabled.

When you select the 100G ports while you're creating the interface map for QFX10002, you are asked if you want to select the disabled interfaces for unused device profile ports. For 100G ports on the

QFX10002, click **OK** so the unused QSFP ports are disabled and can't be used.



SONiC Device Profile

IN THIS SECTION

- Background | 107
- Problem Statement | 107
- Solution | 107
- User Interface | 107
- Selector information | 107
- Capabilities | 108
- Interface naming conventions | 108
- Troubleshooting | 109
- Example: DP and port_config.ini | 110

Background

Devices are recognized in the Apstra environment with device profiles. They capture device-specific semantics, which are required for the Apstra software to discover them 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

Device profiles are vendor semantics-aware data structures. To create a device profile, you need the device specification from the vendor. To create a valid and config-friendly JSON, you'll need to translate these specifications into the Apstra device profile data model.

Solution

The high level data model is the same for all DPs. The same keys are used for every device profile. 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 to consider, 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 |
| 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 port_congi.ini specifies interface names that SONiC uses. The device profile must match interface names which will 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. Use those interface names in your 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 the device is not selecting the device profile, check the four selector fields in the device profile.

If ports are configured with incorrect speeds or if the OS-specific port constraints were not handled in the device profile or interface map, then deploy errors could be raised.

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
- You can find the naming constraints in the official SONiC documentation. For example if you want 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 Sometimes the device might have inter-port constraints. For SONiC, it's generally laid out in the port_config.ini file. A specific platform could have multiple port_config.ini files, and a specific manufacturer with each port_config.ini file residing in their on HWSKU folders in the sonic image (like the one referenced 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 requested of 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 Apstrkr, Inc. All rights reserved.
# Apstrkr, 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\": \"41,42,43,44\"}}",

```

```

\"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\": \"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
        }
    ]
}
]
}
}

```

```

\"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
}
}

```

System Agents

IN THIS SECTION

- [Create Onbox Agent | 151](#)
- [Create Offbox Agent | 154](#)
- [Edit Agent | 160](#)
- [Uninstall and Delete Agent | 162](#)
- [Juniper Device Agent | 164](#)
- [SONiC Device Agent | 169](#)
- [Cisco Device Agent | 177](#)
- [Arista Device Agent | 189](#)
- [Packages \(Devices\) | 210](#)
- [Agent Profiles | 211](#)

Create Onbox Agent

You need **full admin / root** privileges to create onbox agents. We recommend creating a dedicated user on the device using ["Apstra ZTP" on page 230](#) 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 onbox device agents on Cisco NX-OS and Arista EOS, configure the following minimum configuration on them as shown below. (SONiC Enterprise has no specific configuration requirements other than Management Network and privileged user access.)

Cisco NX-OS Onbox 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 Onbox 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>
!

```

Make sure the following configuration is *not* on the device:

- 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 > Managed Devices**, click **Advanced Settings** (top-right), select **Skip Pristine Configuration Validation**, then click **Update**.

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 61.



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.

Onbox agents include the following parameters:

| Parameter | Description |
|-------------------------------------|--|
| Device addresses | Management IP(s) of the device(s) |
| Operation Mode | <ul style="list-style-type: none"> • Full Control - deploys configuration and collects telemetry • Telemetry Only - configuration is not deployed |
| 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"> • Install (default) - installs the agent on the device • Check - creates the agent, but does not install it. It appears in the table view where you can install it later. |
| 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. |

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.
2. From the left navigation menu, navigate to **Devices > Managed Devices** and click **Create Onbox Agent(s)**.
3. Specify agent details as described in the parameters table 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 Offbox Agent

Before installing offbox agents, 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.
- If you're using Juniper offbox agents, ["increase the application memory usage" on page 777](#).
- 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 164](#).)

Before creating/installing offbox device agents on Juniper Junos, Cisco NX-OS and Arista EOS, configure the following minimum configuration on them as shown below.

Juniper Junos Offbox 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 164](#).

Cisco NX-OS Offbox 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 Offbox 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
!

```

Make sure the following configuration is *not* on the device:

- 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 > Managed Devices**, click **Advanced Settings** (top-right), select **Skip Pristine Configuration Validation**, then click **Update**.

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 61](#).



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.

Offbox agents include the following parameters:

| Parameter | Description |
|------------------|-----------------------------------|
| Device addresses | Management IP(s) of the device(s) |

(Continued)

| Parameter | Description |
|-------------------------------------|--|
| Operation Mode | <ul style="list-style-type: none"> Full Control - deploys configuration and collects telemetry Telemetry Only - configuration is not deployed |
| Platform (offbox only) | For offbox 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"> Install (default) - installs the agent on the device Check - creates the agent, but does not install it. It appears in the table view where you can install it later. |
| 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 (offbox only) | <p>Passes configured parameters to offbox agents. For example, to use HTTPS as the API connection from offbox 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"> commit_timeout - 60 (integer: seconds) telemetry_timeout - 100 (integer: seconds) probe_timeout: 5 (integer: seconds) log_config_diff - True (boolean) |

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.
2. From the left navigation menu, navigate to **Devices > Managed Devices** and click **Create Offbox Agent(s)**.

- Specify agent details as described in the parameters table above.
- 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**.

Edit Agent

IN THIS SECTION

- Edit One Agent | 160
- Edit Multiple Agents | 161

You can edit individual agent details one at a time, or (as of Apstra version 4.1.0) you can edit details that are common to multiple agents at the same time.

Edit One Agent

- From the left navigation menu, navigate to **Devices > Managed Devices** to go to devices and agents.
- Click the three dots in the **Actions** column (right side) for the device that you want to edit, then click the **Edit** button in the **Agent** menu.

☆ 🏠 > **Devices** > Managed Devices

[+ Create Onbox Agent\(s\)](#)
[+ Create Offbox Agent\(s\)](#)
[Advanced Settings](#)

Query: All 1-5 of 5 Columns (15/16) Page Size: 25

Filter selected by ☒ all ☐ selected only ☐ unselected only

| Device Information | | | | | | | | | | Agent Information | | | Actions |
|--------------------|--------------|----------------|----------|-------------|------------|-------|---------------|--|-------|-------------------|------------------|--|---------|
| Management IP | Device Key | Device Profile | Hostname | OS | State | Comms | Acknowledged? | Blueprint | Type | Agent Profile | Apstra Version | | |
| 10.29.36.12 | 52540060061B | Cisco NXOSv | spine2 | NXOS 9.3(8) | IS- ACTIVE | 🟢 | 🟢 | zz-kathy-evpn.nxosv.2485377892354-835348458 - evpn-nxosv-virtual | ONBOX | UNASSIGNED | AOS_4.1.0_OB.115 | | 1. |
| 10.29.36.15 | 525400C89964 | Cisco NXOSv | leaf2 | NXOS 9.3(8) | IS- ACTIVE | 🟢 | 🟢 | zz-kathy-evpn.nxosv.2485377892354-835348458 - evpn-nxosv-virtual | ONBOX | UNASSIGNED | AOS_4.1.0_OB.115 | | |
| 10.29.36.11 | 5254001D6537 | Cisco NXOSv | spine1 | NXOS 9.3(8) | IS- ACTIVE | 🟢 | 🟢 | zz-kathy-evpn.nxosv.2485377892354-835348458 - evpn-nxosv-virtual | ONBOX | UNASSIGNED | AOS_4.1.0_OB.115 | | |

2. Edit agent

Delete agent

- Make your changes (device addresses, operation mode, agent profile, packages, open-options, as applicable).



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 824](#).

4. Click **Update** to update the agent and return to the table view.

Edit Multiple Agents

1. From the left navigation menu, navigate to **Devices > Managed Devices** and select one or more check boxes for the device(s) to edit.

☆ 🏠 > [Devices](#) > [Managed Devices](#)

[+ Create Onbox Agent\(s\)](#)
[+ Create Offbox Agent\(s\)](#)
[Advanced Settings](#)

Query: All 1-5 of 5

Device:
 Agent:

Filter selected by ☒ all ☐ selected only ☐ unselected only

Columns (15/16) Page Size: 25

1. Select one or more devices

| Device Information | | | | | | | | | | Agent Information | | | | | |
|-------------------------------------|---------------|--------------|----------------|----------|-------------|--------|-------|---------------|--|-------------------|---------------|------------------|---------------|-----------|---------|
| | Management IP | Device Key | Device Profile | Hostname | OS | State | Comms | Acknowledged? | Blueprint | Type | Agent Profile | Apstra Version | Last Job Type | Job State | Actions |
| <input checked="" type="checkbox"/> | 10.29.36.12 | 521000000618 | Cisco NXOSv | spine2 | NXOS 9.3(8) | ACTIVE | | | zz-kathy-evpn-nxosv-2485377892354-835348458 - evpn-nxosv-virtual | ONBOX | UNASSIGNED | AOS_4.1.0_OB.115 | INSTALL | SUCCESS | |
| <input checked="" type="checkbox"/> | 10.29.36.15 | 525400C89964 | Cisco NXOSv | leaf2 | NXOS 9.3(8) | ACTIVE | | | zz-kathy-evpn-nxosv-2485377892354-835348458 - evpn-nxosv-virtual | ONBOX | UNASSIGNED | AOS_4.1.0_OB.115 | INSTALL | SUCCESS | |

2. Assign Profile

2. Click the **Assign Profile** button (in the menu that appears above the table after making a selection).

Assign System Agent Profile ✕

System Agent Profile
 profile_vqfx

This profile will override the platform of the selected agents.

☐ Clear Existing Packages?

☐ Clear Existing Open Options?

Query: All

1-1 of 1 < >

Page Size: 25

| Device Address | Type | Agent Profile | Operation Mode | Platform | Platform Version | State | Job State | Connection State | System ID | Hostname | Device State | Action Status |
|----------------|--------|---------------|----------------|----------|------------------|-------|-----------|------------------|--------------|----------|--------------|---------------|
| 10.29.16.15 | OFFBOX | profile_vqfx | FULL CONTROL | Junos | 21.2R3.8 | | SUCCESS | CONNECTED | 525400C87D53 | leaf2 | IS-ACTIVE | N/A |

Assign System Agent Profile

3. Make your changes (agent profile, clear existing packages, clear open options).
4. Click **Assign System Agent Profile** to save your changes and return to the table view.

Uninstall and Delete Agent

If you want to remove a device from Apstra management, see ["Remove \(Decommission\) Device from Managed Devices" on page 80](#) for the complete workflow. There are additional steps before and after uninstalling and deleting the agent.

1. From the left navigation menu, navigate to **Devices > Managed Devices** to go to the managed devices table view.

NOTE: When you uninstall a device agent, the pristine configuration is restored on the device by default. If you want to retain existing configuration, click **Advanced Settings** and check the box to **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

2. Check the box(es) for the device(s), then in the **Agent Actions** panel that appears above the table, click the **Uninstall** button, click **Uninstall selected elements**, then click **Close**.

☆ 🏠 > Devices > Managed Devices

+ Create Onbox Agent(s)
 + Create Offbox Agent(s)
 Advanced Settings

Query: All 1-5 of 5

Device: [Icons]
 Agent: [Icons]
 Columns (15/16)
 Page Size: 25

Filter selected by **Uninstall** ☐ selected only ☐ unselected only

| Device Information | | | | | | | | | | Agent Information | | | | |
|--------------------|--------------|----------------|----------|-------------|-----------|-------|---------------|---|-------|-------------------|------------------|---------------|-----------|---------|
| Management IP | Device Key | Device Profile | Hostname | OS | State | Comms | Acknowledged? | Blueprint | Type | Agent Profile | Apstra Version | Last Job Type | Job State | Actions |
| 10.28.52.15 | 525400AC2DDE | Cisco NXOSv | leaf2 | NXOS 9.3(8) | IS-ACTIVE | 🟢 | 🟢 | zz-kathy-evpn.nxosv.2485377892355-3507048024 - evpn-nxosv-virtual | ONBOX | UNASSIGNED | AOS_4.1.0_OB.115 | CHECK | SUCCESS | |



NOTE: If the device is unreachable, the job will fail. You can force delete the agent (in the next step), as of Apstra version 4.1.2.

Prior to Apstra version 4.1.2, if the device is unreachable, you could ["change the agent operation mode" on page 160](#) to **telemetry only**, then uninstall the agent.

3. Check the box for the device(s) again, then in the **Agent** Actions panel that appears above the table, click the **Delete** button, click **Delete selected elements**, then click **Close**.

☆ 🏠 > Devices > Managed Devices

+ Create Onbox Agent(s)
 + Create Offbox Agent(s)

Query: All 1

Device: [Icons]
 Agent: [Icons]
 Columns (15/17)

Filter selected by ☒ all ☐ selected only ☐ unselected only

Delete

If you weren't able to uninstall the agent in the previous step because the device is unreachable, a dialog opens that gives you the option to force delete the agent. With the **Force Delete** box checked,

click **Delete** to force delete the agent and return to the table view.

Delete System Agent



Apstra cannot communicate with this device. It will not be possible to uninstall any agents or revert the device to its Pristine config before deleting. Only select this box if you understand the consequences and cannot reestablish communication with the device.

☒ Force delete?

Delete

Juniper Device Agent

IN THIS SECTION

- [Juniper ZTP | 164](#)
- [Disable ZTP | 164](#)
- [Apply Initial Juniper Junos Configuration | 165](#)
- [Configure super-user User | 167](#)
- [Configure IP address and Management VRF | 168](#)
- [Configure SSH and NETCONF | 168](#)
- [Add Junos License Configuration | 169](#)

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 230](#), 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>;
      }
    }
  }
}
}

```



NOTE: Starting in Apstra 4.1.2, devices with dual routing engines (QFX10008, for example), the minimum configuration needs to include the `commit synchronize` command at the `[edit system]` CLI hierarchy.

To enable the `commit synchronize` command, issue the `set system commit synchronize` command at the `[edit system]` CLI hierarchy.

For example:

```

{master:0}
root> edit
Entering configuration mode

{master:0}[edit]
root# set system commit synchronize

{master:0}[edit]
root# commit and-quit
configuration check succeeds

```

```
commit complete
Exiting configuration mode
```

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 `vme0`), 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 45](#).

SONiC Device Agent

IN THIS SECTION

- [SONiC Device Agent Overview | 169](#)
- [Configure Management IP Manually \(SONiC\) | 170](#)
- [Install Agent Manually \(SONiC\) | 172](#)
- [Uninstall Agent Manually \(SONiC\) | 176](#)

SONiC Device Agent Overview

Although the preferred method of installing device system agents is from the Apstra GUI, you *can* manually install Apstra agents from the CLI. Only in rare exceptions would you need to manually install agents, which requires more effort and is error-prone. Before manually installing agents, you should have an in-depth understanding of the various device states, configuration stages, and agent operations. For assistance, contact ["Juniper Support" on page 824](#).



NOTE: You can also use ["Apstra ZTP" on page 230](#) to automatically boot and install agents and prerequisite configuration on switches. Using Apstra ZTP is simpler and easier to support at scale than manually installing agents.

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 Management 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}}.runcurl` 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" ["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. Update /etc/aos/aos.conf with the `sudo vi /etc/aos/aos.conf` command to set the IP of the Apstra server and enable configuration service.

- 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 | 177](#)
- [Device Configuration Requirements | 178](#)
- [Resize and Enable Guestshell | 179](#)
- [Download Agent Installer | 179](#)
- [Install Cisco Device Agent | 180](#)
- [Update Agent Config File and Start Service | 180](#)
- [Activate Apstra Devices on Apstra Server | 181](#)
- [Deploy Device | 181](#)
- [Reset Apstra Device Agent | 181](#)
- [Uninstall Apstra Device Agent | 181](#)
- [Remove Apstra EEM Scripts | 182](#)
- [Cisco Agent Troubleshooting | 182](#)

Cisco NX-OS Device Agent Overview

Although the preferred method of installing device system agents is from the Apstra GUI, you *can* manually install Apstra agents from the CLI. Only in rare exceptions would you need to manually install agents, which requires more effort and is error-prone. Before manually installing agents, you should have an in-depth understanding of the various device states, configuration stages, and agent operations . For assistance, contact "[Juniper Support](#)" on [page 824](#).



NOTE: You can also use ["Apstra ZTP" on page 230](#) to automatically boot and install agents and prerequisite configuration on switches. Using Apstra ZTP is simpler and easier to support at scale than manually installing agents.

Manually installing an agent for Cisco devices involves the following steps:

- Update the guestshell disk size, memory and cpu, then enable/reboot the guestshell.
- Install the device agent.
- Update the aos.config file.
- Start service.



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.

If you configure NX-OS "system jumbomtu" with a value lower than the MTUs that Apstra uses, then Apstra MTU commands will fail.

Device Configuration Requirements

Configure the device in the following order: VRF, NXAPI, GuestShell, Create Management VRF. To allow for agent-server communication Apstra's device agent uses the VRF name management. 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

1. Run the following commands to resize the disk space, memory and CPU:

```
guestshell resize rootfs 1024
guestshell resize memory 2048
guestshell resize cpu 6
```

2. If the guestshell is not enabled, run the command `guestshell enable` to activate the changes.
3. If the guestshell was already enabled, run the command `guestshell reboot` to restart the shell and activate the changes.
4. Run the command `switch# show guestshell detail` and verify that the guestshell has been activated.

Download Agent Installer

You 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: To retrieve the agent file, the Cisco device connects to the Apstra server using HTTPS. Before proceeding, make sure this connectivity is functioning.

Apstra ships with 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. For the Apstra server version (4.1.0-115 for example), navigate to **Platform > About**).

```
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



NOTE: We recommend that you run the command `copy running-config startup-config` to save your latest changes, in case any issues arise.

From the Cisco NX-OS switch guestshell, run the command to install the agent as shown below:

```
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
```

Update Agent Config File and Start Service

After installing the agent and before starting service, update the `aos.conf` file so it will connect to the server.

Configure the Cisco NX-OS device agent configuration file located at `/etc/aos/aos.conf`. See ["Apstra device agent configuration file" on page 1170](#) for parameters.

After updating the file, run the command `service aos start` to start the Apstra device agent.

Activate Apstra Devices on 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 | 184](#)
- [Confirm Agent Installation | 184](#)
- [Check that Apstra Agent is Running | 185](#)
- [Check for Presence of Files in /etc/aos | 186](#)
- [Check for Apstra Data in /var/log/aos | 186](#)
- [Determine Apstra Agent Version | 187](#)
- [DNS Resolution Failure | 187](#)
- [Apstra Service Takes Long Time to Start on Cisco NX-OS | 188](#)
- [Apstra Stops and ails Without Errors \(MGMT VRF\) | 188](#)
- [Verify MGMT VRF in NX-OS Guest Shell | 189](#)

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

Within the guest shell, ping to the Apstra server to check ICMP Ping. When running commands within the context of a VRF, use the command `chvrf <vrf>` In this case, it's 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_OB.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

It takes a few minutes for the GuestShell on Cisco NX-OS to initialize the NXAPI within the LXC container. This is normal. To account for this delay, a wait-delay has been added to the Apstra script initialization.

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 | 190](#)
- [Decommission Device | 193](#)
- [Remove Apstra Package from Device | 193](#)
- [Restart System | 194](#)
- [Manually Install Arista Device Agent | 195](#)
- [Device Agent Configuration File | 197](#)
- [Arista Agent Troubleshooting | 198](#)

Although the preferred method of installing device system agents is from the Apstra GUI, you *can* manually install Apstra agents from the CLI. Only in rare exceptions would you need to manually install agents, which requires more effort and is error-prone. Before manually installing agents, you should have an in-depth understanding of the various device states, configuration stages, and agent operations . For assistance, contact ["Juniper Support" on page 824](#).



NOTE: You can also use ["Apstra ZTP" on page 230](#) to automatically boot and install agents and prerequisite configuration on switches. Using Apstra ZTP is simpler and easier to support at scale than manually installing agents.

Initial Arista EOS Configuration

IN THIS SECTION

- [Disable ZTP | 190](#)
- [Configure AAA and network-admin User | 190](#)
- [Configure IP Address and Management VRF | 191](#)
- [Configure DNS for EOS | 191](#)
- [Configure HTTP API for EOS | 192](#)
- [Configure multi-agent for EVPN | 192](#)

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 onbox 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 onbox 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 table view.
3. With the device still selected, click the **Delete system(s)** button, then click **Confirm** to remove the device and return to the table view.

Remove Apstra Package from Device

IN THIS SECTION

- [Uninstall Agent using EOS CLI | 193](#)
- [Uninstall Agent using Bash | 193](#)
- [Remove Remaining Apstra Data from System | 194](#)
- [Save Config File | 194](#)

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, the last configuration that was rendered (including any quarantine configuration) replaces the existing 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 | 195
- Install Arista Device Agent | 196



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. To proceed with manually installation see sections below. For assistance, contact "[Juniper Support](#)" on [page 824](#).

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

```
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" [ "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 directly edit the configuration file located at `/mnt/flash/aos-config`. See ["Agent Configuration file" on page 1170](#) 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 | 198](#)
- [Verify Agent is Running | 199](#)
- [DNS Resolution Failure | 201](#)
- [List Running Processes | 201](#)
- ['Unable to Connect' error during Installation | 209](#)

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_eh_0]
509 ?      S     0:00 [scsi_eh_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-cld
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 [kbfd_v4v6_rx]
1980 ?      S      0:00 [kbfd_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 tacleafsystdb --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.

Packages (Devices)

IN THIS SECTION

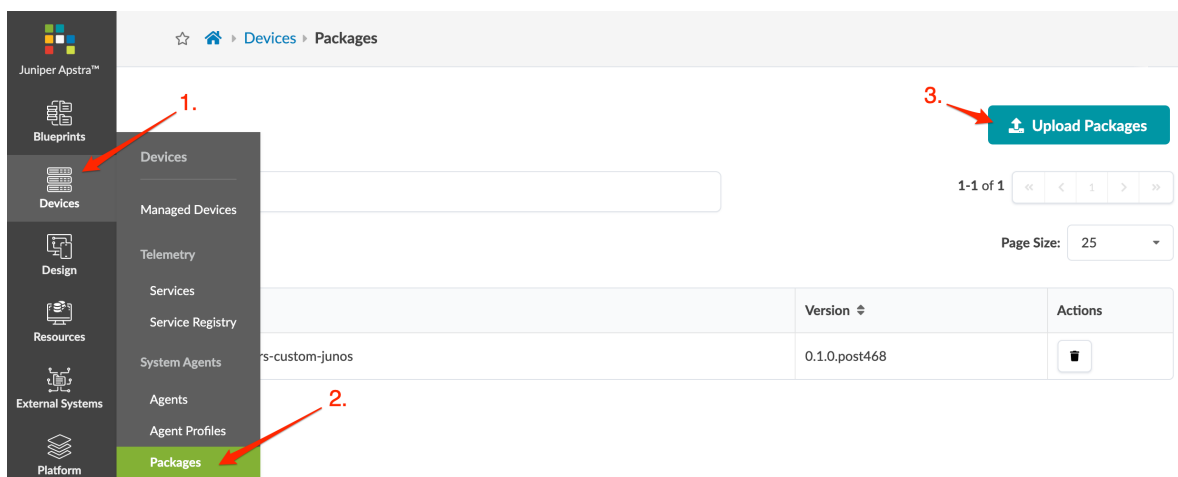
- [Packages Overview | 210](#)
- [Upload Packages | 210](#)

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 and ["agent profiles" on page 211](#). 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 [Juniper Apstra Downloads](#).
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 table view.

Agent Profiles

IN THIS SECTION

- [Agent Profiles \(Devices\) | 211](#)
- [Create Agent Profile | 212](#)
- [Edit / Delete Agent Profile | 212](#)

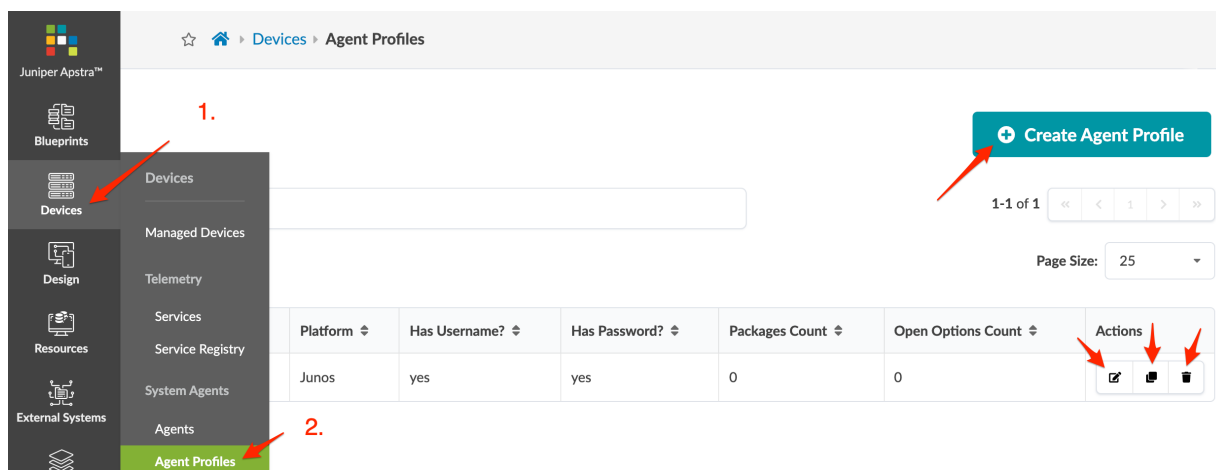
Agent Profiles (Devices)

Agent profiles enable the logical link between device credentials, a device configuration key-value store, 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 13: 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 (offbox only) | <p>Passes configured parameters to offbox agents. For example, to use HTTPS as the API connection from offbox agents to devices, use the key-value pair: proto-https - port-443. You can override the following default values with open options:</p> <ul style="list-style-type: none">● commit_timeout - 60 (integer: seconds)● telemetry_timeout - 100 (integer: seconds)● probe_timeout: 5 (integer: seconds)● log_config_diff - True (boolean) |
| Packages | Admin-provided software packages stored on the Apstra server that you can apply to each device agent that you create using the profile. |

From the left navigation menu, navigate to **Devices > System Agents > Agent Profiles** to go to the agent profile table view. You can create, clone, edit, and delete agent profiles.



Create Agent Profile

Before creating an agent profile, upload any "packages" on page 210 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. Enter a unique agent profile name.
3. Select the platform from the drop-down list (optional).
4. Set a username and password (optional).
5. Add open options (optional).
6. Select package(s) (optional).
7. Click **Create** to create the agent profile and return to the table view.

Edit / Delete Agent Profile

IN THIS SECTION

- [Edit Agent Profile | 213](#)
- [Delete Agent Profile | 213](#)

Edit Agent Profile

1. Either from the table 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 table view.

Delete Agent Profile

1. Either from the table 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 table view.

Pristine Configuration

IN THIS SECTION

- [Edit Pristine Config | 213](#)
- [Update Pristine Config from Device | 214](#)

Edit Pristine Config

Modifying pristine config is a local operation, and does not lead to a change to the running device configuration. Changes are applied on the next full config push. If you want to apply persistent changes to a configuration, use ["configlets" on page 45](#).



CAUTION: Manual modifications to the Pristine Config are not validated. Mistakes can lead to full erasure of the device, potentially causing a service-impacting outage. Never modify the pristine config directly unless there is no alternative. For assistance, contact ["Juniper Support" on page 824](#).

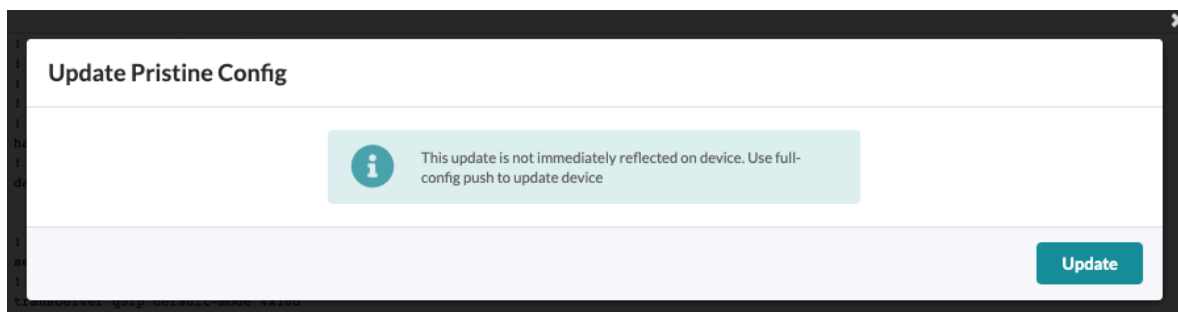
1. From the left navigation menu, navigate to **Devices > Managed Devices** and click the **Management IP** of the device to edit.
2. Click the **Pristine Config** tab (top-left), then click the **Edit pristine config** button (under **checkpoint** on the left).



3. Make your changes.
4. Click **Update** to apply the changes.

Update Pristine Config from Device

1. From the blueprint, unassign the device by removing the system ID (Staged > Physical > Build > Devices). Make sure the device is in the out of service state (OOS-READY or OOS-MAINT).
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 click the **Management IP** of the device to edit.
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.



Verify the Pristine Config. You have copied the running config of the device in the out of service state, which should be **Discovery 1** config. It may include additional configuration such as interface "speed" commands. You can edit Pristine Config again and delete the additional configuration manually. Contact ["Juniper Support" on page 824](#) for assistance as needed.

Telemetry

IN THIS SECTION

- [Services | 215](#)
- [Service Registry | 218](#)
- [Telemetry Collection Statistics | 220](#)
- [Telemetry Streaming | 222](#)
- [Route Anomalies for a Host - Example | 223](#)
- [Juniper Telemetry Commands | 225](#)
- [Cisco Telemetry Commands | 226](#)
- [Arista Telemetry Commands | 227](#)
- [Linux Servers | 228](#)
- [Debugging Telemetry | 229](#)

Services

From the left navigation menu, navigate to **Devices > Telemetry > Services** to go to a summary of telemetry services.

Click a service name to see details

1. ARP

2. Services

| Service Name | Configured on | Errors during enabling | Last collection cycle errors |
|--------------|---------------|------------------------|------------------------------|
| ARP | 5 devices | 0 devices | 0 devices |
| BGP | 7 devices | 0 devices | 0 devices |
| BGP_IBA | 5 devices | 0 devices | 0 devices |
| DISK UTIL | 5 devices | 0 devices | 0 devices |
| HOSTNAME | 7 devices | 0 devices | 0 devices |

Telemetry services include the following:

| Service | Description |
|----------|---|
| ARP | ARP telemetry shows an ARP table. You can query this information via API. Anomalies are not 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>This is the running config.</p> <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 "configlet" on page 45 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 Ready to a blueprint, the device enters the Ready stage (previously known as Discovery 2). Hostname telemetry is collected that validates the device hostname against intent. Mismatches result in anomalies. |

(Continued)

| Service | Description |
|----------------|---|
| Interface | When you assign a device with deploy mode Ready to a blueprint, the device enters the Ready stage (previously known as Discovery 2). Interface telemetry is collected that compares intent with the 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 Ready to a blueprint, the device enters the Ready stage (previously known as Discovery 2). 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 you must correct 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> <p>NOTE: 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> |

(Continued)

| Service | Description |
|---------------------------------|---|
| 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 (Onbox 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 offbox agents (Junos for example). Therefore, the utilization tab contains the error Network Device not found.</p> |

Service Registry

IN THIS SECTION

- [Service Registry Overview | 218](#)
- [Import Service Schemas | 220](#)
- [Delete Service Registry | 220](#)

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 861](#).

Juniper Apstra™

Blueprints

Devices

Design

Resources

External Systems

Platform

Favorites

User: admin

☆

🏠

Devices

Service Registry

Import Service Schemas

1-17 of 17

« < 1 > »

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_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.

Juniper Apstra™

Blueprints

Devices

Design

Resources

External Systems

Platform

Favorites

☆

🏠

Devices

Service Registry

evpn_vxlan_type5

| | |
|---------------------|--|
| 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" } } } } }</pre> |

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 table 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). |

(Continued)

| Collection Statistics | Description |
|-----------------------|---|
| 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 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. |
| Event Data | Event data is a collection of status information that you may need to refer back for troubleshooting your network. syslog is the best reference for example event. 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. |

Data streams are implemented with Google Protocol Buffers (GPB). GPBs define and implement the format of data streams. 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. Example Python code named "AOSOM Streaming" on page 902 is available for GPBs. The AOSOM Streaming demo software is open source and you can download it 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 (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": "547bcb9-963f-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
}

```

Juniper Telemetry Commands

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.

Apstra uses CLI to retrieve telemetry from Junos OS and Junos OS Evolved devices.

| Service | Command |
|--------------------------|---|
| Interface Counters | show interfaces extensive |
| Interface Error Counters | show interfaces extensive |
| Interface Status | show interfaces terse |
| LLDP neighbors | show lldp neighbors |
| BGP sessions | show bgp neighbor |
| Hostname | show system information |
| ARP | show arp no-resolve Provides the ARP information. This is combined with show configuration routing-instances which provides the VRF membership for interfaces. |
| MAC Table | show ethernet-switching table extensive |
| Routing Table | show route table inet |
| Port Channel | show lacp interfaces |
| Environment | show chassis environment |

Cisco Telemetry Commands

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 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 |

(Continued)

| Service | Command |
|--------------------------|--|
| Interface error counters | <code>show interface counters errors json</code> |
| Interface status | <code>show interface status json</code> |
| LLDP neighbors | <code>show lldp neighbors detail json</code> |
| BGP Sessions | <code>show bgp session json</code> |
| Hostname | <code>show hostname json</code> and <code>show hosts json</code> |
| ARP | <code>show ip arp vrf default json</code> |
| MAC Table | <code>show mac address-table json</code> |
| Routing table | <code>show ip route json</code> |
| Port-channel | <code>show port-channel summary json</code> |
| MLAG | <code>show vpc json</code> |

Arista Telemetry Commands

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.

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 189](#).

When the Arista API does not provide information (LLDP statistics), Apstra runs CLI commands at a regular interval to derive telemetry expectations.

| Service | Command |
|--------------------------|--|
| 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 |

Linux Servers

Linux Servers use simple CLI commands and standard Linux sockets for most of the telemetry collection.

| Service | Command |
|--------------------------|------------|
| Interface counters | ethtool -m |
| Interface error counters | ethtool -m |

(Continued)

| Service | Command |
|------------------|---|
| Interface status | Interface status is collected using the netlink api (AF_INET) |
| LLDP neighbors | lldpctl -f xml |
| BGP Sessions | vysh -c 'show ip bgp summary json' |
| Hostname | hostname |
| ARP | ip -4 neigh |
| MAC Table | brctl showmacs |
| Routing table | show ip route and the AF_INET linux socket |
| Port-channel | netshow bondmems --json |
| MLAG | clagctl -j |

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.

Apstra ZTP

IN THIS SECTION

- [Apstra ZTP \(Devices\) | 230](#)
- [Apstra ZTP - Juniper | 247](#)
- [Apstra ZTP - SONiC | 251](#)
- [Apstra ZTP - Cisco | 253](#)
- [Apstra ZTP - Arista | 257](#)
- [Upgrade Apstra ZTP | 260](#)

Apstra ZTP (Devices)

IN THIS SECTION

- [Apstra ZTP Overview | 230](#)
- [Download and Deploy Apstra ZTP VM | 235](#)
- [Configure Static Management IP Address \(Apstra ZTP\) | 236](#)
- [Configure ZTP User | 237](#)
- [Configure DHCP Server | 238](#)
- [Configure Controller IP Address for ZTP | 241](#)
- [Edit Apstra ZTP Configuration File | 241](#)

Apstra ZTP Overview



NOTE: This document applies to Apstra ZTP 4.1 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 system 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
- Onbox or Offbox Device System Agent installation

See also vendor-specific information:

- ["Juniper Junos" on page 247](#)
- ["Enterprise SONiC" on page 251](#)
- ["Cisco NX-OS" on page 253](#)
- ["Arista EOS" on page 257](#)



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. Agent Installation Phase

- The ZTP script makes an API call to install a device system agent on the device.

Apstra ZTP VM Server Resource Requirements

Apstra ZTP runs as an Ubuntu 18.04 LTS server running a DHCP, HTTP, and TFTP server and includes Apstra provided ZTP scripts that you must customize for 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 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 in the [Juniper Apstra Installation and Upgrade Guide](#) for more information.

You can monitor device ZTP status from the Apstra GUI. From the left navigation menu, navigate to **Devices > ZTP Status > Devices**.

| System ID | ZTP Status | ZTP Latest Event | Last Updated | Actions |
|------------|------------|------------------|----------------------|---------------|
| [Redacted] | Completed | Device Ready | 2022-05-01, 11:46:36 | [Eye] [Trash] |
| [Redacted] | Completed | Device Ready | 2022-05-01, 11:46:37 | [Eye] [Trash] |
| [Redacted] | Completed | Device Ready | 2022-05-01, 11:46:42 | [Eye] [Trash] |
| [Redacted] | Completed | Device Ready | 2022-05-01, 11:46:47 | [Eye] [Trash] |
| [Redacted] | Completed | Device Ready | 2022-05-01, 11:46:36 | [Eye] [Trash] |
| [Redacted] | Completed | Device Ready | 2022-05-01, 11:48:40 | [Eye] [Trash] |
| [Redacted] | Completed | Device Ready | 2022-05-01, 11:48:55 | [Eye] [Trash] |

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** button.

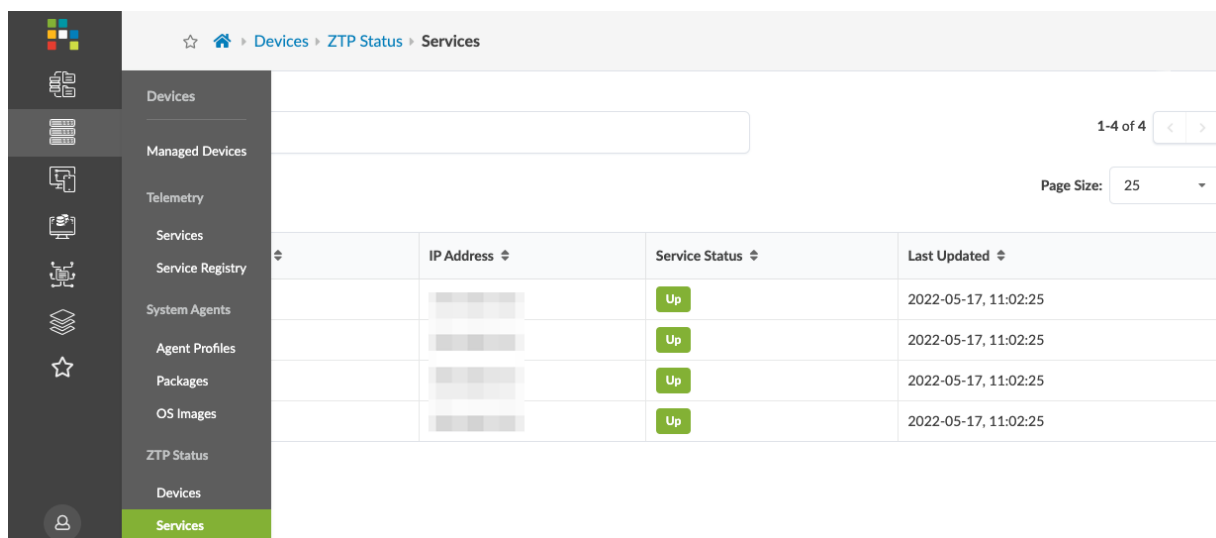
Log files for all processes are in the /containers_data/logs directory.

```

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#

```

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**.



| | IP Address | Service Status | Last Updated |
|--|------------|----------------|----------------------|
| | | Up | 2022-05-17, 11:02:25 |
| | | Up | 2022-05-17, 11:02:25 |
| | | Up | 2022-05-17, 11:02:25 |
| | | Up | 2022-05-17, 11:02:25 |

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-4.1.*-<build-version>.ova (example: apstra-ztp-4.1.0-4.ova) |
| Microsoft Hyper-V | apstra-ztp-4.1.*-<build-version>.vhdx.gz (example: apstra-ztp-4.1.0-115.vhdx.gz) |
| Linux KVM QCOW2 image | apstra-ztp-4.1.*-<build-version>.qcow2.gz (example: apstra-ztp-4.1.0-115.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
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
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 731](#).

The screenshot shows the 'Create User' modal in the Apstra web interface. The modal is centered on a dark background. It has a title bar 'Create User' with a close button. The form inside includes:

- Username**: A text field with 'ztp' entered.
- First Name**: An empty text field.
- Last Name**: An empty text field.
- Email**: An empty text field.
- Password**: A password field with masked characters.
- Repeat Password**: A password field with masked characters.
- Global Roles**: A section with four radio buttons: 'administrator', 'device_ztp' (selected), 'user', and 'viewer'.
- Per-Blueprint Roles**: A section that is currently empty.
- Footer**: A checkbox 'Create Another?' and a green 'Create' button.

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 Junos OS or Junos OS Evolved devices, you must ensure the server contains the following, so the device loads the proper configuration file.

```
option space JUNIPER
option JUNIPER.config-file-name code 1 = text
option JUNIPER-encapsulation code 43 = encapsulate JUNIPER
option user-class-information code 77 = text;
class "juniper" { match if (substring(option vendor-class-identifier, 0, 7) = "Juniper") and
not (suffix(option user-class-information, 4) = "-EVO");
option JUNIPER.config-file-name "junos_apstra_ztp_bootstrap.sh";
}
class "juniper-evo" {
match if (substring(option vendor-class-identifier, 0, 7) = "Juniper") and
(suffix(option user-class-information, 4) = "-EVO");
```

```
option JUNIPER.config-file-name "ztp.py";
}
```

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 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

5. 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

Configure the controller IP and Apstra ZTP username in the `/containers_data/status/app/aos.conf` file on the Apstra ZTP 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. (Cumulus is no longer supported as of Apstra version 4.1.0.)

```
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 ("nxos", "eos", "junos", "sonic") unless more specific keys are defined.

```
"sonic": {
  "sonic-versions": ["SONiC-OS-3.4.0-Enterprise_Advanced"],
  "sonic-image": "http://10.85.24.52/sonic/3.4.0/sonic-3.4.0-GA-adv-bcm.bin",
  "device-root-password": "admin",
  "device-user": "admin",
  "device-user-password": "admin",
  "custom-config": "sonic_custom.sh",
  "system-agent-params": {
    "agent_type": "onbox",
    "job_on_create": "install"
  }
}
```

| | |
|--|---|
| model - Values are used for all devices for a specific device model (for example "QFX10002-36Q"). | <pre> "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" } </pre> |
| serial number - Values are used for a device matching a specific device serial number (for example "TH0TFD6TCET0015G0015"). | <pre> "TH0TFD6TCET0015G0015": { "sonic-versions": ["SONiC-OS-4.0.5-Enterprise_Advanced"], "sonic-image": "http://10.85.24.52/sonic/4.0.5/sonic- broadcom-enterprise-advanced-4.0.5-GA.bin" } </pre> |

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. | <pre>"junos-versions": ["20.2R2-S3.5"]</pre> |
| junos-image - Filename of the Juniper Junos TGZ image to load if the running version does not match a version in the junos-versions list. | <ul style="list-style-type: none"> 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" <p>This example uses HTTP from the controller to transfer the Juniper Junos image.</p> |
| 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. | <pre>"sonic-versions": ["SONiC-OS-3.1.0a-Enterprise_Base"]</pre> |

sonic-image - Filename of the SONiC ONIE BIN image to load 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 load 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 load 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.

`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"`

`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": "sonic_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 onbox 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 offbox agents only) Set to the device platform ("eos", "nxos", "junos"). Lowercase only.

"platform": "junos"

open_options - (offbox agents only) Set to enable HTTPS between offbox 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

IN THIS SECTION

- [Juniper and ZTP Disk Space | 247](#)
- [Example: Juniper Junos - ztp.json | 248](#)
- [Example: Juniper Junos OS Evolved - ztp.json | 248](#)
- [Juniper Junos Bootstrap File | 249](#)
- [Juniper Junos Custom Config File | 249](#)
- [Restart Juniper Junos ZTP | 251](#)
- [Troubleshoot Juniper Junos ZTP | 251](#)

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 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. The ZTP process copies a new OS image 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 4.1

```
{
  "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"
    }
  }
}
```

Example: Juniper Junos OS Evolved - ztp.json

IN THIS SECTION

- [Juniper Junos OS Evolved Offbox Agent / Apstra ZTP 4.1 | 248](#)

Juniper Junos OS Evolved Offbox Agent / Apstra ZTP 4.1

```
{
  "junos-evo": {
    "junos-evo-versions": [ "20.4R3-S1.3-EV0" ],
    "junos-evo-image": "http://192.168.59.4/junos-evo-install-qfx-ms-fixed-x86-64-20.4R3-S1.3-EV0.iso",
    "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"
    }
}
}

```

The following additional fields can be used for dual RE platforms, such as PTX10004.

```

"dual-routing-engine": true,
"management-ip": "10.161.37.7",
"management-gw-ip": "10.161.39.254",
"management-subnet-prefixlen": "21",
"management-master-ip": "10.161.37.8",
"management-backup-ip": "10.161.37.9",

```

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 a device that is not Junos OS Evolved, 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.

Junos OS Evolved devices don't require this bootstrap; they run the Apstra ZTP python script (ztp.py) directly.

Juniper Junos Custom Config File

When configuring `custom-config` for Juniper Junos devices, refer to the example `junos_custom.sh`, a bash executable file executed during the ZTP process. It can set system configuration (such as Syslog, NTP, SNMP authentication) prior to device system agent installation.



NOTE: Junos OS and Junos OS Evolved platforms with dual-RE setups require the `set system commit synchronize` command. Without this configuration, the ZTP process fails. We recommend adding the command to the `junos_custom.sh` file.

```
#!/bin/sh

SOURCE_IP=$(cli -c "show conf interfaces em0.0" | grep address | sed 's/.*address \([0-9.]*\)*/\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.

You can find additional useful messages in `/var/log/messages` (search for 'ztp').

Apstra ZTP - SONiC

IN THIS SECTION

- [Enterprise SONiC and ZTP Overview | 252](#)
- [Example: Enterprise SONiC -ztp.json | 252](#)
- [Enterprise SONiC Custom Config File | 253](#)
- [Restart Enterprise SONiC ZTP | 253](#)

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 onbox agents installed. It uses a custom script to create onbox 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 Onbox 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 | 254](#)
- [Example: Cisco NX-OS - ztp.json | 254](#)

- [Cisco NX-OS Custom Config File | 255](#)
- [Cisco NX-OS Offbox Agent Custom Config File | 255](#)
- [Restart Cisco NX-OS ZTP | 256](#)

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, you must copy the image (nxos.7.0.3.I7.7.bin for example) 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 offbox agent HTTPS (port 443) between the offbox 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 that set system configuration, such as the SSH login banner, before installing the device system agent.



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 Offbox Agent Custom Config File

If using Apstra ZTP to prepare a Cisco NX-OS device for use with offbox 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
```

You can use the following `nxos_custom.sh` 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 177](#) for more information.

To restart Cisco NX-OS ZTP process:

```

switch# write erase
switch# reload

```


Apstra ZTP - Arista

IN THIS SECTION

- [Arista EOS | 257](#)
- [Example: Arista EOS - ztp.json | 258](#)
- [Arista EOS Custom Config File | 258](#)
- [Restart Arista EOS ZTP | 259](#)

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, the ZTP script copies this to the permanent configuration.

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 189](#) 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
```

Upgrade Apstra ZTP

1. As a registered support user, download the Apstra ZTP image for your hypervisor from [Juniper Support Downloads](#). (This example uses ZTP v1.0.0-33.)
2. Extract the apstra-ztp-1.0.0-33.tar.gz file (for this example) in /home/admin, then run docker-compose with the docker-compose.yml file:

```
admin@localhost:~$ tar zxvf apstra-ztp-1.0.0-33.tar.gz
./
./etc/
./etc/apstra_ztp/
./etc/apstra_ztp/docker-compose.yml
./etc/apstra_ztp/docker-compose-nohttp.yml
./containers_data/
./containers_data/dhcp/
./containers_data/dhcp/dhcpd.conf
./containers_data/init
./containers_data/Dockerfile.tftp
./containers_data/tftp/
./containers_data/tftp/eos_custom.sh
./containers_data/tftp/ztp.json
./containers_data/tftp/cumulus_license_file
./containers_data/tftp/nxos_custom.sh
./containers_data/tftp/cumulus_custom.sh
./containers_data/tftp/ztp.py
./containers_data/tftp/poap-md5sum
admin@localhost:~$ cd etc
admin@localhost:~/etc$ ls
apstra_ztp
admin@localhost:~/etc$ cd apstra_ztp/
admin@localhost:~/etc/apstra_ztp$ sudo docker-compose -f ./docker-compose.yml up --detach
WARNING: The CONTAINER_DATA_BASE_DIR variable is not set. Defaulting to a blank string.
Pulling http (nginx:latest)...
latest: Pulling from library/nginx
000eee12ec04: Pull complete
eb22865337de: Pull complete
bee5d581ef8b: Pull complete
Digest: sha256:50cf965a6e08ec5784009d0fccb380fc479826b6e0e65684d9879170a9df8566
Status: Downloaded newer image for nginx:latest
```

```
apstra_ztp_tftp_1 is up-to-date
apstra_ztp_dhcpd_1 is up-to-date
Creating apstra_ztp_http_1 ... done
admin@localhost:~/etc/apstra_ztp$
```

3. Run `docker ps` to see that the `apstra_ztp_http_1` container was created.

```
admin@localhost:~/etc/apstra_ztp$ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS          NAMES
4b879edd355a   nginx:latest "nginx -g 'daemon of..." 25 seconds ago Up 21 seconds 0.0.0.0:80->80/tcp   apstra_ztp_http_1
7f753bdc5853   networkboot/dhcpd "/entrypoint.sh eth0"     3 months ago Up 2 minutes                               apstra_ztp_dhcpd_1
61bfc64f294d   apstra/tftp "sh /init"               3 months ago Up 26 minutes 0.0.0.0:69->69/udp   apstra_ztp_tftp_1
admin@localhost:~/etc/apstra_ztp$
```

4. Copy the updated `ztp.py` and `poap-md5sum` files to the `/containers_data/tftp/` directory.

```
admin@localhost:~$ cd ~/containers_data/tftp/
admin@localhost:~/containers_data/tftp$ sudo cp ztp.py /containers_data/tftp/
admin@localhost:~/containers_data/tftp$ sudo cp poap-md5sum /containers_data/tftp/
admin@localhost:~/containers_data/tftp$
```

5. Update `/containers_data/tftp/ztp.json` accordingly.

Resources Introduction

IN THIS SECTION

- [ASN Pools \(Resources\) | 262](#)
- [VNI Pools \(Resources\) | 264](#)
- [Integers | 266](#)
- [IPv4 Addresses | 267](#)
- [IPv6 Addresses | 270](#)

In Apstra, resources are values that are assigned to various elements of the network. Resources include the following types:

- IPv4 (including Host IPv4)
- IPv6 (including Host IPv6)
- ASN - (autonomous system number)
- VNI (virtual network identifier)
- VLAN (virtual local area network)
- Integer - used for pool type VLAN in local pools in Freeform blueprints (new in Apstra version 4.1.2)

ASN Pools (Resources)

IN THIS SECTION

- [ASN Pool Overview | 262](#)
- [Create ASN Pool | 263](#)
- [Edit ASN Pool | 263](#)
- [Delete ASN Pool | 264](#)

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 | A unique name to identify the resource pool |

(Continued)

| Name | Description |
|-------------|---|
| 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.

| Name | Total Usage | Range Usage | Status | Actions |
|--------------------------------|-------------|-------------|------------|-------------------|
| me | 5% | 5% | IN USE | Edit Clone Delete |
| VNI Pools 64512-65534 | 0% | 0% | NOT IN USE | Edit Clone Delete |
| IP Pools 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 unique 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 table view.

When you're building your blueprint, you'll "assign resources" on page 277 from these pools in the **Staged > Physical** view of the blueprint.

Edit ASN Pool

1. Either from the table 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 table 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 table 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 table view.

VNI Pools (Resources)

IN THIS SECTION

- [VNI Pool Overview | 264](#)
- [Create VNI Pool | 265](#)
- [Edit VNI Pool | 265](#)
- [Delete VNI Pool | 266](#)

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 409](#). When you're building your blueprint you'll specify which resource pool to use for assigning VNIs.



NOTE: If you need to assign a specific VNI to a specific device, you can assign the VNI individually from the staged blueprint in the **Properties** panel of a selection.

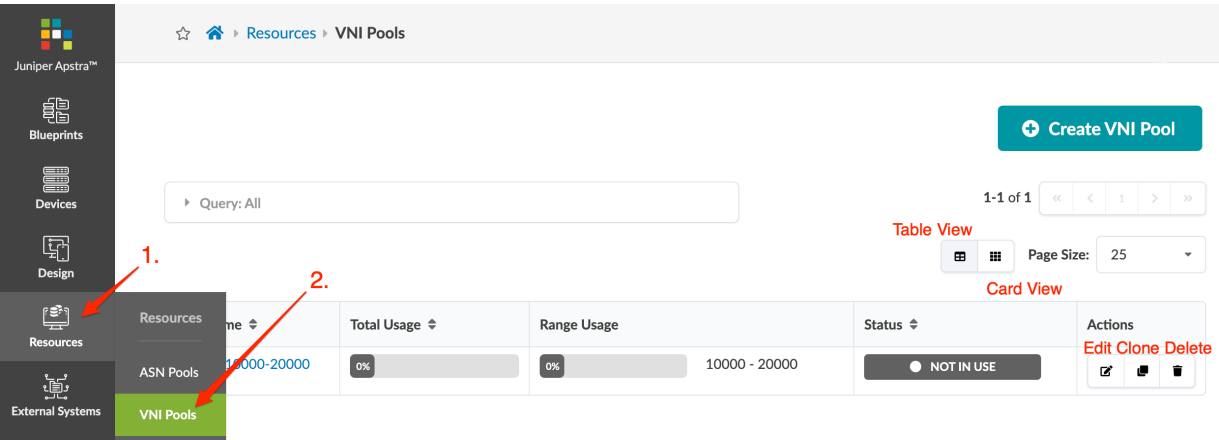
VNI pools include the following details:

| Name | Description |
|-----------|---|
| Pool Name | A unique 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 unique 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 table view.

When you've created your blueprint, you'll ["assign resources" on page 277](#) from these pools in the **Staged > Virtual** view of the blueprint.

Edit VNI Pool

1. Either from the table 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 table 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 table 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 table view.

Integers

IN THIS SECTION

- [Integers in Apstra | 266](#)
- [Create Integer Pool | 267](#)
- [Update Integer Pool | 267](#)
- [Delete Integer Pool | 267](#)

Integers in Apstra

IN THIS SECTION

- [Integer Pool Details | 266](#)
- [Integer Pools in the Apstra GUI | 267](#)

Integers are optional resource types for Freeform blueprints only. In Apstra, you create Integer pools in the global **Resources** catalog. When you're building your blueprint, you can assign an integer pool, then Apstra will automatically pull resources as needed. If you need to assign a specific integer, you can assign it manually.

Integer Pool Details

Integer pools include the following details:

| | |
|-------------|---|
| Name | A unique name to identify the resource pool |
|-------------|---|

Ranges The beginning and ending numbers of the range.

Integer Pools in the Apstra GUI

From the left navigation menu in the Apstra GUI, navigate to **Resources > Integer Pools** to go to the global catalog where you can create, edit and delete integer pools.

Create Integer Pool

1. From the left navigation menu, navigate to **Resources > Integer Pools** and click **Create Integer Pool**.
2. Enter a unique 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 table view.

Update Integer Pool

1. Either from the table view (Resources > Integer 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 integers that are in use.
3. Click **Update** to update the pool and return to the table view.

Delete Integer Pool

1. Either from the table view (Resources > Integer 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 table view.

IPv4 Addresses

IN THIS SECTION

- [IP Pool Overview | 268](#)
- [Create IPv4 Pool | 269](#)
- [Edit IPv4 Pool | 269](#)
- [Delete IPv4 Pool | 269](#)

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 spine devices and leaf devices to build the L3-CLOS fabric. These IPs are necessary for BGP peering between spine devices and leaf devices, 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 14: IPv4 Pool Parameters

| Name | Description |
|------------------|---|
| Pool Name | A unique 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™

Blueprints
Devices
Design
Resources
External Systems
Platform

☆ Home > Resources > IP Pools

Query: All

1-6 of 6

Table View
Card View

Page Size: 25

| Name | Total Usage | Per Subnet Usage | Status | Actions |
|-------------|-------------|---------------------|------------|-------------------|
| AS429702 | 2.37% | 2.37% 172.16.0.0/16 | IN USE | Edit Clone Delete |
| 11.1.0.0/16 | <0.01% | <0.01% 11.1.0.0/16 | IN USE | Edit Clone Delete |
| 10.0.0.0/8 | 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 unique 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 table view.

When you've created your blueprint, you'll "assign resources" on page 277 from these pools in the **Staged > Physical** view of the blueprint.

Edit IPv4 Pool

1. Either from the table 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 table 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 table 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 table view.

IPv6 Addresses

IN THIS SECTION

- [IPv6 Pool Overview | 270](#)
- [Create IPv6 Pool | 271](#)
- [Edit IPv6 Pool | 271](#)
- [Delete IPv6 Pool | 272](#)

IPv6 Pool Overview

To use IPv6 addressing, you must update the ["fabric addressing policy" on page 551](#) to enable IPv6 in the blueprint (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 Apstra Feature Matrix 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 15: IPv4 Pool Parameters

| Name | Description |
|-------------|--|
| Pool Name | A unique 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.) |

Table 15: IPv4 Pool Parameters (Continued)

| Name | Description |
|------------------|---|
| 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.

Create IPv6 Pool

1. From the left navigation menu, navigate to **Resources > IPv6 Pools** and click **Create IPv6 Pool**.
2. Enter a unique 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 table view.

When you've created the blueprint, you'll ["assign resources" on page 277](#) from these pools in the **Staged > Virtual** view of the blueprint.

Edit IPv6 Pool

1. Either from the table 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 table 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 table 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 table view.

Datacenter Reference Design

IN THIS SECTION

- [Create / Delete Datacenter Blueprint | 273](#)
- [Datacenter Blueprint Summary and Dashboard | 274](#)
- [Assign Physical Resources \(Datacenter\) | 277](#)
- [Assign Device Profiles | 280](#)
- [Configlets \(Datacenter Blueprint\) | 281](#)
- [Topology \(Datacenter\) | 287](#)
- [Nodes \(Datacenter\) | 293](#)
- [Links \(Datacenter\) | 338](#)
- [Racks \(Datacenter\) | 387](#)
- [Pods \(Datacenter\) | 391](#)
- [Planes \(Datacenter\) | 405](#)
- [Virtual Networks | 409](#)
- [Routing Zones | 429](#)
- [Static Routes \(Virtual\) | 439](#)
- [Protocol Sessions \(Virtual\) | 440](#)
- [Data Center Interconnect \(DCI\) / Remote EVPN Gateways \(Virtual\) | 441](#)
- [Virtual Infra \(Virtual\) | 454](#)
- [Endpoints Overview \(Virtual\) | 521](#)
- [Policies \(Datacenter\) Staged | 527](#)
- [Logical Devices \(Datacenter Blueprint\) | 559](#)
- [Interface Maps \(Datacenter Blueprint\) | 561](#)

- [Property Sets \(Datacenter Blueprint\) | 562](#)
- [AAA Servers \(Datacenter Blueprint\) | 564](#)
- [Tags \(Datacenter Blueprint\) | 568](#)
- [Tasks \(Datacenter\) Staged | 571](#)
- [Connectivity Templates | 571](#)
- [Active \(Datacenter Blueprint\) | 596](#)
- [BGP Route Tagging | 623](#)

Create / Delete Datacenter Blueprint

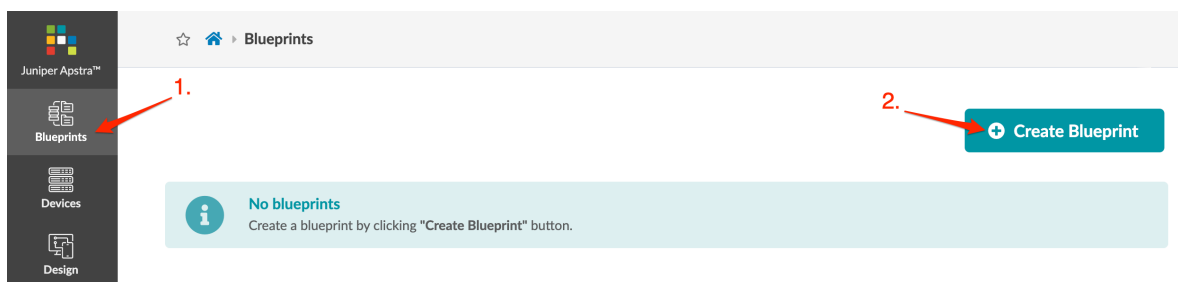
IN THIS SECTION

- [Create Datacenter Blueprint | 273](#)
- [Delete Datacenter Blueprint | 274](#)

Create Datacenter Blueprint

Datacenter blueprints are based on templates. Make sure a suitable ["template" on page 35](#) exists in the global catalog (Design > Templates).

1. From the left navigation menu in the Apstra GUI, click **Blueprints**, then click **Create Blueprint**.



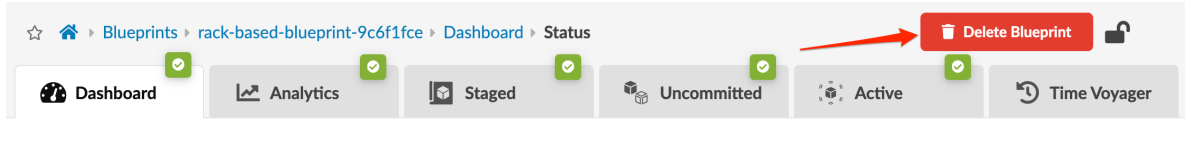
2. Enter a unique name and select a template from the **Template** drop-down list. A preview shows template parameters, topology preview, structure, external connectivity, and policies.
3. Click **Create** to create the blueprint and return to the blueprint summary view.

Next Steps: ["Assign Resources" on page 277](#).

Delete Datacenter Blueprint

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.

Datacenter Blueprint Summary and Dashboard

IN THIS SECTION

- [Blueprints Summary | 274](#)
- [Blueprint Dashboard | 276](#)

Blueprints Summary

The blueprint summary page shows a summary of each individual blueprint. At the top of the page, the different status indicators show various statuses across all blueprints (deployment status, anomalies, root causes, build errors and warnings, and uncommitted changes. This is useful when you have many blueprints in your Apstra instance. To quickly filter to show only blueprints that meet a certain criteria, click a color in one of the indicators. In the example below, clicking the gray part in the **Deployment Status** indicator results in displaying only the blueprints that don't have any nodes deployed yet. If the red part in the **Anomalies** indicator were clicked, only the blueprints that have anomalies would display.

☆

🏠

Blueprints

Click a color to filter blueprints

Deployment Status

Anomalies

Root Causes

Build Errors

Build Warnings

Uncommitted Changes

Create Blueprint

Query: All

1-1 of 1

⚠

Showing only blueprints that don't have any nodes deployed yet. [Reset filter](#)

📄

📊

Page Size: 25

3s-v4-v6

Datacenter

Structure:

2 spines, 4 leafs, 4 access switches, 8 generic systems

Analytics

Deployment Status

N/A

Service Anomalies

N/A

Probe Anomalies

N/A

Root Causes:

N/A

Version 16

Last modified a day ago

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.

Juniper Apstra™

☆ 🏠 > Blueprints

1. Click to go to blueprints dashboard

Deployment Status Anomalies Root Causes Build Errors Build Warnings Uncommitted Changes

Query: All 1-1 of 1

Create Blueprint

2. Click to go to blueprint dashboard

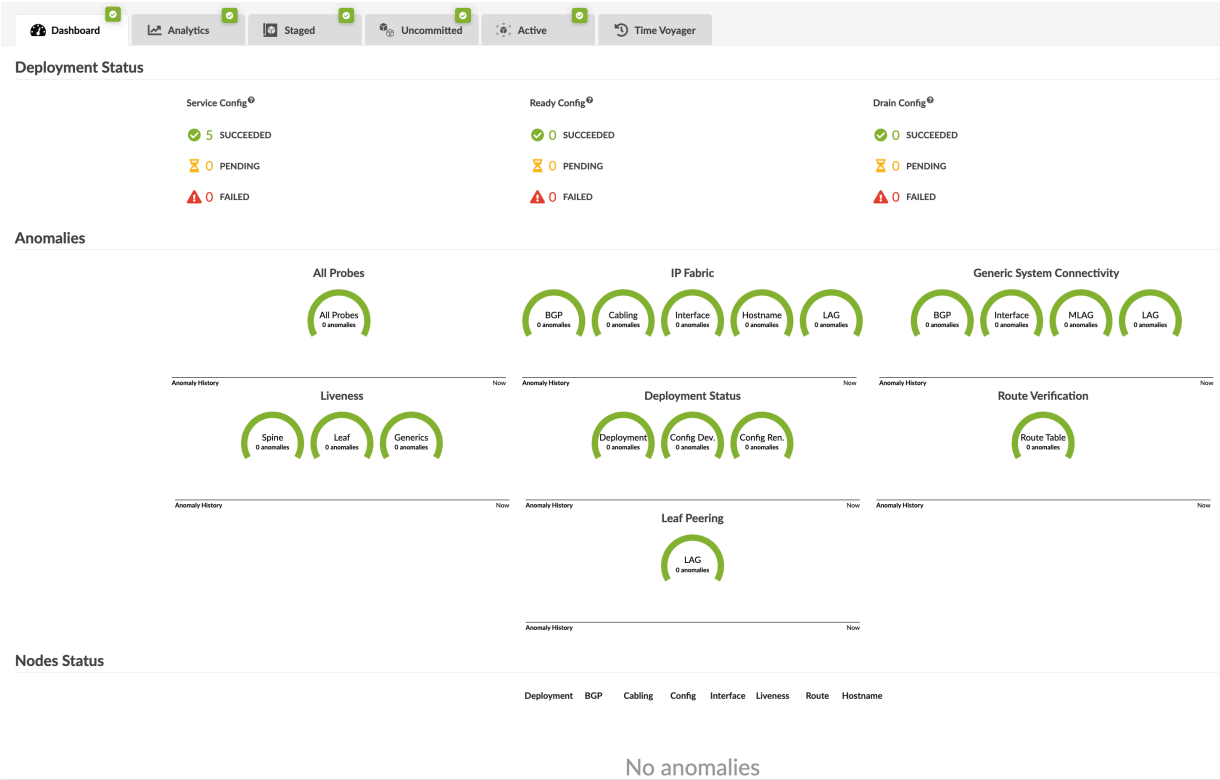
rack-based-blueprint-1d6794e9
Datacenter

| | |
|-------------------|--------------------------------------|
| Structure: | 2 spines, 3 leafs, 3 generic systems |
| Analytics | |
| Deployment Status | 5 |
| Service Anomalies | 0 |
| Probe Anomalies | 0 |
| Root Causes: | 0 |

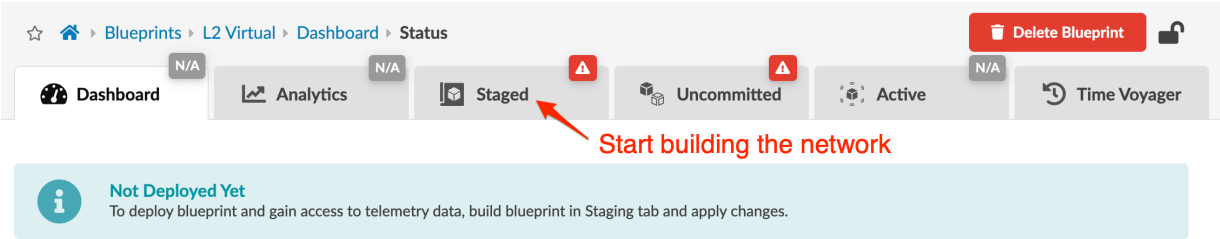
Version 61 Last modified 6 days ago

User: admin

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, ready config (previously called 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 702](#) 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 277 your network in the staging area.



Assign Physical Resources (Datacenter)

IN THIS SECTION

- Update Physical Resource Assignments | 278
- Reset Physical Resource Group Overrides | 279

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 illustrates the resource assignment workflow. The top navigation bar shows the 'Staged' tab selected. The sidebar on the left has 'Physical' selected. The main topology view shows a network diagram with nodes like 'spine1', 'spine2', and various racks. The right-hand 'Build' panel is the focus, showing a list of resources. Red arrows and numbers 1-6 indicate the steps: 1. Click 'Staged' in the top bar. 2. Click 'Physical' in the sidebar. 3. Click 'Build' in the right panel. 4. Click 'Resources' in the right panel. 5. Click the red status indicator for 'ASNs - Spines'. 6. Click the 'Update assignments' button. A 'Manage resource pools' button is also visible. A note at the bottom 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). Since you'd not be using a resource pool to assign from, the **No pools assigned** message remains in the **Build** panel. (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 table view to indicate that they have been retained (but the build section shows that no resources are assigned). Situations like this can (but do not always) result in build errors. Examples of where we want resources to persist include:

- Particular time voyager rollbacks (rack removal/addition and so on).
- ["Revert" on page 692](#) operations.
- Using the **Update Stated Cabling Map from LLDP** feature.

If you don't need to re-use the same resources, reset the resource groups by clicking the **Reset resource group overrides** button (shown in the overview image above). Then you can unallocate resources, and allocate new ones, as applicable.

Assign Device Profiles

1. From the "blueprint" on page 274, navigate to **Staged > Physical > Build > Device Profiles**.

1. From the "blueprint" on page 274, navigate to **Staged > Physical > Build > Device Profiles**.

2. Click a red status indicator, then click the **Change interface maps assignment** button (looks like an edit button). You assign device profiles by assigning interface maps.

3. Select the appropriate interface map from the drop-down list for each node. Or, to assign the same interface map to multiple nodes, 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

2. Click a red status indicator, then click the **Change interface maps assignment** button (looks like an edit button). You assign device profiles by assigning interface maps.
3. Select the appropriate interface map from the drop-down list for each node. Or, to assign the same interface map to multiple nodes, 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

| <input type="checkbox"/> | 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.

Configlets (Datacenter Blueprint)

IN THIS SECTION

- [Import Configlet | 283](#)
- [Edit / Delete Configlet \(Blueprint\) | 285](#)
- [Assign Configlet | 286](#)

From the blueprint, navigate to **Staged > Catalog > Configlets** to go to blueprint configlets. You can import, edit, and delete configlets from the blueprint catalog.

Click configlet name to see preview

| Name | Condition | Actions |
|---------------------------|------------------------|---------|
| Set_speed_1000_for_spine1 | hostname in ["spine1"] | |
| Set_speed_1000_for_leaf1 | hostname in ["leaf1"] | |



NOTE: If an improperly-configured configlet causes Apstra deployment errors (when the device rejects the command), a **service config deployment** failure occurs. See ["Anomalies \(Service\)" on page 613](#) for troubleshooting.



NOTE: If an improperly-configured configlet causes the disruption of connectivity between the device and Apstra controller, the device deployment state remains in PENDING forever and will never time out and fail.

For example, if a configlet with misconfigured routing engine firewall filter entry blocks the NETCONF port (tcp 830), the Junos offbox agent can't 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 45](#) that you want to import is in the design (global) catalog (Design > Configlets).
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 becomes invalidated.

Import Configlet from Global Catalog ✕

Configlet ^{*}

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. For example, you can apply configlets based on node tags (such as for all generic systems tagged as firewalls) in addition to applying them on link tags. This allows for more user-friendly definition of the configlet application scope, where instead of listing all interfaces, you can list a tag.
6. Click **Import Configlet** to stage the configlet and return to the table view.

Edit / Delete Configlet (Blueprint)

IN THIS SECTION

- [Edit Configlet Scope | 285](#)
- [Edit Configlet Generators | 286](#)
- [Delete Configlet | 286](#)

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 table view.

Edit Configlet Generators

You can't change configlet generators (template text, negation template text, filename, as applicable) directly in blueprints. If an existing configlet is no longer relevant, you can delete it and import a new or revised one. If you're changing a configlet in a blueprint catalog because of a configuration deviation, see also the Configlets and Config Deviation section.

1. ["Edit" on page 51](#) or create a ["configlet" on page 50](#) in the design (global) catalog.
2. Delete the configlet from the blueprint catalog. (See section below.)
3. ["Import" on page 283](#) the configlet into the blueprint catalog from the design (global) catalog.
4. ["Commit" on page 692](#) the changes.

Delete 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 table view.

Assign Configlet

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 283](#) 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 (Datacenter)

IN THIS SECTION

- [2D Topology View | 288](#)
- [3D Topology View | 289](#)
- [Neighbors Selection View | 291](#)
- [Links Selection View | 292](#)
- [Virtual Network Endpoints | 292](#)

Before you push your changes to the active blueprint you can view progressive changes in the staged 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.

Many node and link operations are performed from the **Topology** view. See ["Nodes" on page 293](#) and ["Links" on page 338](#) for more information.

You can view topologies in 2D or 3D view, and selections within topologies as neighbors, links, or virtual network endpoints, as applicable.

2D Topology View

From the blueprint, navigate to **Staged > Physical > Topology** to go to the 2D topology view.

The screenshot shows the 2D Topology View interface. At the top, there is a navigation bar with tabs: Dashboard, Analytics, Staged, Uncommitted, Active, and Time Voyager. Below this is a sub-navigation bar with tabs: Physical, Virtual, Policies, Catalog, Tasks, and Connectivity Templates. The Physical tab is selected. Below the sub-navigation bar, there are two dropdown menus: Nodes: All and Links: All. Below these are tabs for Topology, Nodes, Links, Racks, and Pods. The Topology tab is selected. Below the tabs, there is a Layer dropdown menu set to Uncommitted Changes. Below the Layer dropdown, there are two checkboxes: Expand Nodes? (unchecked) and Show Links? (checked). Below the checkboxes, there are two dropdown menus: Selected Rack (set to All) and Selected Node (set to All). Below these are two checkboxes: Has Uncommitted Changes (checked) and Topology Label (set to Name). Below the checkboxes, there is a topology diagram showing nodes and links. The nodes are labeled: spine1, spine2, leaf1, leaf2, leaf3, rack1-server1, switch1-server1, switch2-server1, switch3-server1, and rtr_leaf1_leaf2. The links are labeled: evpn_mlag_001 and evpn_single_001. Red arrows point to the Staged tab (1), the Physical tab (2), the Topology tab (3), the Selected Rack dropdown, the Selected Node dropdown, the Show Links? checkbox, and the leaf2 node. A red arrow also points to the leaf2 node with the text 'Hover over node or link to see more info'.

1. Staged

2. Physical

3. Topology

Select from drop-down lists or ...

... click node or link to go to selection topology

Hover over node or link to see more info

- To make topology elements larger, click the **Expand Nodes** check box.
- To display the links between elements, click the **Show Links** check box.
- To display a different layer, select the layer from the **Layer** drop-down list. **Uncommitted Changes** is an example of one of the layers you could display. The nodes with uncommitted changes are shown in yellow. The changes that apply to this layer are specific to the nodes themselves, such as ASN, loopback IP addresses and deploy modes. It doesn't apply to such changes as adding routing zones, virtual networks or connectivity templates on those nodes.
- To display additional information (node name, hostname, role, link, tags, as applicable), hover over a node or link.
- To display a different label (name, hostname, S/N), select a different label from the **Topology Label** drop-down list.

- To display a specific rack topology, click the rack element or select the rack from the **Selected Rack** drop-down list.
- To display a specific node topology, click the node element in the topology or select the node from the **Selected Node** drop-down list.

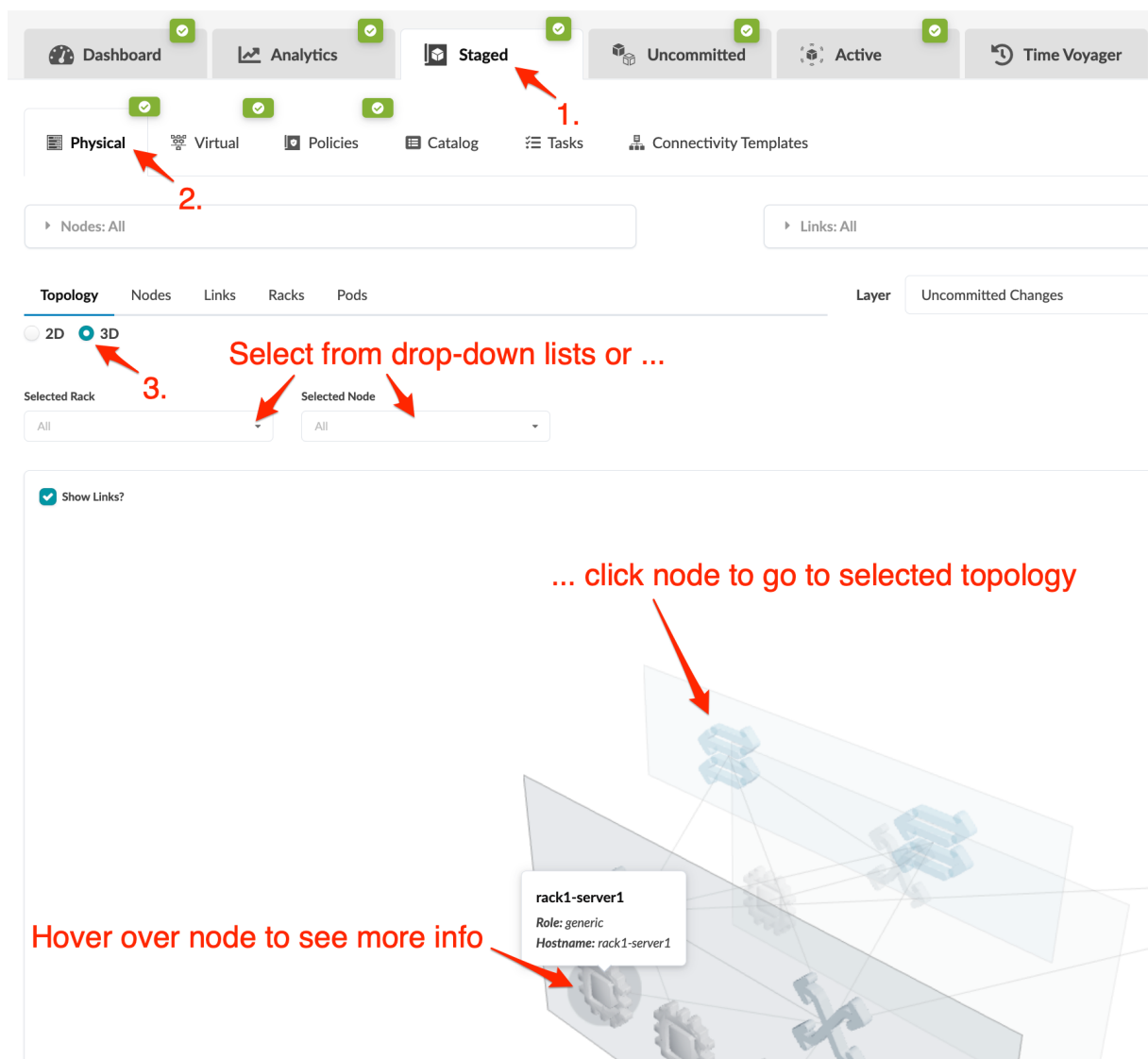
3D Topology View



NOTE: This feature is classified as a Juniper Apstra Technology Preview feature. These features are "as is" and are 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 1191](#) page or contact ["JuniperSupport" on page 824](#).

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 display additional information (node name, hostname, role, as applicable) hover over a node.
- To display rack topology (in 2D), click the rack element or selecting the rack from the **Selected Rack** drop-down list.
- To display node topology (in 2D), click the node element or select the node from the **Selected Node** drop-down list.

Neighbors Selection View

To see the neighbors view of a selection, click **Neighbors**.

Selected Rack: evpn_es1_001

Selected Node: leaf2 (Leaf)

Topology Label: Name

Neighbors | Links

Select check boxes to see available operations

☒ Show Aggregate Links ☐ Show Unused Ports

leaf2

xe-0/0/2

xe-0/0/0

xe-0/0/1

xe-0/0/3

xe-0/0/4

Interface

Label: n/a

Applied CTs: vn_endpoints_red_vxlan_39_v4_one_ep_vlan_tagged, vn_endpoints_blue_vxlan_34_v4_one_ep_vlan_tagged, vn_endpoints_red_vxlan_32_v4_1_vlan_tagged, vn_endpoints_red_302_evpn_es1_001_le_v4_vlan_tagged, vn_endpoints_blue_300_evpn_es1_001_le_v4_vlan_tagged, vn_endpoints_blue_vxlan_31_v4_1_vlan_tagged

n/a switch2-server1

eth2 rtr_leaf1_leaf2

Hover over node for details

- To display aggregate links, click the **Show Aggregate Links** check box.
- To display unused ports, click the **Show Unused Ports** check box.
- To display a different label (name, hostname, S/N), select a different label from the **Topology Label** drop-down list (right side).
- To display a particular neighbor type (all neighbors, generic, leaf, spine, and so on) select it from the **Show** drop-down list.
- To display available operations for a selected node or interface select the check box(es).
- To see details, hover over a node. As of Apstra version 4.1.2 hovering over a generic system shows applied connectivity templates. (Prior versions don't include these details on generics.)

Links Selection View

To see the links view of a selection, click **Links**.

Dashboard

Analytics

Staged

Uncommitted

Active

Time Voyager

Physical

Virtual

Policies

Catalog

Tasks

Connectivity Templates

Nodes: All

Links: All

Topology

Nodes

Links

Racks

Pods

2D

3D

Selected Rack

Selected Node

Topology Label

rack_1_001

rack_1_001_leaf1 (Leaf)

Name

Neighbors

Links

1-4 of 4

Page Size: 25

Filter selected by

all

selected only

unselected only

| | Name | Role | Tags | Speed | Port Channel ID | Endpoint 1 | | | | Endpoint 2 | | | | Actions |
|--------------------------|--|----------------|------|-------|-----------------|------------------|------|-----------|----------|------------------|------|-----------|----------|---------------------|
| | | | | | | Name | Role | Interface | Lag Mode | Name | Role | Interface | Lag Mode | |
| <input type="checkbox"/> | rack_1_001_leaf1<->rack_1_001_leaf2[1] | Leaf Peer Link | | 10G | 2 | rack_1_001_leaf1 | Leaf | swp3 | N/A | rack_1_001_leaf2 | Leaf | swp3 | N/A | <div>>></div> |

Virtual Network Endpoints

To see the virtual network endpoints of a selection, click **Virtual Networks Endpoints**.

Nodes: All

Links: All

Topology

Nodes

Links

Racks

Pods

2D

3D

Selected Rack

Selected Node

Topology Label

I2_virtual_001

I2_virtual_001_sys001 (Generic System)

Name

Neighbors

Links

Virtual Networks Endpoints

Query: All

Page Size: 25

| Virtual Network | Tag Type | Leaf(s) | Port Channel ID | Interface Name(s) |
|-----------------|----------|---------|-----------------|-------------------|
|-----------------|----------|---------|-----------------|-------------------|

Nodes (Datacenter)

IN THIS SECTION

- [Assign Device \(Datacenter\) | 295](#)
- [Unassign Device \(Datacenter\) | 300](#)
- [Set Deploy Mode \(Datacenter\) | 304](#)
- [Generic Systems vs. External Generic Systems | 305](#)
- [Add Generic System | 306](#)
- [Add External Generic System | 314](#)
- [Add Access Switch | 319](#)
- [Update Node Tags | 324](#)
- [Update Port Channel ID Range | 327](#)
- [Edit Hostname \(Datacenter\) | 330](#)
- [Edit Generic System Name | 332](#)
- [Edit Device Properties \(Datacenter\) | 333](#)
- [View Node's Static Routes | 334](#)
- [Delete Node | 335](#)

From the blueprint, navigate to **Staged > Physical > Nodes** to go to the **Nodes** view.

The screenshot shows the 'Nodes' view in a network management interface. The top navigation bar includes 'Dashboard', 'Analytics', 'Staged', 'Uncommitted', and 'Active'. The 'Staged' tab is selected, and the 'Physical' sub-tab is also selected. Below the navigation bar, there are filters for 'Nodes: All' and 'Links: All'. The main view is divided into tabs: 'Topology', 'Nodes', 'Links', 'Racks', 'Pods', and 'Planes'. The 'Nodes' tab is active. A 'Layer' dropdown is set to 'Uncommitted Changes'. Below the tabs, there are filters for 'Selected Plane', 'Selected Pod', 'Selected Rack', and 'Topology Label'. A red arrow points to the 'Selected Plane' dropdown with the text 'For 5-stage topologies'. Below the filters, there are three red arrows pointing to the 'Add external generic systems', 'Edit server names and hostnames', and 'Edit port channel id min max' buttons. A red arrow points to the 'Columns (9/17)' dropdown menu with the text 'Select what to display in table'. The dropdown menu is open, showing options: 'Tags', 'Role', 'External?', 'Pod', and 'Rack'. The table below shows a list of nodes with columns: 'Name', 'Tags', 'Role', 'External?', 'Deploy M', 'Hostname', 'ASN', and 'Loopback IPv4'. The table has 3 rows of data.

For 5-stage topologies

Add external generic systems

Edit server names and hostnames

Edit port channel id min max

Select what to display in table

| Name | Tags | Role | External? | Deploy M | Hostname | ASN | Loopback IPv4 |
|---------|------|------------|-----------|----------|----------|-------|---------------|
| sspine1 | | Superspine | N/A | Deploy | sspine1 | 64512 | 10.0.0.0/32 |
| sspine2 | | Superspine | N/A | Deploy | sspine2 | 64512 | 10.0.0.1/32 |
| spine1 | | Spine | N/A | Deploy | spine1 | 64513 | 10.0.0.2/32 |

- You can view nodes in the 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).

Many node operations are performed from the **Topology** view, and some can also be performed directly in the **Nodes** view. See the following sections for more information.

Assign Device (Datacenter)

IN THIS SECTION

- [Device Assignment Overview | 295](#)
- [Assign Device\(s\) \(from Devices Build Panel\) | 295](#)
- [Assign One Device \(from Devices Build Panel\) | 298](#)
- [Assign One System ID \(from Selection Panel\) | 299](#)

Device Assignment Overview

Before devices can be assigned to a blueprint, they must have interface maps assigned to them (from the Device Profiles tab). 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.

It's common to have a committed blueprint without any deployed devices. You can deploy devices as required, in batches, one by one, or all in one go. If you want to assign devices without deploying them, set the deploy mode to **Ready**, which puts devices in the **In Service Ready** state. This configuration is called **Ready Config** (previously known as Discovery 2 Config).



NOTE: When resetting system IDs (serial number) Discovery 1 configuration is re-applied. Before physically uninstalling the agent, it is good practice to fully erase the device configuration and uninstall the device agent.

Assign Device(s) (from Devices Build Panel)



NOTE: You can also use `apstra-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 the 'Staged' tab in the network management interface. The main area displays a topology diagram with nodes like 'sys001', 'spine1', 'spine2', and racks containing leaf nodes. A sidebar on the right shows the 'Assigned System IDs' panel, which lists nodes and their corresponding system IDs. Red arrows and numbers 1-6 indicate the steps to change system ID assignments:

1. Click the 'Staged' tab.
2. Click the 'Physical' tab.
3. Click the 'Build' button.
4. Click the 'Assigned System IDs' panel.
5. Click the 'Change System ID assignments' button (pencil icon).
6. Change System ID assignments in the list.

The 'Assigned System IDs' panel shows a table with 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 the device (Devices > Managed Devices).)

Assign Systems

Query: All

1-8 of 8

| Name | Role | Hostname | System ID | Deploy Mode |
|------------------|-------|----------|--|---|
| spine1 | Spine | spine-1 | 525400C50088 (10.28.11.13) - some_location ✖  | <input checked="" type="radio"/> Deploy <input type="radio"/> Ready <input type="radio"/> Drain <input type="radio"/> Undeploy |
| spine2 | Spine | spine-2 | Select... | <input type="radio"/> Deploy <input type="radio"/> Ready <input type="radio"/> Drain <input type="radio"/> Undeploy |
| rack_1_001_leaf1 | Leaf | leaf-1-1 | <div style="border: 1px solid #ccc; padding: 2px;"> 505400B41BBC (10.28.11.11) - some_location 505400B41BBC (10.28.11.11) - some_location 5054004B6645 (10.28.11.9) - some_location </div> | <input type="radio"/> Deploy <input type="radio"/> Ready <input type="radio"/> Drain <input type="radio"/> Undeploy |
| rack_1_001_leaf2 | Leaf | leaf-1-2 | Select... | <input type="radio"/> Deploy <input type="radio"/> Ready <input type="radio"/> Drain <input type="radio"/> Undeploy |

Update Assignments

- When you select a system ID, the deploy mode changes to **Deploy** by default. If you don't want to deploy the device yet, change the deploy mode here. When you're ready to deploy the device, return here to set the deploy mode back to **Deploy**.
- 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.
- "Commit" on page 692 changes to the blueprint to deploy device(s) into the active fabric. Device state changes to **In Service Active** and the configuration is called **Service Config**.

As soon as you deploy a device, anomalies may appear on the dashboard. When telemetry data is verified against Intent, anomalies resolve themselves. This can take a fair amount of time in some cases, especially for BGP sessions and advertising routes.

Deploying devices can have different implications depending on the device vendor. Juniper Junos devices, for example, 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 can't 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. If this anomaly is undesirable, we recommend that you remove the interface from the virtual network until the interface is up.

After deploying devices a new running config is collected, called the **Golden Config**, which serves as Intent. Running configuration is continuously collected and compared against this Golden config. When a deployment fails, Golden Config is unset. Protocol related anomalies like BGP or LLDP are only raised if devices at both ends are deployed.

Assign One Device (from Devices Build Panel)

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**.

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. 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

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' panel on the right shows 'rack_2_001_leaf1' with role 'Leaf' and group label 'leaf'. Below this, a 'Device' panel shows 'Deploy Mode' with 'not set' and 'S/N' with 'Not assigned'. A red arrow points to the 'Edit' button for 'S/N' with the text 'Click to assign ID'. Another red arrow points to the 'Selected Node' dropdown with the text 'Alternative method for accessing device details'. The topology view shows a rack with four switches (swp1, swp2, swp3, swp4) connected to a spine (spine1, spine2) and a system (rack_2_001_sys001, sys001). A 'Show Unused Ports' checkbox is visible.



NOTE: You can also select a node name in the **Selected Nodes** drop-down list (left-middle) to go to these device details.

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 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.

Assign One System ID (from Selection Panel)

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.)

- Click the **Device** tab in the right panel (if it's not already selected).

The screenshot shows the 'Staged' view of 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 main area displays a table of devices with columns: Name, Tags, Role, External?, Deploy Mode, Device Profile, Hostname, ASN, and Loopback IPv4. The table lists several devices, including 'leaf1', 'leaf2', 'leaf3', 'leaf_pair001_001_1', 'rack1-server1', 'spine1', 'spine2', 'switch1-server1', and 'switch2-server1'. A red arrow points to the 'leaf1' device in the table, with the text 'Click a node name...' and '... to see its details'. Another red arrow points to the 'Edit' button in the right panel, with the text 'Edit'. The right panel shows the 'Device' tab selected, with fields for 'Name', 'Deploy Mode', 'S/N', and 'Device Info'.

| 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 |

- Enter a different S/N. (You can also access configuration files from here: rendered, incremental, pristine).
- Click the **Save** button to stage the changes.

Unassign Device (Datacenter)

IN THIS SECTION

- Unassign Device (from Device Selection Panel) | 300
- Unassign Device(s) (from Devices Build Panel) | 303

Unassign Device (from Device Selection Panel)

- From the blueprint, navigate to **Staged > Physical > Topology**, and click the device to be removed.

Dashboard Analytics Staged Uncommitted Active

Physical Virtual Policies Catalog Tasks Connectivity Templates

Nodes: All Links: All

Topology Nodes Links Racks Pods Layer System IDs Assignments

2D 3D Assigned Not Assigned

Selected Rack: All Selected Node: All Topology Label: Name

Expand Nodes? Show Links?

Diagram showing nodes: rtr_leaf1_leaf2, spine1, spine2, evpn_mlag_001 (leaf1, leaf2), evpn_single_001 (leaf3). A red arrow points to leaf3 with the text: Click to go to device details

- In the **Device** panel (on the right), click the **Edit** button for deploy mode, and change it to **Undeploy**, then click the **Save** button.

Dashboard Analytics Staged Uncommitted Active Time Voyager

Physical Virtual Policies Catalog Tasks Connectivity Templates

Nodes: All Links: All Selection Build

Topology Nodes Links Racks Pods

2D 3D

Selected Rack: evpn_single_001 Selected Node: leaf3 (Leaf)

Neighbors Links

Diagram showing connections between leaf3 and spine1, spine2, and switch3-server1.

Device Details for leaf3:

- Role: Leaf
- Group label: evpn-single
- Deploy Mode: deploy (Edit button highlighted)
- S/N: 50540087C33A
- Device Info:
 - Management IP: 10.28.173.14
 - OS: EOS 4.27.4M
 - Operation Mode: FULL CONTROL



NOTE: Another way to get to the **Device** selection panel from the Topology view (or Nodes, Links, Racks, or Pods view) is to click the **Devices** tab in the **Build** panel (on the right), click the status indicator for **Assigned System IDs** (to display the nodes and assigned system IDs), then click the node name that you want to unassign.

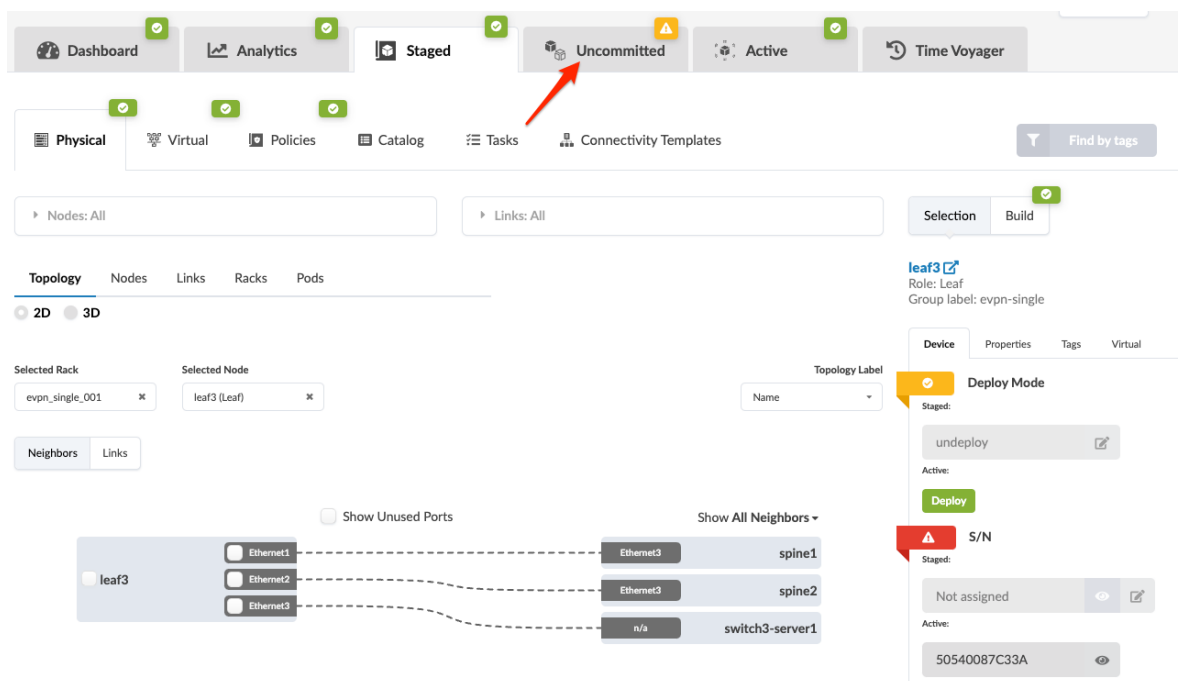
3. In the **S/N** section, click the **Edit** button.

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 main area displays the Topology view with a 2D/3D toggle. The selected rack is 'evpn_single_001' and the selected node is 'leaf3 (Leaf)'. The topology diagram shows 'leaf3' connected to 'spine1' and 'spine2'. The right panel shows the 'Device' configuration for 'leaf3'. The 'Deploy Mode' section has 'Staged' and 'Active' tabs. The 'S/N' section shows the system ID '50540087C33A' with a red square icon next to it. Red arrows point from the text 'Visibility of staged and active' to the 'Staged' and 'Active' sections of the device configuration panel.

4. Click the red square in the **S/N** section to unassign the system ID.

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 main area displays the Topology view with a 2D/3D toggle. The selected rack is 'evpn_single_001' and the selected node is 'leaf3 (Leaf)'. The topology diagram shows 'leaf3' connected to 'spine1' and 'spine2'. The right panel shows the 'Device' configuration for 'leaf3'. The 'Deploy Mode' section has 'Staged' and 'Active' tabs. The 'S/N' section shows the system ID '50540087C33A' with a red square icon next to it. A red arrow points from the text 'Click the red square in the S/N section to unassign the system ID.' to the red square icon.

5. Click **Uncommitted** and "**commit**" on page 692 changes to the blueprint to remove the device from the fabric.

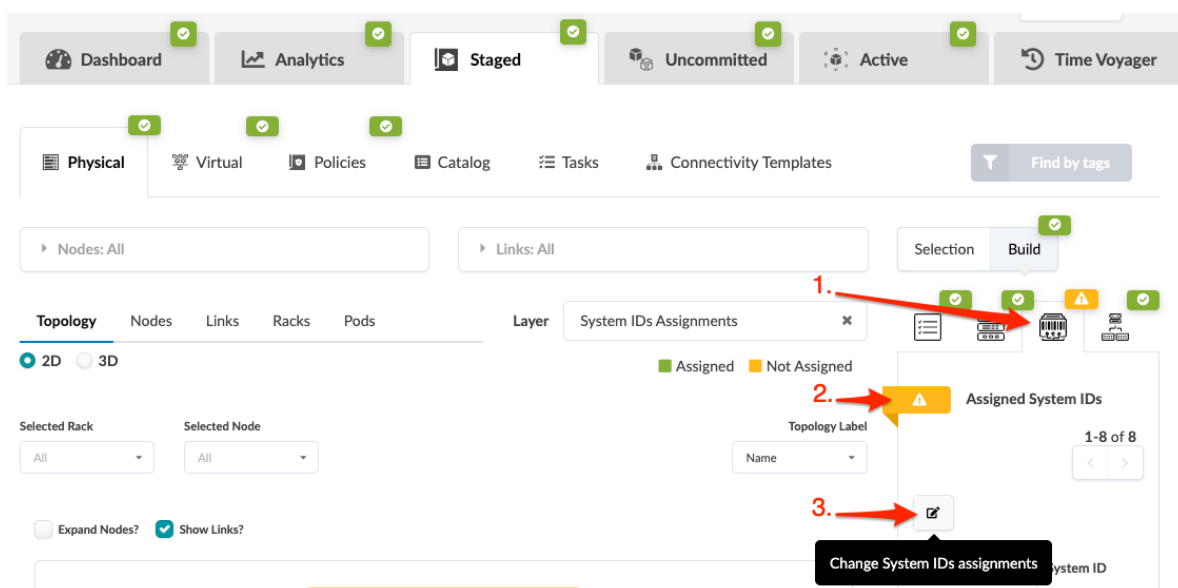


The device is still under Apstra management. It's ready and available to be assigned to any blueprint.

To remove the device completely from Apstra management, ["remove the device from Managed Devices" on page 80.](#)

Unassign Device(s) (from Devices Build Panel)

1. From the blueprint, navigate to **Staged > Physical > Topology**, click the **Devices** tab in the **Build** panel (on the right), then click the status indicator for **Assigned System IDs** to display the nodes and assigned system IDs.



2. Click the **Change System IDs assignments** button (below Assigned System IDs), then in the dialog that opens click the **Remove assignment** button for the device to remove. The deploy mode is automatically unselected.

Assign Systems

| Query: All | | | | 1-8 of 8 | |
|------------|-------|----------|--|---|--------------------------------|
| Name | Role | Hostname | System ID | Deploy Mode | |
| spine1 | Spine | spine-1 | 525400528138 (10.29.12.13) - some_location | <input checked="" type="radio"/> Deploy | <input type="radio"/> Ready |
| | | | | <input type="radio"/> Drain | <input type="radio"/> Undeploy |

3. Click **Update Assignments** (bottom-right in dialog) to stage the change and return to the **Topology** view.
4. Click **Uncommitted** and **"commit"** on page 692 changes to the blueprint to remove the device from the fabric.

The device is still under Apstra management. It's ready and available to be assigned to any blueprint.

To remove the device completely from Apstra management, ["remove the device from Managed Devices" on page 80](#).

Set Deploy Mode (Datacenter)

IN THIS SECTION

- [Set Deploy Mode \(from Build Panel\) | 304](#)
- [Set Deploy Mode \(from Selection Panel\) | 305](#)
- [Set Deploy Mode \(from Nodes View\) | 305](#)

Set Deploy Mode (from Build Panel)

1. From the blueprint, navigate to **Staged > Physical**, then in the **Build** panel (on the right) click the **Devices** tab.
2. If you don't see the nodes list, click the status indicator for **Assigned System IDs**.
3. Click a node name to see device details.
4. 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 Ready configuration (hostnames, interface descriptions, port speeds / breakouts) (previously called Discovery 2 config). Changing from deploy to ready removes service configuration.
- Drain - Takes a device out of service for maintenance. For more information, see ["Draining Device Traffic" on page 82](#).
- 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.

5. Click the **Save** button to stage the change.

When you're ready to activate changes, ["Commit" on page 692](#) them from the **Uncommitted** tab.

Set Deploy Mode (from Selection Panel)

1. From the blueprint, navigate to **Staged > Physical**.
2. Either from the **Topology** view or the **Nodes** view, select a node.
3. If it's not already selected, click the **Device** tab in the **Selection** panel (on the right).
4. Click the **Edit** button for **Deploy Mode** and select a deploy mode.
5. Click the **Save** button to stage the new deploy mode.

Set Deploy Mode (from Nodes View)

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.)
2. Click the **Set Deploy Mode** button (fourth of five buttons above the nodes list) and select a 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.

Generic Systems vs. External Generic Systems

When to use a generic system and when to use an external generic system:

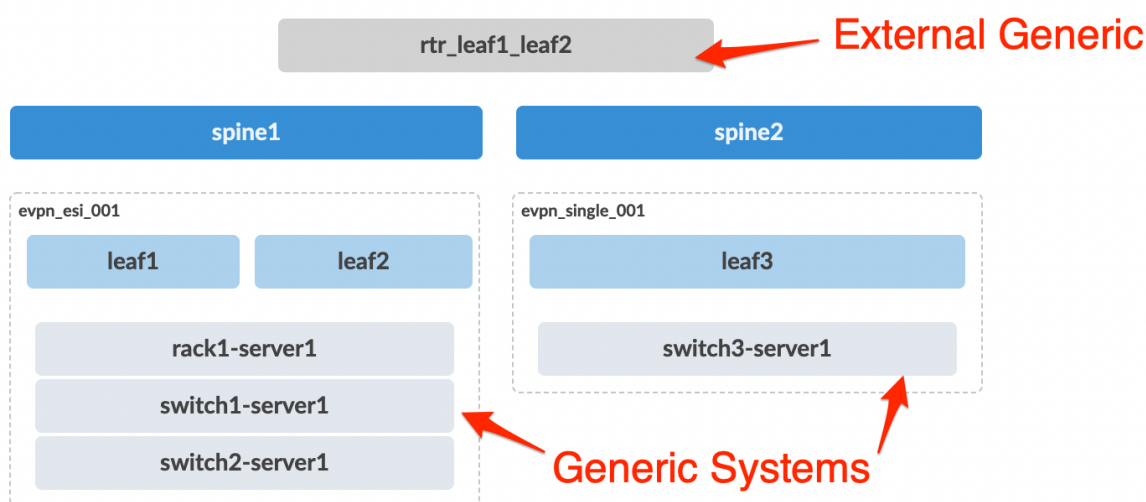
Generic System

- For attaching compute/storage
- Can only be connected to a single rack
- Appears in the topology as part of a rack

External Generic System

- For middleware devices, such as firewalls, load balancers, external routers and so on*
- Can be connected to multiple racks
- Appears in the topology outside of racks for easier identification

* In many cases, middleware boxes only connect to a single *border leaf pair* in a rack, but configuring it as an **external generic** system allows it to be visually separated outside of the rack. However, if there is a requirement such as connecting to an external router (MX) via BGP and you want to provide rack redundancy, then you would use an external generic system to allow this multi-rack connectivity.



Add Generic System

IN THIS SECTION

- [Add Generic System \(from Topology View\) | 307](#)
- [Copy Existing Generic System \(from Topology View\) | 311](#)

When you want to connect your Apstra-managed fabric to a system that's not managed in the Apstra environment, you use generic systems and external generic systems. These systems can be external routers, firewalls, or whatever else you want; you specify their roles with tags. If the system is part of 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.

As of Apstra version 4.1.2, you can add generic systems to access switches.



NOTE: You can also create generic systems during the Design phase before creating your blueprint. For more information, see ["Rack Types" on page 23](#).

Add Generic System (from Topology View)

1. From the blueprint, navigate to **Staged > Physical > Topology** and select the node to add the new generic system to.

The screenshot shows the 'Staged > Physical > Topology' view. The network diagram includes a central spine structure with leaf nodes. A red arrow points to the 'leaf1' node, which is highlighted. A tooltip for 'leaf1' displays the following information:

- Role: Leaf
- Hostname: leaf1
- Tags: n/a

The interface also features a 'Selection' button and a message box indicating 'Nothing selected yet'.

2. Select the node check box to see the operations available for that node (and that you have permissions for).

The screenshot shows the network management interface with the following components:

- Top Navigation Bar:** Dashboard, Analytics, Staged, Uncommitted, Active.
- Left Sidebar:** Physical, Virtual, Policies, Catalog, Tasks, Connectivity Templates.
- Filters:** Nodes: All, Links: All.
- View Options:** Topology, Nodes, Links, Racks, Pods. 2D (selected), 3D.
- Selected Rack:** evpn_mlag_001
- Selected Node:** leaf1 (Leaf)
- Topology Label:** Name
- Neighbors/Links:** Neighbors (selected), Links.
- Context Menu (for leaf1):**
 - Add generic system
 - Copy existing generic
 - Add links to generic system
 - Add external generic
 - Add links to external generic
 - Add leaf peer links
 - Add access switch
 - Update node tags
- Diagram:** A network diagram showing connections between nodes. A red arrow points to the 'Add generic system' option in the context menu.



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).

3. Click **Add generic system** and enter a unique label and (optional) hostname.

Create New System

Create Links

✕

Label

Hostname

Choose a representation for a new device *

☐ None
 ☒ Apstra Logical Device
 ☐ Apstra Logical Device With an Interface Map

☐ Show whole catalog

Select...

Port Channel ID min

Port Channel ID max

0

0

System tags

Select...

Next

- Select the representation for the new node (none, logical device, or logical device with interface map), then select the appropriate logical device or interface map from the drop-down list, as applicable. (Logical devices allow you to define port roles.)
- Enter the port channel ID min and max.
- Enter tags (optional) to identify the role(s) of the new generic system, then click **Next**.

✓ Create New System

Create Links

✕

Select devices and their interfaces to create a link:

Leaf: leaf1

Device profile: Cisco NXOSv

1 2 3 4 5 6 7 8 9

Leaf: leaf2

Device profile: Cisco NXOSv

1 2 3 4 5 6 7 8 9

AOS-4x10-1

4 x 10 Gbps

Leaf • Access

Link tags

Select...

Add Link →

Links

1-4 of 4

| Type | Speed | Leaf | | Generic | | Tags | Actions |
|----------|-------|-------|-------------|-----------------|-----------|------|---------|
| | | Name | Interface | Name | Interface | | |
| Existing | 10G | leaf1 | Ethernet1/5 | rack1-server1 | N/A | | |
| Existing | 10G | leaf1 | Ethernet1/6 | switch1-server1 | N/A | | |
| Existing | 10G | leaf2 | Ethernet1/5 | rack1-server1 | N/A | | |
| Existing | 10G | leaf2 | Ethernet1/6 | switch2-server1 | N/A | | |

Please add at least one link

Back

Create

7. Select an available port and transformation. The gray **Add Link** button turns green.

✓ Create New System Create Links

Select devices and their interfaces to create a link:

Leaf: leaf1
Device profile: Cisco NXOSv

Port #8 Tr. #1 (10 Gbps, default) **Ethernet1/8**

Port #8 Tr. #2 (1 Gbps) Ethernet1/8

Leaf: leaf2
Device profile: Cisco NXOSv

AOS-4x10-1

4 x 10 Gbps
Leaf • Access

Add Link →

Links

1-4 of 4

| Type | Speed | Leaf | Generic | Tags | Actions |
|----------|-------|-------|-------------|-----------------|-----------|
| | | Name | Interface | Name | Interface |
| Existing | 10G | leaf1 | Ethernet1/5 | rack1-server1 | N/A |
| Existing | 10G | leaf1 | Ethernet1/6 | switch1-server1 | N/A |
| Existing | 10G | leaf2 | Ethernet1/5 | rack1-server1 | N/A |
| Existing | 10G | leaf2 | Ethernet1/6 | switch2-server1 | N/A |

Please add at least one link

Back Create

8. Click **Add Link**. The link is added to the link table.

✓ Create New System ✓ Create Links

Select devices and their interfaces to create a link:

Leaf: leaf1
Device profile: Cisco NXOSv

Port #8 Tr. #1 (10 Gbps, default) Ethernet1/8

Port #8 Tr. #2 (1 Gbps) Ethernet1/8

Leaf: leaf2
Device profile: Cisco NXOSv

AOS-4x10-1

4 x 10 Gbps
Leaf • Access

Add Link →

Links (1 will be added)

1-5 of 5

| Type | Speed | Leaf | Generic | Tags | Actions |
|----------|-------|-------|-------------|-----------------|-----------|
| | | Name | Interface | Name | Interface |
| New | 10G | leaf1 | Ethernet1/8 | gen-sys | N/A |
| Existing | 10G | leaf1 | Ethernet1/5 | rack1-server1 | N/A |
| Existing | 10G | leaf1 | Ethernet1/6 | switch1-server1 | N/A |
| Existing | 10G | leaf2 | Ethernet1/5 | rack1-server1 | N/A |
| Existing | 10G | leaf2 | Ethernet1/6 | switch2-server1 | N/A |

New link is added

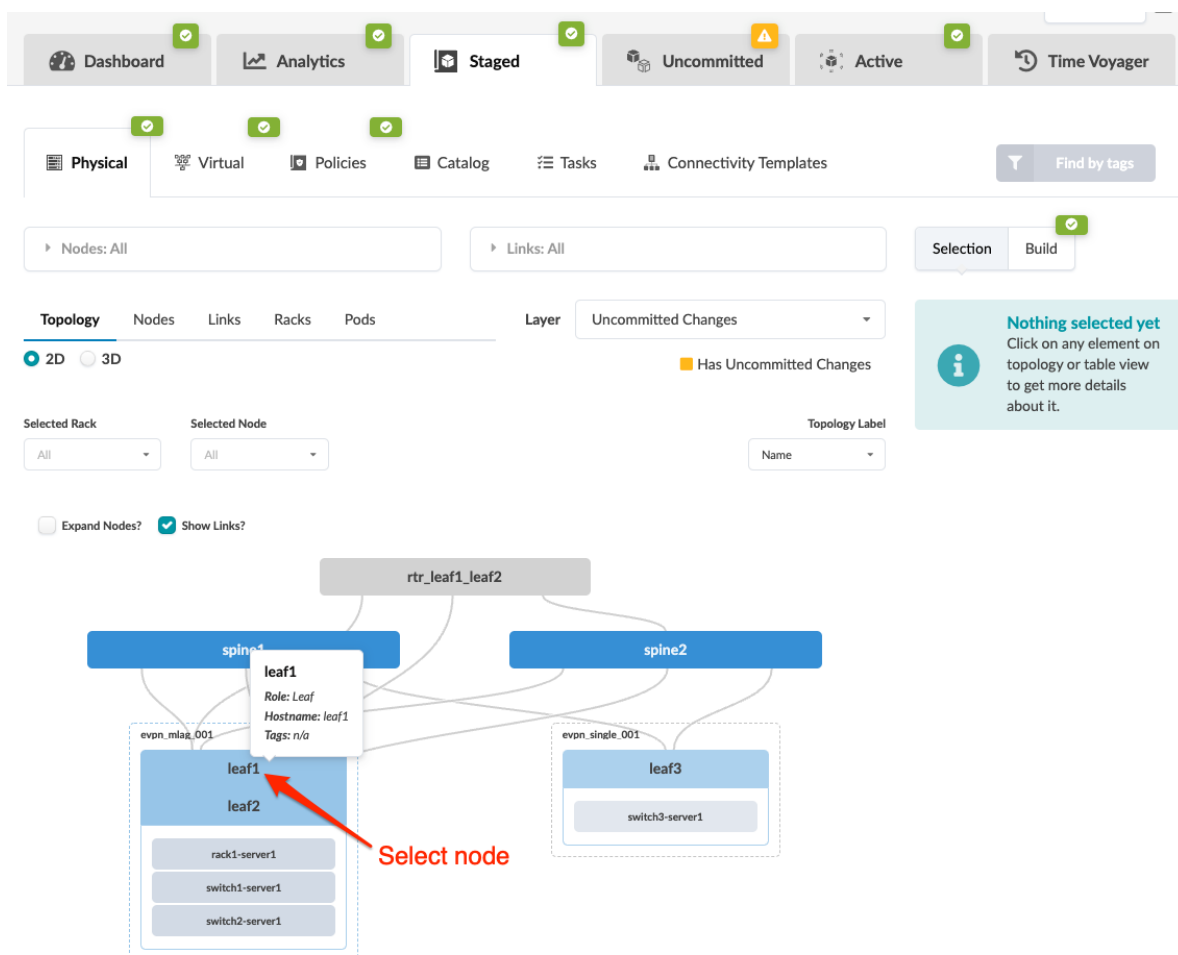
Back Create

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 System (from Topology View)

1. From the blueprint, navigate to **Staged > Physical > Topology** and select a node that has the generic system that you want to copy.



2. Select the node check box to see the operations available for that node (and that you have permissions for).

The screenshot displays the network management interface. 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. The main area shows filters for Nodes and Links, and tabs for Topology, Nodes, Links, Racks, and Pods. The 2D view is selected. Below the tabs, there are dropdowns for Selected Rack (evpn_mlag_001), Selected Node (leaf1 (Leaf)), and Topology Label (Name). A Neighbors/Links toggle is present. A context menu for node 'leaf1' is open, listing options: Add generic system, Copy existing generic, Add links to generic system, Add external generic, Add links to external generic, Add leaf peer links, Add access switch, and Update node tags. A red arrow points to 'Copy existing generic'. The background shows a 2D topology diagram with nodes and links.

Select node for available operations

leaf1

- Add generic system
- Copy existing generic
- Add links to generic system
- Add external generic
- Add links to external generic
- Add leaf peer links
- Add access switch
- Update node tags

Show Unused Ports

Show All Neighbors

| | |
|-------------|-----------------|
| n/a | rack1-server1 |
| Ethernet1/4 | leaf2 |
| Ethernet1/3 | leaf2 |
| Ethernet1/1 | spine1 |
| Ethernet1/1 | spine2 |
| n/a | switch1-server1 |
| eth1 | rtr_leaf1_leaf2 |



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).

- Click **Copy existing generic** and select the generic system from the drop-down list. The link table appear.

Copy Existing Generic



Select existing generic

switch1-server1



Logical device: AOS-1x10-1

Device profile: N/A

Interface map: N/A

System group label: single-server-1

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 | leaf1 | Select Interface | No LAG | New Generic | Select Interface | |

Submit

4. Click **Select interface** to go to ports.

Select interface

Leaf: leaf1

Device profile: Cisco NXOSv



Port #8 Tr. #1 (10 Gbps, default)

Ethernet1/8

Port #8 Tr. #2 (1 Gbps)

Ethernet1/8

Cancel

Confirm



5. Select a port and transformation, then click **Confirm** to return to the dialog.

Copy Existing Generic



Select existing generic

switch1-server1



Logical device: AOS-1x10-1

Device profile: N/A

Interface map: N/A

System group label: single-server-1

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 | leaf1 | Ethernet1/8 | No LAG | New Generic | Select Interface | |

Submit

6. Click **Submit** 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 System

IN THIS SECTION

- [Add External Generic System \(from Topology View\) | 314](#)
- [Add External Generic System \(from Nodes View\) | 319](#)

When you want to connect your Apstra-managed fabric to a system that's not managed in the Apstra environment, you use generic systems and external generic systems. These systems can be external routers, firewalls, or whatever else you want; you specify their roles with tags. If the system is part of 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 System (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 displays a network management dashboard with a top navigation bar containing tabs: Dashboard, Analytics, Staged, Uncommitted, Active, and Time Voyager. Below this is a secondary navigation bar with tabs: Physical, Virtual, Policies, Catalog, Tasks, and Connectivity Templates. A search bar labeled 'Find by tags' is on the right. The main interface features filters for 'Nodes: All' and 'Links: All', a 'Selection' button, and a 'Build' button. A 'Topology' section includes tabs for Nodes, Links, Racks, and Pods, a 'Layer' dropdown set to 'Uncommitted Changes', and a '2D' radio button. A 'Selected Rack' dropdown is set to 'All', and a 'Selected Node' dropdown is set to 'All'. A 'Topology Label' dropdown is set to 'Name'. A 'Has Uncommitted Changes' indicator is present. A 'Nothing selected yet' message box is on the right. The main area shows a network topology diagram with nodes: spine1, spine2, rtr_leaf1_leaf2, leaf1, leaf2, leaf3, rack1-server1, switch1-server1, switch2-server1, and switch3-server1. A red arrow points to leaf1 with the text 'Select node'. A tooltip for leaf1 shows: Role: Leaf, Hostname: leaf1, Tags: n/a.

2. Select the node check box to see the operations available for that node (and that you have permissions for).

The screenshot shows the network management interface with the following components:

- Top Navigation Bar:** Dashboard, Analytics, Staged, Uncommitted, Active.
- Left Sidebar:** Physical, Virtual, Policies, Catalog, Tasks, Connectivity Templates.
- Filters:** Nodes: All, Links: All.
- View Options:** Topology (selected), Nodes, Links, Racks, Pods. 2D (selected), 3D.
- Selection Fields:** Selected Rack: evpn_mlag_001, Selected Node: leaf1 (Leaf), Topology Label: Name.
- Buttons:** Neighbors, Links.
- Context Menu (for leaf1):**
 - Add generic system
 - Copy existing generic
 - Add links to generic system
 - Add external generic
 - Add links to external generic
 - Add leaf peer links
 - Add access switch
 - Update node tags
- Diagram:** A network diagram showing connections between various nodes. A red arrow points to the 'Add external generic' option in the context menu.



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).

3. Click **Add external generic** and enter a unique label and (optional) hostname.

Create New System

Create Links

✕

Label

Hostname

Choose a representation for a new device *

☐ None
☒ Apstra Logical Device
☐ Apstra Logical Device With an Interface Map

☐ Show whole catalog

Select...

System tags

Select...

Next

- Select the representation for the new node (none, logical device, or logical device with interface map), then select it from the drop-down list as applicable. (Selecting a logical device allows you to define port roles.)
- Enter tags (optional) to identify the role(s) of the new external generic system, then click **Next**.

✓ Create New System

Create Links

✕

Select devices and their interfaces to create a link:

Leaf: leaf1

Device profile: Cisco NXOSv

1

2

3

4

5

6

7

8

9

Leaf: leaf2

Device profile: Cisco NXOSv

1

2

3

4

5

6

7

8

9

slider-7x10-1

7 x 10 Gbps

Superspine • Spine • Leaf • Access • Peer • Generic

Link tags

Select...

Add Link →

Links

1-2 of 2

| Type | Speed | Leaf | | External Generic | | Tags | Actions |
|----------|-------|-------|-------------|------------------|-----------|------|---------|
| | | Name | Interface | Name | Interface | | |
| Existing | 10G | leaf1 | Ethernet1/7 | rtr_leaf1_leaf2 | eth1 | | |
| Existing | 10G | leaf2 | Ethernet1/7 | rtr_leaf1_leaf2 | eth2 | | |

Back

Create

- Select an available port and transformation. The gray **Add Link** button turns green.

Create New System

Create Links

Select devices and their interfaces to create a link:

Leaf: leaf1
Device profile: Cisco NXOSv

123456789

Ethernet1/8

Ethernet1/8

Leaf: leaf2
Device profile: Cisco NXOSv

123456789

7 x 10 Gbps

Superspine • Spine • Leaf • Access • Peer • Generic

Link tags

Select...

Add Link →

Links

1-2 of 2

| Type | Speed | Leaf | External Generic | Tags | Actions |
|----------|-------|------------------|---------------------|------|---------|
| | | NameInterface | NameInterface | | |
| Existing | 10G | leaf1Ethernet1/7 | rtr_leaf1_leaf2eth1 | | |
| Existing | 10G | leaf2Ethernet1/7 | rtr_leaf1_leaf2eth2 | | |

7. Click **Add Link**. The link is added to the link table.

✔

Create New System

✔

Create Links

Select devices and their interfaces to create a link:

Leaf: leaf1

Device profile: Cisco NXOSv

1

2

3

4

5

6

7

8

9

Port #8 Tr. #1 (10 Gbps, default)

Ethernet1/8

Port #8 Tr. #2 (1 Gbps)

Ethernet1/8

Leaf: leaf2

Device profile: Cisco NXOSv

1

2

3

4

5

6

7

8

9

slicer-7x10-1

7 x 10 Gbps

Superspine • Spine • Leaf • Access • Peer • Generic

Link tags

Select...

Links (1 will be added)

1-3 of 3

< >

New

Existing

Existing

| Type | Speed | Leaf | | External Generic | | Tags | Actions |
|----------|-------|-------|-------------|------------------|-----------|------|---------|
| | | Name | Interface | Name | Interface | | |
| New | 10G | leaf1 | Ethernet1/8 | N/A | N/A | | |
| Existing | 10G | leaf1 | Ethernet1/7 | rtr_leaf1_leaf2 | eth1 | | |
| Existing | 10G | leaf2 | Ethernet1/7 | rtr_leaf1_leaf2 | eth2 | | |

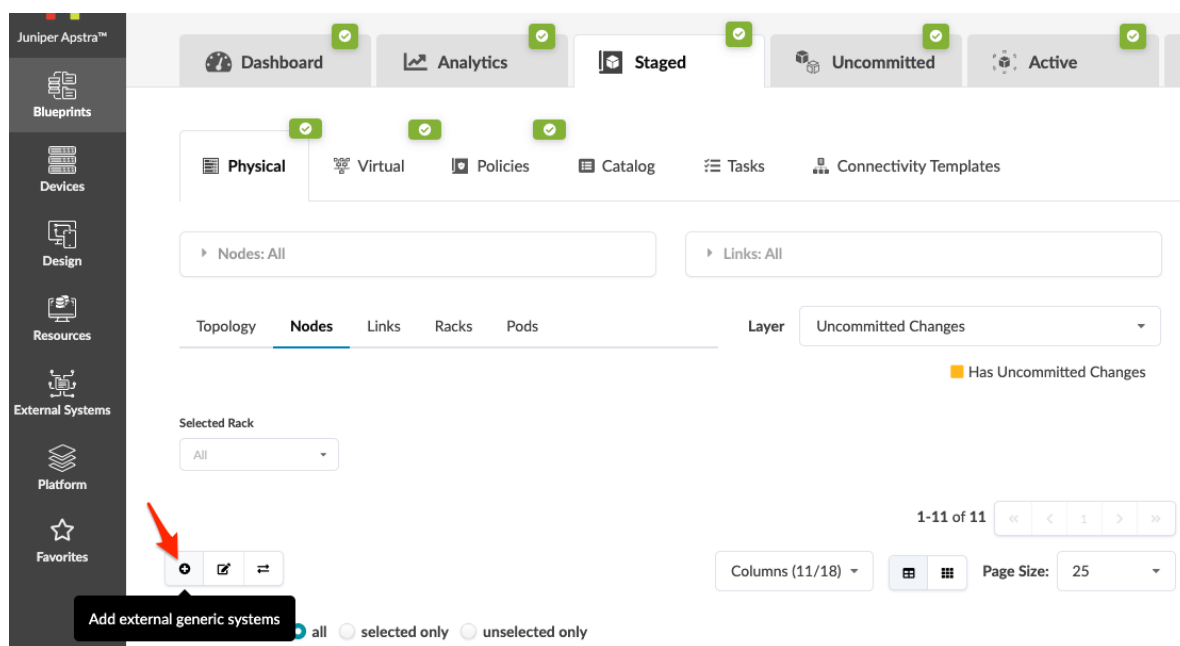
Add Link →

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 System (from Nodes View)

1. From the blueprint, navigate to **Staged > Physical > Nodes** and click the **Add external generic systems** button to open its dialog.

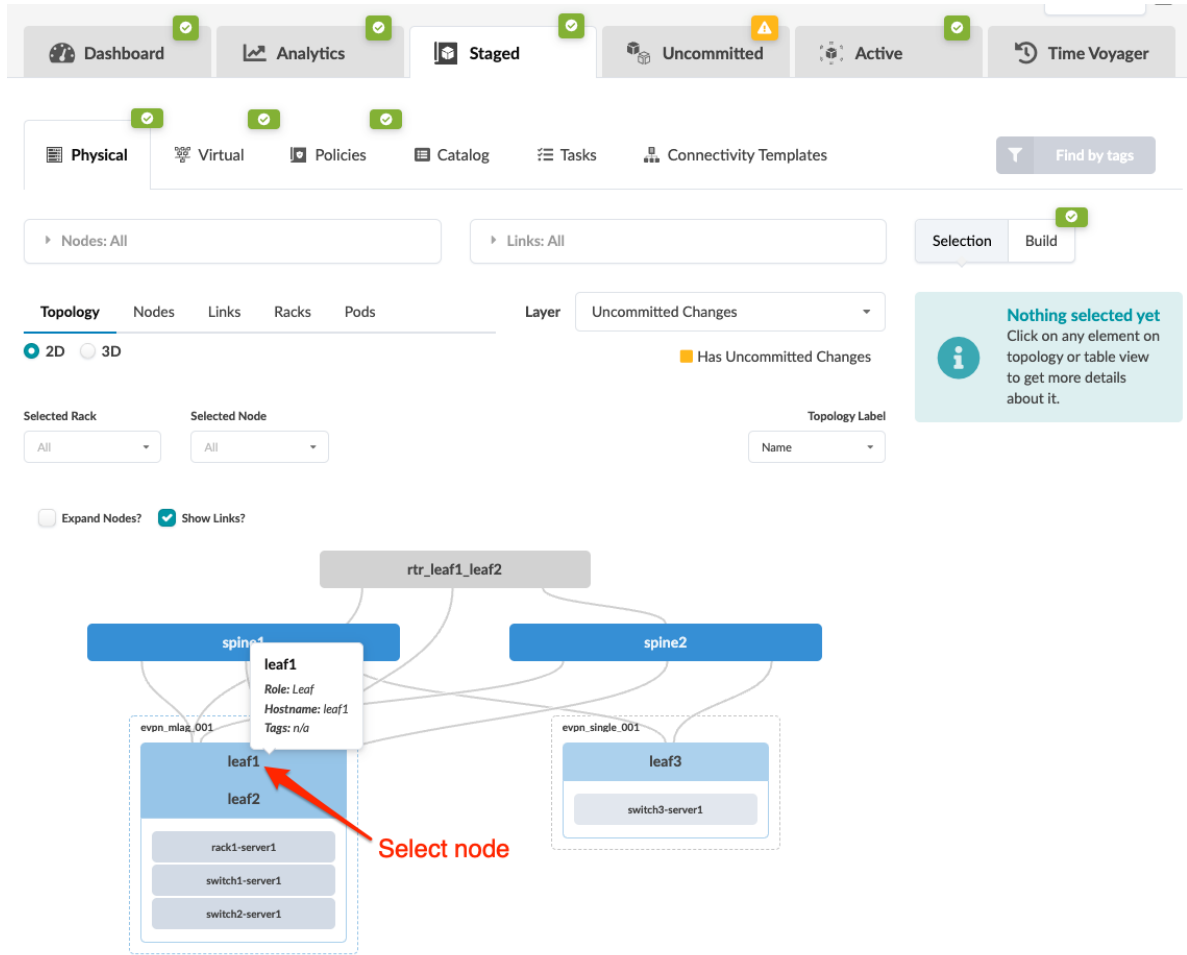


2. Enter a hostname, and if you want to be able to define port roles select a logical device from the drop-down list.
3. Enter tags (optional) to identify the role(s) of the new external generic system.
4. Click **Create** to stage the changes and return to the **Nodes** view.

Add Access Switch

For more information about access switches, see Rack Types.

1. From the blueprint, navigate to **Staged > Physical > Topology** and select the node to add an access switch to.



2. Select the node check box to see the operations available for that node (and that you have permissions for).

The screenshot shows the network management interface with the following components:

- Top Navigation Bar:** Dashboard, Analytics, Staged (active), Uncommitted, Active.
- Left Sidebar:** Physical (active), Virtual, Policies, Catalog, Tasks, Connectivity Templates.
- Filters:** Nodes: All, Links: All.
- View Options:** Topology (selected), Nodes, Links, Racks, Pods. 2D (selected), 3D.
- Selection Fields:** Selected Rack: evpn_mlag_001, Selected Node: leaf1 (Leaf), Topology Label: Name.
- Neighbors/Links Tab:** Neighbors (selected), Links.
- Context Menu for leaf1:**
 - Add generic system
 - Copy existing generic
 - Add links to generic system
 - Add external generic
 - Add links to external generic
 - Add leaf peer links
 - Add access switch (highlighted with a red arrow)
 - Update node tags
- Network Diagram:** Shows connections between leaf1 and various nodes: rack1-server1, leaf2, spine1, spine2, switch1-server1, and rtr_leaf1_leaf2. A 'Show Unused Ports' checkbox is present.



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).

3. Click **Add access switch** and enter a unique label and hostname.

Create New System

Create Links

×

Label

Hostname

Choose a representation for a new device *

☐ None
 ☐ Apstra Logical Device
 ☒ Apstra Logical Device With an Interface Map

☐ Show whole catalog

Select...

Port Channel ID min

Port Channel ID max

0

0

System tags

Select...

Next

- Select the appropriate interface map from the drop-down list.
- Enter the port channel ID min and max.
- Enter tags (optional) to identify the role(s) of the new access switch, then click **Next**.

✓ Create New System

Create Links

×

Select devices and their interfaces to create a link:

Leaf: leaf2

Device profile: Juniper vQFX

0 1 2 3 4 5 6 7 8 9 10 11

Leaf: leaf1

Device profile: Juniper vQFX

0 1 2 3 4 5 6 7 8 9 10 11

Access

Device profile: Juniper vQFX

0 1 2 3 4 5 6 7 8 9 10 11

Link tags

Select...

Links

1-3 of 3

| Type | Speed | Leaf | | Access Switch | | Tags | Actions |
|----------|-------|-------|-----------|---------------|-----------|------|---------|
| | | Name | Interface | Name | Interface | | |
| Existing | 10G | leaf2 | xe-0/0/4 | access1 | xe-0/0/2 | | |
| Existing | 10G | leaf1 | xe-0/0/1 | access1 | xe-0/0/0 | | |
| Existing | 10G | leaf1 | xe-0/0/0 | access2 | xe-0/0/0 | | |

Add Link →

Back

Create

- Select available ports and transformations, as applicable. The gray **Add Link** button turns green.

8. Click **Add Link**. The link is added to the link table.

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.

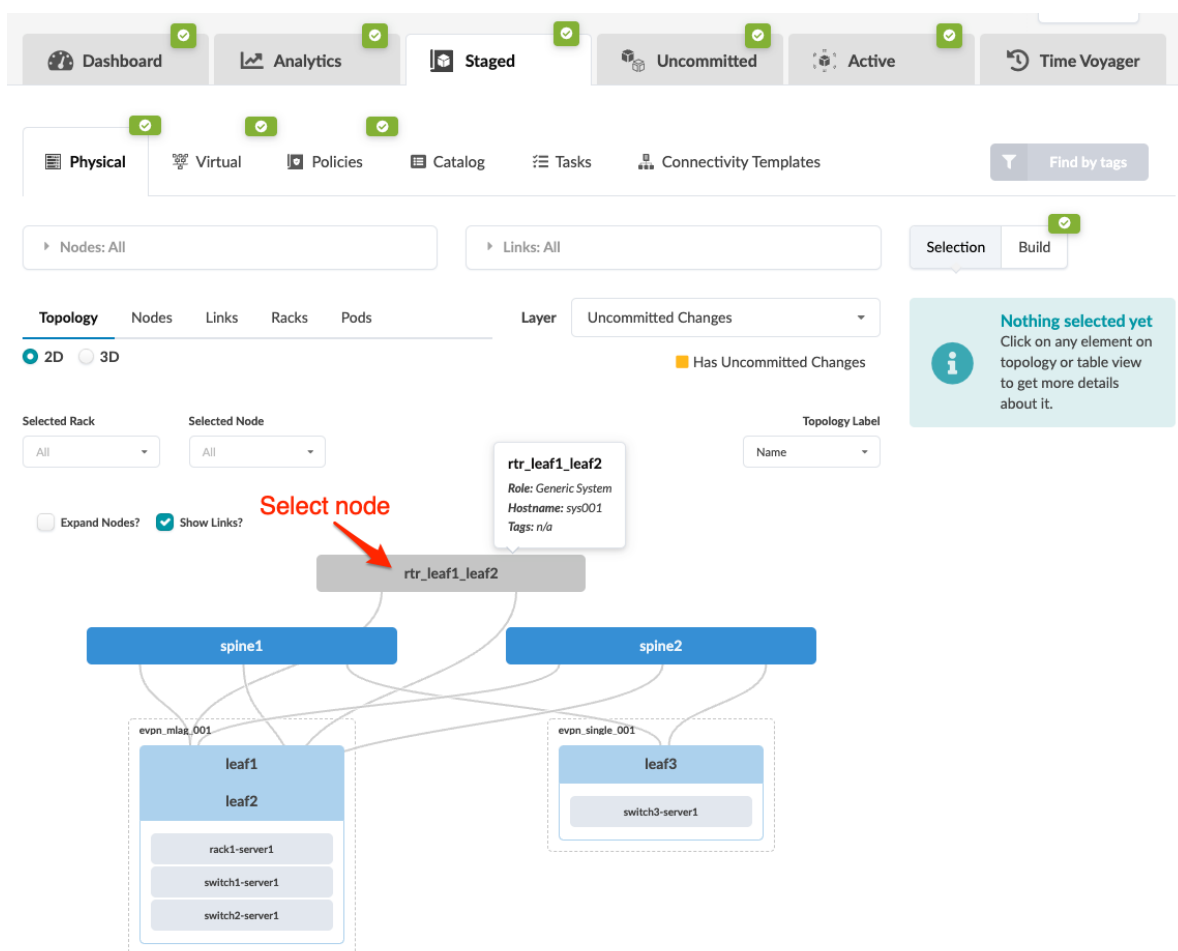
Update Node Tags

IN THIS SECTION

- [Update Node Tags \(One Node\) | 324](#)
- [Update Node Tags \(Multiple Nodes\) | 325](#)

Update Node Tags (One Node)

1. From the blueprint, navigate to **Staged > Physical > Topology** and select the node that needs updated tags.



2. Select the node check box to see the operations available for that node (and that you have permissions for).

The screenshot displays the network management interface. At the top, there are tabs for Dashboard, Analytics, Staged, Uncommitted, and Active. Below these are Physical, Virtual, Policies, Catalog, Tasks, and Connectivity Templates. The Physical tab is selected, showing a list of nodes and links. A red arrow points to the 'Neighbors' tab, which is highlighted. A context menu is open over the 'Neighbors' tab, showing options: 'Add links to leaf', 'Add links to spine', 'Update node tags', and 'Delete node'. The 'Update node tags' option is selected. The context menu also shows the selected node: 'rtr_leaf1_leaf2 (Gen 8 eric System)'. The main area shows a topology diagram with nodes 'leaf1' and 'leaf2' connected by links 'eth1' and 'eth2'. The 'Show All Neighbors' dropdown is visible.

Select node for available operations

3. Click **Update node tags** and update node tags as needed.
4. Click **Update** to update the tags and return to the **Selection** view.

When you're ready to activate your changes, commit them from the **Uncommitted** tab.

Update Node Tags (Multiple Nodes)

1. From the blueprint, navigate to **Staged > Physical > Nodes** and select one or more check boxes for the node(s) that need updated tags. The **Add/Remove Tags** button appears above the table.

1. Select one or more

2.

Add/Remove Tags

Filter selected by ☒ selected ☐ unselected only

| | Name | Tags | Role | External? | Deploy Mode | Device Profile | Hostname | ASN | Loopback IPv4 | Loopback IPv6 | Port Channel ID Range |
|-------------------------------------|--------------------------|------|----------------|-----------|--------------|----------------|---------------|--------------|-----------------|----------------------------|-----------------------|
| <input type="checkbox"/> | spine1 | | Spine | N/A | Deploy | Cisco NXOSv | spine1 | 64512 | 10.0.0.0/32 | fc01:a05:fab::128 | n/a |
| <input type="checkbox"/> | spine2 | | Spine | N/A | Deploy | Cisco NXOSv | spine2 | 64513 | 10.0.0.1/32 | fc01:a05:fab::1/128 | n/a |
| <input type="checkbox"/> | evpn_mlag_001_leaf_pair1 | | Leaf Pair | N/A | N/A | N/A | N/A | N/A | N/A | N/A | n/a |
| <input type="checkbox"/> | leaf1 | | Leaf | N/A | Deploy | Cisco NXOSv | leaf1 | 64514 | 10.0.0.2/32 | fc01:a05:fab::2/128 | n/a |
| <input type="checkbox"/> | leaf2 | | Leaf | N/A | Deploy | Cisco NXOSv | leaf2 | 64515 | 10.0.0.3/32 | fc01:a05:fab::3/128 | n/a |
| <input type="checkbox"/> | leaf3 | | Leaf | N/A | Deploy | Cisco NXOSv | leaf3 | 64516 | 10.0.0.4/32 | fc01:a05:fab::4/128 | n/a |
| <input type="checkbox"/> | rack1-server1 | | Generic System | No | Not assigned | Not assigned | rack1-server1 | Not assigned | Not assigned | Not assigned | 0-0 |
| <input checked="" type="checkbox"/> | rtr_leaf1_leaf2 | node | Generic System | Yes | Not assigned | Not assigned | sys001 | 65533 | 198.51.100.2/32 | fc01:a05:198:51:100::2/128 | 0-0 |

- Click the **Add/Remove Tags** button and update tags as needed. When you create new tags here they are added to the blueprint catalog.

Add/Remove Tags

Add Tags

Select...

Remove Tags

Select...

The following nodes will be affected

Query: All

Page Size: 25

| Name |
|-----------------|
| rtr_leaf1_leaf2 |

Add/Remove Tags

- Click **Add/Remove Tags** to stage the change and return to the **Nodes** view.

When you're ready to activate your changes, commit them from the **Uncommitted** tab.

Update Port Channel ID Range

IN THIS SECTION

- [Update Port Channel ID Range \(from Topology View\) | 327](#)
- [Update Port Channel ID Range \(from Nodes view\) | 329](#)

You can update the port channel ID range for generic systems either from the Topology view (as of Apstra version 4.1.2) or from the Nodes view.



CAUTION: Changing port channel range is an invasive operation and may lead to reassigning existing port channel IDs.

Update Port Channel ID Range (from Topology View)

You can update the port channel ID range from the **Topology** view as of Apstra version 4.1.2.

1. From the blueprint, navigate to **Staged > Physical > Topology** and select the generic system to update.

Dashboard Analytics Staged Uncommitted Active Time Voyager

Physical Virtual Policies Catalog Tasks Connectivity Templates Find by tags

Nodes: All Links: All Selection Build

Topology Nodes Links Racks Pods Layer Uncommitted Changes

2D 3D Has Uncommitted Changes

Selected Rack: All Selected Node: All Topology Label: Name

Expand Nodes? Show Links?

Select generic system

sys001

spine1 spine2

rack_1_001 rack_2_001

rack_1_001_sys001 rack_2_001_sys001

Nothing selected yet
Click on any element on topology or table view to get more details about it.

2. You can hover over the generic system to see the port channel ID range (and other details).

Topology Nodes Links Racks Pods

2D 3D

Selected Rack: rack_1_001 Selected Node: rack_1_001_sys001 (Generic System)

Neighbors Links Virtual Networks Endpoints

Hover over node to see details

rack_1_001_sys001

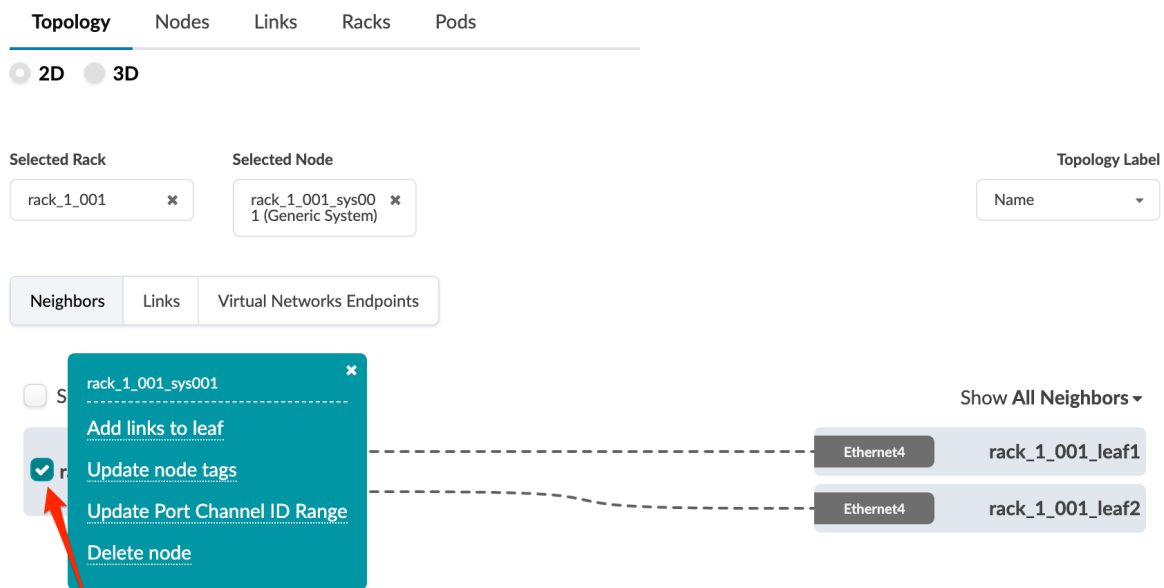
Name: rack_1_001_sys001
Tags: n/a
Role: Generic System
Hostname: rack-1-001-sys001
Port Channel ID Range: 0-0

Show All Neighbors

Ethernet4 rack_1_001_leaf1

Ethernet4 rack_1_001_leaf2

3. Select the generic system check box to see the operations available for that generic system (and that you have permissions for).

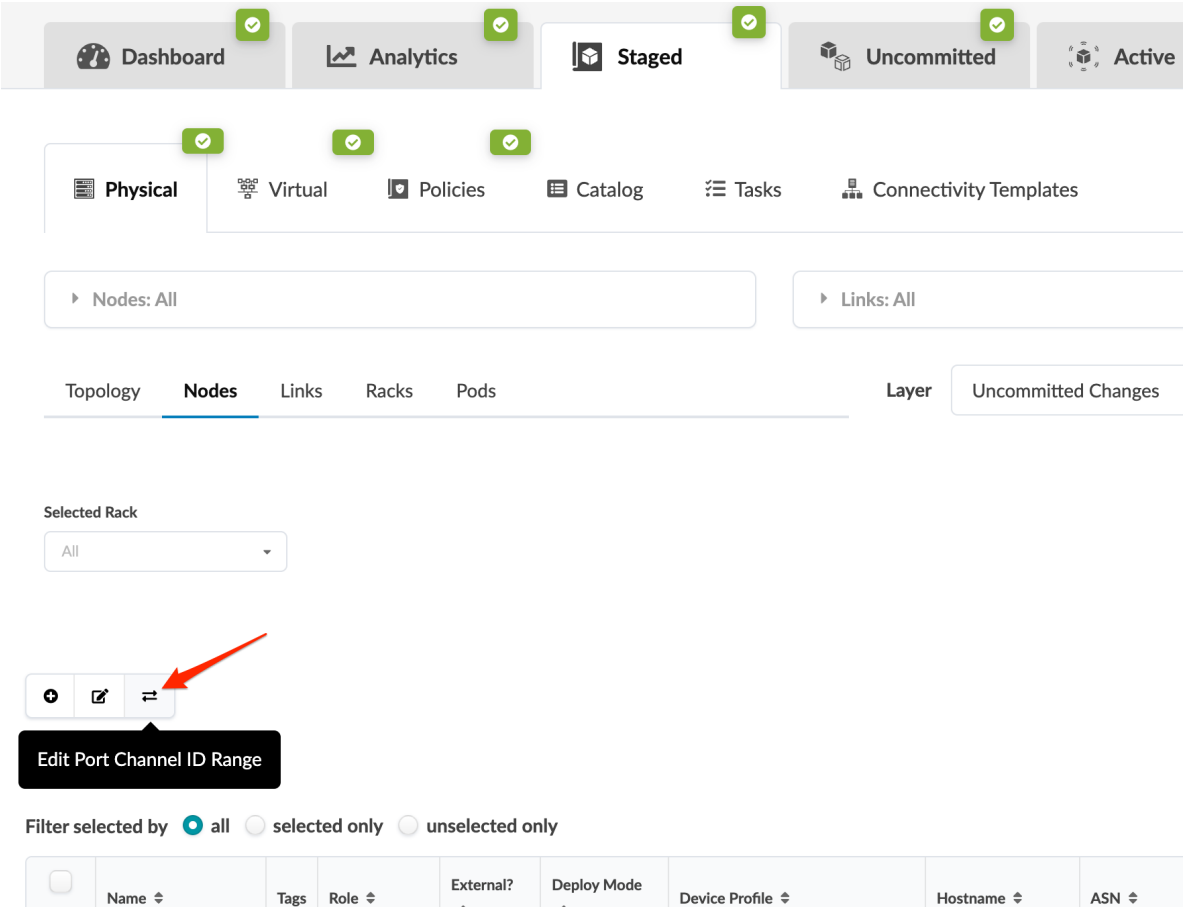


Select node for available operations

4. Click **Update Port Channel ID Range**, then edit the min and/or max values, as needed.
5. Click **Update** to stage your changes and return to the **Topology** view.

Update Port Channel ID Range (from Nodes view)

1. From the blueprint, navigate to **Staged > Physical > Nodes** and click the **Edit Port Channel ID Range** button (previously called **Edit port channel id min max**).



2. In the table of generic systems, edit the min and/or max port channel ID values, as needed.
3. Click **Update** to stage your changes and return to the **Nodes** view.

Edit Hostname (Datacenter)

IN THIS SECTION

- [Edit Hostname \(from Build Panel\) | 330](#)
- [Edit Hostname \(from Selection Panel\) | 331](#)
- [Edit Hostname \(from Nodes View\) | 331](#)

Edit Hostname (from Build Panel)

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.

- 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 692 them from the **Uncommitted** tab.

Edit Hostname (from Selection Panel)

- 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.)
- If it's not already selected, click the **Device** tab in the **Selection** panel (on the right). (You can also access the **Selection** panel from **Staged > Physical > Topology**.)

Click a node name... ... to see its details

Table View

Selection panel tabs: Device, Properties, Tags, Virtual

Selection panel fields: Name, Deploy Mode, S/N, Device Info

| 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 |

- Enter a different hostname. (You can also change deploy mode and system ID and access configuration files from here: rendered, incremental, pristine).
- Click the **Save** button to stage the changes.

Edit Hostname (from Nodes View)

You can edit multiple hostnames at the same time, fetch discovered LLDP data (hostnames), and update names based on hostnames, all from the same dialog.

- From the blueprint, navigate to **Staged > Physical > Nodes** and click the **Edit server names and hostnames** button (second of three buttons above the nodes view).

2. Make your changes.

- To change names, select a name and enter a different unique one.
- To fetch discovered LLDP data (hostnames), click its button.
- To update the names based on hostnames, click its button.

The screenshot shows the 'Edit Server Names and Hostnames' interface. At the top, there is a title bar with a close button (X). Below the title bar, there are two buttons: a refresh button (circular arrow) and a document icon. Red arrows point to these buttons with the text 'Fetch discovered LLDP data (hostnames)' and 'Update the names based on the hostnames' respectively. Below the buttons, there is a search bar with the text 'Query: All', a pagination indicator '1-4 of 4' with left and right arrows, and a 'Page Size: 25' dropdown menu. The main content is a table with three columns: 'Name', 'Hostname', and 'S/N'. Red arrows point to the 'Name' and 'Hostname' columns with the text 'Change names'. The table contains four rows of data, each with a server name, a matching hostname, and a 'Not assigned' status in the 'S/N' column. At the bottom right, there is a large blue 'Update' button.

| Name | Hostname | S/N |
|-----------------|-----------------|--------------|
| switch2-server1 | switch2-server1 | Not assigned |
| rack1-server1 | rack1-server1 | Not assigned |
| switch3-server1 | switch3-server1 | Not assigned |
| switch1-server1 | switch1-server1 | Not assigned |

3. Click **Update** to stage the changes and return to the nodes view.

Any associated link names do not automatically update to match the changed server names and/or hostnames. You can manually ["change the link names" on page 377](#) to match so when you are reviewing an updated cabling map the names align.

Edit Generic System Name

IN THIS SECTION

- Edit Generic System Name (from Nodes View) | 333

Edit Generic System Name (from Nodes View)

You can edit multiple server names and hostnames at the same time, fetch discovered LLDP data (hostnames), and update names based on hostnames, all from the same dialog.

1. From the blueprint, navigate to **Staged > Physical > Nodes** and click the **Edit generic system names and hostnames** button (second of three buttons above the nodes view). (In versions prior to 4.1.2 the button was called **Edit server names and hostnames**.)
2. Make your changes.
 - To change names, select a name and enter a different unique one.
 - To fetch discovered LLDP data (hostnames), click its button.
 - To update the names based on hostnames, click its button.

Edit Server Names and Hostnames ✕

↺ 📄

Fetch discovered LLDP data (hostnames)

Update the names based on the hostnames

Query: All
1-4 of 4
Page Size: 25

Change names

| Name ↕ | Hostname ↕ | S/N ↕ |
|-----------------|-----------------|--------------|
| switch2-server1 | switch2-server1 | Not assigned |
| rack1-server1 | rack1-server1 | Not assigned |
| switch3-server1 | switch3-server1 | Not assigned |
| switch1-server1 | switch1-server1 | Not assigned |

Update

3. Click **Update** to stage the changes and return to the nodes view.

Any associated link names do not automatically update to match the changed server names and/or hostnames. You can manually ["change the link names"](#) on page 377 to match so when you are reviewing an updated cabling map the names align.

Edit Device Properties (Datacenter)

You can change device properties such as name, interface map, ASN, and loopback IP, depending on the node chosen.

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, racks and access groups, as applicable.
2. Click the **Properties** tab in the right panel.

Click a node name... ... to see its properties

Card View

Edit

3. You can change device properties such as name (must be changed to a unique 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.

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.

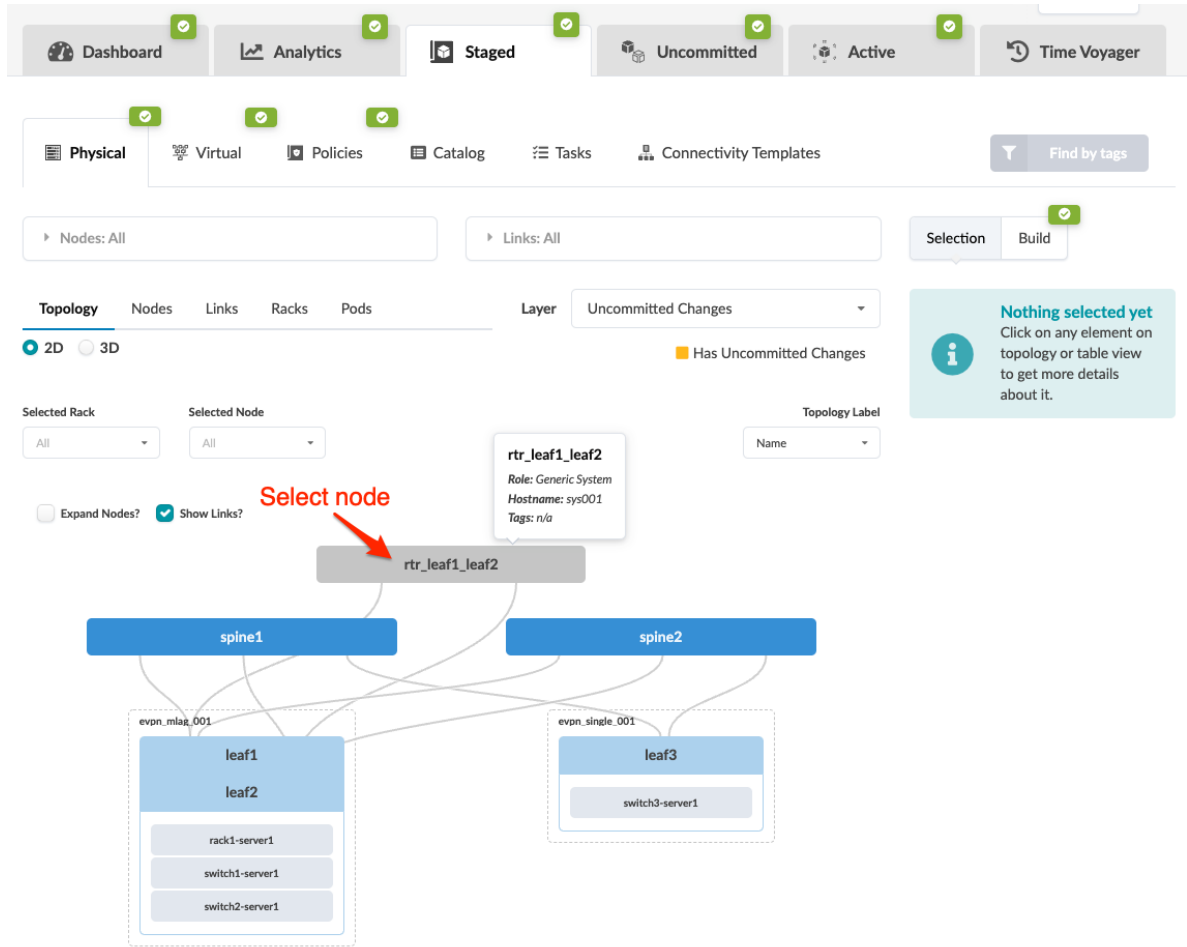
The screenshot shows the network management interface with the **Nodes** tab selected. The interface includes a top navigation bar with tabs like Dashboard, Analytics, Staged, Uncommitted, Active, and Time Voyager. Below this is a secondary navigation bar with Physical, Virtual, Policies, Catalog, Tasks, and Connectivity Templates. The main content area shows a list of nodes with columns for Name, Tags, Role, External?, Deploy Mode, Device Profile, Hostname, ASN, and Loopback IPv4. The node 'leaf1' is highlighted. To the right, a panel for 'leaf1' shows its role as 'Leaf' and group label 'evpn-mlag'. A link labeled 'Node's Static Routes' is visible in this panel. Red arrows indicate the steps: '1. Click a node name' pointing to 'leaf1' and '2. Access static routes' pointing to 'Node's 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.

Delete Node

1. From the blueprint, navigate to **Staged > Physical > Topology** and select the node to delete.



2. Select the check box to see the operations available for that node (and that you have permissions for).

Dashboard Analytics Staged Uncommitted Active

Physical Virtual Policies Catalog Tasks Connectivity Templates

Nodes: All Links: All

Topology Nodes Links Racks Pods

2D 3D

Selected Rack: All Selected Node: rtr_leaf1_leaf2 (Gen 8 eric System) Topology Label: Name

Select node for available operations

Neighbors Links

rtr_leaf1_leaf2

- Add links to leaf
- Add links to spine
- Update node tags
- Delete node

Show All Neighbors

eth1 Ethernet1/7 leaf1

eth2 Ethernet1/7 leaf2



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).

- Click **Delete node** to go to its dialog. All links towards the system will be deleted and connectivity templates will be unassigned for you.

Delete Node

Label: rtr_leaf1_leaf2
 Role: External generic
 Hostname: sys001
 Tags:



All links towards this system will be deleted and connectivity templates will be unassigned.



Delete

4. Click **Delete** to stage the deletion and return to the **Topology** view.

When you're ready to activate your changes, commit them from the **Uncommitted** tab.

Links (Datacenter)

IN THIS SECTION

- Add Links to Leaf | 340
- Add Links to Spine | 343
- Add Links to Generic System | 347
- Add Links to External Generic System | 352
- Add Leaf Peer Links | 357
- Form LAG | 361
- Break LAG | 364

- [Update LAG Mode | 366](#)
- [Update Link Tags | 369](#)
- [Update Link Speed | 374](#)
- [Update Link Properties | 377](#)
- [Delete Link \(Datacenter\) | 379](#)
- [Import / Export Cabling Map \(Datacenter\) | 384](#)
- [Edit Cabling Map \(Datacenter\) | 385](#)
- [Fetch Discovered LLDP Data \(Datacenter\) | 386](#)

From the blueprint, navigate to **Staged > Physical > Links** to go to the **Links** view.

1. Click **Staged**

2. Click **Physical**

3. Click **Links**

Fetch discovered LLDP data

Change link speeds

Edit cabling map

Export cabling map

Import cabling map

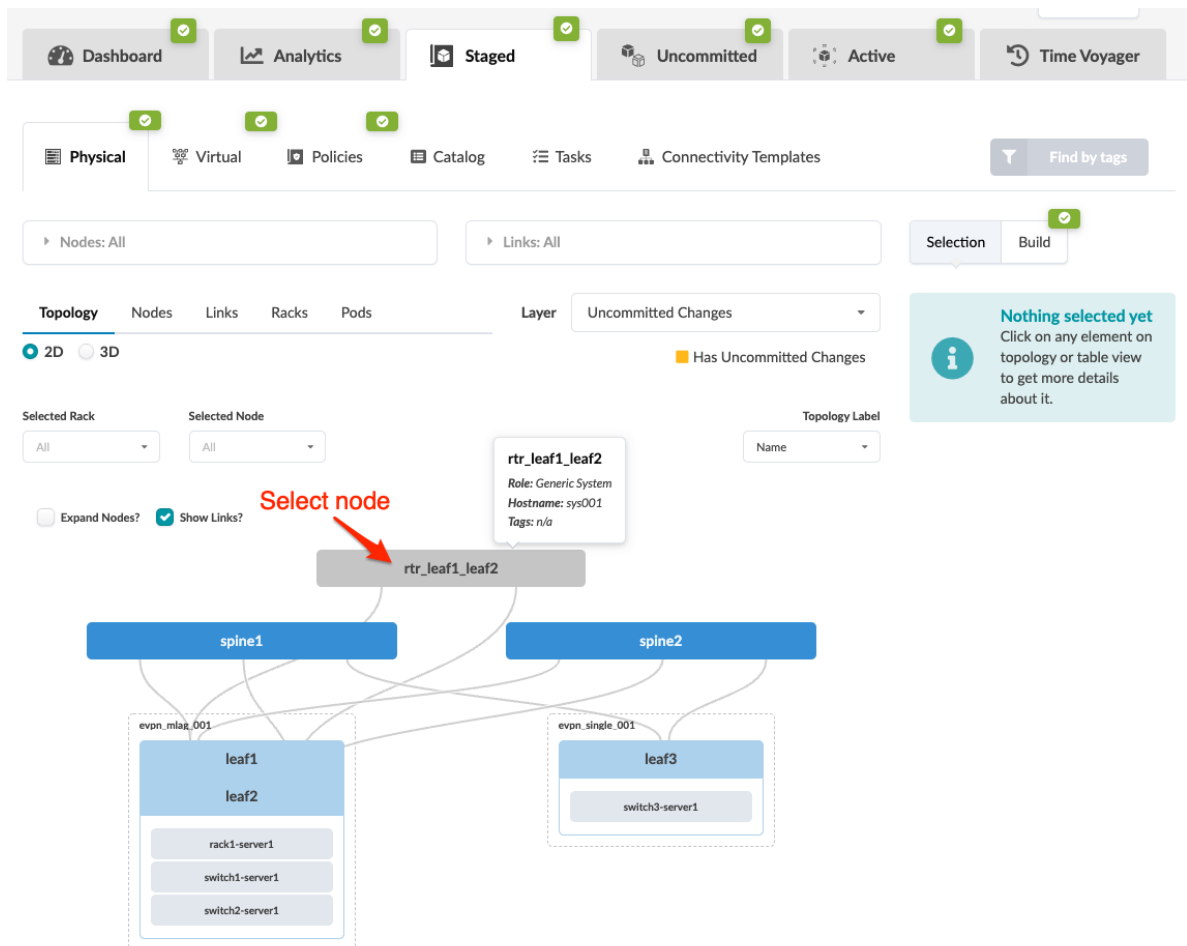
Filter selected by ☒ all ☐ selected only ☐ unselected only

| | Name | Role | Speed | Tags | Endpoint 1 | | | | Endpoint 2 | | | |
|------------|--|-------------------|-------|------|------------|------|-------------|------|------------|------|-------------|------|
| | | | | | Name | Role | Interface | IPv4 | Name | Role | Interface | IPv4 |
| 0 selected | 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 |

Many link operations are performed from the **Topology** view, and some can also be performed directly in the **Links** view. See the following sections for more information.

Add Links to Leaf

1. From the blueprint, navigate to **Staged > Physical > Topology** and select a node that can connect to a leaf.



2. Select the node check box to see the operations available for that node (and that you have permissions for).

Dashboard Analytics Staged Uncommitted Active

Physical Virtual Policies Catalog Tasks Connectivity Templates

Nodes: All Links: All

Topology Nodes Links Racks Pods

2D 3D

Selected Rack: All Selected Node: rtr_leaf1_leaf2 (Generic System) Topology Label: Name

Select node for available operations

Neighbors Links

rtr_leaf1_leaf2

- Add links to leaf
- Add links to spine
- Update node tags
- Delete node

Show All Neighbors

eth1 Ethernet1/7 leaf1

eth2 Ethernet1/7 leaf2



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).

3. Click **Add links to leaf** to go to its dialog.

Create Links

Select Leaf: * Select...

Select devices and their interfaces to create a link:

Generic System: rtr_leaf1_leaf2 Device profile: N/A

Link tags: Select...

Add Link

Links

1-2 of 2

| Type | Speed | External Generic | | Leaf | | Tags | Actions |
|----------|-------|------------------|-----------|-------|-------------|------|---------|
| | | Name | Interface | Name | Interface | | |
| Existing | 10G | rtr_leaf1_leaf2 | eth1 | leaf1 | Ethernet1/7 | | |
| Existing | 10G | rtr_leaf1_leaf2 | eth2 | leaf2 | Ethernet1/7 | | |

Create

4. Select the leaf to link to from the drop-down menu, then select an available port and transformation. The gray **Add Link** button turns green.

Create Links ✕

Select Leaf: ✕

Select devices and their interfaces to create a link:

Leaf: leaf3
Device profile: Cisco NXOSv

Port #4 Tr. #1 (10 Gbps, default)

Port #4 Tr. #2 (1 Gbps)

Generic System: rtr_leaf1_leaf2
Device profile: N/A

Link tags:

Add Link →

Links 1-2 of 2 < >

| Type | Speed | External Generic | | Leaf | | Tags | Actions |
|----------|-------|------------------|-----------|-------|-------------|------|---------|
| | | Name | Interface | Name | Interface | | |
| Existing | 10G | rtr_leaf1_leaf2 | eth1 | leaf1 | Ethernet1/7 | | |
| Existing | 10G | rtr_leaf1_leaf2 | eth2 | leaf2 | Ethernet1/7 | | |

Create

5. Click **Add Link**. The link is added to the link table.

Create Links ✕

Select Leaf: ✕

Select devices and their interfaces to create a link:

Leaf: leaf3
Device profile: Cisco NXOSv

Port #4 Tr. #1 (10 Gbps, default)

Port #4 Tr. #2 (1 Gbps)

Generic System: rtr_leaf1_leaf2
Device profile: N/A

Link tags:

Add Link →

Links (1 will be added) 1-3 of 3 < >

New link is added

| Type | Speed | External Generic | | Leaf | | Tags | Actions |
|----------|-------|------------------|-----------|-------|-------------|------|---------|
| | | Name | Interface | Name | Interface | | |
| New | 10G | rtr_leaf1_leaf2 | N/A | leaf3 | Ethernet1/4 | | |
| Existing | 10G | rtr_leaf1_leaf2 | eth1 | leaf1 | Ethernet1/7 | | |
| Existing | 10G | rtr_leaf1_leaf2 | eth2 | leaf2 | Ethernet1/7 | | |

Create

6. Click **Create** to stage the change and return to the **Topology** view.

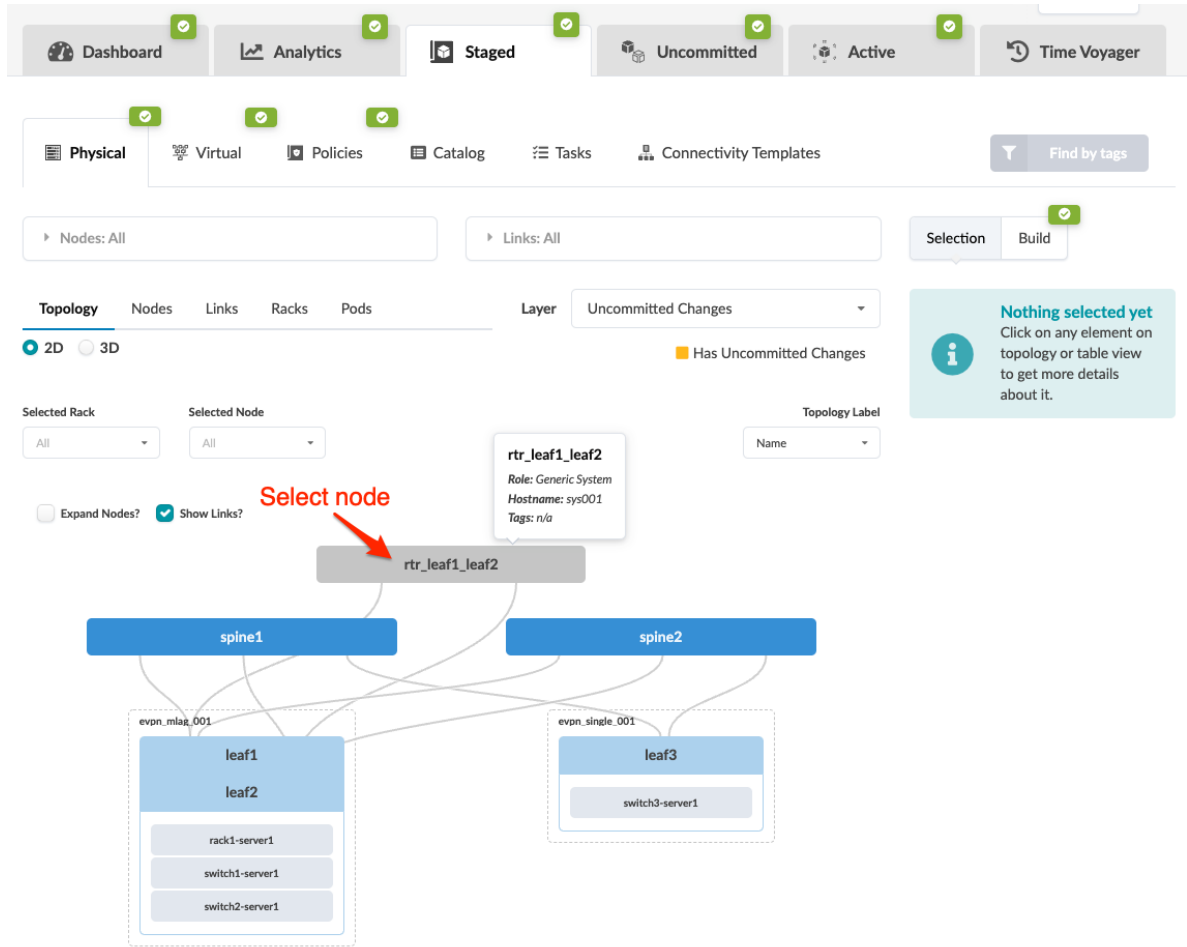
The screenshot shows the network management interface with the following components:

- Top Navigation Bar:** Dashboard, Analytics, Staged, Uncommitted, Active.
- Left Sidebar:** Physical, Virtual, Policies, Catalog, Tasks, Connectivity Templates.
- Filters:** Nodes: All, Links: All.
- Topology View:** Topology (selected), Nodes, Links, Racks, Pods.
- View Mode:** 2D (selected), 3D.
- Selected Rack:** All.
- Selected Node:** rtr_leaf1_leaf2 (Gen eric System).
- Topology Label:** Name.
- Neighbors/Links:** Neighbors (selected), Links.
- Diagram:** A diagram showing the selected node 'rtr_leaf1_leaf2' connected to three spine nodes: 'leaf1', 'leaf2', and 'leaf3'. The connections are labeled 'eth1', 'eth2', and 'n/a' on the left, and 'Ethernet1/7', 'Ethernet1/7', and 'Ethernet1/4' on the right. A red arrow points to the 'n/a' connection, labeled 'New link'.

When you're ready to activate your changes, commit them from the **Uncommitted** tab.

Add Links to Spine

1. From the blueprint, navigate to **Staged > Physical > Topology** and select a node that can connect to a spine.



2. Select the node check box to see the operations available for that node (and that you have permissions for).

Dashboard

Analytics

Staged

Uncommitted

Active

Physical

Virtual

Policies

Catalog

Tasks

Connectivity Templates

Nodes: All

Links: All

Topology

Nodes

Links

Racks

Pods

2D

3D

Selected Rack

Selected Node

Topology Label

All

rtr_leaf1_leaf2 (Gen *
eric System)

Name

Select node for available operations

Neighbors

Links

rtr_leaf1_leaf2

Add links to leaf

Add links to spine

Update node tags

Delete node

Show All Neighbors

eth1

eth2

Ethernet1/7

leaf1

Ethernet1/7

leaf2

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).

3. Click **Add links to spine** to go to its dialog.

Create Links

Select Spine: *

Select...

Select spine to link to

Select devices and their interfaces to create a link:

Generic System: rtr_leaf1_leaf2

Device profile: N/A

Link tags

Select...

Add Link

Links

1-2 of 2

| Type | Speed | External Generic | | Spine | | Tags | Actions |
|----------|-------|------------------|-----------|-------|-------------|------|---------|
| | | Name | Interface | Name | Interface | | |
| Existing | 10G | rtr_leaf1_leaf2 | eth1 | leaf1 | Ethernet1/7 | | |
| Existing | 10G | rtr_leaf1_leaf2 | eth2 | leaf2 | Ethernet1/7 | | |

Create

4. Select the spine to link to from the drop-down menu, then select an available port and transformation. The gray **Add Link** button turns green.

Create Links ✕

Select Spine: spine2 ✕

Select devices and their interfaces to create a link:

Spine: spine2
Device profile: Cisco NXOSv

1. 2. 3. 4.

Port #4 Tr. #1 (10 Gbps, default) Ethernet1/4

Port #4 Tr. #2 (1 Gbps) Ethernet1/4

Generic System: rtr_leaf1_leaf2
Device profile: N/A

Link tags
Select...

Add Link →

Links 1-2 of 2

| Type | Speed | External Generic | | Spine | | Tags | Actions |
|----------|-------|------------------|-----------|-------|-------------|------|---------|
| | | Name | Interface | Name | Interface | | |
| Existing | 10G | rtr_leaf1_leaf2 | eth1 | leaf1 | Ethernet1/7 | | |
| Existing | 10G | rtr_leaf1_leaf2 | eth2 | leaf2 | Ethernet1/7 | | |

Create

5. Click **Add Link**. The link is added to the link table.

Create Links ✕

Select Spine: spine2 ✕

Select devices and their interfaces to create a link:

Spine: spine2
Device profile: Cisco NXOSv

Port #4 Tr. #1 (10 Gbps, default) Ethernet1/4

Port #4 Tr. #2 (1 Gbps) Ethernet1/4

Generic System: rtr_leaf1_leaf2
Device profile: N/A

Link tags
Select...

Add Link →

Links (1 will be added) 1-3 of 3

New link is added

| Type | Speed | External Generic | | Spine | | Tags | Actions |
|----------|-------|------------------|-----------|--------|-------------|------|---------|
| | | Name | Interface | Name | Interface | | |
| New | 10G | rtr_leaf1_leaf2 | N/A | spine2 | Ethernet1/4 | | |
| Existing | 10G | rtr_leaf1_leaf2 | eth1 | leaf1 | Ethernet1/7 | | |
| Existing | 10G | rtr_leaf1_leaf2 | eth2 | leaf2 | Ethernet1/7 | | |

Create

6. Click **Create** to stage the change and return to the **Topology** view.

Dashboard Analytics Staged Uncommitted Active

Physical Virtual Policies Catalog Tasks Connectivity Templates

Nodes: All Links: All

Topology Nodes Links Racks Pods

2D 3D

Selected Rack: All

Selected Node: rtr_leaf1_leaf2 (Generic System)

Topology Label: Name

Neighbors Links

Show All Neighbors

rtr_leaf1_leaf2

eth1 eth2 n/a

Ethernet1/7 Ethernet1/7 Ethernet1/4

leaf1 leaf2 spine2

New link

When you're ready to activate your changes, commit them from the **Uncommitted** tab.

Add Links to Generic System

1. From the blueprint, navigate to **Staged > Physical > Topology** and select a node that can connect to a generic system.

The screenshot displays a network management dashboard with the following components:

- Top Navigation Bar:** Includes tabs for Dashboard, Analytics, Staged, Uncommitted, Active, and Time Voyager.
- Secondary Navigation Bar:** Includes Physical, Virtual, Policies, Catalog, Tasks, and Connectivity Templates.
- Filters:**
 - Nodes: All
 - Links: All
 - Selection/Build toggle
 - Find by tags
- Topology View:**
 - Views: Topology, Nodes, Links, Racks, Pods
 - Layer: Uncommitted Changes
 - Has Uncommitted Changes indicator
 - Selected Rack: All
 - Selected Node: All
 - Topology Label: Name
 - Expand Nodes? (unchecked)
 - Show Links? (checked)
- Topology Diagram:**
 - Nodes: spine1, spine2, rtr_leaf1_leaf2, leaf1, leaf2, leaf3.
 - Leaf1 details: Role: Leaf, Hostname: leaf1, Tags: n/a.
 - Leaf1 is highlighted with a red arrow and the text "Select node".
- Information Panel:** "Nothing selected yet. Click on any element on topology or table view to get more details about it."

2. Select the node check box to see the operations available for that node (and that you have permissions for).

The screenshot shows the network management interface with the following components:

- Top Navigation Bar:** Dashboard, Analytics, Staged (active), Uncommitted, Active.
- Left Sidebar:** Physical (active), Virtual, Policies, Catalog, Tasks, Connectivity Templates.
- Filters:** Nodes: All, Links: All.
- Topology View:** Topology (selected), Nodes, Links, Racks, Pods. View toggle: 2D (selected), 3D.
- Selected Rack:** evpn_mlag_001
- Selected Node:** leaf1 (Leaf)
- Topology Label:** Name
- Neighbors/Links:** Neighbors (selected), Links.
- Selection Panel (Left):**
 - leaf1 (selected)
 - Options: Add generic system, Copy existing generic, Add links to generic system, Add external generic, Add links to external generic, Add leaf peer links, Add access switch, Update node tags.
- Topology Diagram (Right):**
 - Shows connections between nodes and external systems.
 - Nodes: rack1-server1, leaf2, spine1, spine2, switch1-server1, rtr_leaf1_leaf2.
 - Ports: n/a, Ethernet1/4, Ethernet1/3, Ethernet1/1, n/a, eth1.

Select node for available operations



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).

3. Click **Add links to generic system** to go to its dialog.

Create Links

Select devices and their interfaces to create a link:

Leaf: leaf1
Device profile: Cisco NXOSv

1

2

3

4

5

6

7

8

9

Leaf: leaf2
Device profile: Cisco NXOSv

1

2

3

4

5

6

7

8

9

Select Generic: *

Select...

Link tags

Select...

Add Link →

Links

1-4 of 4

| Type | Speed | Leaf | | Generic | | Tags | Actions |
|----------|-------|-------|-------------|-----------------|-----------|------|---------|
| | | Name | Interface | Name | Interface | | |
| Existing | 10G | leaf1 | Ethernet1/5 | rack1-server1 | N/A | | |
| Existing | 10G | leaf1 | Ethernet1/6 | switch1-server1 | N/A | | |
| Existing | 10G | leaf2 | Ethernet1/5 | rack1-server1 | N/A | | |
| Existing | 10G | leaf2 | Ethernet1/6 | switch2-server1 | N/A | | |

Create

4. Select an available port, transformation, and the generic system to link to. The gray **Add Link** button turns green.

Create Links

Select devices and their interfaces to create a link:

Leaf: leaf1
Device profile: Cisco NXOSv

1

2

3

4

5

6

7

8

9

Port #8 Tr. #1 (10 Gbps, default)

Ethernet1/8

Port #8 Tr. #2 (1 Gbps)

Ethernet1/8

Leaf: leaf2
Device profile: Cisco NXOSv

1

2

3

4

5

6

7

8

9

Select Generic: *

rack1-server1

Generic System: rack1-server1
Device profile: N/A

Link tags

Select...

Add Link →

Links

1-4 of 4

| Type | Speed | Leaf | | Generic | | Tags | Actions |
|----------|-------|-------|-------------|-----------------|-----------|------|---------|
| | | Name | Interface | Name | Interface | | |
| Existing | 10G | leaf1 | Ethernet1/5 | rack1-server1 | N/A | | |
| Existing | 10G | leaf1 | Ethernet1/6 | switch1-server1 | N/A | | |
| Existing | 10G | leaf2 | Ethernet1/5 | rack1-server1 | N/A | | |
| Existing | 10G | leaf2 | Ethernet1/6 | switch2-server1 | N/A | | |

Create

5. Click **Add Link**. The link is added to the link table.

Create Links

Select devices and their interfaces to create a link:

Leaf: leaf1
Device profile: Cisco NXOSv

1

2

3

4

5

6

7

8

9

Port #8 Tr. #1 (10 Gbps, default)

Ethernet1/8

Port #8 Tr. #2 (1 Gbps)

Ethernet1/8

Leaf: leaf2
Device profile: Cisco NXOSv

1

2

3

4

5

6

7

8

9

Select Generic:

rack1-server1

×

Generic System: rack1-server1
Device profile: N/A

Link tags

Select...

▼

Links (1 will be added)

1-5 of 5 < >

| Type | Speed | Leaf | | Generic | | Tags | Actions |
|----------|-------|-------|-------------|-----------------|-----------|------|---------|
| | | Name | Interface | Name | Interface | | |
| New | 10G | leaf1 | Ethernet1/8 | rack1-server1 | N/A | | |
| Existing | 10G | leaf1 | Ethernet1/5 | rack1-server1 | N/A | | |
| Existing | 10G | leaf1 | Ethernet1/6 | switch1-server1 | N/A | | |
| Existing | 10G | leaf2 | Ethernet1/5 | rack1-server1 | N/A | | |
| Existing | 10G | leaf2 | Ethernet1/6 | switch2-server1 | N/A | | |

Add Link →

Create

New link is added

- Click **Create** to stage the change and return to the **Topology** view.

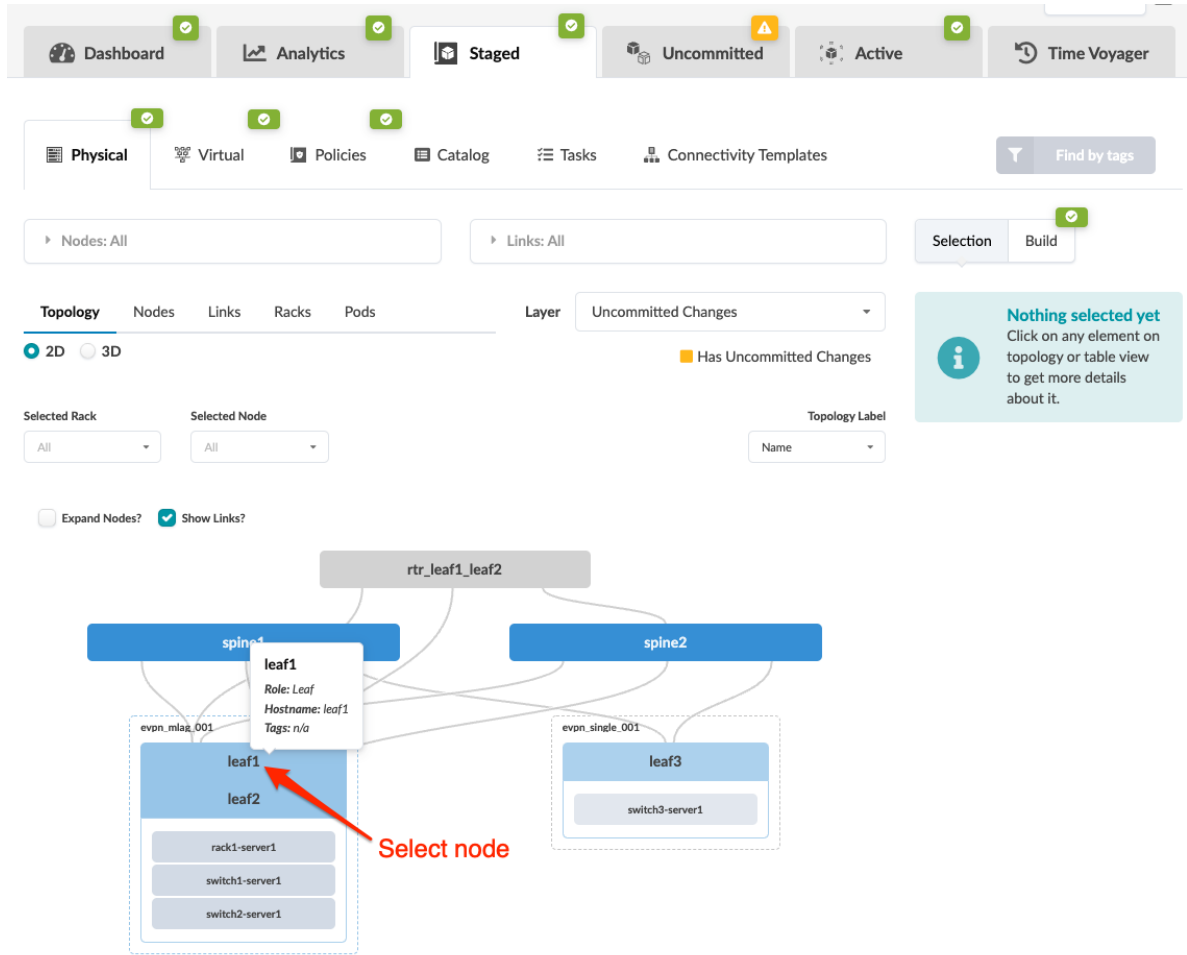
The screenshot displays the network management interface with the following components:

- Navigation Tabs:** Dashboard, Analytics, Staged (active), Uncommitted, Active.
- Sub-menu:** Physical (active), Virtual, Policies, Catalog, Tasks, Connectivity Templates.
- Filters:** Nodes: All, Links: All.
- Topology View:** Topology (selected), Nodes, Links, Racks, Pods.
- View Mode:** 2D (selected), 3D.
- Selected Rack:** evpn_mlag_001.
- Selected Node:** leaf1 (Leaf).
- Topology Label:** Name.
- Buttons:** Neighbors, Links.
- Options:** Show Aggregate Links (checked), Show Unused Ports (unchecked), Show All Neighbors (dropdown).
- Diagram:** A 2D topology diagram showing connections between various nodes. A red arrow points to a new link being added between 'leaf1' and 'rtr_leaf1_leaf2'.

When you're ready to activate your changes, commit them from the **Uncommitted** tab.

Add Links to External Generic System

1. From the blueprint, navigate to **Staged > Physical > Topology** and select a node that can connect to an external generic system.



2. Select the node check box to see the operations available for that node (and that you have permissions for).

The screenshot shows the network management interface with the following components:

- Top Navigation Bar:** Dashboard, Analytics, Staged, Uncommitted, Active.
- Left Sidebar:** Physical (selected), Virtual, Policies, Catalog, Tasks, Connectivity Templates.
- Filters:** Nodes: All, Links: All.
- View Options:** Topology (selected), Nodes, Links, Racks, Pods. 2D (selected), 3D.
- Selection Fields:** Selected Rack: evpn_mlag_001, Selected Node: leaf1 (Leaf), Topology Label: Name.
- Buttons:** Neighbors, Links.
- Context Menu (for leaf1):**
 - Add generic system
 - Copy existing generic
 - Add links to generic system
 - Add external generic
 - Add links to external generic
 - Add leaf peer links
 - Add access switch
 - Update node tags
- Diagram:** A network diagram showing connections between nodes. A red arrow points to the 'Add links to external generic' option in the context menu.



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).

3. Click **Add links to external generic** to go to its dialog.

Create Links

Select devices and their interfaces to create a link:

Leaf: leaf1
Device profile: Cisco NXOSv

1

2

3

4

5

6

7

8

9

Leaf: leaf2
Device profile: Cisco NXOSv

1

2

3

4

5

6

7

8

9

Select External generic: *

Select...

Link tags

Select...

Add Link →

Links

1-2 of 2

| Type | Speed | Leaf | | External Generic | | Tags | Actions |
|----------|-------|-------|-------------|------------------|-----------|------|---------|
| | | Name | Interface | Name | Interface | | |
| Existing | 10G | leaf1 | Ethernet1/7 | rtr_leaf1_leaf2 | eth1 | | |
| Existing | 10G | leaf2 | Ethernet1/7 | rtr_leaf1_leaf2 | eth2 | | |

Create

4. Select an available port, transformation, and the external generic system to link to. The gray **Add Link** button turns green.

Create Links

Select devices and their interfaces to create a link:

Leaf: leaf1
Device profile: Cisco NXOSv

1

2

3

4

5

6

7

8

9

Port #8 Tr. #1 (10 Gbps, default)

Ethernet1/8

Port #8 Tr. #2 (1 Gbps)

Ethernet1/8

Leaf: leaf2
Device profile: Cisco NXOSv

1

2

3

4

5

6

7

8

9

Select External generic: *

rtr_leaf1_leaf2

Generic System: rtr_leaf1_leaf2
Device profile: N/A

Link tags

Select...

Add Link →

Links

1-2 of 2

| Type | Speed | Leaf | | External Generic | | Tags | Actions |
|----------|-------|-------|-------------|------------------|-----------|------|---------|
| | | Name | Interface | Name | Interface | | |
| Existing | 10G | leaf1 | Ethernet1/7 | rtr_leaf1_leaf2 | eth1 | | |
| Existing | 10G | leaf2 | Ethernet1/7 | rtr_leaf1_leaf2 | eth2 | | |

Create

5. Click **Add Link**. The link is added to the link table.

Create Links

Select devices and their interfaces to create a link:

Leaf: leaf1
Device profile: Cisco NXOSv

1 2 3 4 5 6 7 8 9

Port #8 Tr. #1 (10 Gbps, default)

Ethernet1/8

Port #8 Tr. #2 (1 Gbps)

Ethernet1/8

Leaf: leaf2
Device profile: Cisco NXOSv

1 2 3 4 5 6 7 8 9

Select External generic:Ⓢ

rtr_leaf1_leaf2

Generic System: rtr_leaf1_leaf2
Device profile: N/A

Link tags

Select...

Links (1 will be added)

1-3 of 3

| Type | Speed | Leaf | | External Generic | | Tags | Actions |
|----------|-------|-------|-------------|------------------|-----------|------|---------|
| | | Name | Interface | Name | Interface | | |
| New | 10G | leaf1 | Ethernet1/8 | rtr_leaf1_leaf2 | N/A | | |
| Existing | 10G | leaf1 | Ethernet1/7 | rtr_leaf1_leaf2 | eth1 | | |
| Existing | 10G | leaf2 | Ethernet1/7 | rtr_leaf1_leaf2 | eth2 | | |

Add Link →

⚠

Create

New link is added

6. Click **Create** to stage the change and return to the **Topology** view.

Dashboard Analytics Staged Uncommitted Active

Physical Virtual Policies Catalog Tasks Connectivity Templates

Nodes: All Links: All

Topology Nodes Links Racks Pods

2D 3D

Selected Rack: evpn_mlag_001 Selected Node: leaf1 (Leaf) Topology Label: Name

Neighbors Links

☒ Show Aggregate Links ☐ Show Unused Ports Show All Neighbors

| Device | Port | Neighbor | Port |
|--------|-------------|-----------------|-----------------|
| leaf1 | Ethernet1/5 | n/a | rack1-server1 |
| | Ethernet1/4 | Ethernet1/4 | leaf2 |
| | Ethernet1/3 | Ethernet1/3 | leaf2 |
| | Ethernet1/1 | Ethernet1/1 | spine1 |
| | Ethernet1/2 | Ethernet1/1 | spine2 |
| | Ethernet1/6 | Ethernet1/1 | spine2 |
| | Ethernet1/7 | n/a | switch1-server1 |
| | Ethernet1/8 | eth1 | rtr_leaf1_leaf2 |
| | n/a | rtr_leaf1_leaf2 | |

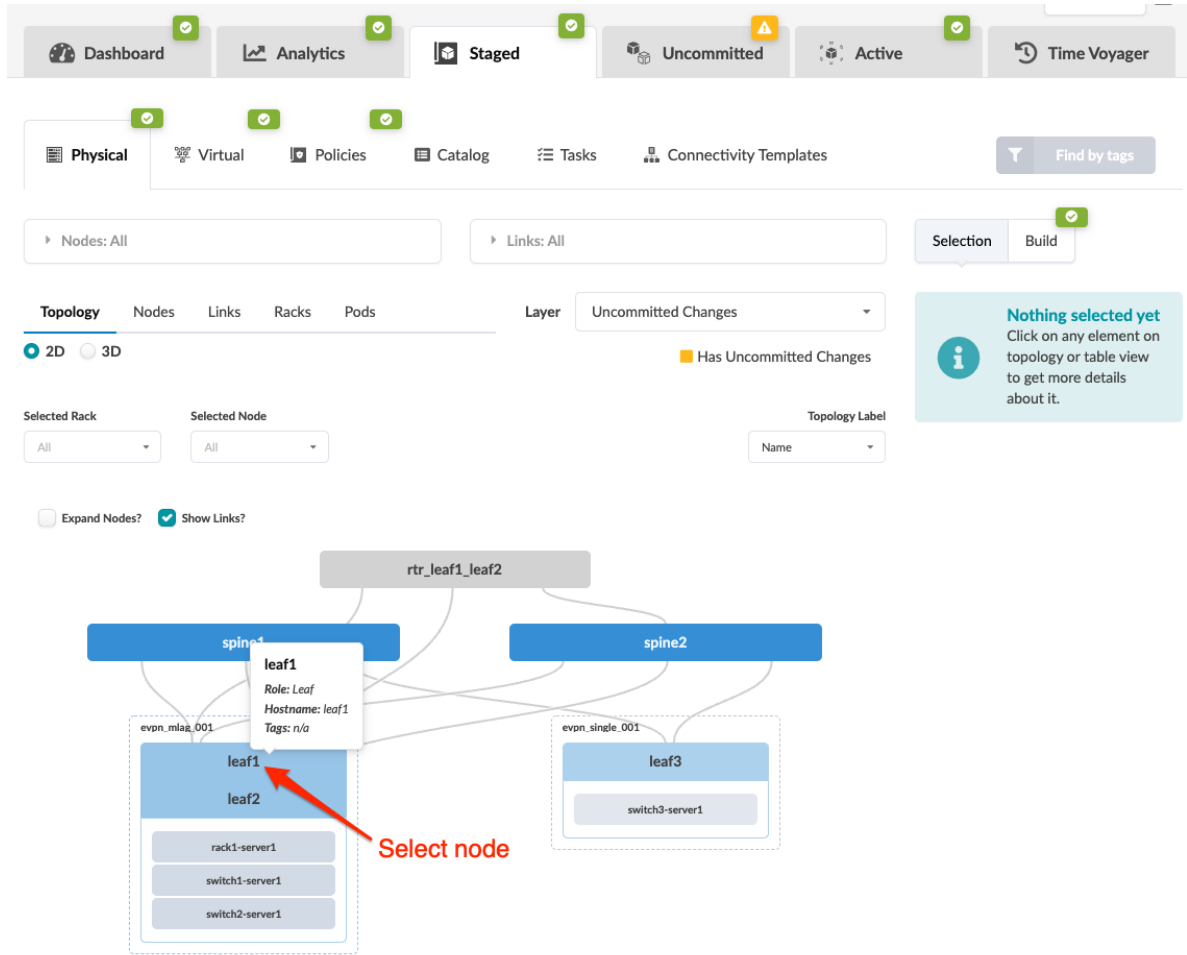
New link

When you're ready to activate your changes, commit them from the **Uncommitted** tab.

Add Leaf Peer Links

If your platform does not support it, do not attempt to create leaf peer links. Currently, Junos devices do not support any peer links, and SONiC devices do not support L3 peer links.

1. From the blueprint, navigate to **Staged > Physical > Topology** and select the MLAG member that needs a peer link.



2. Select the node check box to see the operations available for that node (and that you have permissions for).

The screenshot shows the network management interface with the following components:

- Top Navigation Bar:** Dashboard, Analytics, Staged (active), Uncommitted, Active.
- Left Sidebar:** Physical (active), Virtual, Policies, Catalog, Tasks, Connectivity Templates.
- Filters:** Nodes: All, Links: All.
- Topology View:** Topology (selected), Nodes, Links, Racks, Pods. View toggle: 2D (selected), 3D.
- Selected Rack:** evpn_mlag_001
- Selected Node:** leaf1 (Leaf)
- Topology Label:** Name
- Neighbors/Links:** Neighbors (selected), Links.
- Selection Menu (for leaf1):**
 - Add generic system
 - Copy existing generic
 - Add links to generic system
 - Add external generic
 - Add links to external generic
 - Add leaf peer links (highlighted by a red arrow)
 - Add access switch
 - Update node tags
- Topology Diagram:** Shows connections between leaf1 and other nodes. Connections include:
 - leaf1 to rack1-server1 (n/a)
 - leaf1 to leaf2 (Ethernet1/4, Ethernet1/3)
 - leaf1 to spine1 (Ethernet1/1)
 - leaf1 to spine2 (Ethernet1/1)
 - leaf1 to switch1-server1 (n/a)
 - leaf1 to rtr_leaf1_leaf2 (eth1)



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).

3. Click **Add leaf peer links** to go to its dialog.
4. Select the link type (peer link, L3 peer link) and an available port and transformation for each leaf member. (Only unused ports are selectable.) The gray **Add Link** button turns green.

Add Leaf Peer Links



Link Type

☒ Peer Link ☐ L3 Peer Link

Select ports and interfaces to create a link:

Leaf: leaf1

Device profile: Cisco NXOSv



| | |
|--------------------------------------|-------------|
| Port #8 Tr. #1 (10 Gbps, default) | Ethernet1/8 |
| Port #8 Tr. #2 (1 Gbps) | Ethernet1/8 |

Leaf: leaf2

Device profile: Cisco NXOSv



| | |
|--------------------------------------|-------------|
| Port #8 Tr. #1 (10 Gbps, default) | Ethernet1/8 |
| Port #8 Tr. #2 (1 Gbps) | Ethernet1/8 |

Link tags

Select...

New Links



| Speed | Leaf 1 | | Leaf 2 | | Tags | Link Type | Actions |
|--------------|--------|-----------|--------|-----------|------|-----------|---------|
| | Name | Interface | Name | Interface | | | |
| No new links | | | | | | | |



5. Click **Add Link**. The link is added to the link table.

Add Leaf Peer Links



Link Type

☒ Peer Link ☐ L3 Peer Link

Select ports and interfaces to create a link:

Leaf: leaf1

Device profile: Cisco NXOSv



| | |
|-----------------------------------|-------------|
| Port #8 Tr. #1 (10 Gbps, default) | Ethernet1/8 |
| Port #8 Tr. #2 (1 Gbps) | Ethernet1/8 |

Leaf: leaf2

Device profile: Cisco NXOSv



| | |
|-----------------------------------|-------------|
| Port #8 Tr. #1 (10 Gbps, default) | Ethernet1/8 |
| Port #8 Tr. #2 (1 Gbps) | Ethernet1/8 |

Link tags

Select...

New Links

1-1 of 1

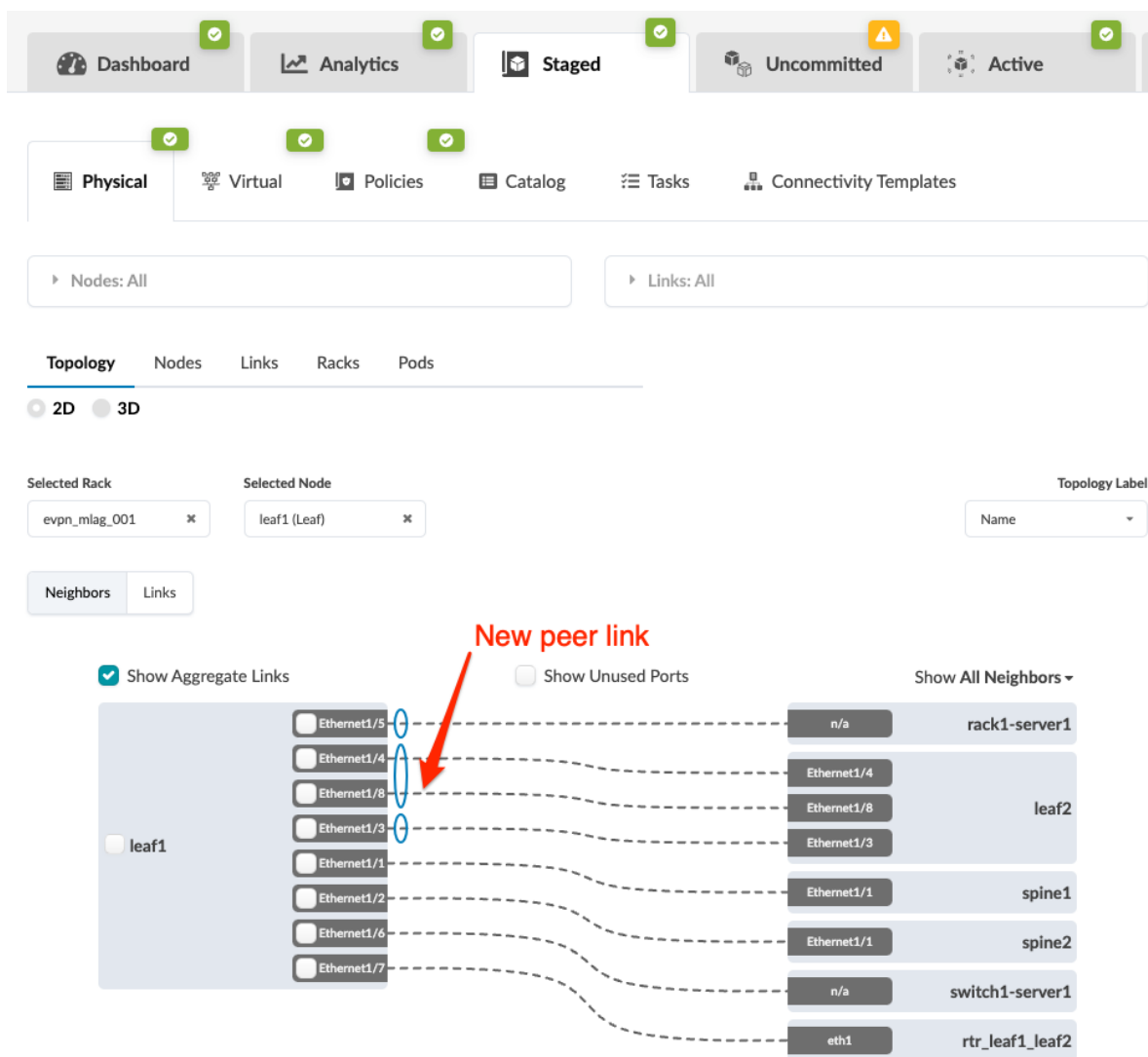


| Speed | Leaf 1 | | Leaf 2 | | Tags | Link Type | Actions |
|-------|--------|-------------|--------|-------------|------|-----------|---------|
| | Name | Interface | Name | Interface | | | |
| 10G | leaf1 | Ethernet1/8 | leaf2 | Ethernet1/8 | | Peer Link | |

Peer link is added



6. Click **Add** to stage the change and return to the **Topology** view. (BGP session is added as applicable.)

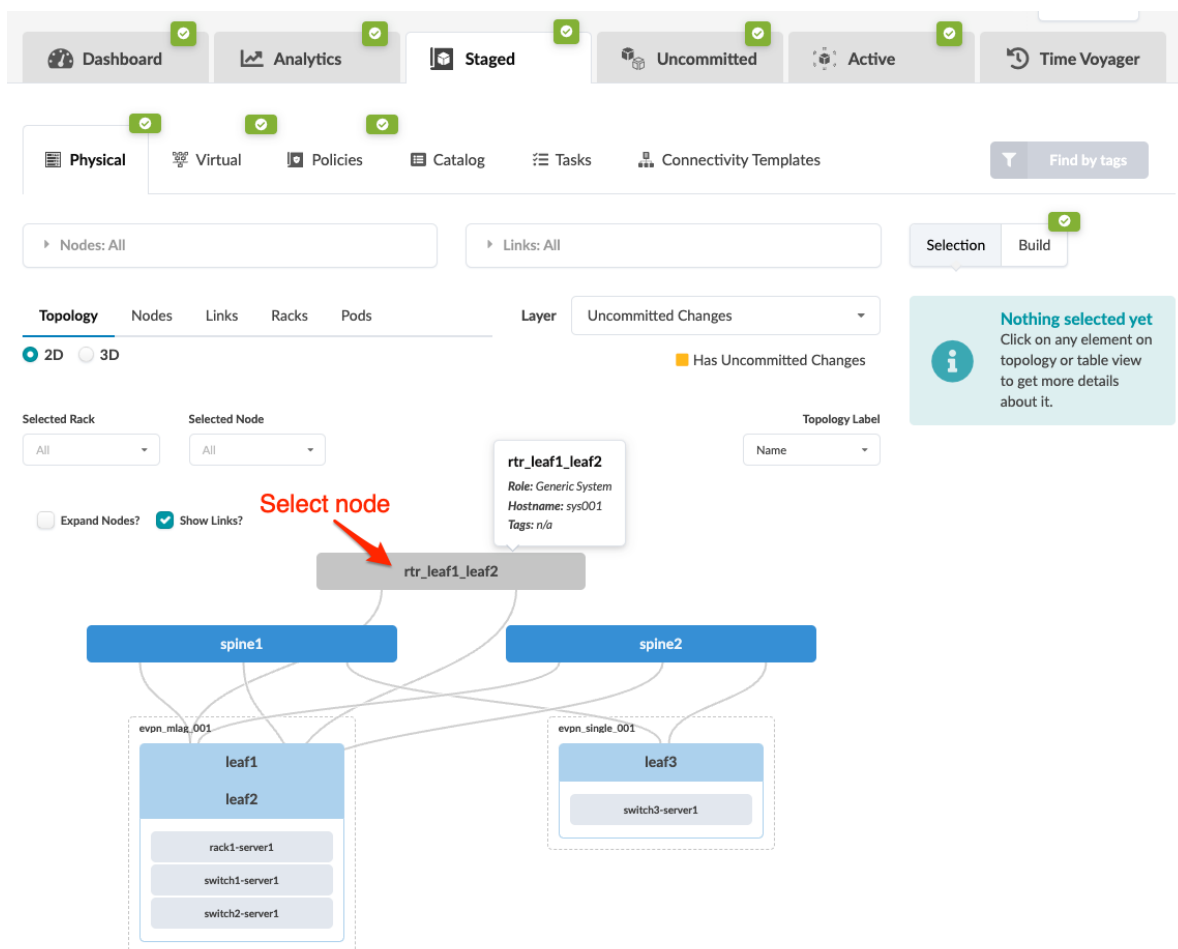


When you're ready to activate your changes, commit them from the **Uncommitted** tab.

Form LAG

It's common to break a LAG towards a server into individual links, then reform the LAG from individual links, all while keeping the same VLAN allocation (when re-bootstrapping the server for example). When you form a LAG, it inherits any connectivity templates assigned on the individual links (as of Apstra version 4.1.0).

1. From the blueprint, navigate to **Staged > Physical > Topology** and select the node to add as a member of a LAG.



2. Select the interface check box to see the operations available for that interface (and that you have permissions for).

Dashboard Analytics Staged Uncommitted Active

Physical Virtual Policies Catalog Tasks Connectivity Templates

Nodes: All Links: All

Topology Nodes Links Racks Pods

2D 3D

Selected Rack: All

Selected Node: rtr_leaf1_leaf2 (Generic System)

Topology Label: Name

Neighbors Links

1 selected

Form LAG

Update LAG mode

Update link tags

Update link speed

Delete link

Interface Label: eth1

eth1

eth2

Ethernet1/7 leaf1

Ethernet1/7 leaf2

Show All Neighbors

Select interface for available operations

3. Click **Form LAG** and select the LAG mode:

- **LACP (Active)** - actively advertises LACP BPDU even when neighbors do not.
- **LACP (Passive)** - doesn't generate LACP BPDU until it sees one from a neighbor.
- **Static LAG (no LACP)** - Static LAGs don't participate in LACP and will conditionally operate in forwarding mode.

Form LAG

LAG Mode*

- ☒ LACP (Active)Ⓢ
 ☐ LACP (Passive)Ⓢ
 ☐ Static LAG (no LACP)Ⓢ

The following links are going to form a LAG. There is a link with CTs assigned. The newly created LAG will inherit these CTs:

| | | | | |
|---|------------|-------|--------------------------------------|--|
| rtr_leaf1_leaf2 Generic System, Interface eth1 | | ↔ | leaf1 Leaf, Interface Ethernet1/7 | |
| Speed: 10G | PC ID: n/a | Tags: | | |
| Applied CTs: rtr_leaf1_leaf2:3:ct_bgp_subintf_to_subintf:ipv4_ipv6 | | | | |

Create

4. Click **Update** to stage your changes and return to the **Topology** view.

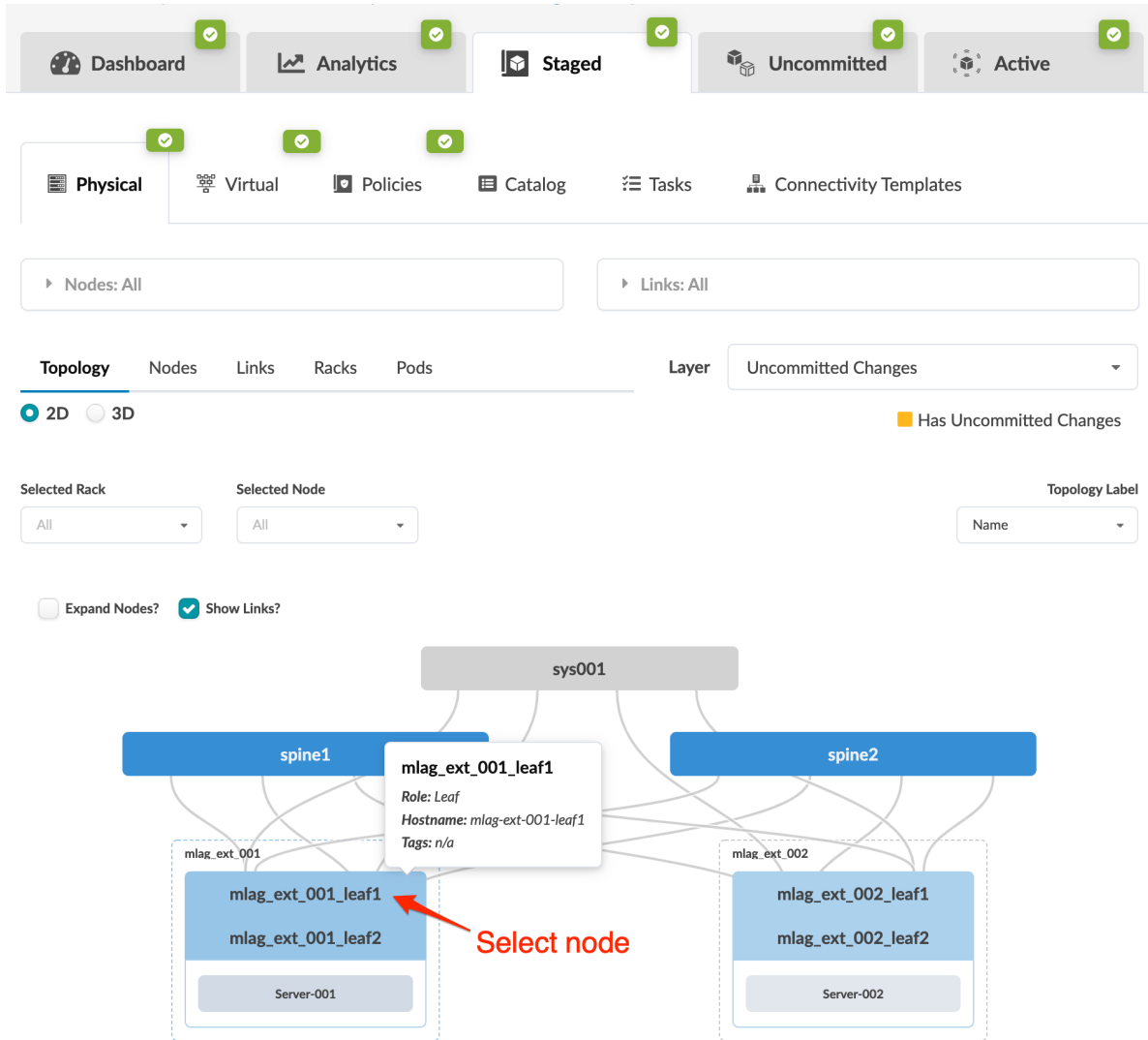
The LAG is created, but LACP configuration won't be pushed to the device until connectivity templates are applied.

When you're ready to activate your changes, commit them from the **Uncommitted** tab.

Break LAG

It's common to break a LAG towards a server into individual links, then reform the LAG from individual links, all while keeping the same VLAN allocation (when re-bootstrapping the server for example). You can break a LAG while preserving any assigned connectivity templates (as of Apstra version 4.1.0).

1. From the blueprint, navigate to **Staged > Physical > Topology** and select the node with the LAG to break.



2. Select the interface check boxes for the LAG (or click the port-channel representation) to see the operations available for those interfaces (and that you have permissions for).

Dashboard Analytics Staged Uncommitted Active

Physical Virtual Policies Catalog Tasks Connectivity Templates

Nodes: All Links: All

Topology Nodes Links Racks Pods

2D 3D

Selected Rack: mlag_ext_001 Selected Node: mlag_ext_001_leaf1 (Leaf) Topology Label: Name

Neighbors Links

Select interfaces

Show Aggregate Links Show Unused Ports Show All Neighbors

2 selected Break LAG Update LAG mode Delete links

Ethernet3 Ethernet1 Ethernet2 Ethernet4 Ethernet49/1 Ethernet50/1

n/a mlag_ext_001_leaf2 sys001 spine1 spine2

3. Click **Break LAG** to go to its dialog with details on the LAG to break.

Break LAG

mlag_ext_001_leaf1 Leaf, Interface Ethernet1

mlag_ext_001_leaf2 Leaf, Interface Ethernet1

Speed: 10G PC ID: 2 Tags:

mlag_ext_001_leaf1 Leaf, Interface Ethernet2

mlag_ext_001_leaf2 Leaf, Interface Ethernet2

Speed: 10G PC ID: 2 Tags:

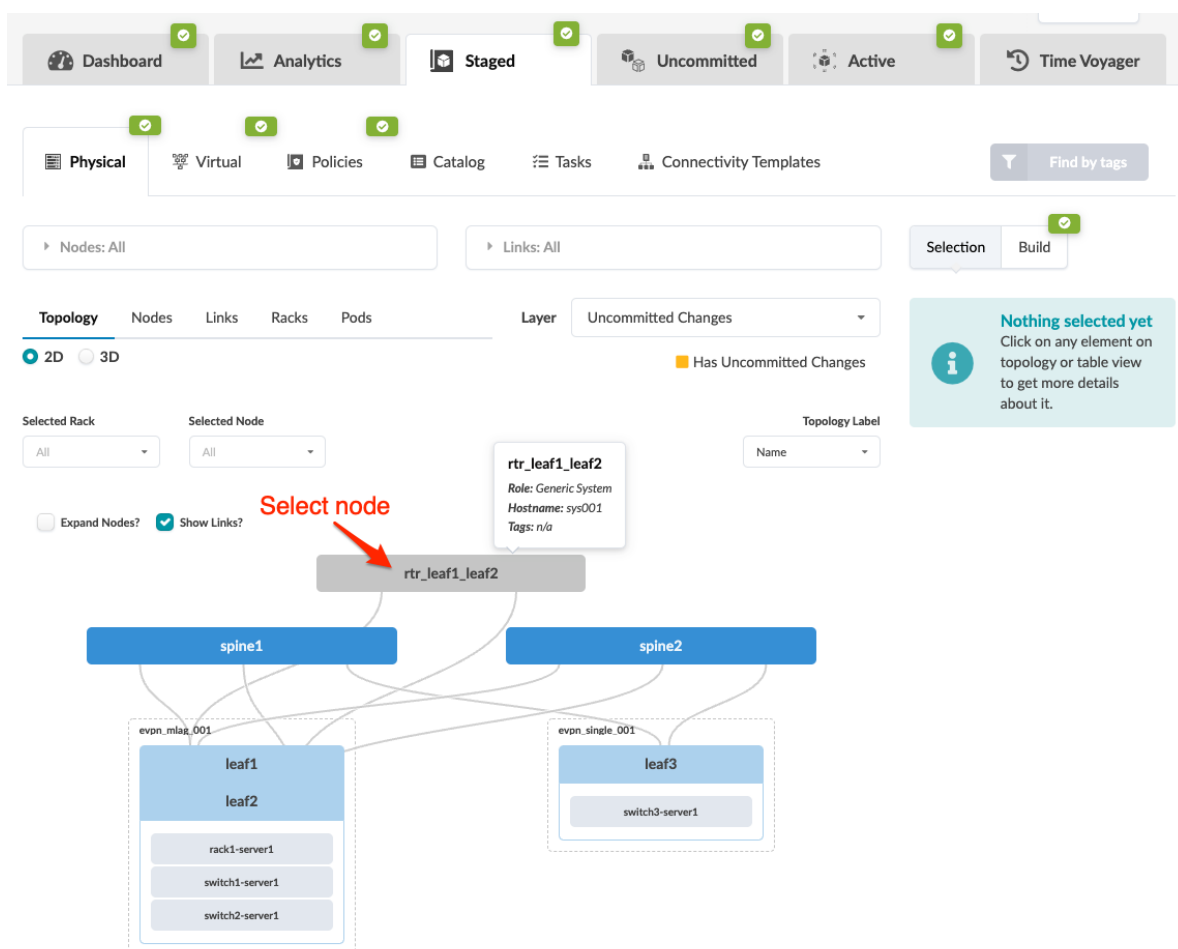
Break

4. Click **Break** to stage your changes and return to the **Topology** view.

When you're ready to activate your changes, commit them from the **Uncommitted** tab.

Update LAG Mode

1. From the blueprint, navigate to **Staged > Physical > Topology** and select the MLAG member that needs an updated link LAG mode.



2. Select the interface check box to see the operations available for that interface (and that you have permissions for).

The screenshot shows the network management interface with the following components:

- Navigation Bar:** Dashboard, Analytics, Staged, Uncommitted, Active.
- Sub-Navigation Bar:** Physical, Virtual, Policies, Catalog, Tasks, Connectivity Templates.
- Filters:** Nodes: All, Links: All.
- Topology View:** Topology (selected), Nodes, Links, Racks, Pods.
- View Mode:** 2D (selected), 3D.
- Selected Rack:** All.
- Selected Node:** rtr_leaf1_leaf2 (Generic System).
- Topology Label:** Name.
- Neighbors List:**
 - 1 selected
 - Form LAG
 - Update LAG mode
 - Update link tags
 - Update link speed
 - Delete link
- Interface Details:**
 - Interface Label: eth1
 - Selected interface: eth1 (checked)
 - Other interfaces: eth2
- Neighbors Table:**

| Neighbor | Interface | Label |
|----------|-------------|-------|
| leaf1 | Ethernet1/7 | leaf1 |
| leaf2 | Ethernet1/7 | leaf2 |

3. Click **Update LAG mode** and select the new LAG mode:

- **LACP (Active)** - actively advertises LACP BPDUs even when neighbors do not.
- **LACP (Passive)** - doesn't generate LACP BPDUs until it sees one from a neighbor.
- **Static LAG (no LACP)** - Static LAGs don't participate in LACP and will conditionally operate in forwarding mode.
- **No LAG** - The link is not part of a LAG.

Update Link LAG Mode

The 'Update Link LAG Mode' dialog box displays the following information:

- Link Details:**
 - Left Node: rtr_leaf1_leaf2 (Generic System, Interface eth1)
 - Right Node: leaf1 (Leaf, Interface Ethernet1/7)
 - Speed: 10G
 - PC ID: n/a
 - Tags:
- LAG Mode Selection:**
 - ☐ LACP (Active)
 - ☐ LACP (Passive)
 - ☐ Static LAG (no LACP)
 - ☒ No LAG
- Action:** Update

4. Click **Update** to stage your changes and return to the **Topology** view.

When you're ready to activate your changes, commit them from the **Uncommitted** tab.

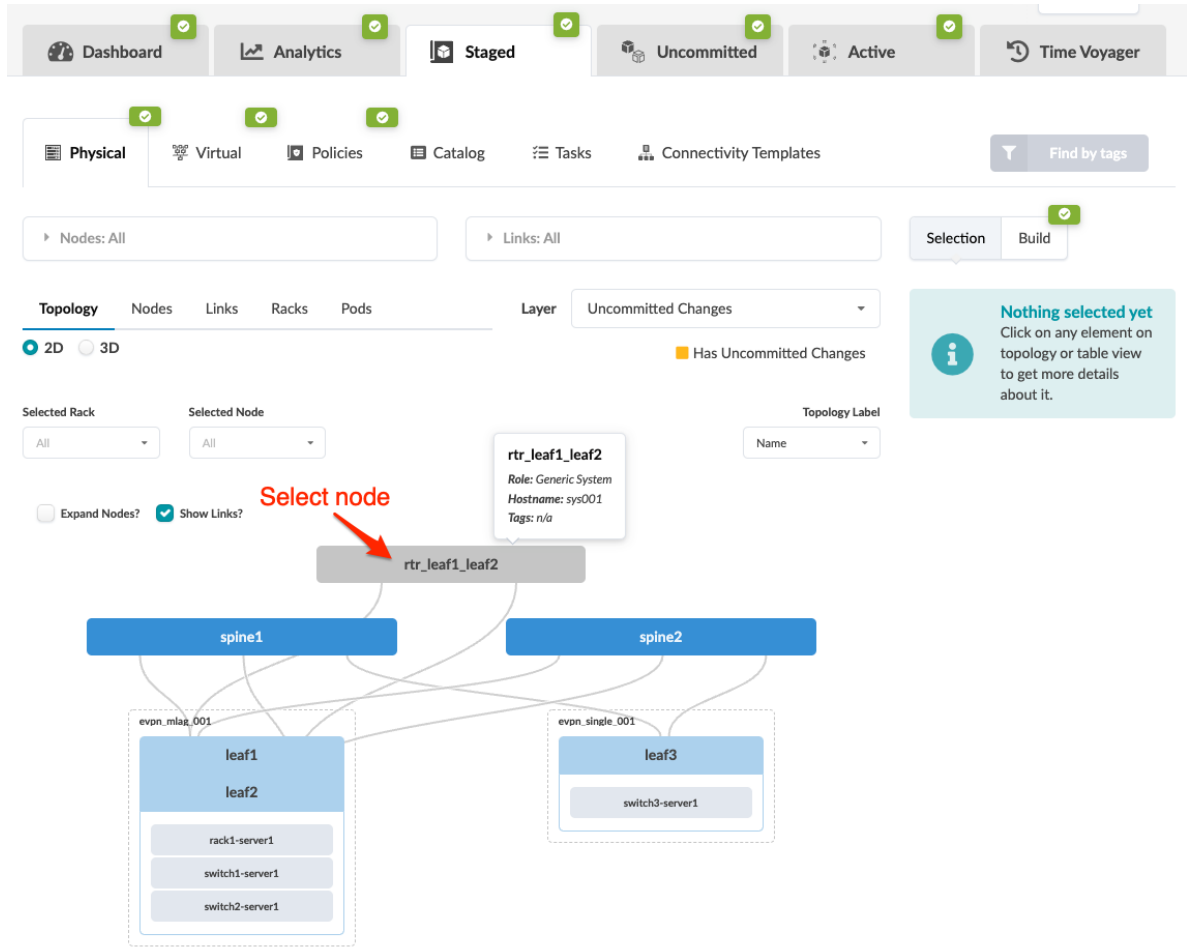
Update Link Tags

IN THIS SECTION

- [Update Link Tags \(One Link - Topology View\) | 369](#)
- [Update Link Tags \(One Link - Links View\) | 372](#)
- [Update Link Tags \(Multiple Link - Links View\) | 372](#)

Update Link Tags (One Link - Topology View)

1. From the blueprint, navigate to **Staged > Physical > Topology** and select the node connected to the link that needs tags updated.



2. Select the interface check box to see the operations available for that interface (and that you have permissions for).

The screenshot shows the network management interface with the following components:

- Top Navigation Bar:** Dashboard, Analytics, Staged, Uncommitted, Active.
- Left Sidebar:** Physical (selected), Virtual, Policies, Catalog, Tasks, Connectivity Templates.
- Filters:** Nodes: All, Links: All.
- Topology View:** Topology (selected), Nodes, Links, Racks, Pods. View toggle: 2D (selected), 3D.
- Selected Rack:** All.
- Selected Node:** rtr_leaf1_leaf2 (Generic System).
- Topology Label:** Name.
- Neighbors Tab:** Shows a list of neighbors. A red arrow points to the 'eth1' interface in the 'Neighbors' list, with the text "Select interface for available operations".
- Interface Details:** Label: eth1. A dropdown menu is open showing options: Form LAG, Update LAG mode, Update link tags, Update link speed, Delete link.
- Neighbors List:**

| Interface | Label |
|--|-------------------|
| <input checked="" type="checkbox"/> eth1 | Ethernet1/7 leaf1 |
| <input type="checkbox"/> eth2 | Ethernet1/7 leaf2 |

3. Click **Update link tags** and update link tags as needed.

Update Link Tags

The 'Update Link Tags' dialog box displays the following information:

- Connection:** rtr_leaf1_leaf2 (Generic System, Interface eth1) ↔ leaf1 (Leaf, Interface Ethernet1/7).
- Speed:** 10G
- PC ID:** n/a
- Tags:** Select...
- Update Button:** A blue button labeled 'Update' is located at the bottom right.

4. Click **Update** to update link tags and return to the **Selection** view.

When you're ready to activate your changes, commit them from the **Uncommitted** tab.

Update Link Tags (One Link - Links View)

1. From the blueprint, navigate to **Staged > Physical > Links** and select the link name (not the check box) for the link that needs updated tags.

The screenshot shows the 'Staged > Physical > Links' 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 displays a table of links. A red arrow labeled '1.' points to the link name 'evpn_mlag_001_leaf1<->evpn_mlag_001_leaf2[i3_peer_link][1]' in the table. To the right of the table, there is a panel with 'Add/Remove Tags' and 'No tags assigned' buttons. A red arrow labeled '2.' points to the 'Add/Remove Tags' button.

| Name | Role | Speed | Tags | Endpoint 1 | | Endpoint 2 | |
|--|-------------------|-------|-------|------------|-------|------------|------|
| | | | | Name | Role | Name | Role |
| evpn_mlag_001_leaf1<->evpn_mlag_001_leaf2[i3_peer_link][1] | Leaf L3 Peer Link | 10G | leaf1 | Leaf | leaf2 | Leaf | |

2. Click **Add/Remove Tags** to see tags that are in the blueprint catalog.
3. Select existing tag(s) or create new one(s) that will be tagged to the link and added to the blueprint catalog.
4. Click **Update Tags** to update the tags and return to the **Links** view.

When you're ready to activate your changes, commit them from the **Uncommitted** tab.

Update Link Tags (Multiple Link - Links View)

1. From the blueprint, navigate to **Staged > Physical > Links** and select one or more check boxes for the link(s) that need updated tags. The **Add/Remove Tags** button appears above the table.

Dashboard Analytics Staged Uncommitted Active Time

Physical Virtual Policies Catalog Tasks Connectivity Templates

Nodes: All Links: All

Topology Nodes **Links** Racks Pods Layer Uncommitted Changes

Has Uncommitted Changes

Selected Rack: All

1-15 of 15

Columns (12/15) Page Size: 25

Filter selected by: ☒ all ☐ selected only ☐ unselected only

| | Name | Role | Speed | Tags | Endpoint 1 | | | | Endpoint 2 | | | |
|-------------------------------------|--|-------------------|-------|------|------------|------|-------------|------|------------|------|-------------|------|
| | | | | | Name | Role | Interface | IPv4 | Name | Role | Interface | IPv4 |
| <input checked="" type="checkbox"/> | evpn_mlag_001_leaf1<->evpn_mlag_001_leaf2[03_peer_link][1] | Leaf L3 Peer Link | 10G | | leaf1 | Leaf | Ethernet1/3 | N/A | leaf2 | Leaf | Ethernet1/3 | N/A |
| <input checked="" type="checkbox"/> | evpn_mlag_001_leaf1<->evpn_mlag_001_leaf2[1] | Leaf Peer Link | 10G | | leaf1 | Leaf | Ethernet1/4 | N/A | leaf2 | Leaf | Ethernet1/4 | N/A |

- Click the **Add/Remove Tags** button and update tags as needed. When you create new tags here they are added to the blueprint catalog.

Add/Remove Tags

Add Tags

Select...

Remove Tags

Selected nodes don't have any tags assigned to them

The following nodes will be affected

▸

Query: All

1-2 of 2

<

>

Page Size:

25

▾

| Name ↕ |
|--|
| evpn_mlag_001_leaf1<->evpn_mlag_001_leaf2(l3_peer_link)[1] |
| evpn_mlag_001_leaf1<->evpn_mlag_001_leaf2[1] |

Add/Remove Tags

3. Click **Add/Remove Tags** to stage the change and return to the **Links** view.

When you're ready to activate your changes, commit them from the **Uncommitted** tab.

Update Link Speed

IN THIS SECTION

●

Update Link Speed (Topology View) | 374

●

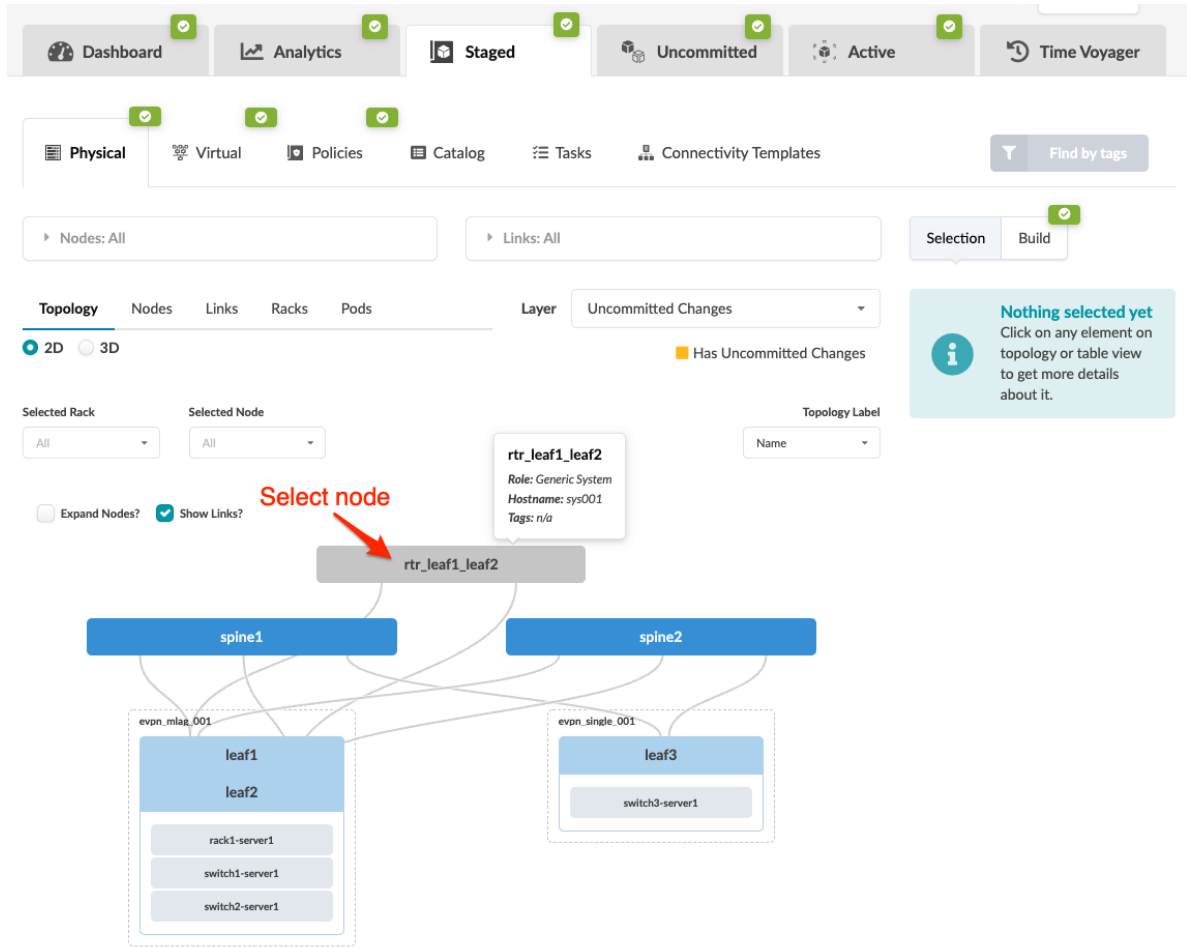
Update Link Speed (Links View) | 376

To change link speeds between spine-leaf and superspine-spine you must ["change the rack" on page 390](#).

Update Link Speed (Topology View)

From the **Topology** view, you can update one link speed at a time. (You can update more than one link speed at a time from the **Links** view; see the next section.)

1. From the blueprint, navigate to **Staged > Physical > Topology** and select the node where you want to change link speed.



2. Select the interface check box to see the operations available for that interface (and that you have permissions for).

Dashboard Analytics Staged Uncommitted Active

Physical Virtual Policies Catalog Tasks Connectivity Templates

Nodes: All Links: All

Topology Nodes Links Racks Pods

2D 3D

Selected Rack: All

Selected Node: rtr_leaf1_leaf2 (Generic System)

Topology Label: Name

Neighbors Links

1 selected

Form LAG

Update LAG mode

Update link tags

Update link speed

Delete link

Interface Label: eth1

eth1

eth2

Ethernet1/7 leaf1

Ethernet1/7 leaf2

Show All Neighbors

Select interface for available operations

- Click **Update link speed** and select the new link speed from the drop-down list.

Update Link Speed

rtr_leaf1_leaf2
Generic System, Interface eth1

leaf1
Leaf, Interface Ethernet1/7

Speed: 10G PC ID: n/a Tags:

Speed

10 Gbps

Update

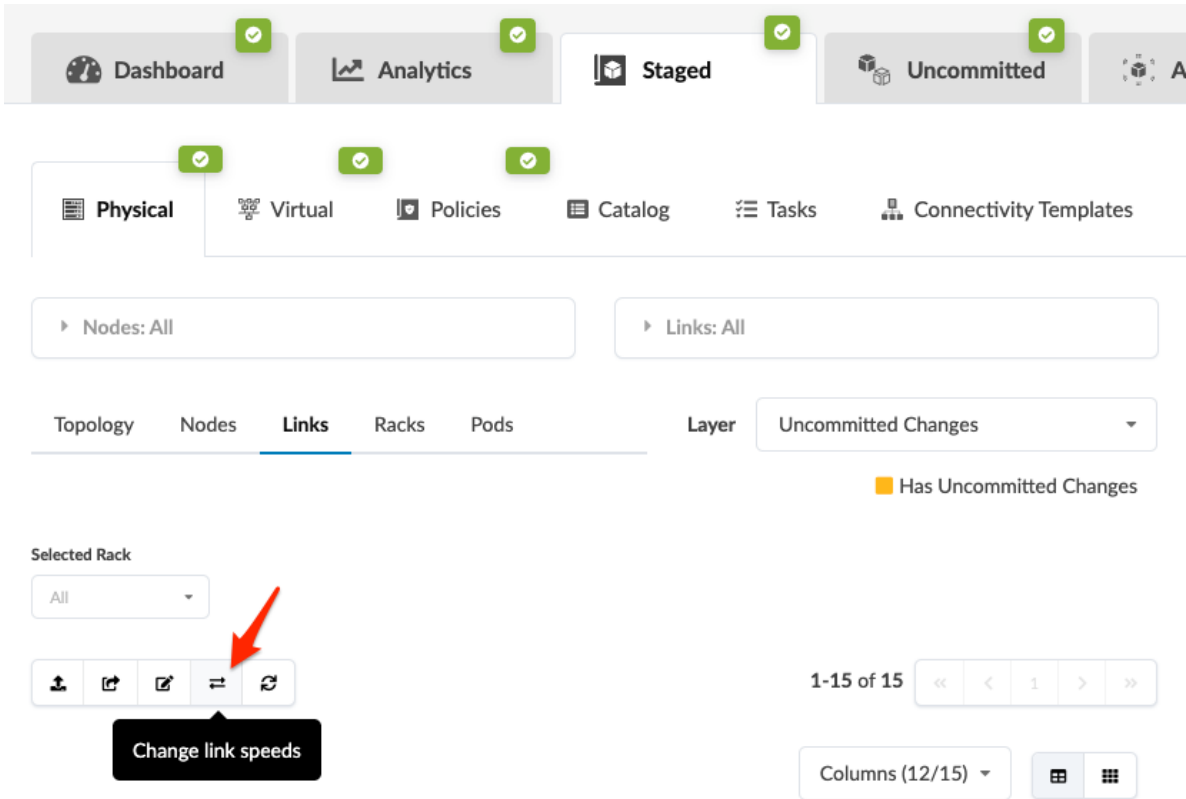
- Click **Update** to stage your changes and return to the **Topology** view.

When you're ready to activate your changes, commit them from the **Uncommitted** tab.

Update Link Speed (Links View)

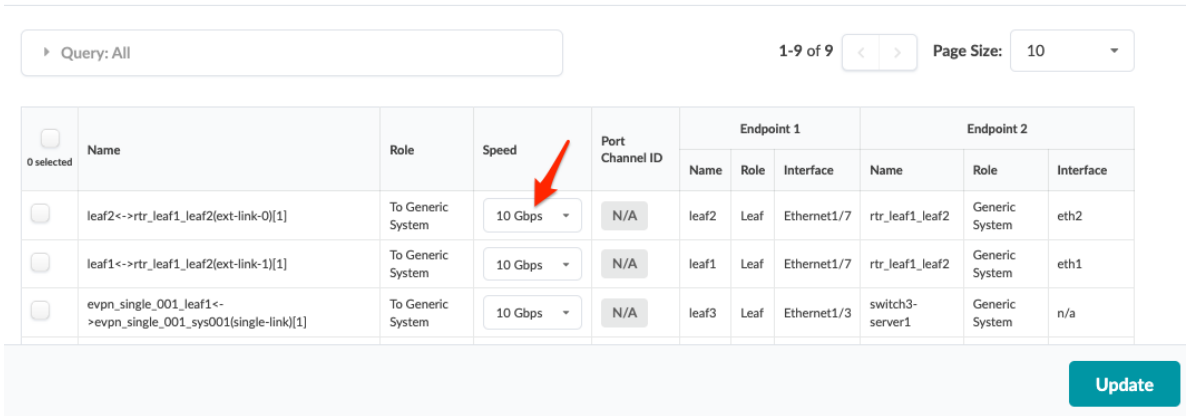
From the **Links** view you can update one or more link speeds at the same time.

1. From the blueprint, navigate to **Staged > Physical > Links** and click the **Change link speeds** button.



2. Select new link speeds for one or more links from the drop-down lists.

Change Link Speeds



3. Click **Update** to stage the changes and return to the **Links** view.

When you're ready to activate your changes, commit them from the **Uncommitted** tab.

Update Link Properties

If you have changed server names and/or hostnames 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.

- Click the **Save** button to stage the changes.

Delete Link (Datacenter)

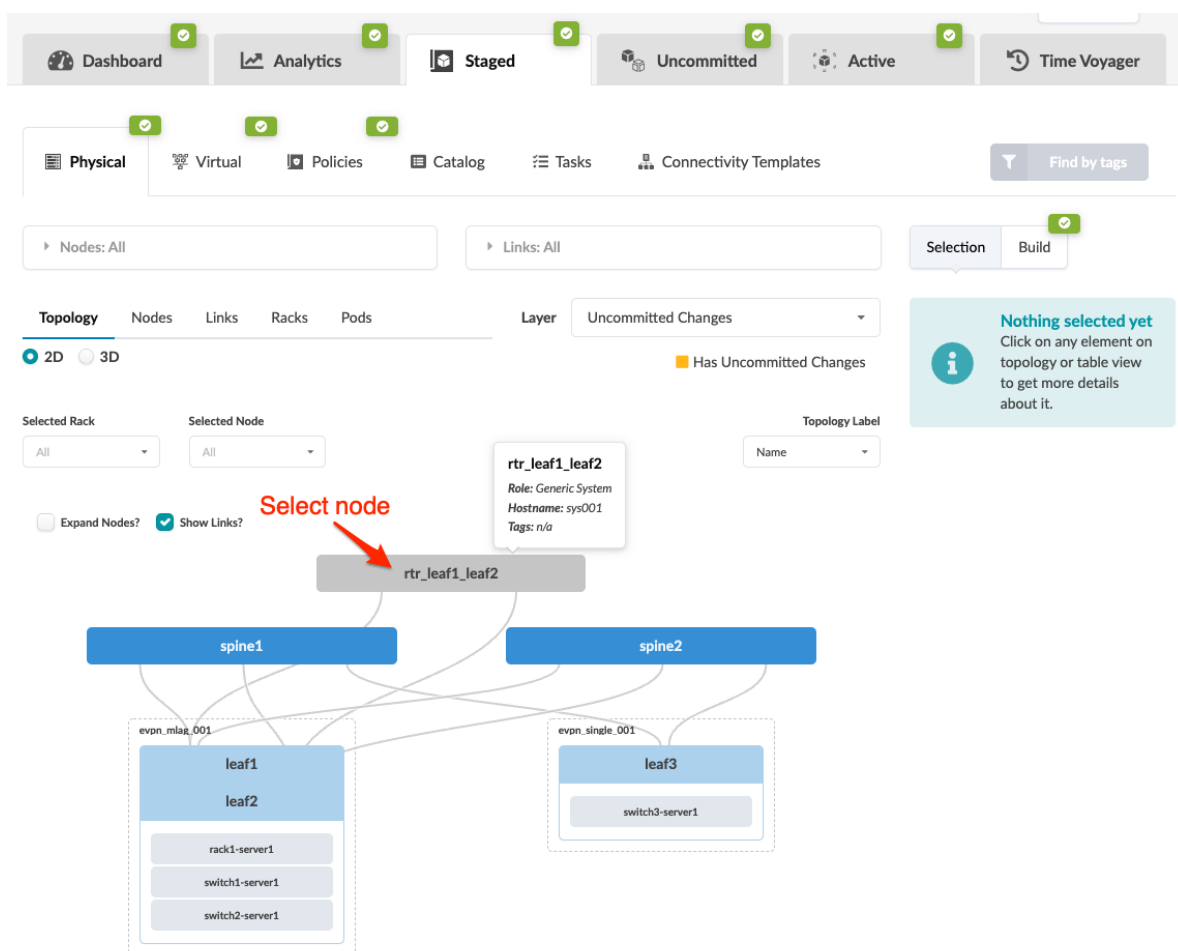
IN THIS SECTION

- Delete Link (Neighbors View) | 379
- Delete Link (Links View) | 381

You can delete links from the **Neighbors** view or the **Links** view of a selection in a blueprint.

Delete Link (Neighbors View)

- From the blueprint, navigate to **Staged > Physical > Topology** and select the node where you want to delete a link.




- From the **Neighbors** view, select the node check box to see the operations available for that node (and that you have permissions for).

The screenshot shows the network management interface. 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. The Physical tab is selected. Below the tabs are filters for Nodes: All and Links: All. The Topology section shows a 2D/3D view. Below this are filters for Selected Rack (All), Selected Node (rtr_leaf1_leaf2 (Generic System)), and Topology Label (Name). The Neighbors view is active, showing a list of interfaces. The 'eth1' checkbox is selected, and a red arrow points to it with the text 'Select interface for available operations'. The interface also shows a list of links with columns for Label, Interface, and Name. The 'eth1' link is highlighted, and a red arrow points to it with the text 'Select interface for available operations'.

- Click **Delete Link** to go to its dialog and review deletion details. Any connectivity templates that are applied on the link will be unassigned.

Delete Link





The following CTs are applied on the link and will be unassigned:
rtr_leaf1_leaf2:l3:ct_bgp_subintf_to_subintf:ipv4_ipv6



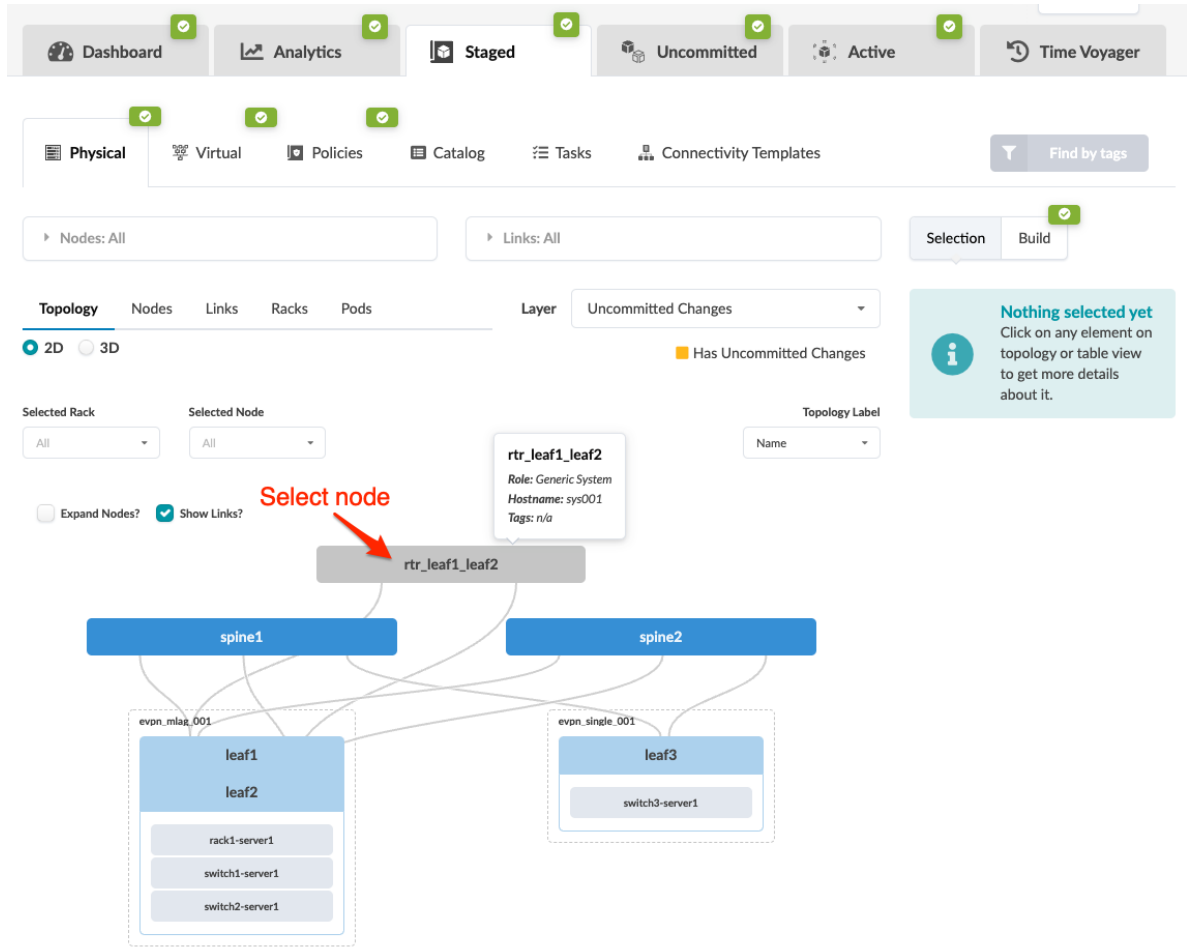
4. Click **Delete** to stage the deletion and return to the **Neighbors** view of the selected node.

When you're ready to activate your changes, commit them from the **Uncommitted** tab.

Delete Link (Links View)

From the **Links** view of your selected node you can delete one or more links at the same time.

1. From the blueprint, navigate to **Staged > Physical > Topology** and select the node where you want to delete a link.



2. Click **Links** to go to the Links table.

| | Name | Role | Tags | Speed | Port Channel ID | Endpoint 1 | | | | Endpoint 2 | | | | Actions |
|-------------------------------------|--|-------------------|------|-------|-----------------|-----------------|----------------|-----------|----------|------------|------|-------------|----------|--------------------------|
| | | | | | | Name | Role | Interface | Lag Mode | Name | Role | Interface | Lag Mode | |
| <input checked="" type="checkbox"/> | leaf1<->rtr_leaf1_leaf2(ext-link-1)[1] | To Generic System | | 10G | N/A | rtr_leaf1_leaf2 | Generic System | eth1 | No LAG | leaf1 | Leaf | Ethernet1/7 | No LAG | <input type="checkbox"/> |
| <input checked="" type="checkbox"/> | leaf2<->rtr_leaf1_leaf2(ext-link-0)[1] | To Generic System | | 10G | N/A | rtr_leaf1_leaf2 | Generic System | eth2 | No LAG | leaf2 | Leaf | Ethernet1/7 | No LAG | <input type="checkbox"/> |

3. Select link(s) to delete in one of the following ways:

- Select one or more links in the left column and click the **Delete** button above the table.
- Click the **Delete** button in the right column for the one link to delete.

4. Review deletion details in the dialog that opens. Any connectivity templates that are applied on the link(s) will be unassigned.

5. Click **Delete** to stage the deletion and return to the **Links** view of the selected node.

When you're ready to activate your changes, commit them from the **Uncommitted** tab.

Import / Export Cabling Map (Datacenter)

IN THIS SECTION

- [Import Cabling Map | 384](#)
- [Export Cabling Map | 384](#)

Import Cabling Map

You can export a cabling map either as a JSON file or a CSV file, but you can only import it as a JSON file.

1. From the blueprint, navigate to **Staged > Physical > Links** and click the **Import cabling map** button (first of five buttons above the links list).
2. Either click **Choose File** and navigate to the JSON 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 and JSON formats. You can copy the contents or download the file to your local computer. If you're planning on importing the cabling map back into your blueprint, use the JSON format; you can't import a CSV file back into a blueprint.

1. From the blueprint, navigate to **Staged > Physical > Links** and click the **Export cabling map** button (second of five buttons above the links list), then select **JSON** or **CSV**. (Use the JSON format if you're planning on importing the cabling map back into your blueprint.)
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 **Active > Physical > Links**.

Edit Cabling Map (Datacenter)

IN THIS SECTION

- [Edit Cabling Map \(GUI\) | 385](#)
- [Edit Cabling Map \(JSON\) | 386](#)

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 824](#).

Edit Cabling Map (GUI)

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. In the cabling map editor, change interface names and/or IP addresses, as applicable.
 - You can use **Batch clear override** to clear all Interface and IPv4/IPv6 values for a specific device type.
 - 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

Batch clear overrides

Query: Role = Spine to Leaf

1-6 of 6

Page Size: 25

| | 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

3. Click **Update** to stage your changes and return to the **Links** view.

Next Steps:

When you're ready to activate your changes, commit them from the **Uncommittedd** tab.

Edit Cabling Map (JSON)

To change the cabling map using JSON, you'll export the JSON file, edit the file, then import it back into the Apstra environment.

1. From the blueprint, navigate to **Staged > Physical > Links** and click the **Export cabling map** button to see the dialog for exporting a cabling map.
2. Select **JSON** and click **Save As File** to download the file.
3. 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**.

Next Steps:

When you're ready to activate your changes, commit them from the **Uncommitted** tab.

Fetch Discovered LLDP Data (Datacenter)

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.

Racks (Datacenter)

IN THIS SECTION

- [Change Rack Name | 388](#)
- [Add Rack | 389](#)
- [Export Rack Type | 389](#)
- [Edit Rack | 390](#)
- [Delete Rack | 391](#)

From the blueprint, navigate to **Staged > Physical > Racks** to go to the **Racks** view.

1.

2.

3.

Table List

Export rack type to global catalog

Edit rack

Delete rack

Change rack name

timestamp

To change rack name click it, then change name in rack properties in right panel

| 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] |

- You can view racks in table view or card view.
- You can filter racks to show **all**, **selected only**, or **unselected only**.

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 391](#), and (as of version 4.0.1) ["increase the number of superpines per plane" on page 406](#). Although, you cannot add or remove planes themselves.) You can also ["change rack names" on page 388](#).

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.

See the following sections for more information on rack operations.

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 to a unique one 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 table view.
7. ["Assign device profiles" on page 280](#) and ["system IDs" on page 295](#) (serial numbers) to the new rack(s).
8. Commit the changes to your blueprint to configure the rack(s) and complete the fabric expansion.

Next Steps:

To assign virtual networks to your new rack, see ["Assign / Unassign Virtual Networks" on page 419](#). As of Apstra version 4.1.2 you can assign many VNs at the same time to one or more nodes.

Export Rack Type

If you can't make certain changes directly in the blueprint rack, you can export the rack type to the global catalog and update it there.

1. From the blueprint, navigate to **Staged > Physical > Racks** and click the **Export rack to global catalog** button (first of three buttons).



NOTE: If the rack type is inconsistent with the same-named one in the global (design) catalog, you won't be able to export the rack type. Rack types are embedded in blueprints from the global catalog. When you use Flexible Fabric Expansion (FFE) operations (for example to change link speeds, add generic systems or add/remove

links) the blueprint rack type is modified. The rack type name in the global catalog and the blueprint are still the same, but their contents are now different from each other. When rack types are inconsistent, you can create a rack type in the global catalog that meets your new requirements.

2. Enter a unique **Rack Type** name.
3. Click **Export** to export the rack type to the global catalog.

Next Steps: From the left navigation menu, navigate to **Design > Rack Types** and edit the rack type in the global catalog. (Or, if you couldn't export the rack type, create one that meets your new requirements.) Then from the blueprint, "[Update the rack](#)" on page 390 to use the revised (or new) rack type from the global catalog.

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 391 and "[add a replacement rack](#)" on page 389, 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 23 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 280 and "[system IDs](#)" on page 295 (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 by draining them. For information, see ["Drain Device Traffic" on page 82](#).

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 table 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 (Datacenter)

IN THIS SECTION

- [Add Pod \(5-Stage Only\) | 392](#)
- [Change Pod Name | 393](#)
- [Add Spine per Pod | 394](#)
- [Add Link per Superspine \(5-Stage\) | 397](#)
- [Change Link Speed per Superspine \(5-Stage\) | 399](#)
- [Change Spine Logical Device \(Pod\) | 401](#)
- [Delete Pod | 404](#)

From the blueprint, navigate to **Staged > Physical > Pods** to go to the **Pods** view.

1.

2.

3.

Select layer to see build details, deploy modes, and uncommitted changes

Click rack type name to see preview

| 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 |

| 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 |

You can search for specific nodes or links.

From the **Pods** view, you can view pod capacity and change pod names. 3-stage topologies can have only one pod. If your topology is for 5-stage, you can add and remove entire pods. 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 ["5-stage Clos Architecture" on page 877](#).

See the following sections for more information about adding, editing and deleting pods.

Add Pod (5-Stage Only)

You can add pods to 5-stage topologies, but not to 3-stage topologies.

1. From the blueprint, navigate to **Staged > Physical > Pods**, and click the **Add Pods** button (+) (center-left). (This button is disabled on 3-stage topologies.)
2. 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

3. Enter the number of pods to add.
4. Click **Add** to stage the pod addition and return to the table view.
5. ["Commit" on page 692](#) the changes to your blueprint to complete the fabric expansion.

Change Pod Name

1. From the blueprint, navigate to **Staged > Physical > Pods** and click the pod name to change.

1. Click pod name...

2. ...to change pod name

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 692](#) the changes to your blueprint to activate the name change.

Add Spine per Pod

As a Day 2 operation, you can add spines per pod on both 3-stage and 5-stage blueprints (as of Apstra version 4.1.2).



CAUTION: Plan carefully. After you've added spines, you won't be able to remove them.

Make sure you have enough ports with specific roles and speeds for additional spine(s).

1. From the blueprint, navigate to **Staged > Physical > Pods**.
2. Click the **Update spine config** button. The location of the button is different for different Apstra versions.
 - On Apstra version 4.1.2, click the **Update spine config** button on the bottom-right of the card for the pod to change

Topology Nodes Links Racks **Pods** Layer Uncommitted Changes ✕

Has Uncommitted Changes

1-1 of 1 < > Page Size: 25 ▾

pod1

Capacity:

Query: All 1-5 of 5 < >

| Name ↕ | Type ↕ | Used ↕ | Available ↕ |
|---------------|----------|--------|-------------|
| L2 One Access | global | 0 | 1 |
| L2 Virtual | global | 0 | 1 |
| rack_1 | embedded | 1 | 0 |
| rack_2 | global | 0 | 1 |
| rack_2 | embedded | 1 | 1 |

Update spine config

Active Tasks: 0

- On Apstra versions 4.1.1 and 4.1.0, check the box on the top-left of the card for the pod to change. Then click the **Update spine config** button that appears above the card(s).

Topology Nodes Links Racks **Pods** Layer Uncommitted Changes ▾

Has Uncommitted Changes

1-1 of 1 < > Page Size: 25 ▾

Update spine config

pod1

Capacity:

Query: All 1-5 of 12 < >

3. In the **Count** field, enter the total number of spines you want:

- You can only *increase* the number of spines.
- On 5-stage blueprints, the number of spines must be a multiplier of the number of superspine planes.



CAUTION: Plan carefully. After you've added spines, you won't be able to remove them.

Update Spine Config

Count[?] *

4

Link per superspine[?] *

1

Link per superspine speed

40 Gbps



Spine Logical Device

AOS-32x40-3 (embedded)



PANEL #1

TOTAL

PORT GROUPS

Connected to ▾

32 ports

32 x 40
Gbps

Superspine •
Spine • Leaf •
Generic

| | | | | | | | | | | | | | | | |
|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|
| 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 |

Update

4. Click **Update** to stage your changes and return to the **Pods** view.

When you're ready to activate changes, commit them from the ["Uncommitted"](#) on [page 692](#) tab.

Add Link per Superspine (5-Stage)

As a Day 2 operation, you can add links per superspine on 5-stage blueprints (as of Apstra version 4.1.0).

1. From the blueprint, navigate to **Staged > Physical > Pods**.
2. Click the **Update spine config** button. The location of the button is different for different Apstra versions.
 - On Apstra version 4.1.2, click the **Update spine config** button on the bottom-right of the card for the pod to change

Topology Nodes Links Racks **Pods** Layer Uncommitted Changes ✕

Has Uncommitted Changes

1-1 of 1 < > Page Size: 25 ▾

pod1

Capacity:

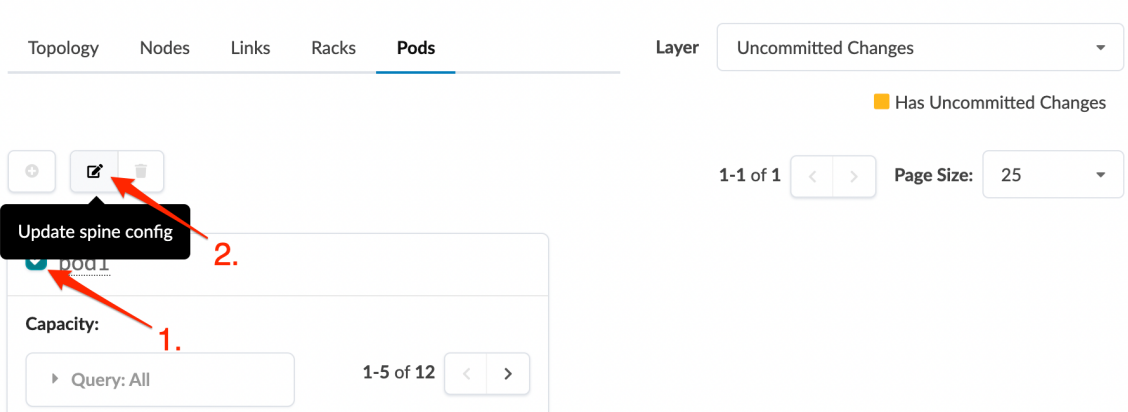
Query: All 1-5 of 5 < >

| Name | Type | Used | Available |
|---------------|----------|------|-----------|
| L2 One Access | global | 0 | 1 |
| L2 Virtual | global | 0 | 1 |
| rack_1 | embedded | 1 | 0 |
| rack_2 | global | 0 | 1 |
| rack_2 | embedded | 1 | 1 |

Update spine config

Active Tasks: 0

- On Apstra versions 4.1.1 and 4.1.0, check the box on the top-left of the card for the pod to change. Then click the **Update spine config** button that appears above the card(s).



3. In the **Link per superspine** field, enter the total number of links you want between spines and superspines. You can only add links. Plan carefully. After you add links, you won't be able to remove them later.

Update Spine Config

Count ⓘ *

4

Link per superspine ⓘ *

1

Link per superspine speed

40 Gbps



Spine Logical Device

AOS-32x40-3 (embedded)



PANEL #1

TOTAL

PORT GROUPS

Connected to ▾

32 ports

32 x 40
Gbps

Superspine •
Spine • Leaf •
Generic

| | | | | | | | | | | | | | | | |
|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|
| 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 |

Update

4. Click **Update** to stage your changes and return to the **Pods** view.

When you're ready to activate changes, commit them from the ["Uncommitted" on page 692](#) tab.

Change Link Speed per Superspine (5-Stage)

As a Day 2 operation, you can change the link speed between spines and superspines on 5-stage blueprints (as of Apstra version 4.1.0).

Make sure link speed is supported on the links / ports (speeds must be part of the port transformations).

1. From the blueprint, navigate to **Staged > Physical > Pods**.
2. Click the **Update spine config** button. The location of the button is different for different Apstra versions.
 - On Apstra version 4.1.2, click the **Update spine config** button on the bottom-right of the card for the pod to change

The screenshot shows the 'Pods' tab in the Apstra interface. At the top, there are tabs for 'Topology', 'Nodes', 'Links', 'Racks', and 'Pods'. A 'Layer' dropdown is set to 'Uncommitted Changes'. Below the tabs, there's a 'Has Uncommitted Changes' indicator. A pagination bar shows '1-1 of 1' and 'Page Size: 25'. The main content area displays a card for 'pod1'. Inside the card, there's a 'Capacity' section with a query filter set to 'All' and a table showing capacity details. The table has columns: Name, Type, Used, and Available. The rows are: L2 One Access (global, 0 used, 1 available), L2 Virtual (global, 0 used, 1 available), rack_1 (embedded, 1 used, 0 available), rack_2 (global, 0 used, 1 available), and rack_2 (embedded, 1 used, 1 available). At the bottom-right of the card, there is a button with a pencil icon, which is labeled 'Update spine config' in a tooltip. A red arrow points to this button.

| Name | Type | Used | Available |
|---------------|----------|------|-----------|
| L2 One Access | global | 0 | 1 |
| L2 Virtual | global | 0 | 1 |
| rack_1 | embedded | 1 | 0 |
| rack_2 | global | 0 | 1 |
| rack_2 | embedded | 1 | 1 |

- On Apstra versions 4.1.1 and 4.1.0, check the box on the top-left of the card for the pod to change. Then click the **Update spine config** button that appears above the card(s).

This screenshot shows the 'Pods' tab for Apstra versions 4.1.1 and 4.1.0. The layout is similar to the previous one, but the 'Update spine config' button is located above the card instead of at the bottom-right. A red arrow points to this button. Another red arrow points to a checkbox on the top-left of the 'pod1' card, which is labeled '1.' in a red number. A second red arrow points to the 'Update spine config' button, labeled '2.' in a red number. The 'Capacity' section and table are also visible.

| Name | Type | Used | Available |
|---------------|----------|------|-----------|
| L2 One Access | global | 0 | 1 |
| L2 Virtual | global | 0 | 1 |
| rack_1 | embedded | 1 | 0 |
| rack_2 | global | 0 | 1 |
| rack_2 | embedded | 1 | 1 |

3. In the **Link per superspine speed** drop-down list, select the new link speed.

Update Spine Config

Count[?] *

4

Link per superspine[?] *

1

Link per superspine speed

40 Gbps

✕

Spine Logical Device

AOS-32x40-3 (embedded)

✕

PANEL #1

TOTAL

32 ports

PORT GROUPS

32 x 40 Gbps

Superspine •

Spine • Leaf •

Generic

Connected to ▾

| | | | | | | | | | | | | | | | |
|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|
| 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 |

Update

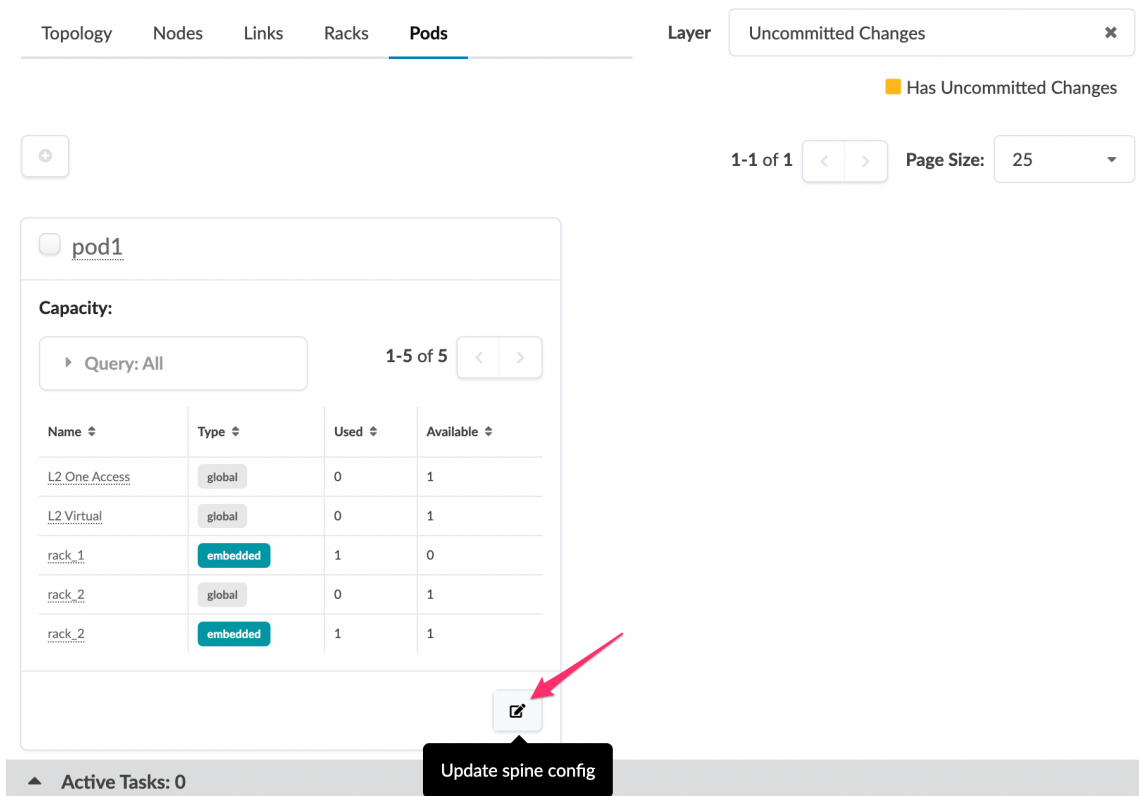
4. Click **Update** to stage your changes and return to the **Pods** view.
- When you're ready to activate changes, commit them from the ["Uncommitted" on page 692](#) tab.

Change Spine Logical Device (Pod)

As a Day-2 operation, you can increase capabilities with a different spine logical device on both 3-stage and 5-stage blueprints (as of Apstra version 4.1.0). (On 5-stage topologies you can also ["change the](#)

[superspine logical device" on page 408.](#)) Changes affect the entire pod, not just a node. Based on the change, this could be disruptive.

1. From the blueprint, navigate to **Staged > Physical > Pods**.
2. Click the **Update spine config** button. The location of the button is different for different Apstra versions.
 - On Apstra version 4.1.2, click the **Update spine config** button on the bottom-right of the card for the pod to change



Topology Nodes Links Racks **Pods** Layer Uncommitted Changes ✕

Has Uncommitted Changes

1-1 of 1 < > Page Size: 25 ▾

☐ pod1

Capacity:

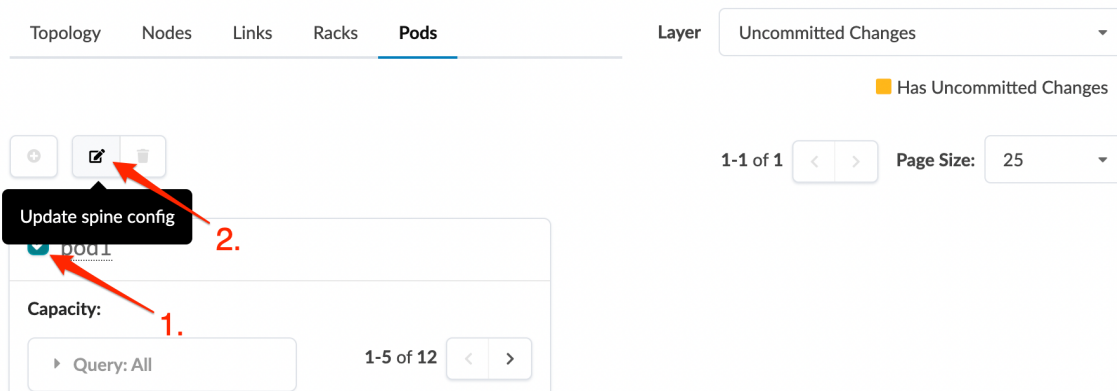
Query: All 1-5 of 5 < >

| Name ↕ | Type ↕ | Used ↕ | Available ↕ |
|---------------|----------|--------|-------------|
| L2 One Access | global | 0 | 1 |
| L2 Virtual | global | 0 | 1 |
| rack_1 | embedded | 1 | 0 |
| rack_2 | global | 0 | 1 |
| rack_2 | embedded | 1 | 1 |

Active Tasks: 0

Update spine config

- On Apstra versions 4.1.1 and 4.1.0, check the box on the top-left of the card for the pod to change. Then click the **Update spine config** button that appears above the card(s).



- From the **Spine Logical Device** drop-down list, select a different logical device. (The image below also shows that you can increase the count to add a spine to your topology. This became available in Apstra version 4.1.2.)

Update Spine Config

Count [ⓘ] *

2

Spine Logical Device

slicer-4x10-1 (embedded)

PANEL #1

| TOTAL | PORT GROUPS | Connected to ▾ |
|---------|--|----------------|
| 4 ports | 4 x 10 Gbps Superspine • Spine • Leaf • Access • Peer • Generic | |

1 3
2 4

Update

- Click **Update** to stage your changes and return to the **Pods** view.
Build errors appear because interface maps need to be assigned.
- Click the **Device Profiles** tab in the right panel and assign interface maps, as needed.

Delete Pod

When you delete a pod, all of its devices are removed from the blueprint; this could be highly impactful. Before deleting a pod that has live traffic on it, you may want to take its devices out-of-service by draining them. For more information, see the ["Drain Device Traffic" on page 82](#) page.

1. From the blueprint, navigate to **Staged > Physical > Pods**.
2. Select the check box(es) for the pod(s) to delete. (You must keep at least one pod.)

The screenshot shows the 'Staged' view of a blueprint, specifically the 'Physical' tab under 'Pods'. The interface includes a top navigation bar with 'Dashboard', 'Analytics', 'Staged', 'Uncommitted', and 'Active' tabs. Below this is a sub-navigation bar with 'Physical', 'Virtual', 'Policies', 'Catalog', 'Tasks', and 'Connectivity Templates'. The main content area shows a table of pods. The first pod, 'pod1', is selected with a checkmark. A red arrow points to the 'Delete' button (trash can icon) in the top left of the pod list. Another red arrow points to the 'pod1' checkbox, labeled '1. Select pod to delete'. The second pod, 'pod2', is not selected. The interface also shows filters for 'Nodes: All' and 'Links: All', a 'Layer' dropdown set to 'Uncommitted Changes', and a 'Page Size' of 25. The pod details for 'pod1' and 'pod2' are shown below the table, including capacity and a list of devices.

| 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 |

| 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 |

3. Click the **Delete** button (trash can) for the pod(s) to delete.
4. Click **Delete Pod** to stage the deletion and return to the table view.
5. ["Commit" on page 692](#) the changes to your blueprint. Configuration on any running devices is erased and the devices are ready to be decommissioned.

Planes (Datacenter)

IN THIS SECTION

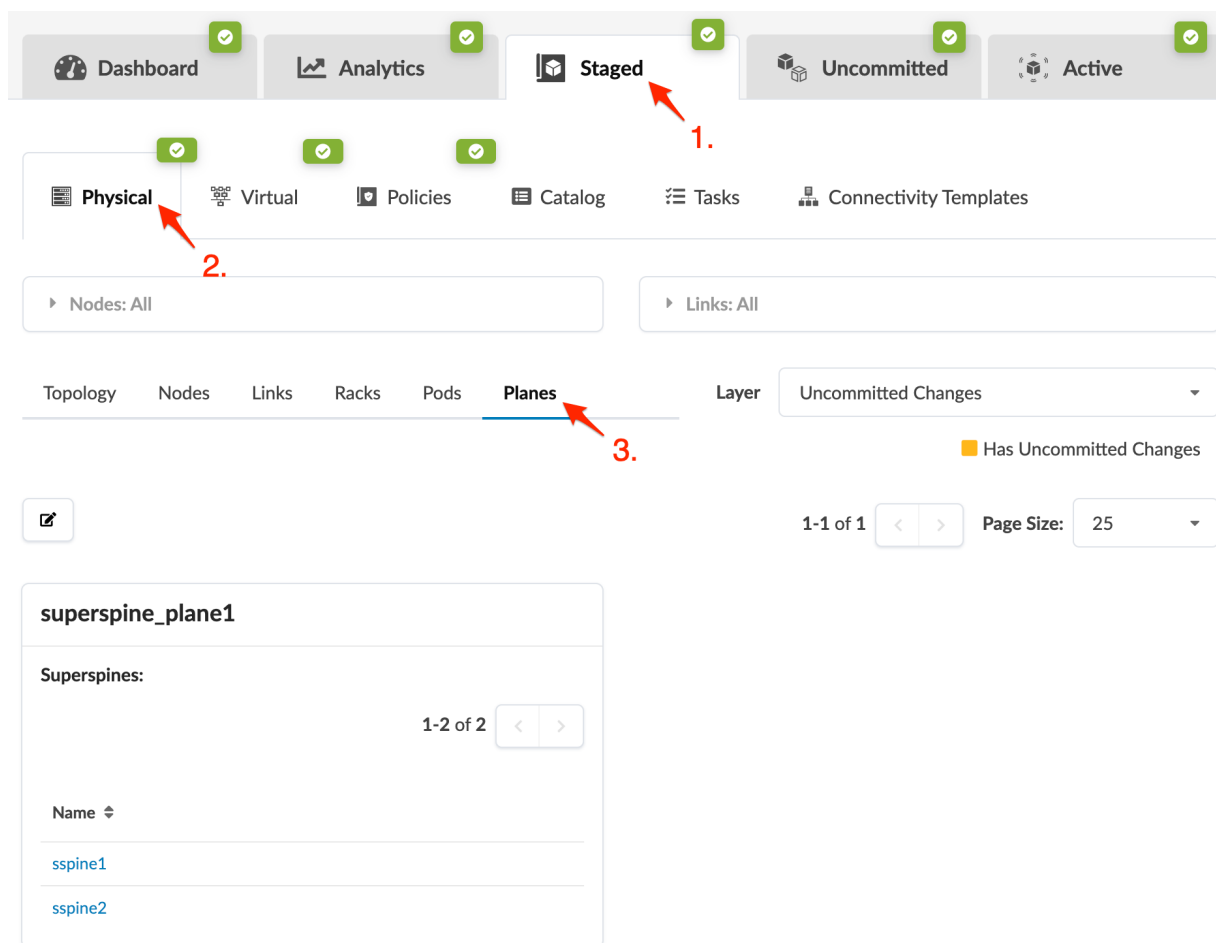
- [Add Superspine per Plane | 406](#)
- [Change Superspine Logical Device \(Plane\) | 408](#)

Planes are groups of superspines in 5-stage blueprints. Every 5-stage topology has at least one plane.

As a Day 2 operation, you can add superspines to planes in 5-stage Clos networks. The maximum number of superspines 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 280](#) of each new node. When the devices are physically ready, you can ["assign" on page 295](#) each node with their corresponding system IDs (serial numbers). When you ["commit pending changes" on page 692](#), the superspines are configured and they become part of the control and data plane, taking part of forward traffic between pods.

You can also change the superspine logical device on planes to add or update superspine port capacity on 5-stage blueprints. This change is for *all* planes (not per plane) which, based on the change, could be disruptive. Changing the logical device requires that you specify a different interface map, and possibly a new device profile.

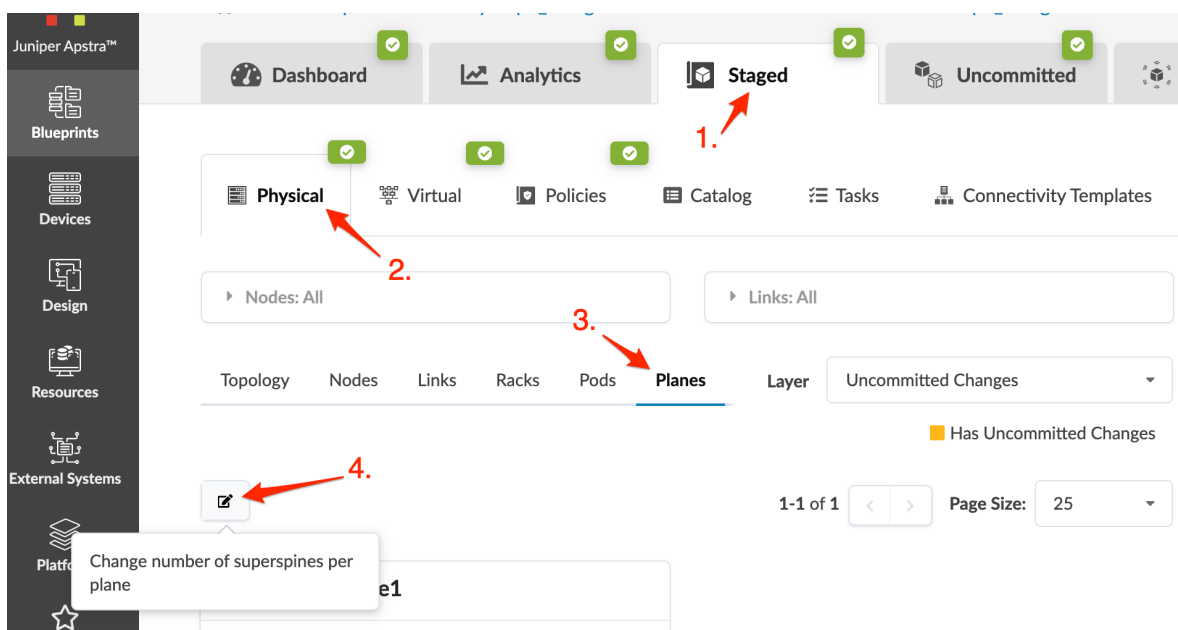
From the blueprint, navigate to **Staged > Physical > Planes** to go to the **Planes** view.



Add Superspine per Plane

As a Day 2 operation, you can add superspines per plane on 5-stage blueprints (as of Apstra version 4.1.0).

1. From the 5-stage blueprint, navigate to **Staged > Physical > Planes** and click the **Change number of superspines per plane** button.



2. In the **Superspines per plane** field, enter the total number of superspines you want. You can only add superspines per plane. Plan carefully. After you add superspines, you won't be able to remove them later.

Change number of superspines per plane

Plane Count

1

Superspines per plane ⓘ *

4

Superspine Logical Device

AOS-32x40-3 (embedded)

✕

PANEL #1

TOTAL

PORT GROUPS

Connected to ▾

32 ports

32 x 40
Gbps

Superspine •
Spine • Leaf •
Generic

| | | | | | | | | | | | | | | | |
|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|
| 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 |

Update

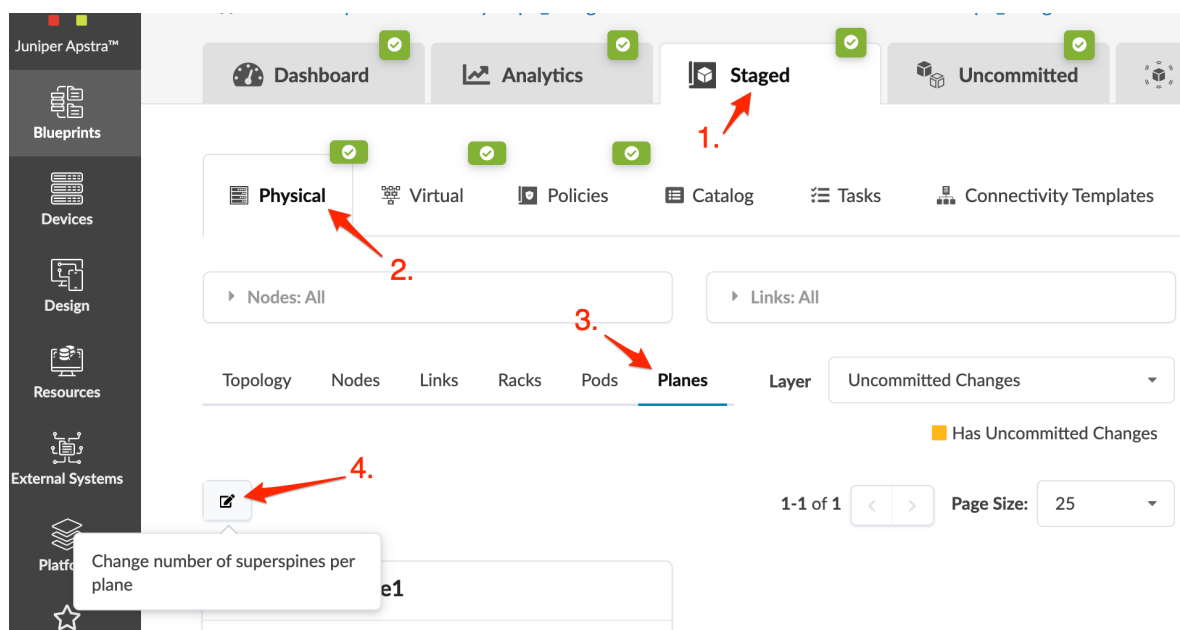
3. Click **Update** to stage your changes and return to the **Planes** view.

When you're ready to activate changes, commit them from the ["Uncommitted" on page 692](#) tab.

Change Superspine Logical Device (Plane)

As a Day 2 operation, you can change the superspine logical device on planes to add or update superspine ports capacity on 5-stage blueprints (as of Apstra version 4.1.0). This change is for *all* planes (not per plane) which, based on the change, could be disruptive. Changing the logical device requires that you specify a different interface map, and possibly a new device profile.

1. From the 5-stage blueprint, navigate to **Staged > Physical > Planes** and click the **Change number of superspines per plane** button.



2. Select a different logical device from the **Superspine Logical Device** drop-down list.
3. Click **Update** to stage your changes and return to the **Planes** view.
Build errors appear because interface maps need to be assigned.
4. Click the **Device Profiles** tab in the right panel and assign interface maps, as needed.

Virtual Networks

IN THIS SECTION

- [Assign Virtual Resources | 414](#)
- [Create Virtual Networks | 415](#)
- [Assign / Unassign Virtual Networks | 419](#)
- [Import / Export Virtual Networks | 422](#)
- [Edit Virtual Networks | 424](#)
- [Delete Virtual Networks | 426](#)

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 are either VLANs or VXLANs.

VLANs have the following characteristics:

- Single rack (rack-local)
- Single leaf devices or leaf pairs
- Can deploy in Layer 2-only mode (for example, isolated cluster networks for database replication)
- Can deploy 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 leaf devices or leaf pairs (MLAG)
- Can deploy in Layer 2-only mode
- Can deploy 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 section). For detailed capability information for a device, contact your network device vendor or ["Juniper Support" on page 824](#).

VNs contain the following details:

Table 16: Virtual Network Parameters

| Name | Description |
|------|--|
| Type | <ul style="list-style-type: none"> • VLAN (rack-local VN) • VXLAN (EVPN) (inter-rack VN) |
| Name | 30 characters or fewer. Underscore, dash, and alphanumeric characters only. |

Table 16: Virtual Network Parameters *(Continued)*

| Name | Description |
|---------------------------------------|--|
| Routing Zone | <ul style="list-style-type: none"> VLAN - default routing zone only (used for the underlay network) VXLAN - default routing zone or user-defined routing zone |
| Default VLAN ID (VLAN only) | <ul style="list-style-type: none"> Layer 2 VLAN ID on the switch that the VN is assigned to. If left blank, it's auto-assigned from static pool (2-4094). If you assign it, we don't recommend assigning VLAN ID 1 for active VNs. Cisco NX-OS reserves VLAN IDs 3968-4094. 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 SYSTEM 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> <ul style="list-style-type: none"> Using reserved VLAN IDs may cause deployment errors, but not build errors. |
| 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. |
| VLAN ID (on leaf devices) | VLAN ID |
| Reserve across blueprint (VXLAN only) | Option to use same VLAN ID on all leaf devices |

Table 16: Virtual Network Parameters *(Continued)*

| Name | Description |
|--|--|
| DHCP server | Enabled/Disabled - DHCP relay forwarder configuration on SVI. Implies L3 routing on SVI |
| IPv4 Connectivity | Enabled/Disabled - for SVI routing |
| IPv4 subnet (if connectivity is enabled) | <ul style="list-style-type: none"> IPv4 subnet - (for example: 192.168.100.0/24) (can't use batching VLANs) IPv4 CIDR length - automatically assigns a subnet with the specified length (for example: /26) If left blank, it's auto-assigned a /24 subnet network from resource pools |
| Virtual Gateway IPv4 | The IPv4 address, if enabled |
| IPv6 Connectivity | Enabled/Disabled - IPv6 connectivity for SVI routing. You must enable IPv6 in blueprint. If the template uses IPv4 spine-to-leaf link types, you can't use IPv6 in default routing zone and for VLAN type VNs. |
| IPv6 subnet (if connectivity is enabled) | <ul style="list-style-type: none"> IPv6 subnet (for example: 2001:4de0::/64) IPv6 CIDR length - automatically assigns a subnet with the specified length (for example: /56) If left blank, it's auto-assigned a /64 subnet network from resource pools. If assigned automatically, the IP is derived from the assigned VNs SVI pools. To assign multiple VLAN networks, leave blank or specify CIDR length. |
| Virtual Gateway IPv6 | The IPv6 address, if enabled |
| Create connectivity templates for | <ul style="list-style-type: none"> Tagged Untagged |
| Assigned to | The racks that the VN is assigned to. For more information, see table below. |

Table 17: Virtual Network Rack (or Pod) Details

| Assigned To Details | Description |
|------------------------------|---|
| Pod Name (5-stage) | 5-stage Clos networks include pods, and you can select leaf devices within each pod to extend VNs 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 use for batch creating VNs |
| Secondary IP Allocation Mode | <ul style="list-style-type: none"> • Enabled (default) - Apstra decides whether a secondary IP address is needed. <ul style="list-style-type: none"> • Automatically allocate if an assigned connectivity template requires an address (BGP, Static routes). • VN of type VXLAN: <ul style="list-style-type: none"> • Some NOS types automatically allocate unicast IPv4 addresses when an anycast IPv4 gateway is present: (Junos when in an ESI pair). • If a NOS type forbids co-existence of an anycast IPv4 address with an unicast IPv4 address, a blueprint error will be raised (Sonic). • VN of type VLAN - All NOS types require unicast IPv4 addresses when the IPv4 anycast address is enabled. • Forced - A secondary IP address is rendered irrespective of whether or not a connectivity template requires it. <ul style="list-style-type: none"> • If a NOS type forbids co-existence of an anycast IPv4 address with a unicast IPv4 address, a blueprint error will be raised. • Permits you to manually create an optional unicast IPv4 address for purposes such as BGP peering or static routing. |
| IPv4 Address / IPv6 Address | You can 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 virtual network table view. You can create, edit and delete virtual networks, and as of Apstra version 4.1.2 you can also import and export virtual networks.

1. Click **Staged** in the top navigation bar.

2. Click **Virtual** in the left sidebar.

3. Click **Virtual Networks** in the left sidebar.

Click VN name for details

| Name | Routing Zone | Type | VN ID | Assigned to | IPv4 Connectivity | IPv4 Subnet | IPv6 Connectivity | IPv6 Subnet | Actions |
|-----------------------------|--------------|-------|-------|-------------|-------------------|-------------|-------------------|-------------|---------|
| blue_300_evpn_mlag_001_l_v4 | blue | VXLAN | 40000 | 1 nodes | Enabled | 20.1.0.0/24 | Disabled | N/A | |

Build

- 19/19 red: Virtual Network SVI Subnets
- 19/19 blue: Virtual Network SVI Subnets
- 16/16 red: Virtual Network SVI Subnets (IPv6)
- 16/16 blue: Virtual Network SVI Subnets (IPv6)

Assign Virtual Resources

IN THIS SECTION

- [Update Virtual Resources Assignments | 414](#)
- [Reset Virtual Resource Group Overrides | 415](#)

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.

Create Virtual Networks

IN THIS SECTION

- Create Virtual Networks (using GUI) | 416
- Create Virtual Networks (using CSV File) | 417

Create Virtual Networks (using GUI)

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 unique name.

Create Virtual Network

Virtual Network Parameters

Type
☐ VLAN ☒ VXLAN

i Will create single VXLAN for all selected nodes

Name ^{*} Routing Zone

VNI(s) [?] VLAN ID (on leafs) Reserve across blueprint ☐

Route Target [?]

| | | | | |
|--|---|--|--|---|
| DHCP Service <input checked="" type="radio"/> Disabled <input type="radio"/> Enabled | IPv4 Connectivity <input type="radio"/> Disabled <input checked="" type="radio"/> Enabled | IPv4 Subnet <input type="text" value="From resource pool"/> | Virtual Gateway IPv4 Enabled? <input checked="" type="checkbox"/> | Virtual Gateway IPv4 <input type="text" value="From resource pool"/> |
| | IPv6 Connectivity <input checked="" type="radio"/> Disabled <input type="radio"/> Enabled | | | |

Create Connectivity Templates for
☐ Tagged ☐ Untagged

3. Select the "routing zone" on page 429 to associate with the VN(s). (VLANs must use the default routing zone.)
4. If you're creating VLANs, you can specify the default VLAN ID(s) or leave it blank to automatically assign it from a resource pool.
5. If you're creating VXLANs, you can specify VNIs or leave it blank to automatically assign it from a resource pool.
6. If you're creating VXLANs and you enter a VLAN ID (on leaf devices), you can select the check box to **Reserve across blueprint**. This enforces the same rule across the fabric and helps you to honor the same VLAN policy across racks when adding new racks.
7. 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.)
8. 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.
9. If you enable **Virtual Gateway IPv4**, enter an IPv4 address.
10. 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.
11. If you enable **Virtual Gateway IPv6**, enter an IPv6 address.

12. To create connectivity templates for the VN(s), check the box for **Tagged** and/or **Untagged**, as applicable.
13. Select and configure racks to assign to the VN. See [Virtual Networks on page 413](#) overview for details.

Assigned To

Query: All 1-2 of 2 Page Size: 25

| <input type="checkbox"/> | Bound To | Link Labels | VLAN ID | Secondary IP Allocation Mode | IPv4 Address |
|--------------------------|--------------------------------|-------------|--------------------|--|--|
| <input type="checkbox"/> | I2_esl_2x_links_001_leaf_pair1 | link | From resource pool | I2_esl_2x_links_001_leaf1 Enabled Enabled Forced | I2_esl_2x_links_001_leaf1 From resource pool I2_esl_2x_links_001_leaf2 From resource pool |
| <input type="checkbox"/> | I2_esl_2x_links_002_leaf_pair1 | link | From resource pool | I2_esl_2x_links_002_leaf1 Enabled I2_esl_2x_links_002_leaf2 Enabled | I2_esl_2x_links_002_leaf1 From resource pool I2_esl_2x_links_002_leaf2 From resource pool |

Route Target Policies

Import Route Targets

[Add Import Route Target](#)

Export Route Targets

[Add Export Route Target](#)

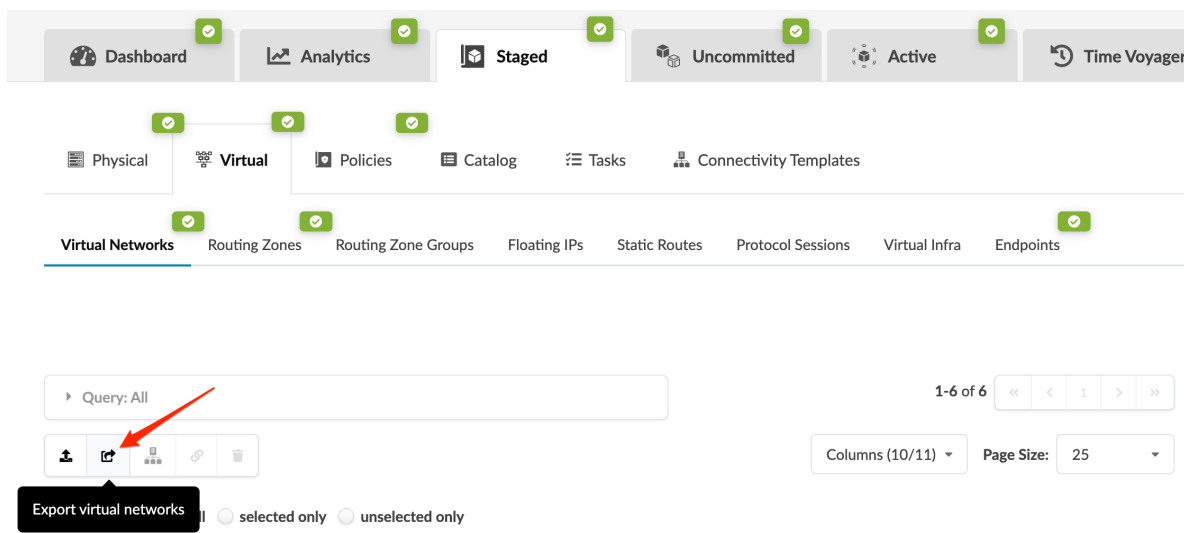
☐ Create Another? [Create](#)

14. Click **Create** to stage the VN and return to the table view.
15. Assign IPv4 (IPv6) resources for SVI subnets. Navigate to **Staged > Virtual > Virtual Networks** and ["assign resources" on page 277](#) in the **Build** panel (right-side).
16. 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.
17. To deploy changes to the active blueprint, click the ["Uncommitted" on page 692](#) tab to review and commit (or discard) changes.

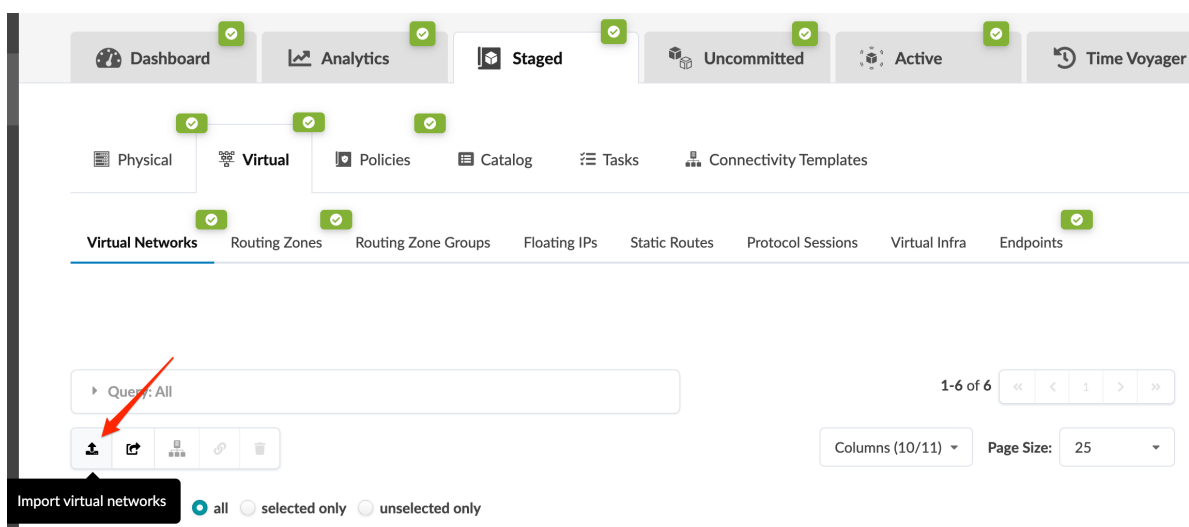
Create Virtual Networks (using CSV File)

You can create many virtual networks at once with a CSV file, as of Apstra version 4.1.2. First, you'll export the virtual network schema from your blueprint, then open and populate the file in a spreadsheet program. And finally, you'll import the file back into your blueprint.

1. From the blueprint, navigate to **Staged > Virtual > Virtual Networks** and click **Export virtual networks**.



2. Click **Copy** to copy the contents or click **Save As File** to download the file. (Close the dialog to return to the table view.)
3. Paste the contents, or open the CSV file, in a spreadsheet program (such as Google Sheets or Microsoft Excel). (Any virtual networks that were previously created are included in the file.)
4. Enter virtual networks details into the spreadsheet leaving the `vn_node_id` field blank for new VNs, then save the file.
5. In the Apstra GUI, navigate to **Staged > Virtual > Virtual Networks** and click **Import virtual networks**.



6. Either click **Choose File** and navigate to the file on your computer, drag and drop the file onto the dialog window, or as shown in the screenshot below, directly paste CSV file contents. Virtual network details are displayed for your review.

Headers marked with an asterisk (*) are mandatory.

```
vn_node_id - virtual network graph node ID (string): leave empty to create a new VN, automatically populated for existing VN and must be kept unchanged to update an existing VN
vn_name* - virtual network name (string)
rz_name* - routing zone name (string)
vn_type* - virtual network type: 'vlan' or 'vxdm'
vn_id - virtual network ID (number)
reserved_vlan_id - reserved virtual network ID (number)
dhcp_service - 'X' or 'x' - for DHCP service enabled, or leave an empty string if not
ipv4_enabled - 'X' or 'x' - for IPv4 enabled, or leave an empty string if not
ipv6_enabled - 'X' or 'x' - for IPv6 enabled, or leave an empty string if not
virtual_gateway_ipv4_enabled - 'X' or 'x' - for Virtual gateway IPv4 enabled, or leave an empty string if not
virtual_gateway_ipv6_enabled - 'X' or 'x' - for Virtual gateway IPv6 enabled, or leave an empty string if not
ipv4_subnet - should be a valid IPv4 subnet (an empty string for IPv4 disabled)
ipv6_subnet - should be a valid IPv6 subnet (an empty string for IPv6 disabled)
virtual_gateway_ipv4_subnet - should be a valid IPv4 subnet (an empty string for Virtual gateway IPv4 disabled)
virtual_gateway_ipv6_subnet - should be a valid IPv6 subnet (an empty string for Virtual gateway IPv6 disabled)
bound_to - **** - for virtual network bound to node, where '*' is a node label; a number in the range 1-4094 OR 'X' or 'x' for access switches, only 'X' or 'x' are allowed
```

► Query: All

1-3 of 3

Page Size: 10

| node_id | vs_name | v2_name | vs_type | vs_id | reserved_vlan_id | dhcp_service | ipv4_enabled | ipv6_enabled | virtual_gateway_ipv4_enabled | virtual_gateway_ipv6_enabled | ipv4_subnet | ipv6_subnet | virtual_gateway_ipv4 | virtual_gateway_ipv6 | bound_to_v2_virtual_001_host1 | bound_to_v2_virtual_002_host1 |
|--------------|---------|---------|---------|-------|------------------|--------------|--------------|--------------|------------------------------|------------------------------|--------------|-------------|----------------------|----------------------|-------------------------------|-------------------------------|
| blue-net-302 | blue | vlan | 30002 | | | | x | | x | | 10.0.32.0/24 | | 10.0.32.1 | | 302 | 302 |
| blue-net-303 | blue | vlan | 30003 | | | | x | | x | | 10.0.33.0/24 | | 10.0.33.1 | | 303 | 303 |
| blue-net-304 | blue | vlan | 30004 | | | | x | | x | | 10.0.34.0/24 | | 10.0.34.1 | | 304 | 304 |

Import

Assign virtual resources.

Assign / Unassign Virtual Networks

- Assign / Unassign One Virtual Network | 419
- Assign / Unassign Multiple Virtual Networks | 420

1. Either from the table view (Staged > Virtual > Virtual Networks) or the details view, click the **Edit** button for the virtual network to update.

2. In the dialog that opens, scroll past the **Virtual Network Parameters** section to the **Assigned To** section:

- Assign the VN to one or more nodes by selecting the applicable node check box(es).
- Unassign the VN from one or more nodes by deselecting the applicable node check box(es).

Edit Virtual Network

Assigned To

Query: All 1-2 of 2 Page Size: 25

Select to assign, deselect to unassign

| <input checked="" type="checkbox"/> | Bound To | VLAN ID | Secondary IP Allocation Mode | IPv4 Address |
|-------------------------------------|--------------------------------|---------|---|--|
| <input checked="" type="checkbox"/> | I2_es1_2x_links_001_leaf_pair1 | 6 | I2_es1_2x_links_001_leaf1 <input type="text" value="Enabled"/> | I2_es1_2x_links_001_leaf1 <input type="text" value="From resource pool"/> |
| <input checked="" type="checkbox"/> | I2_es1_2x_links_002_leaf_pair1 | 6 | I2_es1_2x_links_002_leaf1 <input type="text" value="Enabled"/> | I2_es1_2x_links_002_leaf1 <input type="text" value="From resource pool"/> |

3. Click **Update** to stage the changes and return to the table view.

Assign / Unassign Multiple Virtual Networks

You can assign/unassign many virtual networks at the same time (as of Apstra version 4.1.2). This is especially useful when you've added a rack as a Day 2 operation and you need to assign a lot of virtual networks to it.

1. From the table view (Staged > Virtual > Virtual Networks) select one or more check boxes for the VNs to update.
2. Click the **Assign selected VXLAN networks** button that becomes available above the table (fourth of five buttons).

Dashboard

Analytics

Staged

Uncommitted

Active

Physical

Virtual

Policies

Catalog

Tasks

Connectivity Templates

Virtual Networks

Routing Zones

Routing Zone Groups

Floating IPs

Static Routes

Protocol Sessions

Remote EVPN Gateways

Query: All

1-3 of 3

Columns (10/11)

Page Size: 25

Assign selected VXLAN networks

unselected only

| | Name | Routing Zone | Type | VN ID | Assigned to | IPv4 Connectivity | IPv4 Subnet | IPv6 Connectivity | IPv6 Subnet | Actions |
|-------------------------------------|--------|--------------|-------|-------|-------------|-------------------|-------------|-------------------|-------------|---------|
| <input checked="" type="checkbox"/> | blue-1 | blue | VXLAN | 10004 | 2 nodes | Enabled | 10.0.3.0/24 | Disabled | N/A | |
| <input checked="" type="checkbox"/> | red-1 | red | VXLAN | 10002 | 2 nodes | Enabled | 10.0.1.0/24 | Disabled | N/A | |

3. In the dialog that opens, you can see the associated routing zone, VN type and VN ID by hovering over the VNs that are already assigned.

Assign Selected VXLAN Networks

Pod: All

Rack: All

Node: All

Assigned VNs: All

Tags: All

Selected VXLANs

blue-1 red-1

Bulk assign VXLANs

Bulk unassign VXLANs

1-2 of 2

Page Size: 25

| | Pod | Rack | Node | Role | Tags | Assigned VNs |
|--------------------------|------|---------------------|---------------------------------|-----------|------|--------------------|
| <input type="checkbox"/> | pod1 | I2_esl_2x_links_001 | I2_esl_2x_links_001_I_eaf_pair1 | Leaf Pair | | blue-1 red-1 |
| <input type="checkbox"/> | pod1 | I2_esl_2x_links_002 | I2_esl_2x_links_002_I_eaf_pair1 | Leaf Pair | | blue-1 red-1 red-2 |

Hover over VN name for info

Assign

4. Your selected VXLANs appear above the table on the left. The table shows the VNs that are already assigned to nodes in the network. Select the check boxes for one or more nodes. The **Bulk assign VXLANs** and **Bulk unassign VXLANs** buttons become available.

Assign Selected VXLAN Networks

Pod: All | Rack: All | Node: All | Assigned VNs: All | Tags: All

Selected VXLANs: blue-1 | red-1

Bulk assign VXLANs | **Bulk unassign VXLANs** | 1-2 of 2 | Page Size: 25

| <input type="checkbox"/> | Pod | Rack | Node | Role | Tags | Assigned VNs (to be removed to be added) |
|-------------------------------------|------|---------------------|---------------------------------|-----------|------|--|
| <input checked="" type="checkbox"/> | pod1 | I2_esl_2x_links_001 | I2_esl_2x_links_001_I_eaf_pair1 | Leaf Pair | | blue-1 <input checked="" type="checkbox"/> red-1 <input checked="" type="checkbox"/> red-2 |
| <input type="checkbox"/> | pod1 | I2_esl_2x_links_002 | I2_esl_2x_links_002_I_eaf_pair1 | Leaf Pair | | blue-1 <input type="checkbox"/> red-1 <input type="checkbox"/> red-2 |

Assign

5. Assign and unassign virtual networks, as needed:

- To assign your selected VXLANs to the nodes you just selected, click the **Bulk assign VXLANs** button. The VNs to be assigned turn green.
- To unassign your selected VXLANs that are already assigned to the nodes you just selected, click the **Bulk unassign VXLANs** button. The VNs to be unassigned turn red (as shown in the screenshot example above).

6. Click **Assign** to stage your changes and return to the table view.

Import / Export Virtual Networks

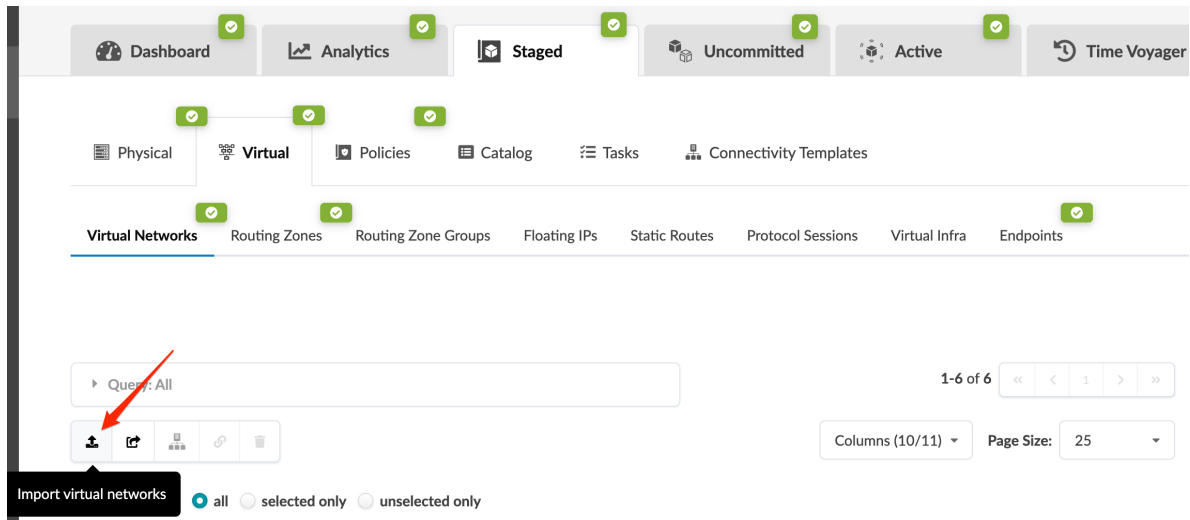
IN THIS SECTION

- [Import Virtual Networks | 422](#)
- [Export Virtual Networks | 423](#)

Import Virtual Networks

You can import multiple virtual networks (as a CSV file) into your blueprint, as of Apstra version 4.1.2. (Tip: First export virtual networks so you'll have the schema set up for you in the CSV file.)

- From the blueprint, navigate to **Staged > Virtual > Virtual Networks** and click the **Import virtual networks** button.



2. Either click **Choose File** and navigate to the file on your computer, drag and drop the file onto the dialog window, or as shown in the screenshot below, directly paste CSV file contents. Virtual network details are displayed for your review.

Import Virtual Networks (CSV) ✕

* See help to get more information about allowed CSV headers and values.

Headers marked with an asterisk (*) are mandatory.

vn_node_id - virtual network graph node ID (string); leave empty to create a new VN, automatically populated for existing VN and must be kept unchanged to update an existing VN
 vn_name(*) - virtual network name (string)
 rz_name(*) - routing zone name (string)
 vn_type(*) - virtual network type: 'vlan' or 'vxlan'
 vn_id - virtual network ID (number)
 reserved_vlan_id - reserved virtual network ID (number)
 dhcp_service - 'x' or 'X' - for DHCP service enabled, or leave an empty string if not
 ipv4_enabled - 'x' or 'X' - for IPv4 enabled, or leave an empty string if not
 ipv6_enabled - 'x' or 'X' - for IPv6 enabled, or leave an empty string if not
 virtual_gateway_ipv4_enabled - 'x' or 'X' - for Virtual gateway IPv4 enabled, or leave an empty string if not
 virtual_gateway_ipv6_enabled - 'x' or 'X' - for Virtual gateway IPv6 enabled, or leave an empty string if not
 ipv6_subnet - should be a valid IPv6 subnet (an empty string for IPv6 disabled)
 virtual_gateway_ipv4 - should be a valid IPv4 subnet (an empty string for Virtual gateway IPv4 disabled)
 virtual_gateway_ipv6 - should be a valid IPv6 subnet (an empty string for Virtual gateway IPv6 disabled)
 bound_to_***(*) - for virtual network bounded to node, where *** is a node label - should be a number in the range 1-4094 OR 'x' or 'X' (for access switches, only 'x' or 'X' are allowed)

Drag and drop file here, choose it by clicking the button or paste its contents in the field below. Choose File

```

1 vn_node_id,vn_name,rz_name,vn_type,vn_id,reserved_vlan_id,dhcp_service,ipv4_enabled,ipv6_enabled,virtual_gateway_ipv4_enabled,virtual_gateway_ipv6_enabled,ipv6_subnet,virtual_gateway_ipv4,virtual_gateway_ipv6,bound_to_12_virtual_001_leaf1,bound_to_12_virtual_002_leaf1
2 blue-net-302,blue,vxlan,30002,,x,x,,10.0.32.0/24,,10.0.32.1,,302,302,302,302
3 blue-net-303,blue,vxlan,30003,,x,x,,10.0.33.0/24,,10.0.33.1,,303,303,303,303
4 blue-net-304,blue,vxlan,30004,,x,x,,10.0.34.0/24,,10.0.34.1,,304,304,304,304
  
```

Query: All 1-3 of 3

Page Size: 10

| vn_node_id | vn_name | rz_name | vn_type | vn_id | reserved_vlan_id | dhcp_service | ipv4_enabled | ipv6_enabled | virtual_gateway_ipv4_enabled | virtual_gateway_ipv6_enabled | ipv6_subnet | virtual_gateway_ipv4 | virtual_gateway_ipv6 | bound_to_12_virtual_001_leaf1 | bound_to_12_virtual_002_leaf1 |
|--------------|---------|---------|---------|-------|------------------|--------------|--------------|--------------|------------------------------|------------------------------|-------------|----------------------|----------------------|-------------------------------|-------------------------------|
| blue-net-302 | blue | vxlan | 30002 | | | x | x | | 10.0.32.0/24 | 10.0.32.1 | | 302 | 302 | | |
| blue-net-303 | blue | vxlan | 30003 | | | x | x | | 10.0.33.0/24 | 10.0.33.1 | | 303 | 303 | | |
| blue-net-304 | blue | vxlan | 30004 | | | x | x | | 10.0.34.0/24 | 10.0.34.1 | | 304 | 304 | | |

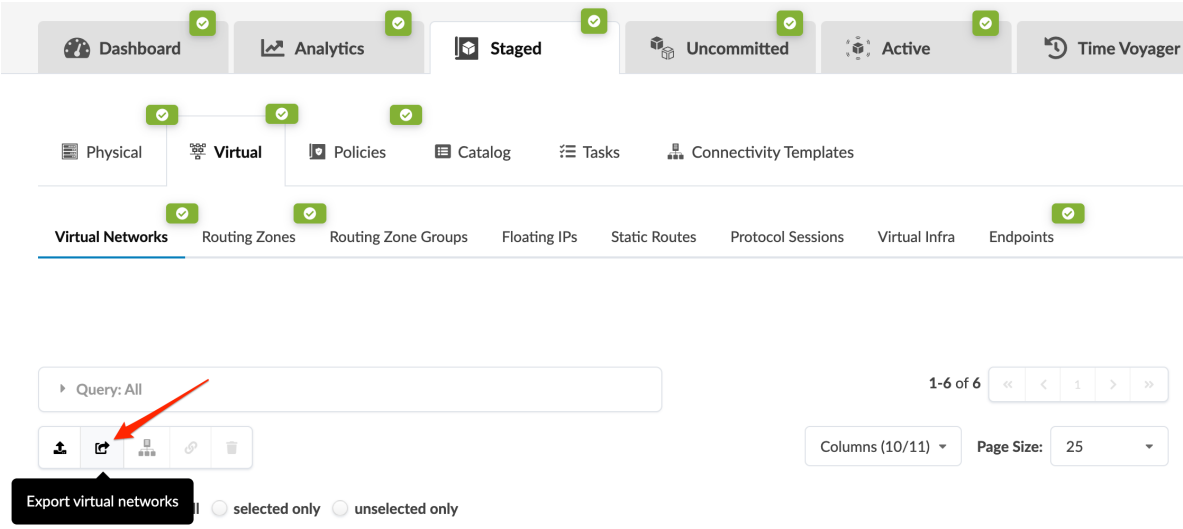
Import

3. Click **Import** to import the virtual networks, stage the changes, and return to the table view.

Export Virtual Networks

You can export virtual networks (as a CSV file) from your blueprint, as of Apstra version 4.1.2.

1. From the blueprint, navigate to **Staged > Virtual > Virtual Networks** and click **Export virtual networks**.



2. Click **Copy** to copy the contents or click **Save As File** to download the file.
3. When you've copied or downloaded the virtual networks, close the dialog to return to the table view.

Edit Virtual Networks

IN THIS SECTION

- [Edit One Virtual Network | 424](#)
- [Edit Multiple Virtual Networks | 425](#)

Edit One Virtual Network

1. From the blueprint, navigate to **Staged > Virtual > Virtual Networks** and click the **Edit** button in the **Actions** panel for the virtual network to edit.

The screenshot shows the Apstra GUI with the 'Staged' tab selected. The 'Virtual' section is active, and the 'Virtual Networks' sub-tab is selected. Below the navigation bar, there is a search bar with 'Query: All' and pagination controls showing '1-6 of 6'. Below the search bar, there are icons for various actions and a 'Filter selected by' section with radio buttons for 'all', 'selected only', and 'unselected only'. The main table displays virtual network information. A red arrow points to the 'Edit' button in the Actions column for the first row.

| | Name | Routing Zone | Type | VN ID | Assigned to | IPv4 Connectivity | IPv4 Subnet | IPv6 Connectivity | IPv6 Subnet | Actions |
|--------------------------|--------------------------------|--------------|------|-------|-------------|-------------------|---------------|-------------------|-------------|---------|
| <input type="checkbox"/> | vnet_10_on_rack_1_001_leaf_pai | default | VLAN | 10 | 1 nodes | Enabled | 172.16.1.0/24 | Disabled | N/A | |
| <input type="checkbox"/> | vnet_10_on_rack_2_001_leaf1 | default | VLAN | 10 | 1 nodes | Enabled | 172.16.2.0/24 | Disabled | N/A | |

2. Make your changes.
3. Click **Update** to stage the changes and return to the table view.

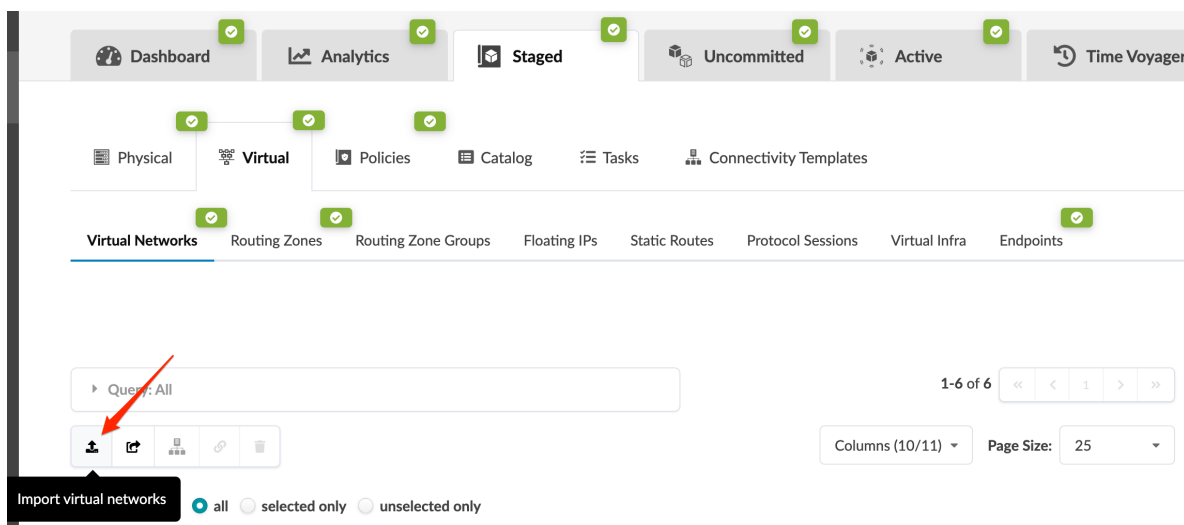
Edit Multiple Virtual Networks

As of Apstra version 4.1.2, you can edit multiple virtual networks using the Apstra GUI. You'll export all virtual networks from your blueprint as a CSV file, edit the file in a spreadsheet program, then import the file back into your blueprint.

1. From the blueprint, navigate to **Staged > Virtual > Virtual Networks** and click **Export virtual networks**.

The screenshot shows the Apstra GUI with the 'Staged' tab selected. The 'Virtual' section is active, and the 'Virtual Networks' sub-tab is selected. Below the navigation bar, there is a search bar with 'Query: All' and pagination controls showing '1-6 of 6'. Below the search bar, there are icons for various actions. A red arrow points to the 'Export virtual networks' button, which is labeled 'Export virtual networks' in a tooltip.

2. Click **Copy** to copy the contents, or click **Save As File** to download the file. (Close the dialog to return to the table view.)
3. Paste the contents, or open the CSV file, in a spreadsheet program (such as Google Sheets or Microsoft Excel).
4. Update virtual networks as needed, then save the file.
5. In the Apstra GUI, navigate to **Staged > Virtual > Virtual Networks** and click **Import virtual networks**.



6. Either click **Choose File** and navigate to the file on your computer, or drag and drop the file onto the dialog window. Virtual network details are displayed for your review.
7. Click **Import** to import the virtual networks, stage the changes, and return to the table view.

Delete Virtual Networks

IN THIS SECTION

- [Delete One Virtual Network | 426](#)
- [Delete Multiple Virtual Networks | 427](#)

Delete One Virtual Network

Virtual networks with assigned connectivity templates can't be deleted.

1. From the blueprint, navigate to **Staged > Virtual > Virtual Networks** and click the **Delete** button in the **Actions** panel for the VN to delete.

The screenshot shows the Apstra GUI navigation menu with tabs: Dashboard, Analytics, Staged, Uncommitted, Active, and Time Voyager. Under the Staged tab, there are sub-tabs: Physical, Virtual, Policies, Catalog, Tasks, and Connectivity Templates. The Virtual Networks sub-tab is selected. Below the navigation, there is a search bar with 'Query: All' and a pagination control showing '1-17 of 17'. There are also icons for upload, share, list, link, and delete, along with 'Columns (10/11)' and 'Page Size: 25' dropdowns. A filter section shows 'Filter selected by' with radio buttons for 'all' (selected), 'selected only', and 'unselected only'. The main table has columns: Name, Routing Zone, Type, VN ID, Assigned to, IPv4 Connectivity, IPv4 Subnet, IPv6 Connectivity, IPv6 Subnet, and Actions. The table contains four rows of virtual networks. A red arrow points to the Delete icon (trash can) in the Actions column of the last row.

| Name | Routing Zone | Type | VN ID | Assigned to | IPv4 Connectivity | IPv4 Subnet | IPv6 Connectivity | IPv6 Subnet | Actions |
|-----------------------------|--------------|-------|-------|-------------|-------------------|-------------|-------------------|-------------|-----------------------------|
| blue_300_evpn_esi_001_le_v4 | blue | VXLAN | 40000 | 1 nodes | Enabled | 20.1.0.0/24 | Disabled | N/A | [Edit] [Up] [Down] [Delete] |
| blue_301_leaf3_v4 | blue | VXLAN | 40001 | 1 nodes | Enabled | 20.1.1.0/24 | Disabled | N/A | [Edit] [Up] [Down] [Delete] |
| blue_vxlan_31_v4_1 | blue | VXLAN | 30000 | 2 nodes | Enabled | 10.1.4.0/24 | Disabled | N/A | [Edit] [Up] [Down] [Delete] |
| blue_vxlan_33_v4_no_eps | blue | VXLAN | 30002 | 2 nodes | Enabled | 10.1.6.0/24 | Disabled | N/A | [Edit] [Up] [Down] [Delete] |

2. Click **Delete** to stage the deletion and return to the table view.

Delete Multiple Virtual Networks

Virtual networks with assigned connectivity templates can't be deleted.

You can delete multiple virtual networks from one dialog in the Apstra GUI, as of Apstra version 4.1.2.

1. From the blueprint, navigate to **Staged > Virtual > Virtual Networks** and select the check boxes for the virtual networks to delete. (Tip: Use the Query function to filter specific virtual networks.)

Query: Type = VXLAN 1-16 of 16 Columns (10/11) Page Size: 25

Filter selected unselected only

| | Name | Routing Zone | Type | VN ID | Assigned to | IPv4 Connectivity | IPv4 Subnet | IPv6 Connectivity | IPv6 Subnet | Actions |
|-------------------------------------|------------------------------|--------------|-------|-------|-------------|-------------------|--------------|-------------------|-------------|----------------------------|
| <input type="checkbox"/> | blue_vxlan_31_v4_ep | blue | VXLAN | 30000 | 2 nodes | Enabled | 10.1.4.0/24 | Disabled | N/A | [Edit] [Up] [Down] [Trash] |
| <input checked="" type="checkbox"/> | red_vxlan_32_v4_1 | red | VXLAN | 30001 | 2 nodes | Enabled | 10.1.5.0/24 | Disabled | N/A | [Edit] [Up] [Down] [Trash] |
| <input checked="" type="checkbox"/> | blue_vxlan_33_v4_no_eps | blue | VXLAN | 30002 | 2 nodes | Enabled | 10.1.6.0/24 | Disabled | N/A | [Edit] [Up] [Down] [Trash] |
| <input type="checkbox"/> | blue_vxlan_34_v4_one_ep | blue | VXLAN | 30003 | 2 nodes | Enabled | 10.1.7.0/24 | Disabled | N/A | [Edit] [Up] [Down] [Trash] |
| <input type="checkbox"/> | blue_vxlan_35_v4_one_ep | blue | VXLAN | 30004 | 2 nodes | Enabled | 10.1.8.0/24 | Disabled | N/A | [Edit] [Up] [Down] [Trash] |
| <input type="checkbox"/> | blue_vxlan_36_v4_one_ep | blue | VXLAN | 30005 | 2 nodes | Enabled | 10.1.9.0/24 | Disabled | N/A | [Edit] [Up] [Down] [Trash] |
| <input type="checkbox"/> | blue_vxlan_37_v4_one_ep_mlag | blue | VXLAN | 30006 | 2 nodes | Enabled | 10.1.10.0/24 | Disabled | N/A | [Edit] [Up] [Down] [Trash] |
| <input checked="" type="checkbox"/> | red_vxlan_38_v4_no_eps | red | VXLAN | 30007 | 2 nodes | Enabled | 10.1.11.0/24 | Disabled | N/A | [Edit] [Up] [Down] [Trash] |

- Click the **Delete selected virtual networks** button that becomes available. Click the drop-down triangle to show (or hide) selected VN names. The virtual networks that will be deleted are listed. If you've selected virtual networks that have connectivity templates assigned to them, they are listed as not being available for deletion. They'll be ignored.

Delete this resource?

Click to show (or hide) VN names

The following virtual networks (2) are going to be deleted:

- Virtual networks to be deleted
 - blue_vxlan_33_v4_no_eps
 - red_vxlan_38_v4_no_eps

The following virtual networks (1) have connectivity template(s) assigned and cannot be deleted:

- Virtual networks not available for deletion
 - red_vxlan_32_v4_1

A VN with assigned CTs was selected.

Delete

- Click **Delete** to stage the deletion and return to the table view.

Routing Zones

IN THIS SECTION

- [Create Routing Zones | 431](#)
- [Assign DHCP Server to Routing Zone | 433](#)
- [Assign Resources to Routing Zone | 434](#)
- [Import / Export Routing Zones | 435](#)
- [Edit Routing Zones | 437](#)
- [Delete Routing Zones | 439](#)

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:

| Parameter | Description |
|-----------|---|
| 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. |

(Continued)

| Parameter | Description |
|-----------------------|---|
| 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 Data Center 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 "Routing Policies" on page 543 . |
| Route Target Policies | <ul style="list-style-type: none"> • Import Route Targets • Export Route Targets |
| Resources | |
| Virtual Networks | |
| Interfaces | |

From the blueprint, navigate to **Staged > Virtual > Routing Zones** to go to the routing zones table view. You can create, edit and delete routing zones and assign DHCP servers to them. As of Apstra version

4.1.2 you can also import and export routing zones.

1. Staged

2. Virtual

3. Routing Zones

Query: All

1-3 of 3

Page Size: 25

| 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 |

Create Routing Zone

- 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 Zones

IN THIS SECTION

- Create Routing Zones (using GUI) | 431
- Create Routing Zones (using CSV File) | 432

Create Routing Zones (using GUI)

You can create routing zones if your blueprint is using **MP-EBGP EVPN** overlay control protocol. If it's using **Static VXLAN**, you must use the default routing zone. (Overlay control protocol is specified in "templates" on page 35.)

- From the blueprint, navigate to **Staged > Virtual > Routing Zones** and click **Create Routing Zone**.
- Enter a unique VRF name (15 characters or fewer).

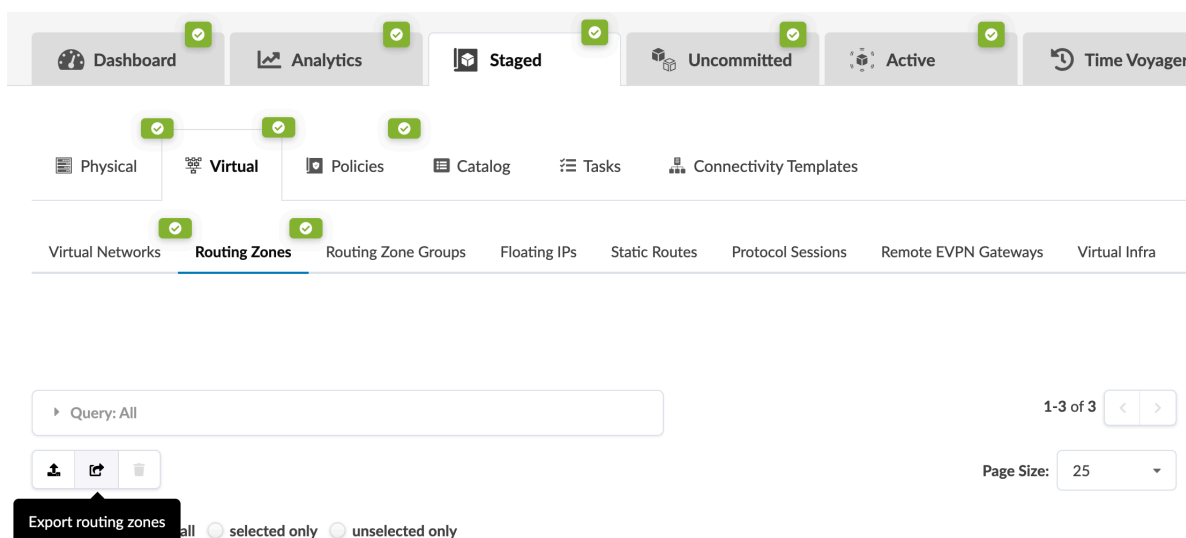
3. You can leave the remaining fields as is to use default values and have resources assigned from pools, or you can configure them manually. See the ["routing zone" on page 429](#) overview for details.
4. Click **Create** to create the routing zone and return to the table view.

Assign resources (leaf loopback IPs, leaf L3 peer links) to the new routing zone.

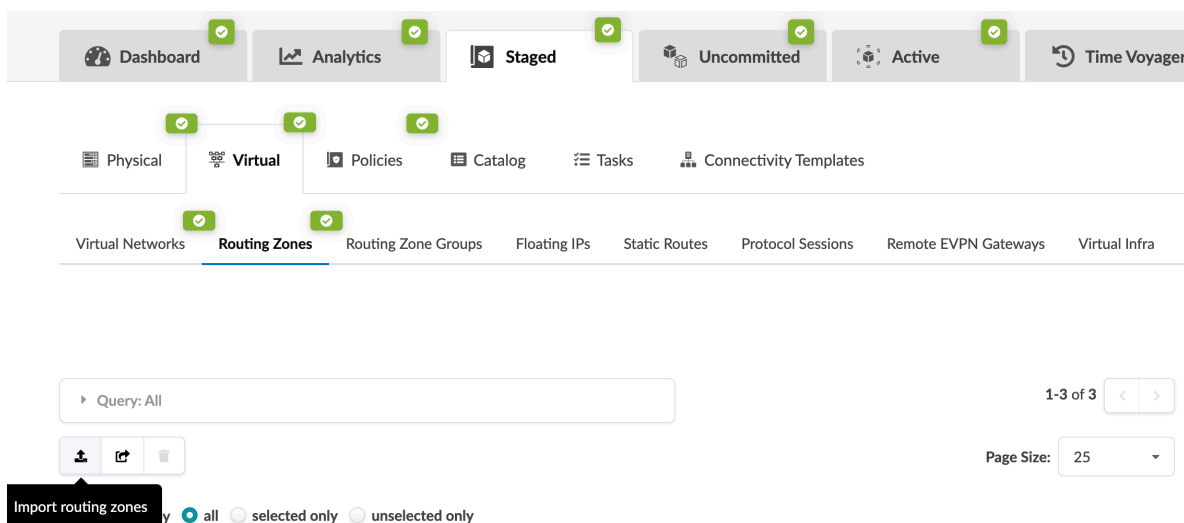
Create Routing Zones (using CSV File)

You can create many routing zones at once with a CSV file, as of Apstra version 4.1.2. First, you'll export the routing zone schema from your blueprint, then open and populate the file in a spreadsheet program. And finally, you'll import the file back into your blueprint.

1. From the blueprint, navigate to **Staged > Virtual > Routing Zones** and click **Export routing zones**.



2. Click **Copy** to copy the contents or click **Save As File** to download the file.
3. Paste the contents, or open the CSV file, in a spreadsheet program (such as Google Sheets or Microsoft Excel).
4. Enter routing zones details into the spreadsheet, then save the file.
5. In the Apstra GUI, navigate to **Staged > Virtual > Routing Zones** and click **Import routing zones**.



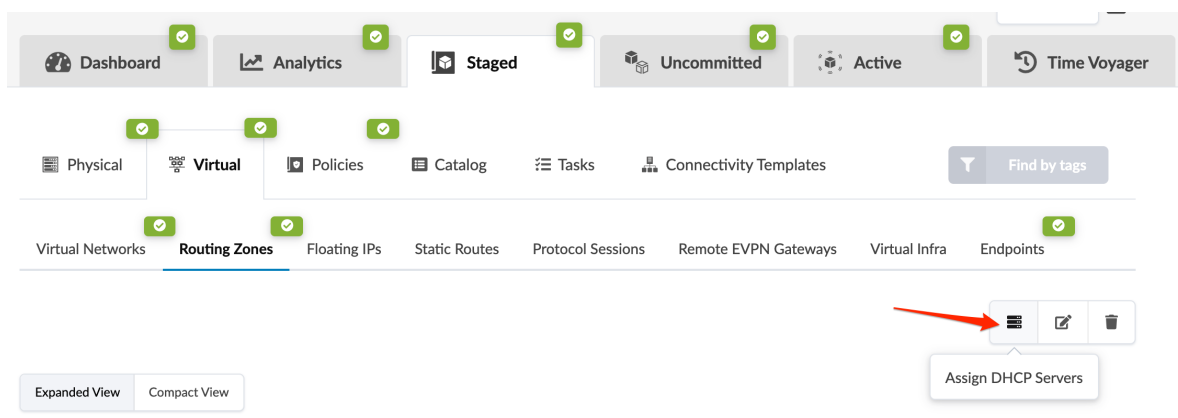
6. Either click **Choose File** and navigate to the file on your computer, drag and drop the file onto the dialog window, or directly paste CSV file contents into the dialog window. Routing zone details are displayed for your review.
7. Click **Import** to import the routing zones, stage the changes, and return to the table view.

Next Steps:

Assign resources. 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.

Assign DHCP Server 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) and enter the IPv4 address (or IPv6 address) for one or more DHCP servers.



3. Click **Update** to stage the assignment and return to the routing zone detail view.

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 network management interface. The 'Build' panel on the right displays the following resource assignment status:

- blue: Leaf L3 Peer Links (2/2)
- blue: Leaf Loopback IPs (5/5)
- blue: To Generic Link IPs (4/4)
- red: Leaf L3 Peer Links (2/2)
- red: Leaf Loopback IPs (5/5)
- red: To Generic Link IPs (0/4)

A red arrow points to the 'Update assignments' button (represented by a pencil icon) in the 'Build' panel. Below the 'Build' panel, a message states: 'No pools 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 267](#).) When the red status indicator turns green, the required resources are 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 table view, scrolling down to the **Interfaces** section, clicking the **Edit**

IP addresses button, and entering them from there.

Dashboard

Analytics

Staged

Uncommitted

Active

Time Voyager

Interfaces 4

1-4 of 4

Page Size: 25

Edit IP Addresses

by ☒ all ☐ selected only ☐ unselected only

| | | Endpoint 1 | | | | Interface 1 | | Endpoint 2 | | | Interface 2 | |
|-------------------------------------|--------------|------------|-------|------|-----------------|----------------|-------------------|-----------------|----------------|-----------|--------------|-------------------|
| <input type="checkbox"/> | Routing Zone | VLAN ID | Name | Role | Interface | IPv4 Address | IPv4 Address Type | Name | Role | Interface | IPv4 Address | IPv4 Address Type |
| <input checked="" type="checkbox"/> | red | 2 | leaf1 | Leaf | Ethernet1/1.3 | Not assigned | Numbered | rtr_leaf1_leaf2 | Generic System | n/a | Not assigned | Nur |
| <input checked="" type="checkbox"/> | red | 2 | leaf2 | Leaf | Ethernet1/5.3 | Not assigned | Numbered | rtr_leaf1_leaf2 | Generic System | n/a | Not assigned | Nur |
| <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 | | | | | |

Import / Export Routing Zones

IN THIS SECTION

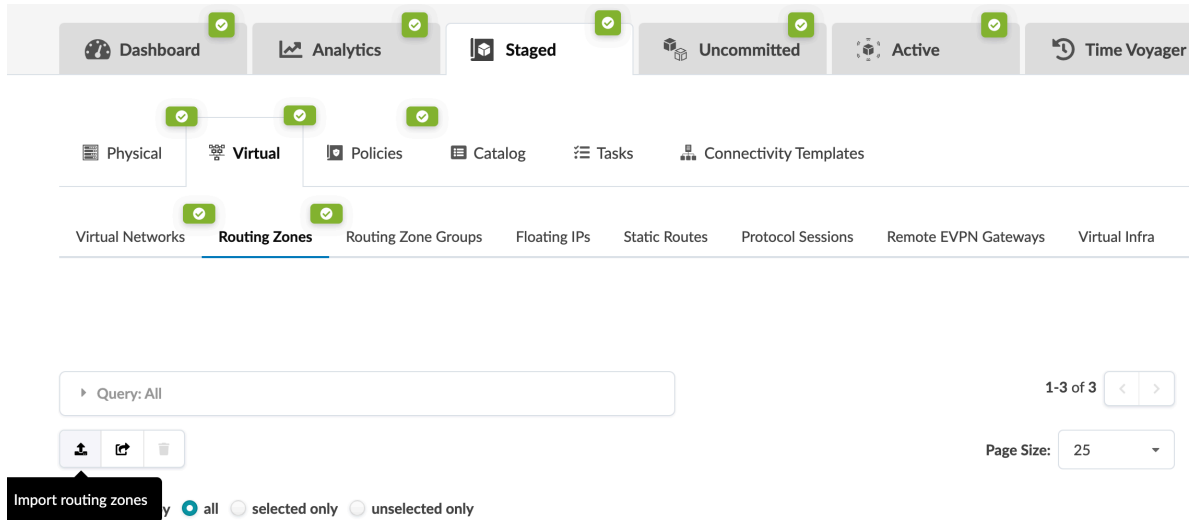
Import Routing Zones | 435

Export Routing Zones | 436

Import Routing Zones

You can import multiple routing zones (as a CSV file) into your blueprint, as of Apstra version 4.1.2. (Tip: First export routing zones so you'll have the schema set up for you in the CSV file.)

1. From the blueprint, navigate to **Staged > Virtual > Routing Zones** and click the **Import routing zones** button.

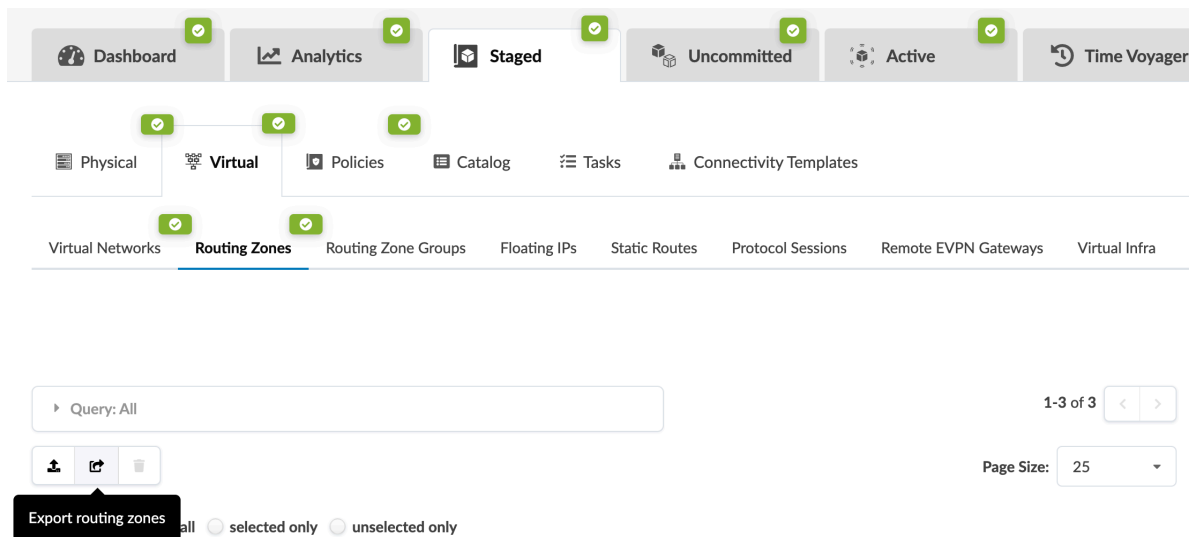


2. Either click **Choose File** and navigate to the file on your computer, drag and drop the file onto the dialog window, or directly paste CSV file contents into the dialog window. Routing zone details are displayed for your review.
3. Click **Import** to import the routing zones, stage the changes, and return to the table view.

Export Routing Zones

You can export routing zones (as a CSV file) from your blueprint, as of Apstra version 4.1.2.

1. From the blueprint, navigate to **Staged > Virtual > Routing Zones** and click **Export routing zones**.



2. Click **Copy** to copy the contents or click **Save As File** to download the file.
3. When you've copied or downloaded the routing zones, close the dialog to return to the table view.

Edit Routing Zones

IN THIS SECTION

- [Edit One Routing Zone | 437](#)
- [Edit Multiple Routing Zones | 437](#)

Edit One Routing Zone

1. From the blueprint, navigate to **Staged > Virtual > Routing Zones** and click the name of the routing zone to edit.

The screenshot shows the Apstra GUI navigation menu. The path **Staged > Virtual > Routing Zones** is highlighted. Below the menu, a table lists routing zones. A red arrow points to the 'blue' routing zone in the 'VRF Name' column.

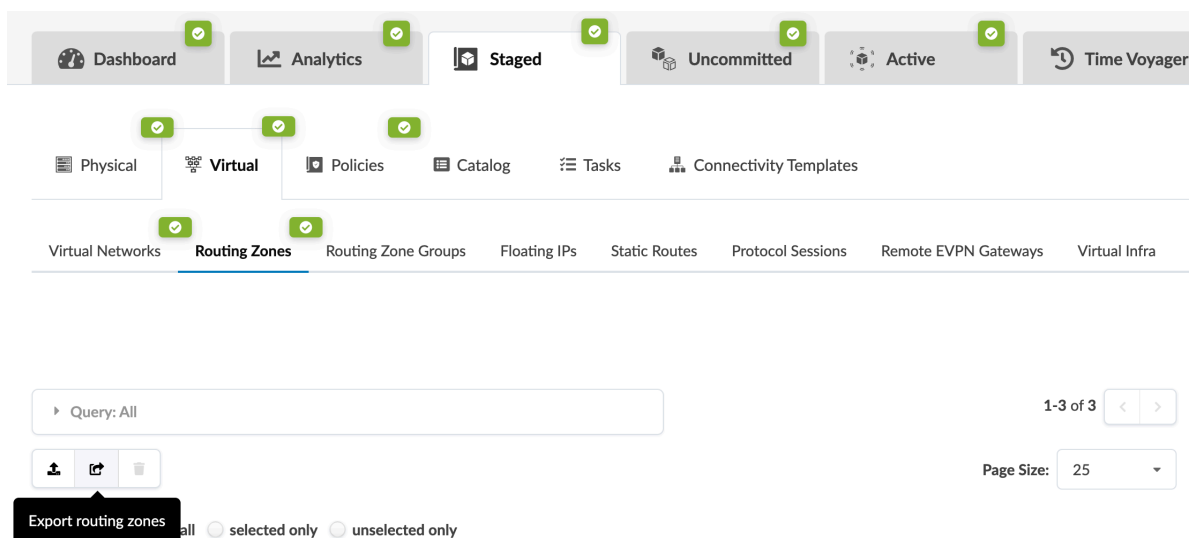
| | VRF Name | Type | VLAN ID | Route Target | VNI | DHCP Servers | Actions |
|------------|----------|------|---------|--------------|-------|--|---------|
| 0 selected | blue | EVPN | 201 | 20002:1 | 20002 | 198.51.100.2 fc01:a05:198:51:100::2 | |

2. Click the **Edit** button (upper-right) and make your changes.
3. Click **Update** to stage your changes and return to the routing zone details.

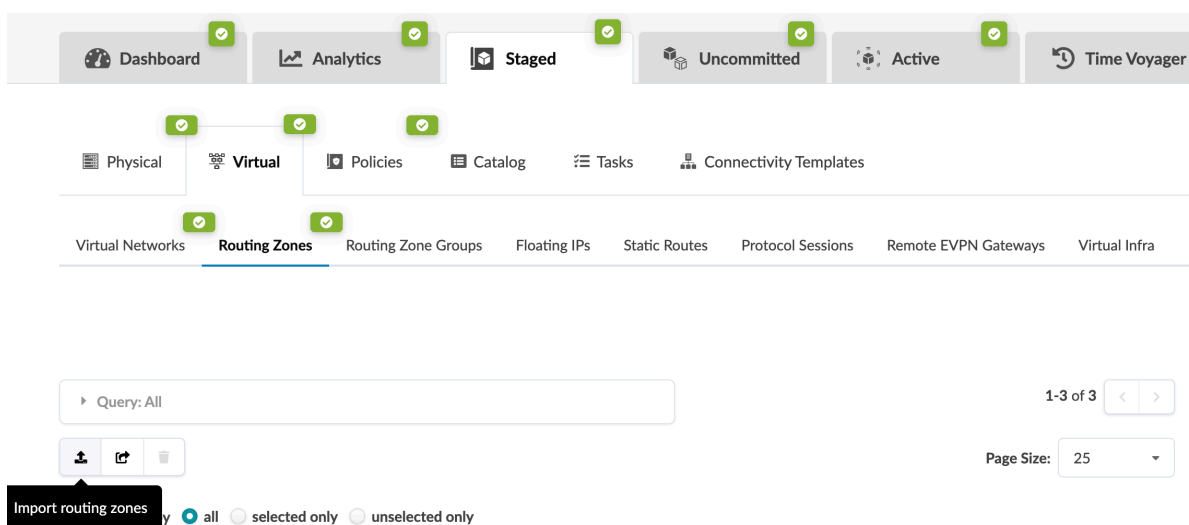
Edit Multiple Routing Zones

You can edit multiple routing zones using the Apstra GUI, as of Apstra version 4.1.2. You'll export all routing zones from your blueprint as a CSV file, edit the file in a spreadsheet program, then import the file back into your blueprint.

1. From the blueprint, navigate to **Staged > Virtual > Routing Zones** and click **Export routing zones**.



2. Click **Copy** to copy the contents, or click **Save As File** to download the file. (Close the dialog to return to the table view.)
3. Paste the contents, or open the CSV file, in a spreadsheet program (such as Google Sheets or Microsoft Excel).
4. Update routing zones as needed, then save the file.
5. In the Apstra GUI, navigate to **Staged > Virtual > Routing Zones** and click **Import routing zones**.



6. Either click **Choose File** and navigate to the file on your computer, or drag and drop the file onto the dialog window. Routing zone details are displayed for your review.
7. Click **Import** to import the routing zones, stage the changes, and return to the table view.

Delete Routing Zones

IN THIS SECTION

- [Delete One Routing Zone | 439](#)
- [Delete Multiple Routing Zones | 439](#)

Delete One Routing Zone

Routing zones with assigned virtual networks with connectivity templates can't be deleted.

1. From the blueprint, navigate to **Staged > Virtual > Routing Zones** and click the **Delete** button in the **Actions** panel for the routing zone to delete.
2. Click **Delete** to stage the deletion and return to the table view.

Delete Multiple Routing Zones

Routing zones with assigned virtual networks with connectivity templates can't be deleted.

You can delete multiple routing zones from one dialog in the Apstra GUI, as of Apstra version 4.1.2.

1. From the blueprint, navigate to **Staged > Virtual > Routing Zones** and select the check boxes for the routing zones to delete. (Tip: Use the Query function to filter specific routing zones.)
2. Click the **Delete selected routing zones** button that becomes available.
3. Click **Delete** to stage the deletion and return to the table view.

Static Routes (Virtual)

When you create connectivity templates, static routes are created. From the blueprint, navigate to **Staged > Virtual > Static Routes** to go to static routes.

1. Staged

2. Virtual

3. Static Routes

Query: All

1-6 of 6

Columns (12/13) Page Size: 25

| Routing Zone | System | | | Destination Network | Next Hop | | Source Interface | | | Connectivity Templates | |
|--------------|---------------------------|---------------------------|------|---------------------|--------------|----------------|------------------|--------------|----------------|------------------------|-----------------------------------|
| | Label | Hostname | Role | | Address | Interface Name | Interface Type | Address | Interface Name | | Interface Type |
| green | mlag_rack_ext_0_001_leaf2 | mlag-rack-ext-0-001-leaf2 | Leaf | 198.51.100.2/32 | 10.0.3.41/31 | N/A | subinterface | 10.0.3.40/31 | Ethernet43.202 | subinterface | evpn_bgp_sessions |

Protocol Sessions (Virtual)

When you create connectivity templates, protocol sessions (BGP sessions) are created. (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. Staged

2. Virtual

3. Protocol Sessions

Query: All

1-6 of 6

Columns (12/18) Page Size: 25

| Endpoint 1 | | | | Endpoint 2 | | | | | | | |
|---------------------------|-------------------|-------------------|-----|------------|-------------------|-------------------|-------|----------|--------------|-----------------------------------|-------------------------------------|
| Name | Peer IPv4 Address | Peer IPv6 Address | ASN | Name | Peer IPv4 Address | Peer IPv6 Address | ASN | Protocol | Routing Zone | Connectivity Template | Protocol Session ID |
| mlag_rack_ext_0_001_leaf2 | 10.0.0.21/32 | N/A | 521 | sys001 | 198.51.100.2/32 | N/A | 65533 | BGP | blue | evpn_bgp_sessions | o_4ICAJ_RbKAnJnRj5g |

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

Routing Zone Groups

Floating IPs

Static Routes

Protocol Sessions

Remote EVPN Gateways

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 | blue |
| Connectivity Template | evpn_bgp_sessions |

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 |
| 1 system(s) mlag_rack_ext_0_001_leaf2 | loopback4 | Loopback | 521 | N/A | Addressed | 10.0.0.21/32 | Static | N/A | N/A | N/A | Static | N/A |
| 1 system(s) sys001 | N/A | Loopback | 65533 | N/A | Addressed | 198.51.100.2/32 | Static | N/A | N/A | N/A | Static | N/A |

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

IN THIS SECTION

- DCI / EVPN Gateway Overview | 442

- [DCI Deployment Options | 443](#)
- [Implementation | 445](#)
- [Apstra Workflow | 449](#)

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:** 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.

Limitations:

- EVPN-GW (DCI) between different vendors' EVPN fabric is not supported.
- IPv6 is not supported on Remote EVPN Gateways. (Actual EVPN routes can contain IPv6 Type 2 and Type 5.)

DCI Deployment Options

IN THIS SECTION

- [Over the Top | 444](#)
- [Gateway \(GW\) | 444](#)
- [Autonomous System Border Router \(ASBR\) | 445](#)

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:

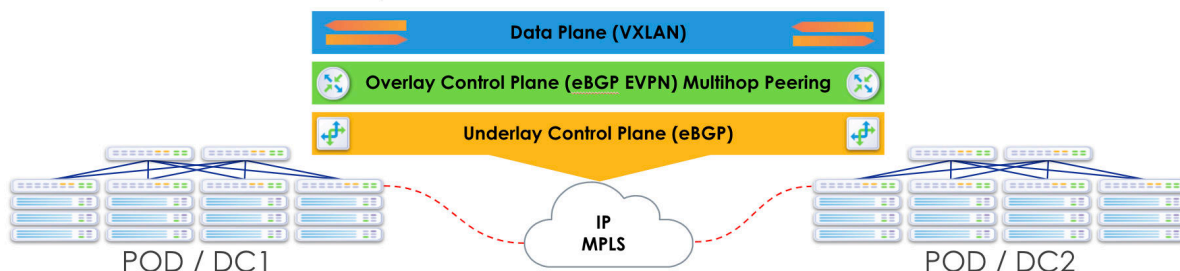
- You can extend Apstra DCI to other Apstra-managed data centers, non-Apstra managed data centers, or even to legacy non-spine-leaf devices.
- Apstra implementation and behavior is the same in all three cases.
- Whether the remote end is another DCI GW or an ASBR, it is transparent to Apstra.
- Apstra 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 data center teams can implement it 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

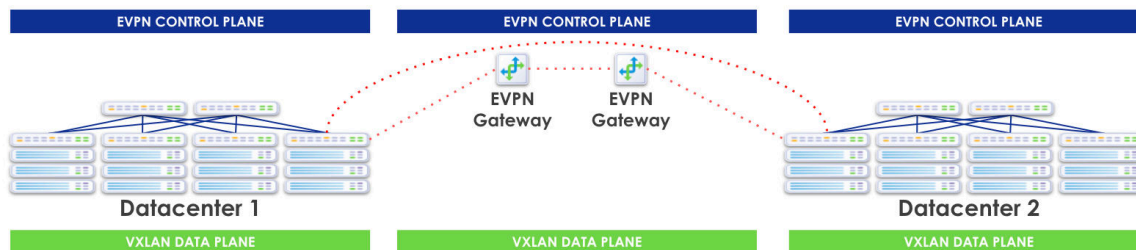


Gateway (GW)

Building upon the Apstra **Remote EVPN Gateway** capability, you can optionally specify that the **Remote EVPN Gateway** is 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 that the transport technology makes available (MPLS TE for example). 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

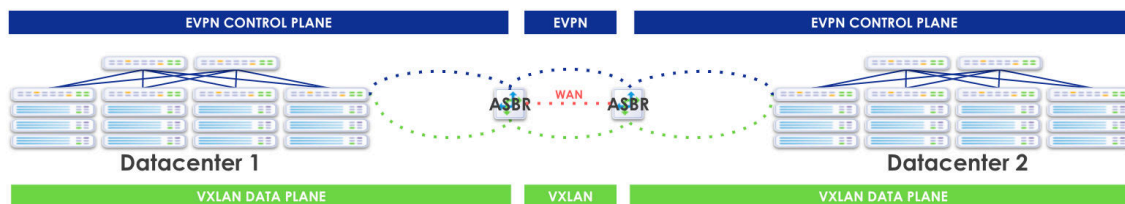


Autonomous System Border Router (ASBR)

Using the Apstra **Remote EVPN Gateway** capability, you can optionally specify that the **Remote EVPN Gateway** is 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 | 445](#)
- [Over the Top | 446](#)
- [Data Plane Extension: Layer 3 | 447](#)
- [Data Plane Extension: Layer 2 | 448](#)

You can extend routing zones and virtual networks (VN) to span across Apstra-managed blueprints (across pods) or to remote networks (across data centers) that Apstra doesn't manage. 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.

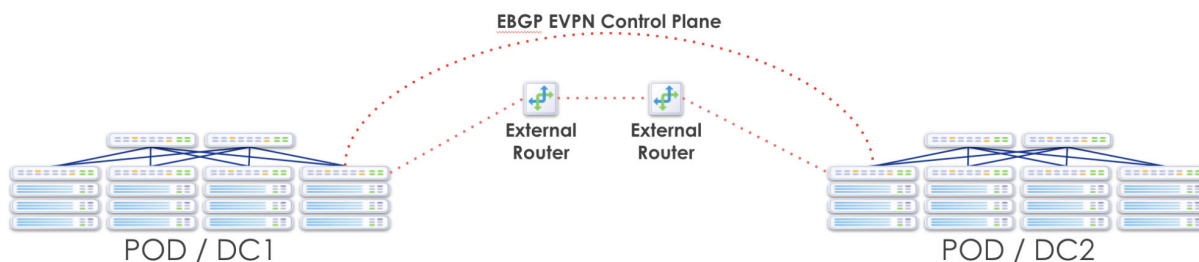
- No VXLAN traffic termination on the spine devices - connect external generic systems (tagged as external routers) on spine devices. This is to support IPv4 (underlay) external connectivity. Here spine devices don't need to terminate VXLAN traffic, unlike border leaf devices, when connected to external 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 separate Apstra blueprints manage individual sites (or when only one site is Apstra-managed) you must create and manage extended routing zones (VRFs) and virtual networks (Layer 2 and/or Layer 3 defined VLANs/subnets) independently in each

site. You must manually map VRFs and VNs between sites (creating administrative overhead).



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).

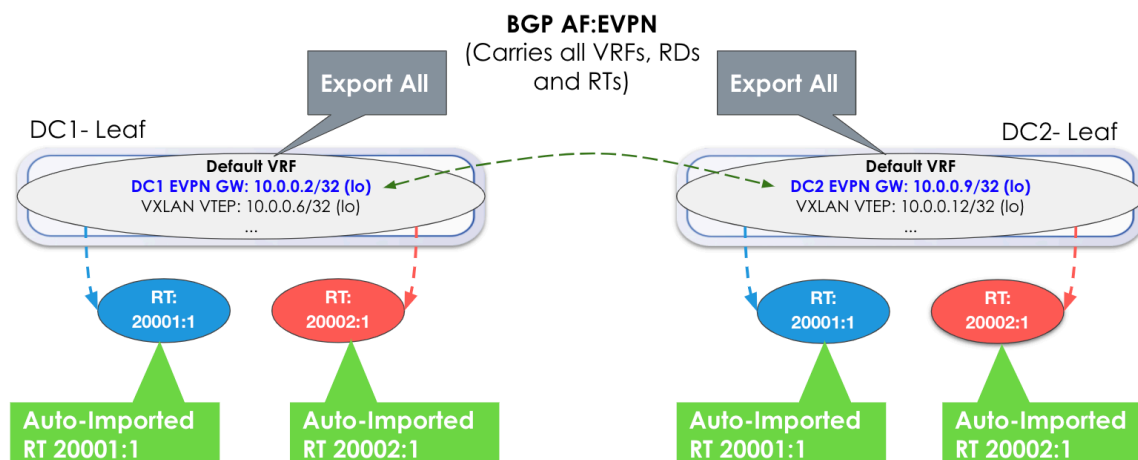
With any DCI implementation, careful resource planning and coordination is required. Adding more sites requires an exponential increase in such planning and coordination. VTEP loopbacks in the underlay need to be leaked. VNIDs must match between sites and in some cases, additional Route Targets (RTs) 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.

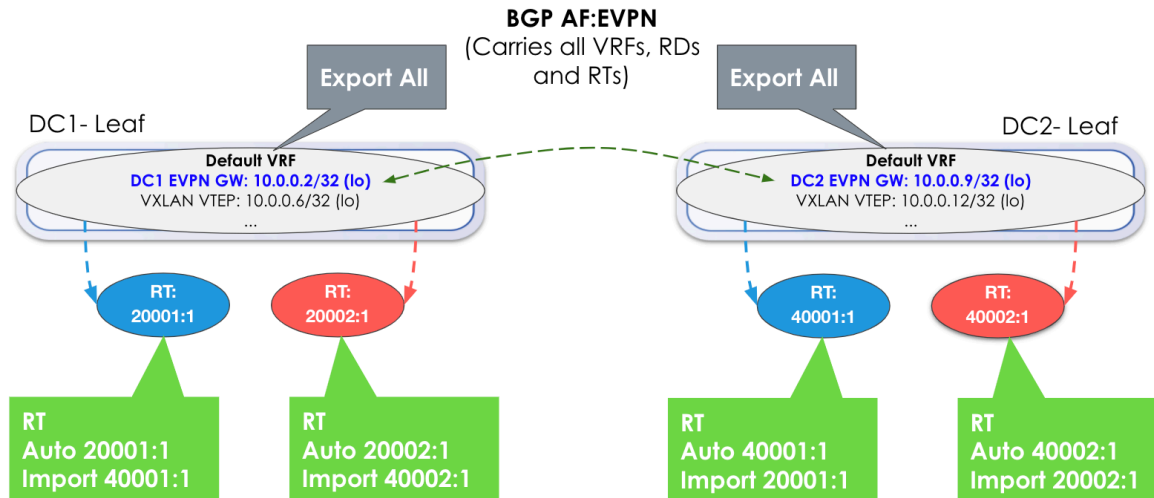
It is important to understand that a particular VNID will be associated with only one VRF (or routing zone in Apstra terminology). VNIDs exist within a VRF. They are tied to a VRF. For Layer 3 services, the stitching, or extending, of each VNID is done with the export and import of RTs within a routing 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

A virtual network can be a pure Layer 2 service (Layer 3 anycast gateway is not instantiated). It can be rack-local (VLAN on server-facing ports contained within a rack) or VXLAN (select the racks to extend the Layer 2 flood and broadcast domain between racks. 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 | 449](#)
- [Underlay VTEP Route Advertisements | 450](#)
- [Create Remote EVPN Gateways | 450](#)
- [Enhanced Routing Zone | 452](#)
- [Enhanced Virtual Networks | 453](#)
- [Remote Gateway Topology Representation | 454](#)

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 when creating 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, you can configure a Local EVPN Gateway (always an Apstra-managed switch) 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 the Apstra environment, each blueprint represents a data center. 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. We recommend 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



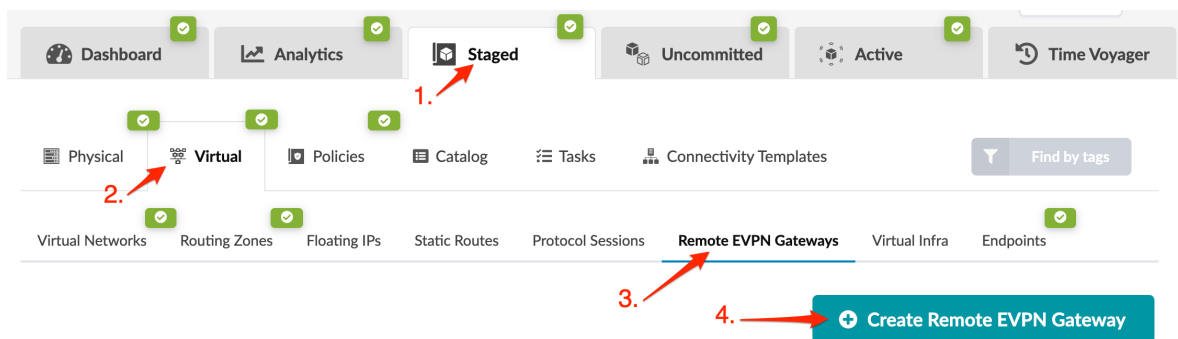
CAUTION: By default, ESI MAC msb (most significant byte) is set to 2 on all blueprints. Every Apstra blueprint that's connected must have a unique msb to prevent service-impacting issues. Before creating gateways, ["change ESI MAC msb" on page 552](#) accordingly. (You can leave one of them at the default value.)

Remote EVPN Gateway is a logical function that you could instantiate anywhere and on any device. It requires BGP support 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 (IANA allocates BGP TCP ports), 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 *

Remote EVPN gateway name

IP Address *

IP address for Remote EVPN gateway

ASN *

BGP autonomous system number for remote EVPN gateway

TTL

BGP multi-hop time-to-live (max number of L3 hops) - optional
If not specified, default is used

Password

Optional BGP TCP authentication password



Keep-alive Timer

BGP keep-alive timer - optional. If not specified, default is used

Hold-time Timer

BGP hold-time timer - optional. If not specified, default is used

EVPN Route Types *

☒ All Routes (I2+I3 mode) [?] ☐ Type-5 Only (I3-only mode) [?]

When extending L2 networks between data center fabrics you have the option (starting with Apstra version 4.1.0) to exchange only EVPN Route Type RT-5 prefixes (interface-less model). This is useful when there is no need to exchange all host routes between data center locations. This results in smaller requirements for the routing information base (RIB), also known as the routing table, and the forwarding information base (FIB), also known as the forwarding table, on DCI equipment.

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-leaf devices 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

Label¹

Role

Group Label¹

Hostname¹

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

Select local gateway nodes

You can filter the nodes list below to find them easier.

4. Click **Create** to stage the gateway and return to the table view.

5. When you are ready to deploy the devices in the blueprint, ["commit" on page 692](#) 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

RT (route-target) import/export policies on devices that are part of extended service govern EVPN route installation. Specify route target policies to add import and export route-targets that Apstra uses for routing zones/VRFs. You do this when you create routing zones. Navigate to **Staged > Virtual > Routing**

Zones and click **Create Routing Zone**. For more information, see ["Routing Zones" on page 429](#).

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 generated default route-target for routing zones is **<L3 VNI>:1**. You can't change this default value.

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.

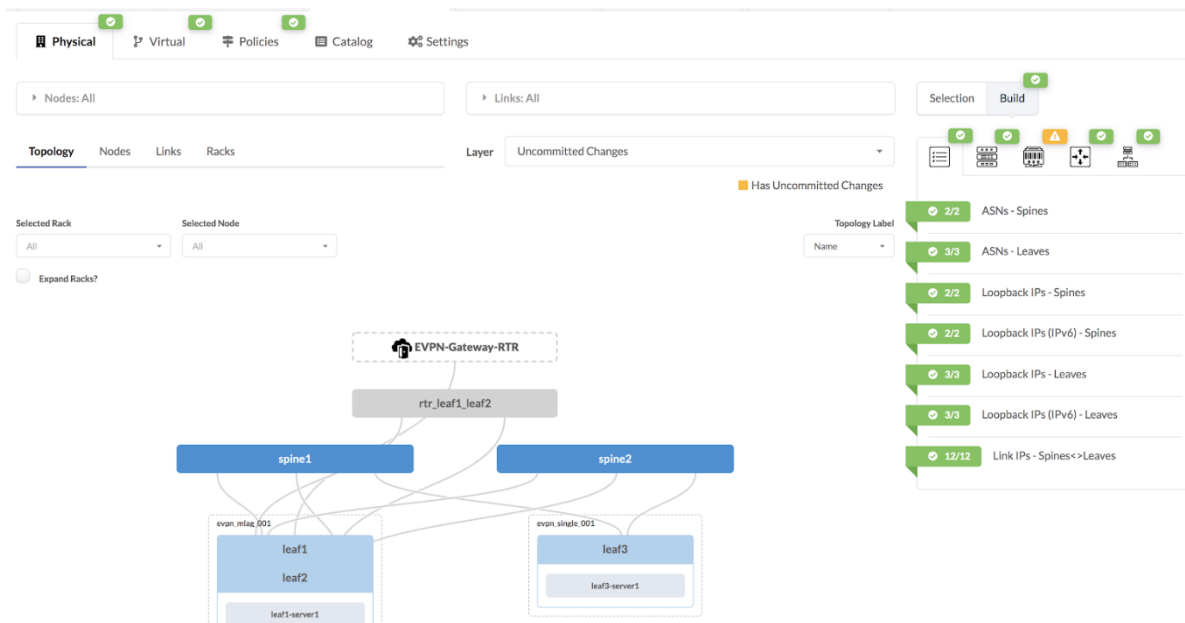


NOTE: The default route target that Apstra generates for virtual networks is **<L2 VNI>:1**. You can't alter this.

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. To establish connectivity, 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

- [vCenter Virtual Infra | 455](#)
- [NSX-T Integration | 462](#)
- [NSX-T Edge and Connectivity Templates | 474](#)
- [NSX-T Inventory Mapping to Apstra Virtual Infrastructure | 485](#)

vCenter Virtual Infra

IN THIS SECTION

- [VMware vSphere Integration Overview | 455](#)
- [Enable vCenter Integration | 456](#)
- [VM Visibility | 457](#)
- [Validate Virtual Infra Integration | 458](#)
- [Auto-Remediation Overview | 460](#)
- [Enable Auto-Remediation | 460](#)
- [Remediate Probe Anomalies | 461](#)
- [Disable Virtual Infra Integration | 461](#)

VMware vSphere Integration Overview

IN THIS SECTION

- [Supported Versions | 456](#)
- [Limitations | 456](#)

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 accomplish 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 offbox agent and uses pyVmmomi 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 7.0U1 (as of Apstra version 4.1.0)
- vCenter Server/vSphere 6.7
- vCenter Server/vSphere 6.5

The specific test and qualification for version 7.0 is three vCenter servers on three different routing zones: zone 1 supports 3000 VMs, zone 2 supports 1000 VMs, and zone 3 supports 1000 VMs. We support vCenter managed data center stretched clusters. vCenter segregation is based on workload, not location.

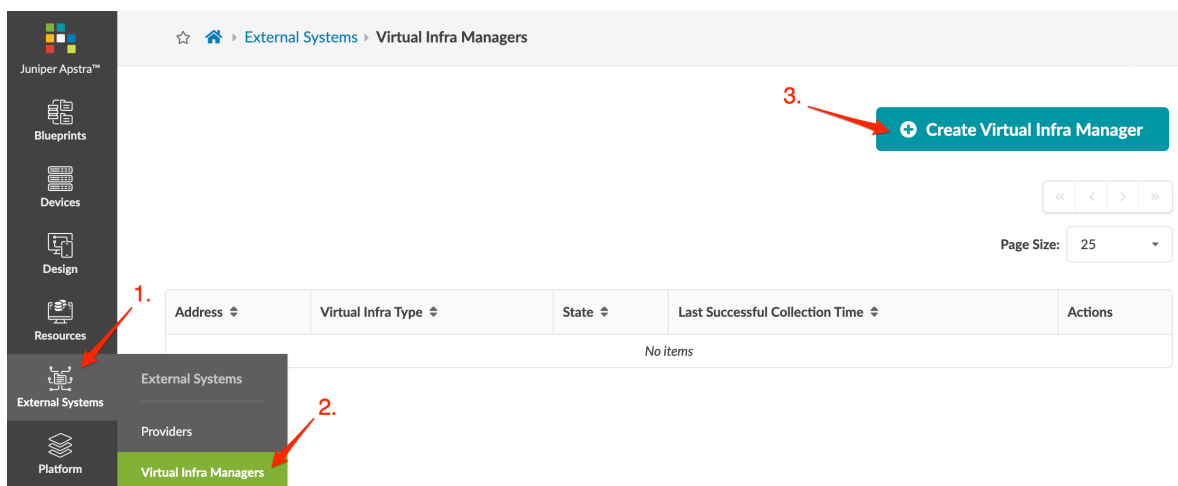
Limitations

vCenter integration does not support DVS port group with VLAN type Trunking.

Enable vCenter Integration

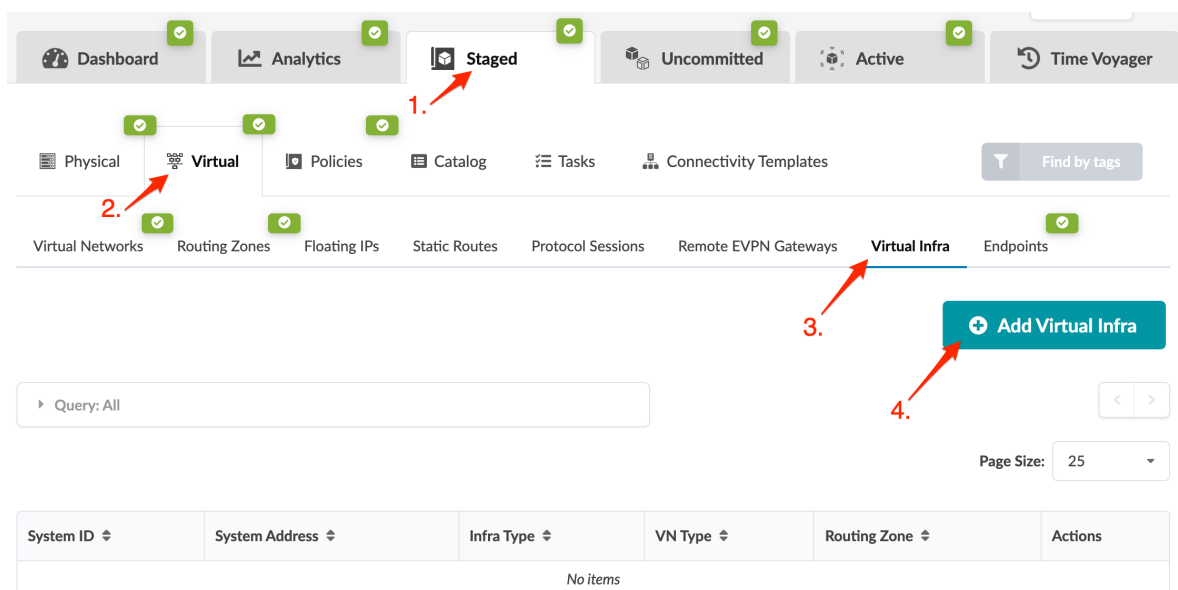
You only need **Read** permissions to enable vSphere Integration.

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.

- Click **Create** to launch an offbox 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:

| Parameter | Description |
|----------------|---|
| 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 |

(Continued)

| Parameter | Description |
|-------------------------|--|
| 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 | 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 | spine-1 | 4 | 1.1.0.3 | spine-1 | lo0 | 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 | spine-2 | 5 | 1.1.0.4 | spine-2 | lo0 | 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 1046
- "Hypervisor MTU Threshold Check Probe" on page 1046
- "Hypervisor & Fabric LAG Config Mismatch Probe" on page 1038
- "Hypervisor & Fabric VLAN Config Mismatch Probe" on page 1039
- "Hypervisor Missing LLDP Config Probe" on page 1047
- "VMs without Fabric Configured VLANs Probe" on page 1072

Virtual Infra Redundancy Check Dashboard

- ["Hypervisor Redundancy Checks Probe" on page 1048](#)

For more information, see ["Analytics Dashboard" on page 702](#) and ["Probes" on page 709](#).

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 leaf devices 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 leaf devices 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

The **Remediate Anomalies** feature works in conjunction with the **Virtual Network (Single)** primitive in connectivity templates. It can't be used with the **Virtual Network (Multiple)** primitive.

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 Intra Managers** and click the **Delete** button for the virtual infra to disable.

NSX-T Integration

IN THIS SECTION

- [VMware NSX-T Integration Overview | 462](#)
- [Enable NSX-T Integration | 463](#)
- [Virtual Infrastructure Visibility | 468](#)
- [Validate Virtual Infra Integration | 472](#)
- [Disable Virtual Infra Integration | 474](#)

VMware NSX-T Integration Overview

IN THIS SECTION

- [Supported Versions | 463](#)
- [Limitations | 463](#)

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.

When NSX-T VM is attached into VLAN Transport, VM query shows TOR switch/interface information together. When NSX-T VM is attached into Overlay Transport, VM query doesn't show TOR switch/interface information. Be sure to add ESXi host in generic systems, not external generic systems.

As of Apstra version 4.1.2, you can create Virtual Infra Managers for NSX-T Manager version 3.2.x using DVS mode. You can also add multiple Virtual Infra Managers per blueprint. This is useful when you have multiple NSX-T Managers or multiple vCenter Servers hosted in the same fabric blueprint. You'll need to provide the vCenter compute managers information (address and credentials) when you add the NSX-T Virtual Infra.

Supported Versions

Apstra version 4.1.2: VMware NSX-T Manager version 3.2.x

Apstra version 4.1.1 and 4.1.0: VMware NSX-T version 3.0.2

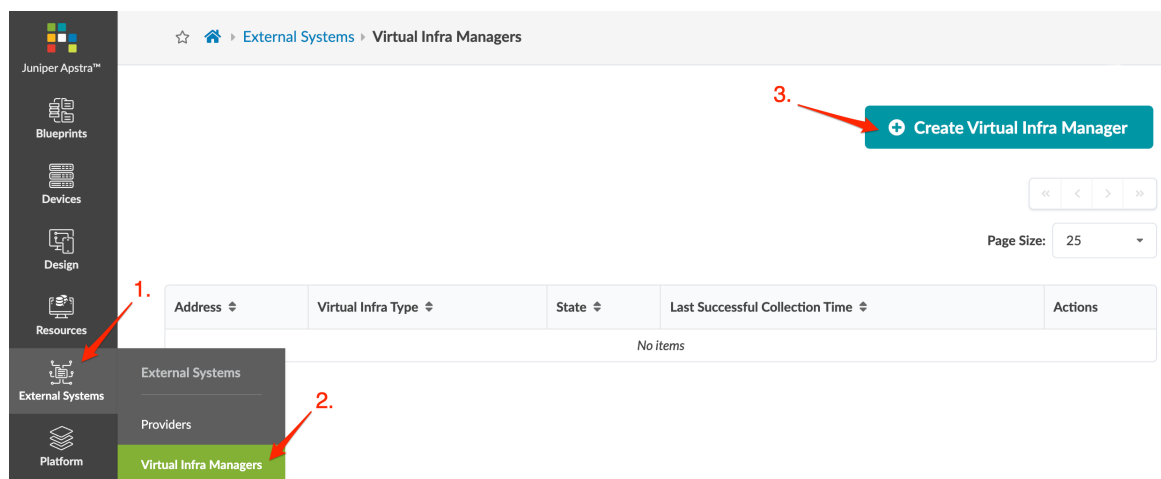
Limitations

- NSX-T Edge VM migration is supported only within a rack. Attempting to migrate between racks results in BGP disruption. You can migrate the NSX-T Edge VM from the ESXi host connected to leaf pair (that is, ToR-Leaf and ToR-Right) to the other ESXi host which is connected to single leaf with the rack.
- (Apstra versions 4.1.1 and 4.1.0 only) Having more than one NSX-T virtual infra in a blueprint is not supported. We recommend only one virtual infra per blueprint.
- (Apstra versions 4.1.1 and 4.1.0 only) NSX-T integration does not support DVS port group with VLAN-type trunking.

Enable NSX-T Integration

We recommend that you ["create a user profile" on page 739](#) dedicated to managing NSX-T integration activities.

1. From the left navigation menu, navigate to **External Systems > Virtual Infra Managers > 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.

Create Virtual Infra Manager

Summary

Address *

192.168.100.1

Virtual Infra Type

☐ VMware vCenter Server ☒ VMware NSX-T Manager

Username *

admin

Password *

••••••••

vCenters

✕

Address *

192.168.100.2

Username *

administrator@vsphere.local

Password *

••••••••

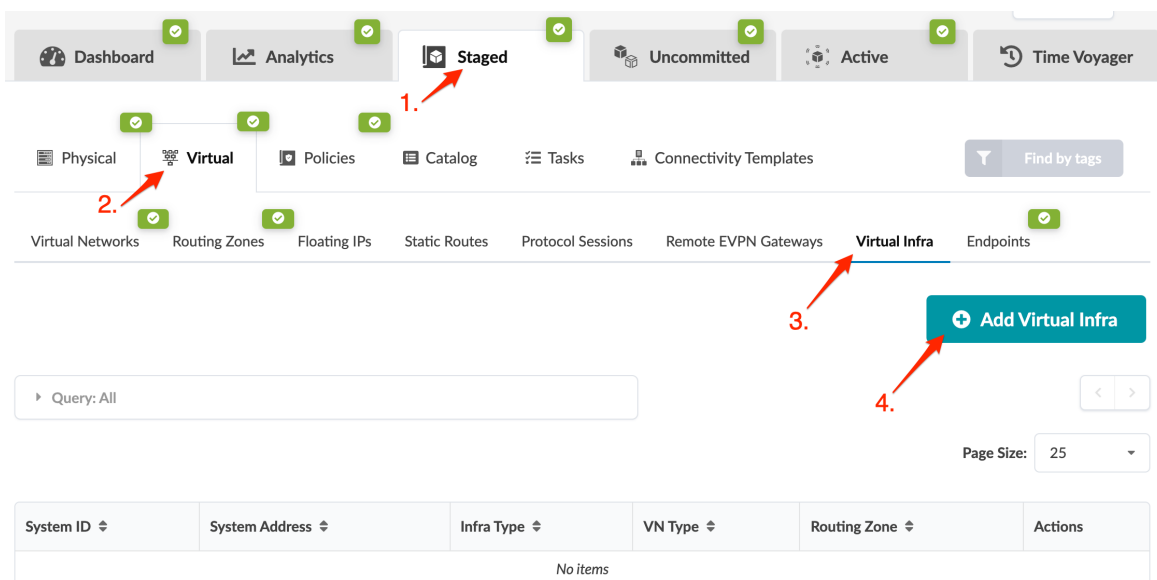


☐ Create Another?

Create

3. Click **Create** to create the virtual infra manager and return to the table 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 > 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 that can remediate anomalies.
6. Select the VN type and routing zone.
 - If VLAN (rack-local) is selected, you must use 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 table view. The new virtual infra manager appears in the table.
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[?]

11011:1

Routing Policies

imp_all

| | |
|-----------------------------------|--------------|
| Name | imp_all |
| Import Policy [?] | All |
| Extra Import Routes [?] | Not provided |
| Spine Leaf Links [?] | yes |
| L3 Edge Server Links [?] | yes |
| L2 Edge Subnets [?] | yes |
| Loopbacks [?] | yes |
| Static routes [?] | yes |
| Extra Export Routes [?] | Not provided |
| Aggregate Prefixes [?] | Not provided |

Update

10. For the GENEVE Tunnels to come up between the Transport Nodes in NSX-T, connectivity must be established via Juniper Apstra Fabric. This will be ensured by creating VXLAN VN in Apstra and assigning correct port mapping in ToR leaf devices towards Transport Node. VLAN ID for Overlay VXLAN VN defined in Apstra must match the one mapped in Overlay Profile in NSX-T for Transport

Nodes. Also, the 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 VXLAN for all selected nodes

Name

overlay-tep-pool-vn

Routing Zone

NSX_VRF

VNI

11050

Do I want to create a single VXLAN for all nodes?

☒

VLAN ID (per host)

50

Route Target

11050:1

DHCP Service

Disabled

IPv4 Connectivity

Enabled

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

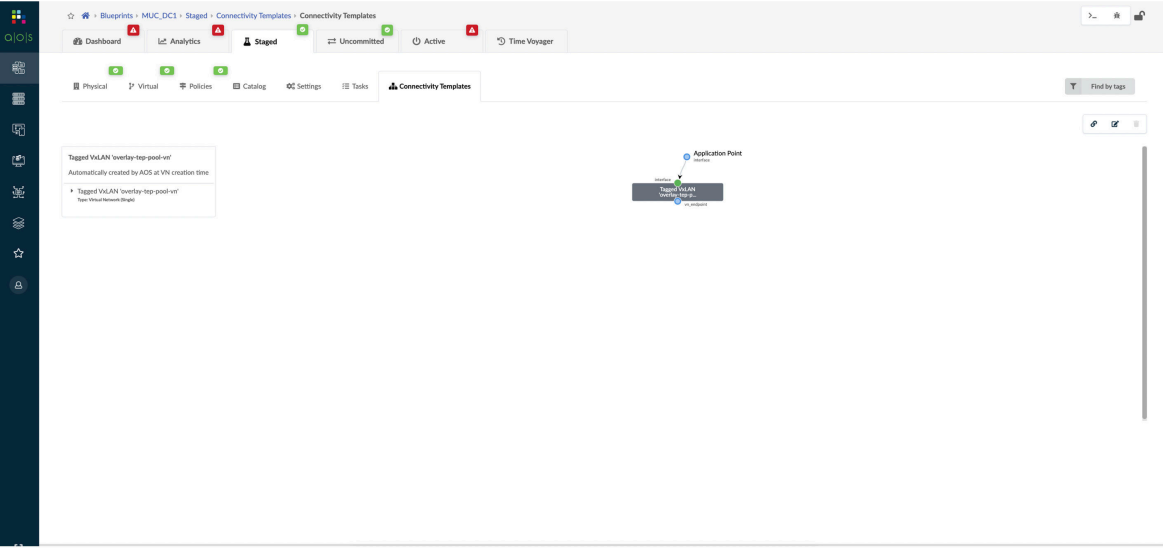
| | Bound To | Link Labels | VLAN ID | IPv4 Mode | IPv4 Address |
|--|----------------------------|------------------|---------|----------------------------|----------------------------|
| | mac_host_5100_001_host_001 | 5100-Server-Link | 50 | Enabled | mac_host_5100_001_host_001 |
| | | | | | From resource pool |
| | | | | mac_host_5100_001_host_002 | 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:



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' |
|---|------|-----------------------------------|
| pod1 (Pod) | | |
| * mac_leaf_5100_001 (Rack) | | |
| * mac_leaf_5100_001_leaf1 / mac_leaf_5100_001_leaf2 (Leaf pair) | | |
| ae1 -> mac_leaf_5100_001_svy001 (Interface) | | |
| ae2 -> mac_leaf_5100_001_svy002 (Interface) | | |
| ae3 -> mac_leaf_5100_001_svy003 (Interface) | | |
| ae4 -> mac_leaf_5100_001_svy004 (Interface) | | |
| * mac_leaf_5110_001 (Rack) | | |
| * mac_leaf_5110_001_leaf1 / mac_leaf_5110_001_leaf2 (Leaf pair) | | |
| ae1 -> mac_leaf_5110_001_svy001 (Interface) | | |
| ae2 -> mac_leaf_5110_001_svy002 (Interface) | | |
| ae3 -> mac_leaf_5110_001_svy003 (Interface) | | |
| ae4 -> mac_leaf_5110_001_svy004 (Interface) | | |
| * mac_leaf_5120_001 (Rack) | | |
| * mac_leaf_5120_001_leaf1 / mac_leaf_5120_001_leaf2 (Leaf pair) | | |
| ae1 -> mac_leaf_5120_001_svy001 (Interface) | | |
| ae2 -> mac_leaf_5120_001_svy002 (Interface) | | |
| ae3 -> mac_leaf_5120_001_svy003 (Interface) | | |
| ae4 -> mac_leaf_5120_001_svy004 (Interface) | | |
| * mac_rack_border_001 (Rack) | | |
| * mac_rack_border_001_leaf1 (Leaf) | | |
| ae0/0/0/3 -> mac_rack_border_001_svy004 (Interface) | | |
| ae0/0/0/0 -> mac_rack_border_001_svy001 (Interface) | | |
| * mac_rack_border_001_leaf1 / mac_rack_border_001_leaf2 (Leaf pair) | | |
| ae2 -> mac_rack_border_001_svy002 (Interface) | | |
| ae3 -> mac_rack_border_001_svy003 (Interface) | | |

Assign

13. Once the configuration 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.

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 leaf devices. From the blueprint, navigate to **Active > Query > VMs**.

☆ Blueprints > MUC_DC1 > Active > Query > VMs

Dashboard Analytics Staged Uncommitted Active Time Voyager

Physical Virtual Policies Catalog Settings Query Anomalies Root Causes Find by tags

MAC ARP VMs

Query: All 1-23 of 23 Page Size: 25

| VM Name | Hosted On | Hypervisor Hostname | Hypervisor Version | VM IPs | LeafInterface | 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:

| Parameter | Description |
|-------------------------|--|
| 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 shows the VMware NSX-T Lab interface. The top navigation bar includes tabs for Dashboard, Analytics, Staged, Uncommitted, and Active. The 'Active' tab is selected. Below the navigation bar, the 'Physical' tab is selected in the left sidebar. The main content area displays a search filter for 'Nodes: Has VMs?=yes'. The filter is applied, and the results are shown in a table. The table has two columns: 'Selection' and 'Status'. The results are as follows:

| Selection | Status |
|-----------|------------------------------|
| 1 | 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 |
| 0 | Anomalies: Route |
| 3/0/0 | Deployment Mode |
| 0/0/0 | Deployment Status: Discovery |
| 3/0/0 | Deployment Status: Service |

If 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)**. Select the **VMs** tab as shown 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. If BGP neighborhood in NSX-T Manager is deleted, then respective anomalies are displayed in the 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 1046
- "Hypervisor MTU Threshold Check Probe" on page 1046
- "Hypervisor & Fabric LAG Config Mismatch Probe" on page 1038
- "Hypervisor & Fabric VLAN Config Mismatch Probe" on page 1039
- "Hypervisor Missing LLDP Config Probe" on page 1047
- "VMs without Fabric Configured VLANs Probe" on page 1072

Virtual Infra Redundancy Check Dashboard

- ["Hypervisor Redundancy Checks Probe" on page 1048](#)

For more information, see ["Analytics dashboard" on page 702](#) and ["Instantiate Predefined Probe" on page 715](#).

Disable Virtual Infra Integration

To disable virtual infra integrations, delete 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 Edge and Connectivity Templates

IN THIS SECTION

- [Overview | 474](#)
- [Set Up NSX-T Tier-0 Router BGP peering | 474](#)
- [Set Up NSX-T VRF Lite | 480](#)
- [Set Up Default Static Route towards NSX-T Edge | 483](#)
- [Set Up BGP IPv6 towards NSX-T Edge | 484](#)
- [Un-assign BGP on VXLAN VN towards NSX-T Edge | 485](#)

Overview

Juniper Apstra supports NSX-T Edge connectivity requirements using connectivity templates. Connectivity templates can be used both where NSX-T Edge is hosted on Bare Metal or when used as a virtual machine.

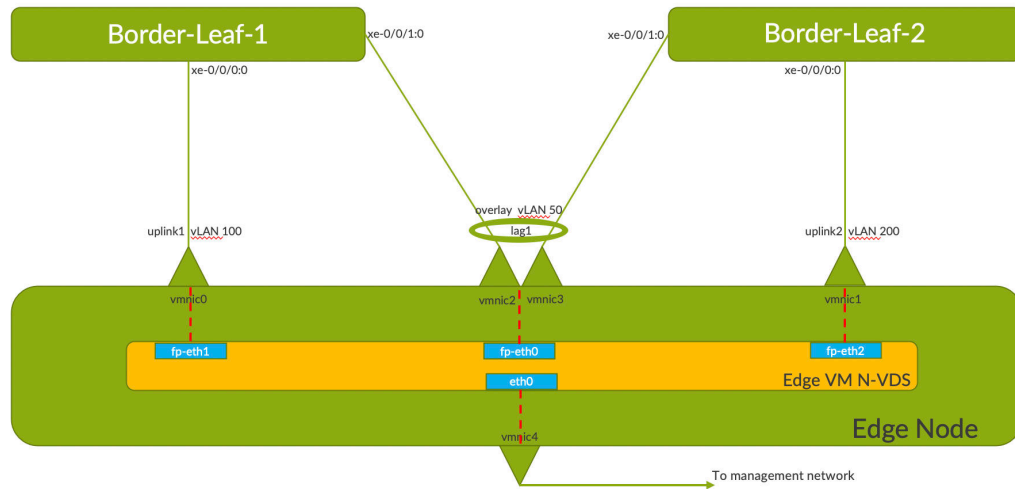
We support VRF lite enabled Tier-0 edge Gateway using connectivity templates.

The use cases below relate to connectivity templates for NSX-T 3.0 Edge:

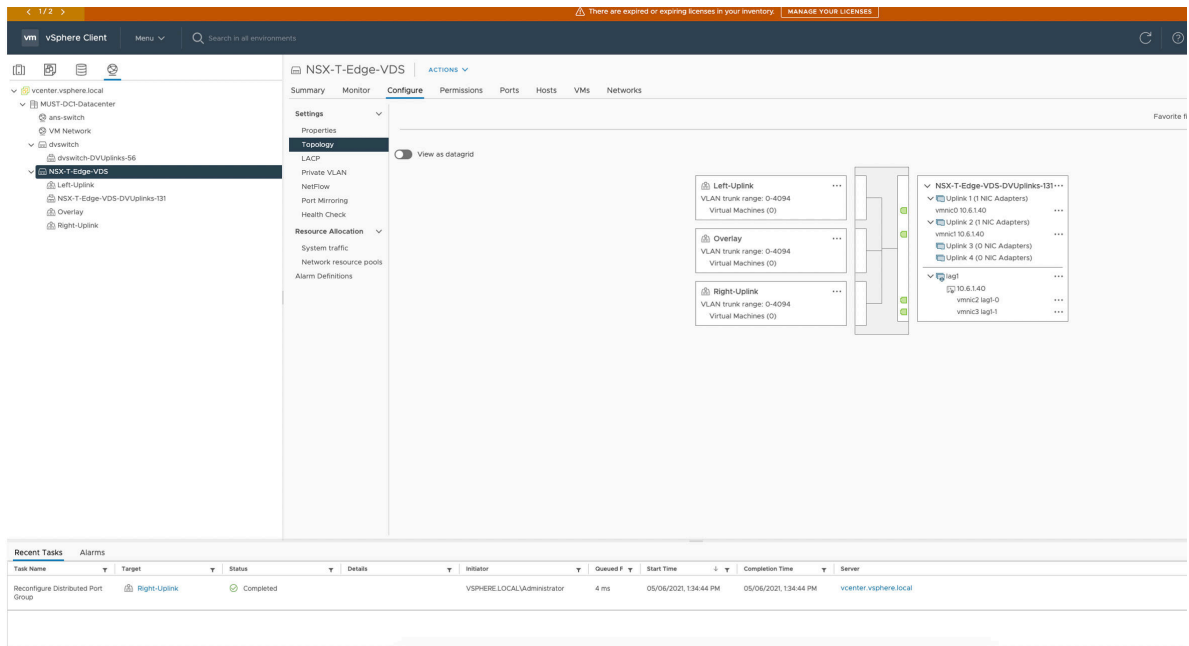
Set Up NSX-T Tier-0 Router BGP peering

Let's say NSX-T Edge VM uplinks are connected to ToR leaf devices 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-left-uplink-profile | c352_3733 | Follower Order | lag | | 0 | 1600 (Global MTU) |
| nsx-edge-multiple-edges-uplink-profile | c482_1007 | Load Balance Source | uplink-Luplink-2 | | 0 | 1600 (Global MTU) |
| nsx-edge-single-ec-uplink-profile | c732_430c | Follower Order | uplink-1 | | 0 | 1600 (Global MTU) |
| nsx-default-loadbalance-uplink-profile | 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 IP addresses of the transport nodes, which are part of the TEP IP 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 |

A green callout box points to the 'TEP IP Addresses' column, stating: "IPs from the TEP Pool".

- 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.

The screenshot shows the 'Tier-0 Gateways' configuration page in the Juniper Apstra GUI. The 'Edge Cluster' field is set to 'edge-cluster1'. A green callout box points to this field with the text 'Specify the edge-cluster created previously'.

- Add External interfaces to the T0 GW which maps to the Uplink segments

The screenshot shows the 'Set Interfaces' dialog for a Tier-0 Gateway. The 'Edge Node' field is set to 'edge01'. The 'Connected To(Segment)' field is set to 'uplink-seg-100'. A green callout box points to the 'Edge Node' field with the text 'Specify the edge VM'. Another green callout box points to the 'Connected To(Segment)' field with the text 'Used for the link between T0-GW and ToR'. A third green callout box points to the 'uplink-seg-100' dropdown with the text '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 462](#)

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.nxst_edge.vrf_lite.vqfx | None | testing_rfe1729.nxst_edge.vrf_lite.vqfx_vlan VLAN | Not Set | 0 |
| L2 VPN | Not Set | VPN Tunnel ID | Not Set | |
| VLAN | 10 | Uplink Teaming Policy | TOR-LEFT | |
| | 11 | | | |
| | 12 | | | |
| | View Less | | | |
| 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 | | | |

| | | | | |
|---|---------------------------|---|-----------------------------------|---|
| seg-red-TOR-R-testing_rfe1729.nxst_edge.vrf_lite.vqfx | None | testing_rfe1729.nxst_edge.vrf_lite.vqfx_vlan VLAN | Not Set | 0 |
| L2 VPN | Not Set | VPN Tunnel ID | Not Set | |
| VLAN | 20 | Uplink Teaming Policy | TOR-RIGHT | |
| | 21 | | | |
| | 22 | | | |
| | View Less | | | |
| 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](#)

[EXPAND ALL](#)

| | 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 🔄 ℹ️ |
| > | 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 🔄 ℹ️ |

BGP Neighbors

Tier-O Gateway blue-testing_... [#Neighbors](#)

[EXPAND ALL](#)

| | IP Address | BFD | Remote AS number | Route Filter | Allowas-in | Status |
|---|-------------|----------|------------------|--------------|------------|---|
| > | 10.100.30.1 | Disabled | 2 | 1 | Disabled | Down 🔄 ℹ️ |
| > | 10.100.40.1 | Disabled | 3 | 1 | Disabled | Down 🔄 ℹ️ |

- From the Apstra GUI, setup the **Routing Zone** and the respective VNs on which BGP session will be established towards ToR leaf devices as below:

Expanded ViewCompact View

Parameters

| | |
|---------------|---------------------------|
| VRF Name | nsxt |
| Type | EVPN |
| VLAN ID | 2 |
| VNI | 20000 |
| Route Target® | 20000:1 |
| DHCP Servers | DHCP Relay not configured |

Routing Policy

| | |
|----------------|-------------------|
| Name | Default Immutable |
| Import Policy® | All |

Query: All

1-4 of 4

Columns (10/11)Page Size: 25

| 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> <div>nsx.edge.compute_001_sw1</div> <div>nsx.edge.compute_001_sw2</div> <div>nsx.compute_001_sw1</div> </div> | Enabled | 10.10.200.0/24 | Disabled | N/A | |
| <input type="checkbox"/> | nsxt_host_traffic | nsxt | VLAN | 10000 | <div>3 nodes</div> <div> <div>nsx.edge.compute_001_sw1</div> <div>nsx.edge.compute_001_sw2</div> <div>nsx.compute_001_sw1</div> </div> | Enabled | 10.10.100.0/24 | Disabled | N/A | |
| <input type="checkbox"/> | vn150 | default | VLAN | 150 | <div>1 nodes</div> <div> <div>nsx.edge.compute_001_sw1</div> </div> | Enabled | 10.100.150.0/24 | Disabled | N/A | |
| <input type="checkbox"/> | vn160 | default | VLAN | 160 | <div>1 nodes</div> <div> <div>nsx.edge.compute_001_sw2</div> </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 (0) 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 (0) will be applied only to the loaded connectivity templates.

| Fabric | Tags | Internet-Server-link |
|---|------|----------------------|
| pod1 (Pod) | | |
| mxr_leaf_5110_001 (Rack) | | |
| mxr_leaf_5110_001_leaf1 / mxr_leaf_5110_001_leaf2 (Leaf-pair) | | |
| ae1 -> mxr_leaf_5110_001_syn001 (Interface) | | |
| ae2 -> mxr_leaf_5110_001_syn002 (Interface) | | |
| ae3 -> mxr_leaf_5110_001_syn003 (Interface) | | |
| ae4 -> mxr_leaf_5110_001_syn004 (Interface) | | |

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 'Custom Static Route' tab selected. The configuration includes:

- Parameters:** static-Internet
- Description:** (empty)
- Tags:** No tags
- Custom Static Route:**
 - Routing Zone:** NSX_VRF
 - Network:** 0.0.0.0/0
 - Next Hop IP Address:** 9.9.9.100

Callouts from the configuration:

- 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)

Buttons at the bottom: Revert Changes, Update.

Set Up BGP IPv6 towards NSX-T Edge

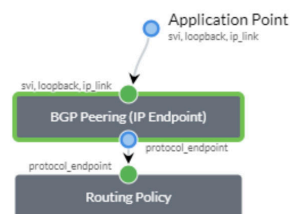
We can enable IPv6-based BGP neighborhood between T0 Gateway and ToR leaf using connectivity templates.

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 'BGP Peering (IP Endpoint)' configuration form with the following fields:

- TTL:** 2
- Single-hop BFD:** OFF
- Password:** (empty)
- Keep Alive Timer (sec):** (empty)
- Hold Time Timer (sec):** (empty)
- IPv4 Address:** 198.51.100.5/24
- IPv6 Address:** 2021:310:310::2/64
- Routing Policy:** (link)



Un-assign BGP on VXLAN VN towards NSX-T Edge

Let's say BGP neighborhood 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 |

NSX-T Inventory Mapping to Apstra Virtual Infrastructure

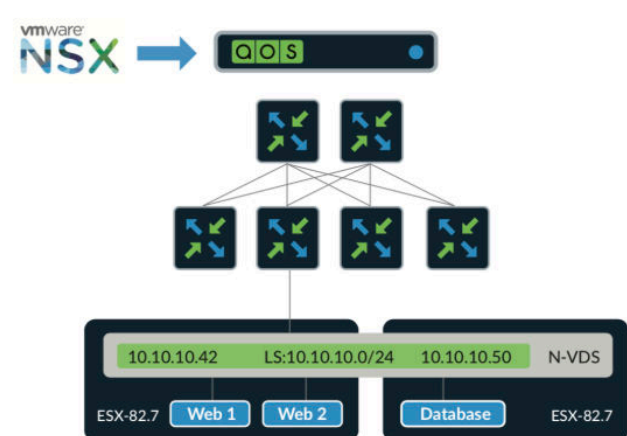
IN THIS SECTION

- Overview | 485
- NSX-T Networking Terminology and correlation | 486
- NSX Inventory Model | 494
- Model Details and Relationship | 495

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

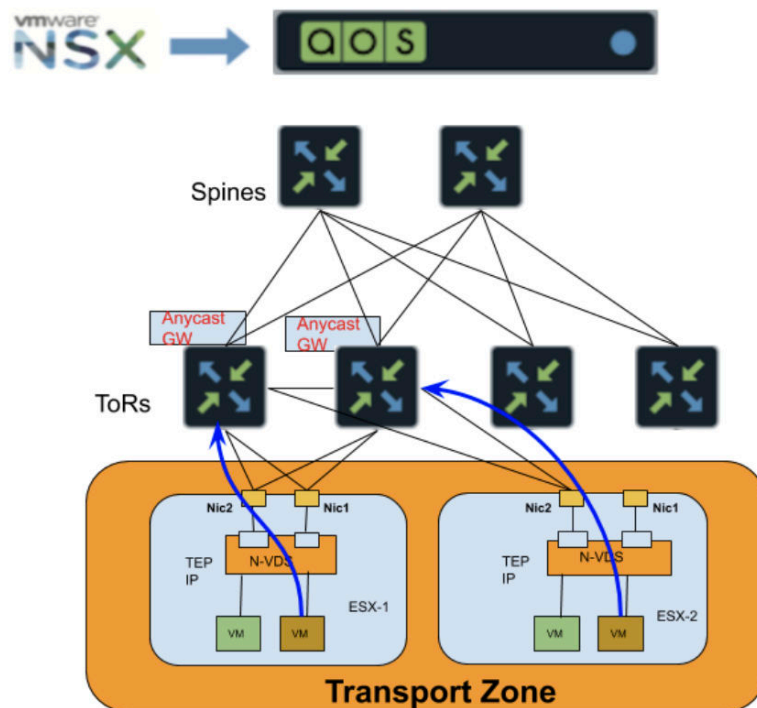
IN THIS SECTION

- [Transport Zones | 487](#)
- [N-VDS | 489](#)
- [Transport Node | 492](#)
- [NSX Edge Node | 493](#)
- [NSX Controller Cluster | 493](#)
- [NSX Manager | 493](#)

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

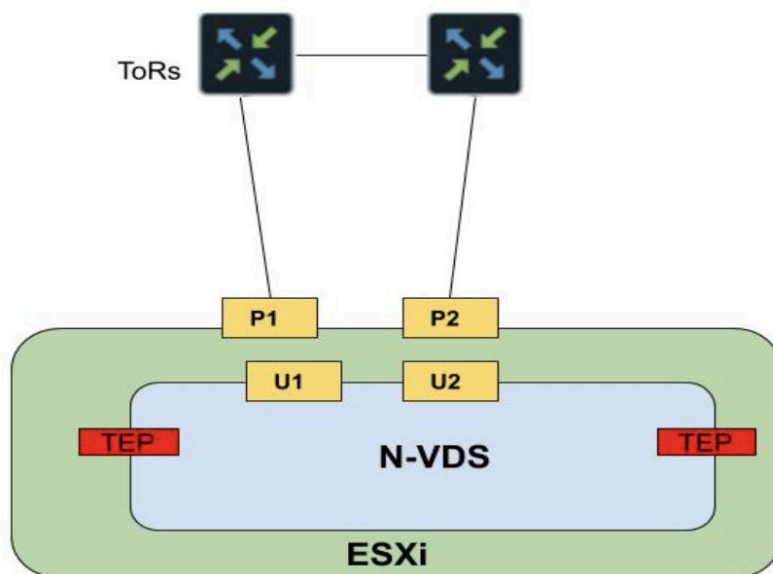
N-VDS

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:

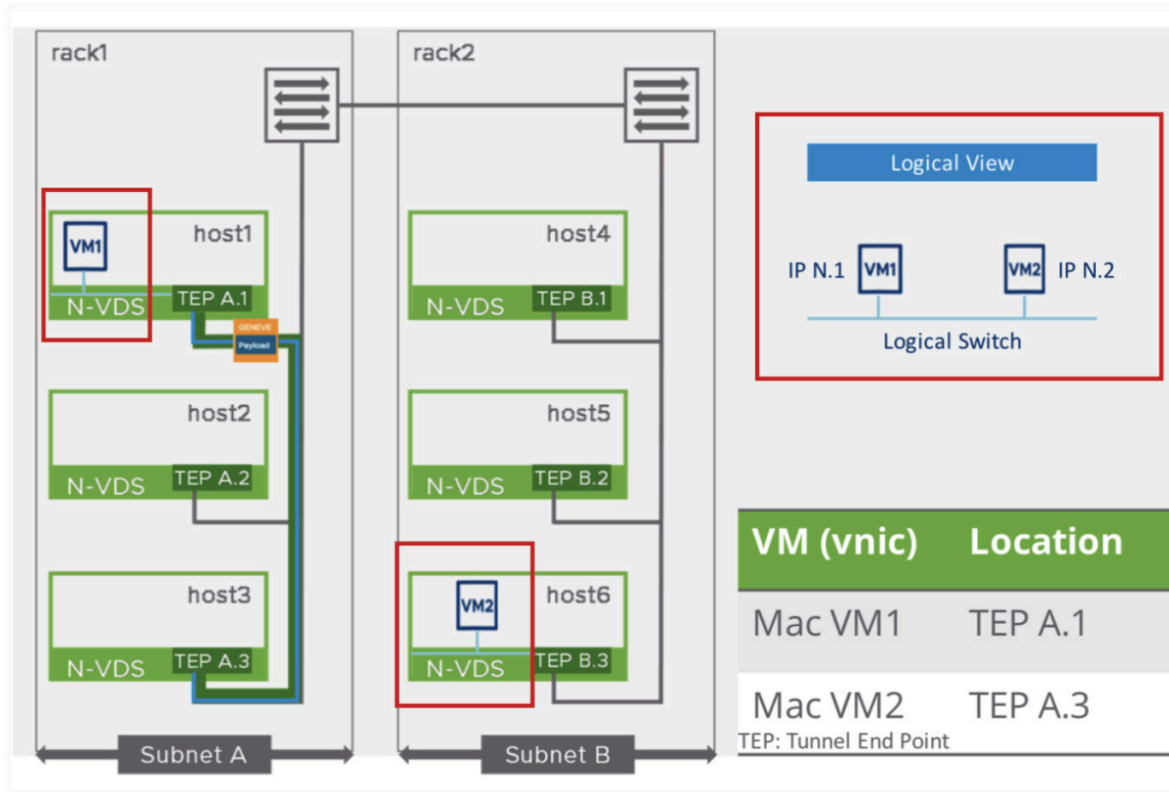
- pnics are physical ports on the ESXi host
- pnics can be bundled to form a link aggregation (LAG)
- uplinks are logical interfaces of an N-VDS

- uplinks are assigned pnic's or LAGs



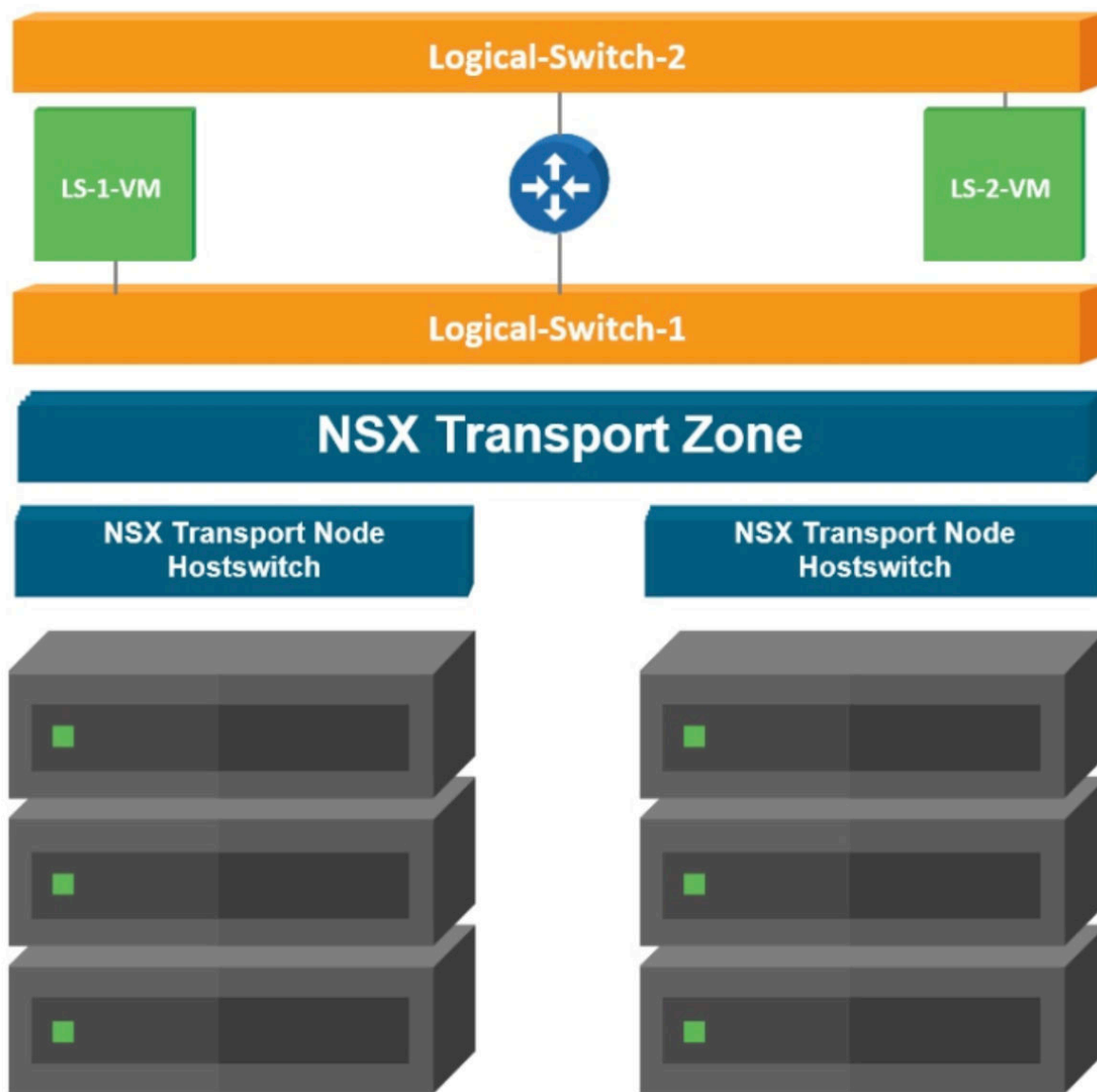
Here TEP are Tunnel Endpoints used for the NSX overlay networking (geneve encapsulation/decapsulation). P1/P2 are pNICs mapped to the uplink profile(U1/U2).

N-VDS are instantiated at the Hypervisor level and can be thought of Virtual switch connected to the ToR physical leaf devices 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:

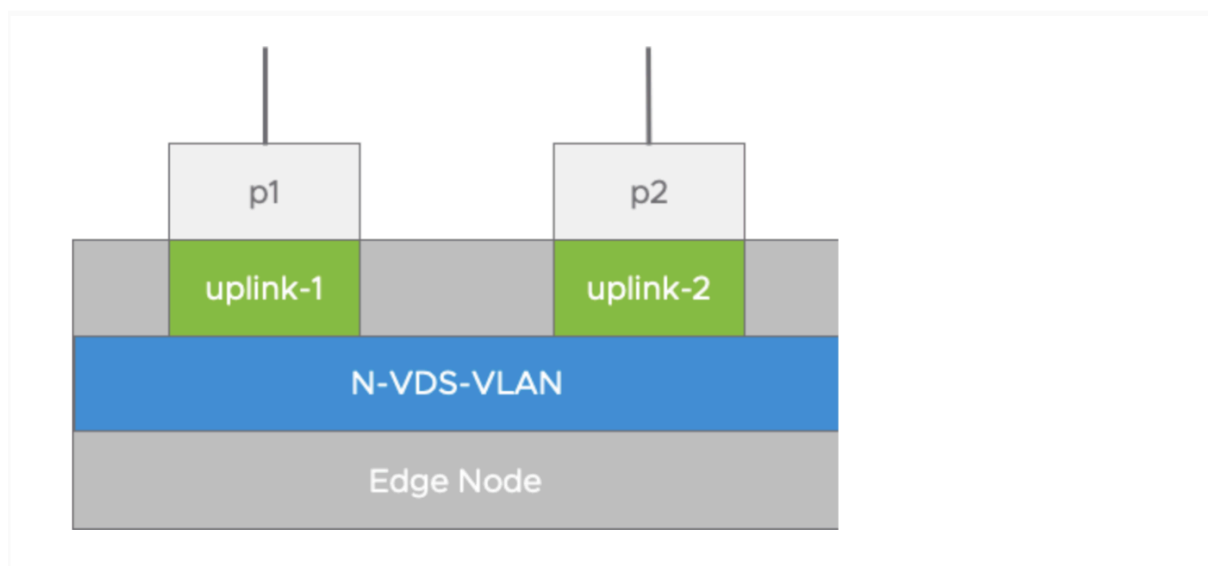
- Multiple VLAN transport zones.
- At most one overlay transport zone with a standard N-VDS.

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.

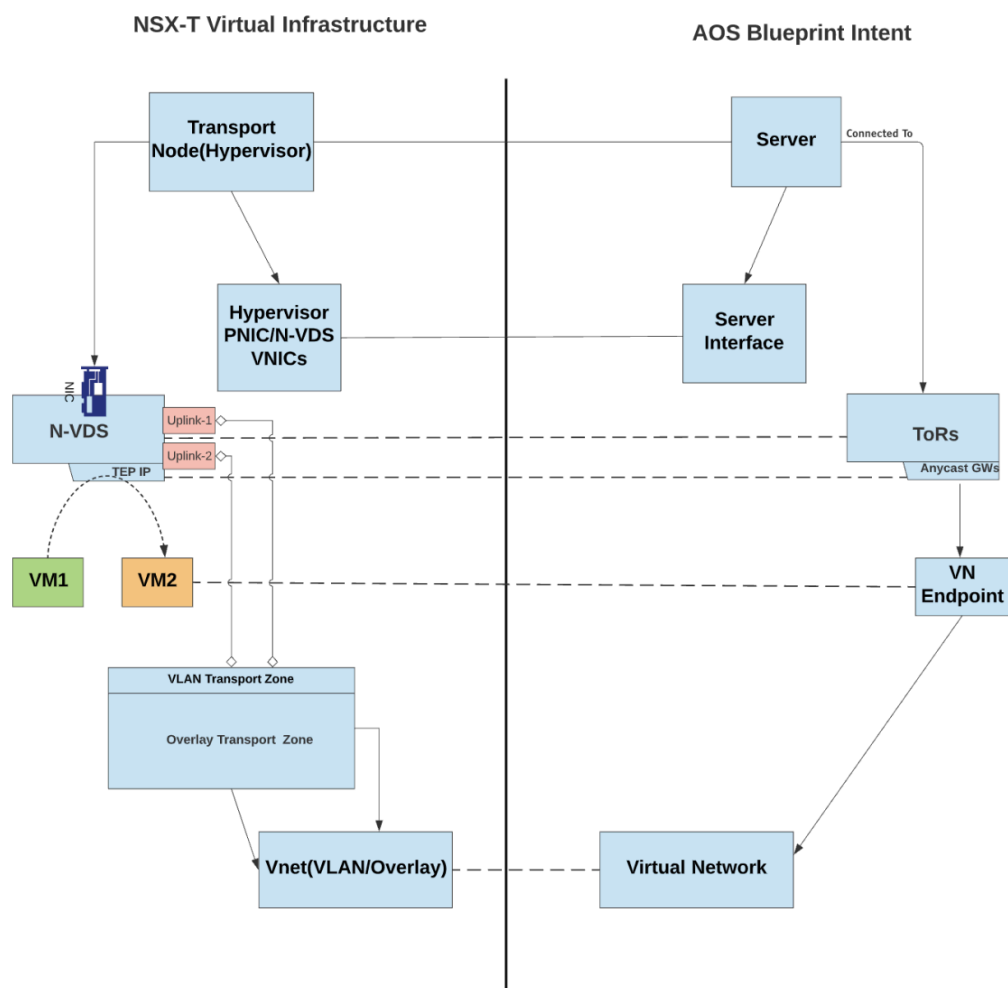
NSX Controller Cluster

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 leaf devices. 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 leaf devices 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

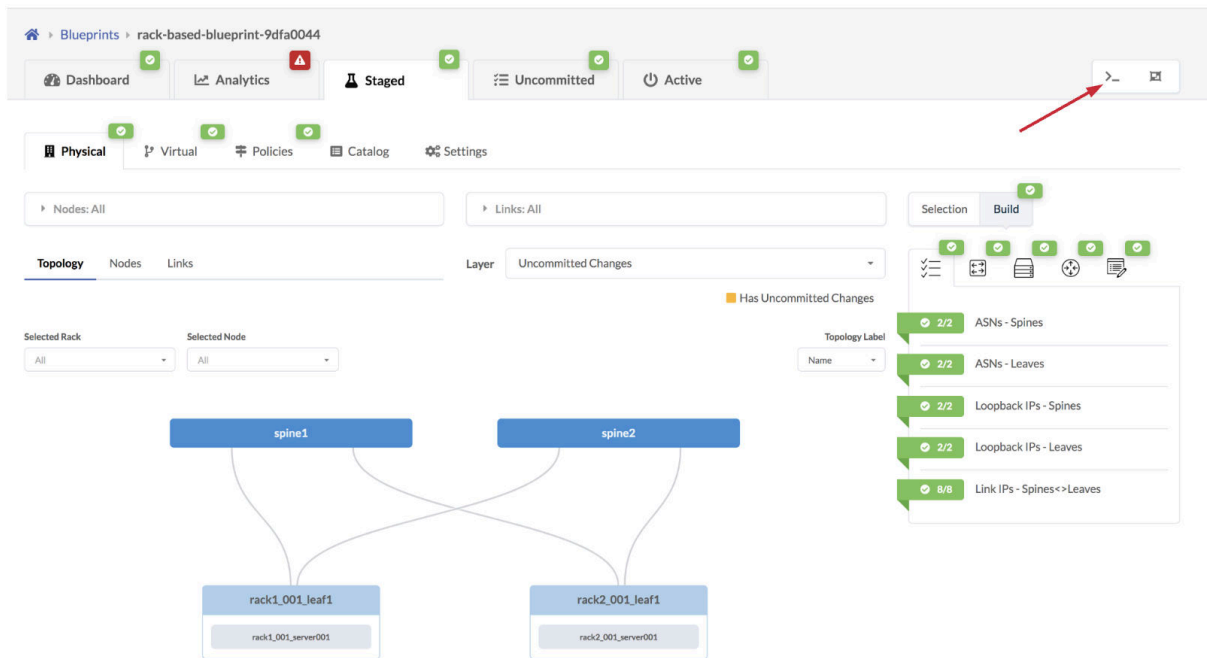
IN THIS SECTION

- [Hypervisor | 495](#)
- [Hypervisor PNIC | 498](#)
- [VNIC | 505](#)
- [Port Channel Policy | 511](#)
- [Vnet | 516](#)

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 web interface. The top navigation bar includes tabs for Dashboard, Analytics, Staged, Uncommitted, and Active. The 'Active' tab is selected. Below the navigation bar, there are filters for Physical, Virtual, Policies, Settings, Query, Anomalies, and Root Causes. The 'Nodes: Has Hypervisor? = yes' filter is applied. The 'Links: All' filter is also applied. The 'Layer' dropdown is set to 'Has VMs?'. The table displays two nodes, rack1_001_server001 and rack2_001_server001, both with the role 'L2 server' and deployment mode 'Deploy'. The 'Hypervisor' column shows 'zz-cvx-nsxt.cvx.2485377892354-2902673742-TN-1' and 'zz-cvx-nsxt.cvx.2485377892354-2902673742-TN-2' respectively. The 'VMs Count' column shows '0' for both nodes. On the right side, there is a sidebar with a list of anomalies: 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), and Deploy Mode (4/0/0).

| 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 |

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 ▼
[Create New Uplink Profile](#)

IP Assignment * Use IP Pool ▼

IP Pool * TEP IPs ▼
[OR Create and Use a new IP Pool](#)

Physical NICs

| | | |
|----------|------------|----|
| vmnic2 ▼ | uplink-1 ▼ | 🗑️ |
| vmnic3 ▼ | 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 leaf devices 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": {
    "pnict_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 ;

| Interface Id | Admin Status | Link Status | MTU | Interface Details | Stats |
|--------------|--------------|-------------|-------|-------------------|-------|
| nsx-switch.0 | 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-ip6tnl0 | Down | ? Unknown | 1452 | 1 | |
| erspan0 | Down | ? Unknown | 1450 | 1 | |

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> <input checked="" type="checkbox"/> <div>zz-cvx-nsxt.cvx.2485377892354-29...</div> </div> <div> <input type="checkbox"/> <div>zz-cvx-nsxt.cvx.2485377892354-29...</div> </div> <div> <input type="checkbox"/> <div>zz-gmat-nsxt.veos.2485377892355...</div> </div> <div> <input type="checkbox"/> <div>zz-leblon-dep-nsxt.veos.248537789...</div> </div> <div> <input type="checkbox"/> <div>zz-virt-nsxt.cvx.2485377892354-26...</div> </div> <div> <input type="checkbox"/> <div>zz-virt-nsxt.cvx.2485377892354-34...</div> </div> <div> <input type="checkbox"/> <div>zz-virt-nsxt.nxosv.2485377892354-2...</div> </div> <div> <input type="checkbox"/> <div>zz-virt-nsxt.nxosv.2485377892354-2...</div> </div> <div> <input type="checkbox"/> <div>zz-virt-nsxt.veos.2485377892354-2...</div> </div> <div> <input type="checkbox"/> <div>zz-virt-nsxt.veos.2485377892354-2...</div> </div> <div> <input type="checkbox"/> <div>zz-virt-speci-nsxt.cvx.24853778923...</div> </div> <div> <input type="checkbox"/> <div>zz-virt-speci-nsxt.cvx.24853778923...</div> </div> <div> <input type="checkbox"/> <div>zz-virt-speci-nsxt.cvx.24853778923...</div> </div> <div> <input type="checkbox"/> <div>zz-virt-speci-nsxt.veos.2485377892...</div> </div> <div> <input type="checkbox"/> <div>zz-virt-speci-nsxt.veos.2485377892...</div> </div> <div> <input type="checkbox"/> <div>zz-virt-speci-nsxt.veos.2485377892...</div> </div> <div> <input type="checkbox"/> <div>zz-virt-speci-nsxt.veos.2485377892...</div> </div> <div> <input type="checkbox"/> <div>zz-virt-speci-nsxt.veos.2485377892...</div> </div> </div> </div> </div> | | | | | |
| 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 | |
| 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
  }
}
```

- **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
  }
}
```

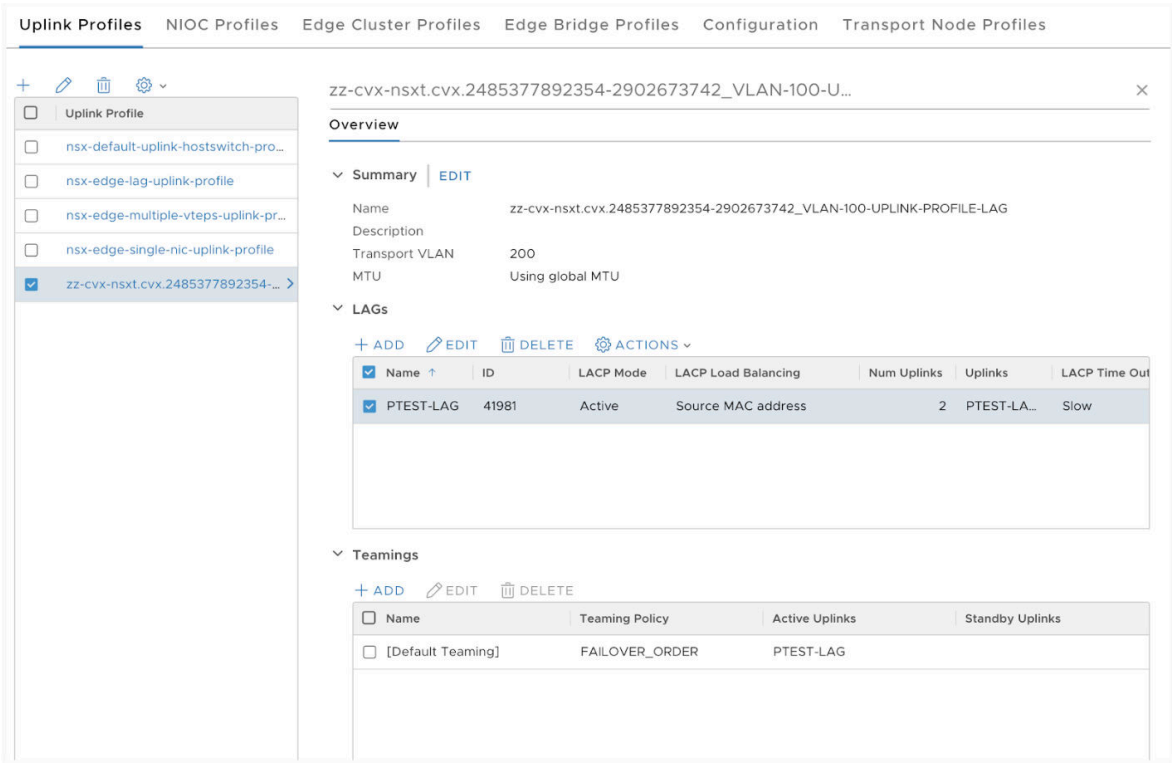


```
}
}
```

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.

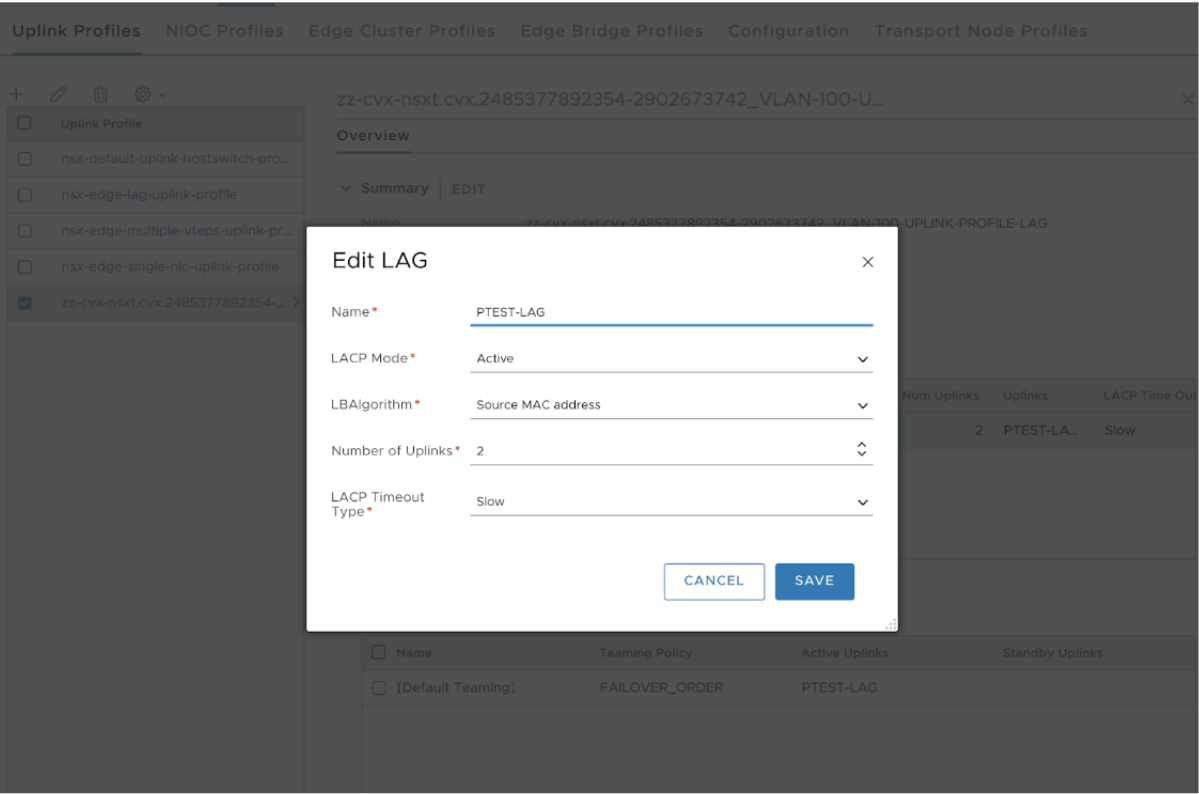
Request:

```
{
  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 | tags-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... |
| tags-test-segment | 8a53...d00c | Up | 0 | Overlay : 67... | Success | tags-76-test |
| web | 9ca1...90cc | Up | 0 | Overlay : 67... | Success | tags-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

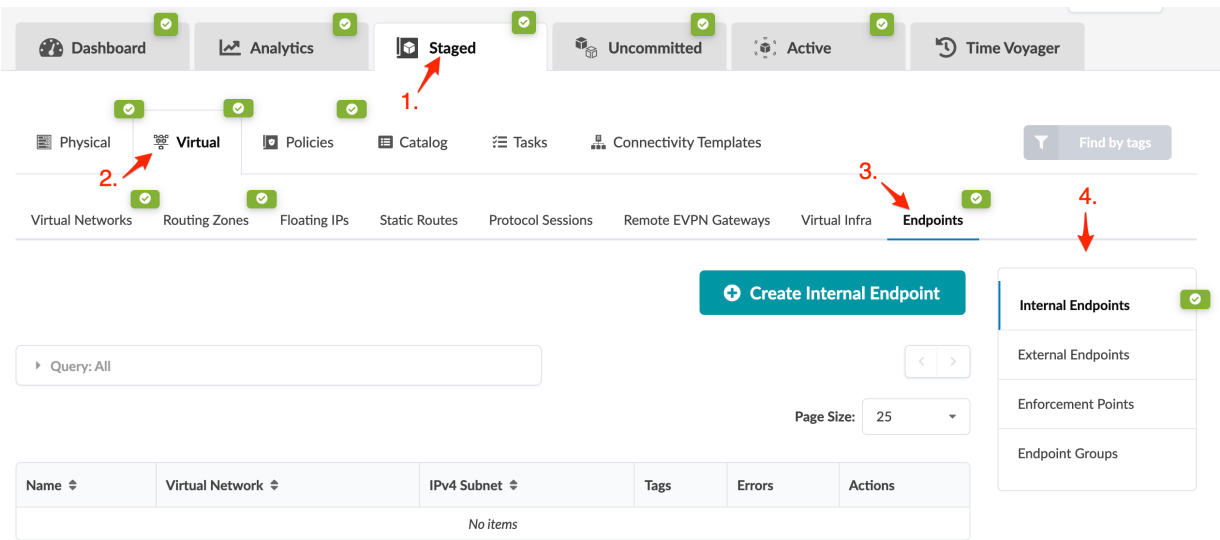
- [Internal Endpoints \(Virtual\) | 522](#)
- [External Endpoints \(Virtual\) | 524](#)

- [Enforcement Points \(Virtual\) | 525](#)
- [Endpoint Groups \(Virtual\) | 525](#)

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.) For more information about working with security policies, see ["Security Policies" on page 527](#).

From the blueprint, navigate to **Staged > Virtual > Endpoints** to go to endpoints. Click the name of a section to go to its table 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 | 523](#)
- [Edit Internal Endpoint | 523](#)
- [Delete Internal Endpoint | 523](#)

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:

| Parameter | Description |
|-----------------|--|
| Name | A unique 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 table 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 table 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 table view.

External Endpoints (Virtual)

IN THIS SECTION

- [Create External Endpoint | 524](#)
- [Edit External Endpoint | 524](#)
- [Delete External Endpoint | 525](#)

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:

| Parameter | Description |
|-------------------------------|--|
| Name | A unique 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 system, external connectivity points and enforcement groups can be added. |

3. Click **Create** to stage the endpoint addition and return to the table 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 table 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 table 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 generic systems 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 | 525
- Edit Endpoint Group | 526
- Delete Endpoint Group | 526

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:

| Parameter | Description |
|-----------|--|
| Name | A unique name, 32 characters or fewer. Alphanumeric characters, underscores and dashes only |

(Continued)

| Parameter | Description |
|-----------|---|
| Type | Select the type of endpoint group to create: Internal Endpoint Group , External Endpoint Group , or Enforcement Point Group . |
| 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"> • Internal Endpoint Group - Select multiple internal endpoints or other internal endpoint groups. • External Endpoint Group - Select multiple external endpoints or other external endpoint groups, then select enforcement points or enforcement point groups to associate with the external endpoint group. • Enforcement Points Group - Select multiple enforcement points or other enforcement point groups. |

3. Click **Create** to stage the endpoint group addition and return to the table 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 table 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 table view.

Policies (Datacenter) Staged

IN THIS SECTION

- [Security Policies | 527](#)
- [Interface Policies | 535](#)
- [Routing Policies | 543](#)
- [Routing Zone \(VRF\) Constraints | 549](#)
- [Fabric Addressing Policy | 551](#)
- [Virtual Network Policy | 553](#)
- [Anti-Affinity Policy | 556](#)
- [Validation Policy | 558](#)

Security Policies

IN THIS SECTION

- [Security Policy Overview | 527](#)
- [Security Policy Parameters | 529](#)
- [Create Security Policy | 531](#)
- [Policy Errors | 532](#)
- [Edit Security Policy | 533](#)
- [Delete Security Policy | 533](#)
- [Security Policy Search | 533](#)
- [Security Policy Conflicts | 534](#)
- [Security Policy Settings | 535](#)

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 leaf devices, 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.

Endpoints are not supported in security policies when:

- Source point is an external endpoint or external endpoint group
- Destination point is internal (internal endpoint, internal endpoint group, virtual network, routing zone)

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:

| Parameter | Description |
|-------------|--|
| Name | 32 characters or fewer, underscore, dash and alphanumeric characters only |
| Description | optional |
| Enabled | <ul style="list-style-type: none"> • ON to enable security policy (default) • OFF to disable security policy |

(Continued)

| Parameter | Description |
|------------------------|--|
| Tags | optional |
| Source Point Type | <ul style="list-style-type: none"> • Internal Endpoint (associated with VNs - contain IP /32 address) • External Endpoint (contains /32 or subnet) • External Endpoint Group • Internal Endpoint Group • Virtual Network (contains subnet) • Routing Zone (logical collection of all virtual networks and internal IP endpoints) |
| Source Point | <ul style="list-style-type: none"> • Internal Endpoint • External Endpoint • External Endpoint Group • Internal Endpoint Group • Virtual Network • Routing Zone |
| Destination Point Type | Source point (previously created) |
| Destination Point | Destination point (previously created) |
| Rule Actions | <ul style="list-style-type: none"> • Deny • Deny & Log • Permit • Permit & Log |

(Continued)

| Parameter | Description |
|------------------|--|
| Rule Protocols | <ul style="list-style-type: none"> TCP UDP IP ICMP |
| 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 displays the 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 '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 active, showing a 'Create Security Policy' button and a table of existing policies. The table has columns: Name, Source Application Point, Destination Application Point, Rule Count, Rule Conflicts, Tags, Enabled, Errors, and Actions. An example policy is shown with a 'vnet_10_on_rack_2_001_leaf1' source and 'Default routing zone' destination, with 2 rule counts. The Actions column contains icons for create, edit, and delete.

Create Security Policy

Before creating security policies, create "routing zones" on page 431, "virtual networks" on page 409, "endpoints and endpoint groups" on page 521, 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 56](#), 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 table view.

Policy Errors

1. Check the security policy in the table view for errors, which are highlighted in red.

+ Create Security Policy

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 <u>Webservers and Databases</u> | 2 | | | <div> <div style="position: relative; height: 1em;"> <div style="position: absolute; top: -10px; left: 50%; transform: translate(-50%, -100%); font-size: 0.8em;">Show errors</div> <div style="position: absolute; bottom: -10px; left: 50%; transform: translate(-50%, -100%); font-size: 0.8em;">View Log</div> </div> </div> | <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></div> |

To activate staged changes, "commit" on page 692 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 table 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 table 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 |
|---|--|--|------------|----------------|------|-------------------------------------|--------|---------|
| External to Compute | External Endpoint Group External | Virtual Network red_125_leaf3_v4 | 2 | | | <input checked="" type="checkbox"/> | | |
| Permit External Clients | 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</div> <div>any</div> </div> <div>→</div> <div> <div>Virtual Network red_125_leaf3_v4</div> <div>443</div> </div> | <div>Permit External Clients / Deny</div> <div> <div>External Endpoint External Clients</div> <div>any</div> </div> <div>→</div> <div> <div>Virtual Network red_125_leaf3_v4</div> <div>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 692](#) them to the blueprint.

Interface Policies

IN THIS SECTION

- [802.1X Server Port Authentication | 536](#)
- [Common Scenarios | 538](#)
- [802.1X Interface Policy Workflow | 539](#)
- [Create Virtual Networks for Interfaces | 539](#)
- [Create AAA Server for Interface Policy | 540](#)
- [Create 802.1x Interface Policy | 540](#)
- [Assign Ports and Fallback VNs to Interface Policy | 541](#)

802.1X Server Port Authentication

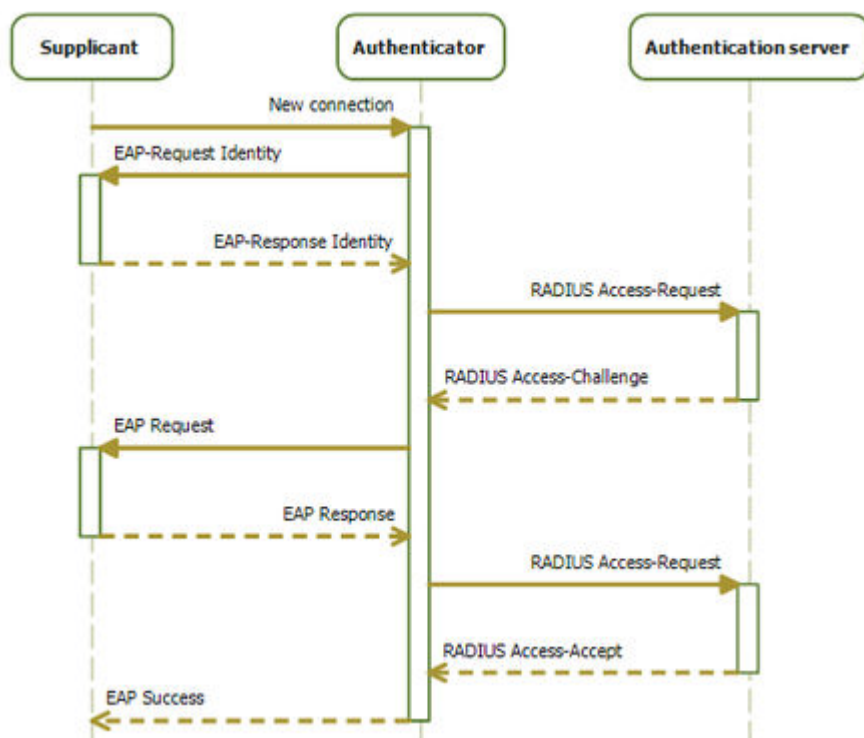
IEEE 802.1X is an IEEE Standard for network port-based Network Access Control. It is part of the IEEE 802.1 group of networking protocols. It provides an authentication mechanism to devices wishing to attach to a LAN.

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 (as a Tech Preview) and Arista EOS physical network devices only. Juniper Evolved does not at this time support this feature.



NOTE: 802.1X interface policy on Junos 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 1191](#) or contact ["Juniper Support" on page 824](#).

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 leaf devices (instead of using a resource pool). For more information about creating VLANs, see ["Virtual Networks" on page 415](#).

| Parameter | Description |
|-------------------------------|--|
| 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. |

(Continued)

| Parameter | Description |
|---|--|
| Dynamic VLAN (optional, assigned to leaf devices, 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 leaf devices, 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 564](#).

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**.

Blueprints > L2V > Staged > Policies > Interface Policies

Dashboard Analytics Staged Uncommitted Active Time Voyager

Physical Virtual Policies Catalog Settings Tasks

Security Policies Interface Policies

Create Interface Policy

Query: All 1-1 of 1 Page Size: 25

| Label | 802.1x | | | | Actions |
|-------|---------------|------------|------------------|-----------------|---------|
| | Port Control | Host Mode | MAC Auth Bypass? | Re-auth Timeout | |
| dot1x | dot1x enabled | Multi-host | no | 30 | [Icons] |

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 table 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 leaf devices, 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 |

Ports: ☒ Assigned to the current policy ☐ Assigned to another policy ☐ Unavailable for assignment
☐ Not assigned to any policy ☐ Partial

Select port and choose interfaces you want to assign to the current policy Assign All Unassign

☐ ☐ ☐ ☒ ☐ ☐ ☐

Port #5 Tr. #1 (10G, default) swp5

Fallback VN fallback_99 x

Save Change

blue-2
fallback_99
red-1

| | | |
|---------------------------|---------------------------|--------------|
| l2_virtual_mlag_001_leaf2 | l2-virtual-mlag-001-leaf2 | 525400589A65 |
|---------------------------|---------------------------|--------------|

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 leaf devices. 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 | 543](#)
- [Create Routing Policy | 548](#)
- [Edit Routing Policy | 548](#)
- [Delete Routing Policy | 548](#)

Routing Policy Overview

Routing policies include the following details:

| Parameter | Description |
|-----------|---|
| Name | 18 characters or fewer. Alphanumeric, _ and - only. |

(Continued)

| Parameter | Description |
|------------------------------------|---|
| Import Policy | <ul style="list-style-type: none"> • Default - The default BGP route (0.0.0.0/0, ::/0) is permitted. If extra import routes are defined, they are also permitted. • All - Any BGP route is permitted. • Extra Only - Only user-defined extra import routes are permitted (or denied). |
| Extra Import Routes (user-defined) | <ul style="list-style-type: none"> • Prefix - IPv4 or IPv6 network address (format: network/prefixlen) or IP address (interpreted as /32 network address). • GE Mask and LE Mask - 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. • Action - Permit or Deny |

(Continued)

| Parameter | Description |
|---------------|---|
| Export Policy | <ul style="list-style-type: none">• Spine Leaf Links - 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.• L3 Edge Server Links - Exports all leaf to L3 server links within a routing zone (VRF). On layer 2 blueprints this is an empty list.• L2 Edge Subnets - Exports all virtual networks (VLANs) that have L3 addresses within a routing zone (VRF).• Loopbacks - Exports all loopbacks within a routing zone (VRF) across spine, leaf, and L3 servers.• Static Routes - Exports all subnets in a VRF associated with static routes from all fabric systems to generic systems associated with this routing policy. |

(Continued)

| Parameter | Description |
|------------------------------------|---|
| Extra Export Routes (user-defined) | <p>User-defined export routes. These policies are additive. To advertise extra routes only, unselect all export policies.</p> <p>NOTE: To enable default route for EVPN host routes, go to Staged > Settings > Virtual Network Policy and enable the Generate EVPN host routes option.</p> <ul style="list-style-type: none"> • Prefix - IPv4 or IPv6 network address (format: network/prefixlen) or IP address (interpreted as /32 network address). • GE Mask and LE Mask - 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. • Action - Permit or Deny |
| Aggregate Prefixes | <p>If you have routing zones associated with your routing policy, and aggregate prefixes are supported on the platform (see the "Feature Matrix" on page 918) 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> <p>CAUTION: Routing policies with aggregate prefixes are applied 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> |

(Continued)

| Parameter | Description |
|-------------------------------|---|
| Expect Default IPv4 Route | To add the expectation that the default route is used in the default routing zone, check the box when you create the policy. (This field applies to the default route in the default routing zone only.) Checking this box does not change any configuration; it generates the expectation and raises an anomaly when the default route is not present. |
| Expect Default IPv6 Route | To add the expectation that the default route is used in the default routing zone, check the box when you create the policy. (This field applies to the default route in the default routing zone only.) Checking this box does not change any configuration; it generates the expectation and raises an anomaly when the default route is not present. |
| 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 table 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 table 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 table view.

Routing Zone (VRF) Constraints

IN THIS SECTION

- [Create Routing Zone Groups \(Optional\) | 549](#)
- [Create Routing Zone Constraint Policy | 549](#)
- [Edit / Delete Routing Zone Constraint Policy | 551](#)
- [Apply Routing Zone Constraint | 551](#)

Routing zone constraints allow you to constrain server-facing interfaces that connect to specific routing zones. Day-2 operators would be prevented from connecting a server to the wrong network, and assure that a given server never gets added to the wrong network. The constraint can be defined in various ways such as a list of allowed VRFs, a list of excluded VRFs, a maximum number of VRFs allowed, and so on. Once the constraint is defined, you can enforce the constraint on server-facing interfaces using connectivity templates of the type **Routing Zone Constraint**.

Create Routing Zone Groups (Optional)

If you want to constrain more than one routing zone to a single port, you can group them, then specify the group as a constraint when you create the routing zone constraint policy.

1. From the blueprint, navigate to **Staged > Virtual > Routing Zone Groups** and click **Create Routing Zone Group**.
2. Enter a group name and (optional) tags.
3. In the **Routing Zone** drop-down list, select a routing zone to add to the group and click **Add**. The routing zone is added to the **Members** list.
4. Repeat the previous step until you've added all the routing zones that you want in the group.
5. Click **Create** to create the group and return to the table view.

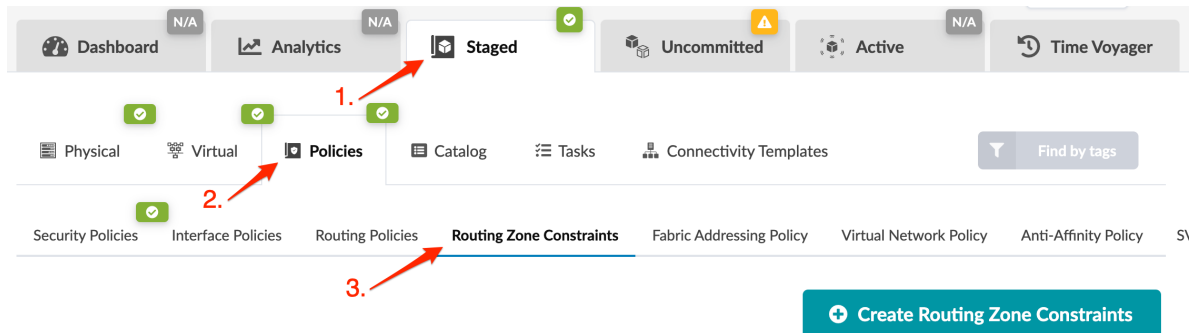
Create Routing Zone Constraint Policy

You can create a routing zone constraint policy, then later when you create a connectivity template you can apply the policy to an application point. Some examples of how you could constrain VRFs include:

- One VRF maximum
- Any VRF except Management
- Only VRFs Blue and Red

- Only VRF Group Orange

1. From the blueprint, navigate to **Staged > Policies > Routing Zone Constraints** and click **Create Routing Zone Constraints**.



Create Routing Zone Constraints

Name *

Max Count Constraint[?]

Routing Zones List Constraint *

☒ Allow[?] ☐ Deny[?] ☐ None[?]

Constraints[?]

Routing Zone Group orange

Routing Zone blue

Routing Zone **▼**

Routing Zone

Routing Zone Group

Select...

☐ Create Another?

2. Enter a name and (optional) maximum number of routing zones that the application point can be part of.
3. Set the (optional) **Routing Zones List Constraint**.
 - a. **Allow** - only allow the specified routing zones (add specific routing zones to allow)
 - b. **Deny** - denies allocation of specified routing zones (add specific routing zones to deny)
 - c. **None** - no additional constraints on routing zones (any routing zones)
4. Click **Create** to create the policy and return to the table view.

Edit / Delete Routing Zone Constraint Policy

If you need to, you can change or delete the policy after you've created it.

- If you edit the policy to increase the number of routing zones, you don't need to unassign participating ports from the restriction.
- If you edit the policy to reduce the number of routing zones, ensure that all participating ports are in compliance with the new restrictions before you save. Otherwise, you will receive an error.
- You can delete a constraint policy to free up any restrictions on the participating ports. These ports should behave as if the constraint was never applied.

Apply Routing Zone Constraint

When you want to apply the constraint to an application point, add the **Routing Zone Constraint** primitive to the connectivity template and specify the routing zone or routing zone group. For more information about connectivity templates, see ["Connectivity Templates" on page 571](#).

Create Connectivity Template

Parameters
Primitives
User-defined
Pre-defined

Summary

Title *

The New CT

Description

Tags

No tags

Routing Zone Constraint ⚠

Routing Zone Constraint *

Value is required

interface

Routing Zone Constraint

Application Point interface

Fabric Addressing Policy

IN THIS SECTION

- [Enable IPv6 Applications | 552](#)
- [ESI MAC Most Significant Byte | 552](#)

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 leaf devices and spine devices, 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.
 - 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 277](#). For more information about IPv6 configuration, see ["Virtual Networks" on page 409](#).

ESI MAC Most Significant Byte

To enable ESI (EVPN) LAG multihoming, an Ethernet segment identifier (ESI) is mandatory. ESIs identify ESI LAGs. Apstra automatically generates ESI MAC addresses using most significant byte (msb) values. Configuration of the ESI value is rendered as 10 octets. The first octet is 0. The second octet is the most significant byte value. To ensure that multicast MACs are not generated, the second octet must be an even number between 0 and 254. The second through sixth octets are 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
```

The msb value in each Apstra blueprint defaults to the value **2**. If you aren't connecting blueprints (IP fabrics) you can leave the value as is. If you're connecting blueprints via data center interconnect (DCI) and ESI, then each blueprint must have a unique most significant byte. Change the msb values so they are unique across the multiple blueprints.



CAUTION: Updating the Most Significant Byte (MSB) value regenerates all existing ESI MACs in the blueprint.

- 1. From the blueprint, navigate to **Staged > Policies > Fabric Addressing Policy** and click **Modify Settings**.
- 2. Change the ESI MAC msb value to an even number between 0 and 254. Each blueprint in the DCI must have a unique value.
- 3. Click **Save Changes** to save your changes and return to the **Fabric Addressing Policy** page.

Virtual Network Policy

IN THIS SECTION

- [Virtual Network Policy Overview | 553](#)
- [Modify Virtual Network Policy | 556](#)

Virtual Network Policy Overview

Virtual network policies include the following details:

| Parameter | 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. |

(Continued)

| Parameter | Description |
|---------------------------|--|
| 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 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. |

(Continued)

| Parameter | Description |
|--|--|
| Max Fabric Routes Count | <p>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).</p> |
| EVPN Type 5 Routes | <p>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) 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 leaf devices. 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.</p> |
| Generate EVPN host routes from ARP/IPV6 ND ARP | <p>Setting "Generate EVPN host routes from ARP/IPV6 ND ARP" adds a policy-statement to the export policy used within the fabric.</p> |

Modify Virtual Network Policy

- 1. From the blueprint, navigate to **Staged > Policies > Virtual Network Policy** and click **Modify Settings** (right side).

Physical

Virtual

Policies

Catalog

Tasks

Connectivity Templates

Find by tags

Security Policies

Interface Policies

Routing Policies

Fabric Addressing Policy

Virtual Network Policy

Anti-Affinity Policy

SVI IP Validation Policy

Modify Settings

| | |
|---|-----------|
| IP Links to Generic Systems MTU [?] | Default |
| Max External Routes Count [?] | Unlimited |
| Max MLAG Routes Count [?] | Unlimited |
| Max EVPN Routes Count [?] | Unlimited |
| Max Fabric Routes Count [?] | Unlimited |
| Generate EVPN host routes from ARP/IPV6 ND ARP [?] | Disabled |

- 2. Make your changes.
- 3. Click **Save Changes**.

Anti-Affinity Policy

IN THIS SECTION

[Anti-Affinity Policy Overview | 556](#)

[Enable/Disable Anti-Affinity Policy | 557](#)

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:

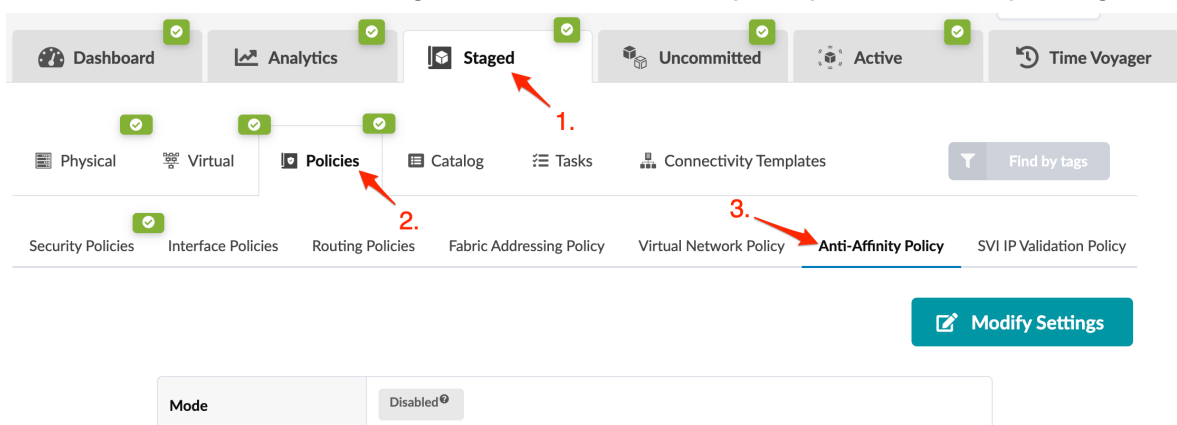
- **Disabled** (default) - ports selection is based on assigned interface maps and interface names (provided or auto-assigned). Port breakouts could terminate on the same physical ports.
- **Enabled (loose)** - controls interface names that were not defined by the user. Does not control or override user-defined 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**.

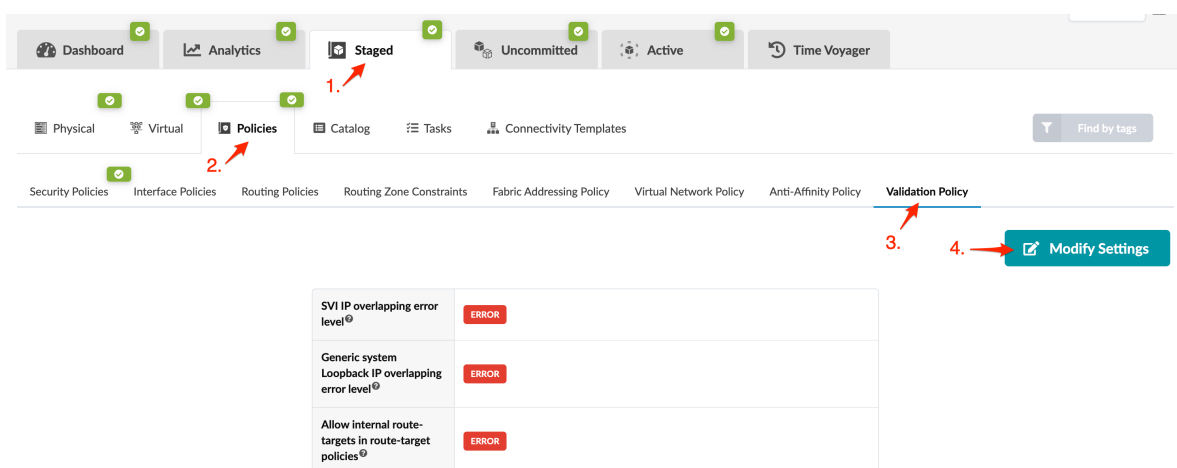


2. Change the policy mode, and if you're enabling the policy, enter a maximum number of links, as applicable.
3. Click **Save Changes** to stage the change and return to the policies view.

To activate staged changes, commit them from the **Uncommitted** tab.

Validation Policy


1. From the blueprint, navigate to **Staged > Policies > Validation Policy** and click **Modify Settings**.



2. Change settings, as applicable:
 - **No Warning** - If validation fails, no warning or error will be generated.
 - **Warning** - If validation fails, warnings will be raised; you can commit changes.
 - **Error** (default) - If validation fails, errors will be raised that must be resolved before you can commit changes.

Generic system Loopback IP overlapping error level and **Allow internal route-targets in route-target policies** are new in Apstra version 4.1.1.

Modify Validation Policy Settings



Experimental Features

Modifying the severity level of blueprint validation errors must be done with caution. Relaxation of each setting to levels lower than 'ERROR' carries its own risk. Please contact support for clarifications.

SVI IP overlapping error level

☐ NO WARNING ☐ WARNING ☒ ERROR

Defines the severity of the error which is raised when there are duplicated SVI IP addresses within a single virtual network. This setting could cause unexpected traffic flow for routed traffic.

Generic system Loopback IP overlapping error level

☐ NO WARNING ☐ WARNING ☒ ERROR

Support for uncontrolled generic loopbacks IP overlap errors. This relaxes validation errors and allows IP addresses to be shared between external generic loopbacks and fabric objects both in default and EVPN routing zones. These fabric objects include loopback IPs, physical link IPs, logical link IPs, virtual network subnets and VTEP addresses.

Allow internal route-targets in route-target policies

☐ NO WARNING ☐ WARNING ☒ ERROR

Severity of errors raised on overlap of a user-defined route-target which overlaps with an internal virtual network or routing zone route-target. This can be used for a form of full table inter-vrf route leaking. This feature is currently marked as experimental and is not a validated design.

Save Changes

3. Click **Save Changes** to stage the changes and return to the **Validation Policy** page.

To activate staged changes, commit them from the **Uncommitted** tab.

Logical Devices (Datacenter Blueprint)

IN THIS SECTION

- [Logical Devices Overview \(Blueprint Catalog\) | 559](#)
- [Export Logical Device | 560](#)

Logical Devices Overview (Blueprint Catalog)

The logical devices in the blueprint catalog are determined by the template that was used to create the blueprint. 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.

The screenshot shows the network management interface. The top navigation bar includes Dashboard, Analytics, Staged, Uncommitted, Active, and Time Voyager. The sub-navigation bar includes Physical, Virtual, Policies, Catalog, Tasks, and Connectivity Templates. The 'Catalog' section is expanded, showing Logical Devices, Interface Maps, Property Sets, Configlets, AAA Servers, and Tags. The 'Logical Devices' section is selected, showing a table of logical devices. The table has columns: Name, Capabilities, Panels Count, Ports Count, Ports Summary, and Actions. The first row shows a logical device named 'AOS-1x10-1' with capabilities '1 x 10 Gbps', 1 panel, and 1 port. The 'Ports Summary' column shows 'AOS-1x10-1' and '1 x 10 Gbps Leaf • Access'. The 'Actions' column has a button labeled 'Export to global catalog'.

| Name | Capabilities | Panels Count | Ports Count | Ports Summary | Actions |
|------------|--------------|--------------|-------------|--|--------------------------|
| AOS-1x10-1 | 1 x 10 Gbps | 1 | 1 | AOS-1x10-1 1 x 10 Gbps Leaf • Access | Export to global catalog |
| AOS-2x10-1 | | | | | |

Export Logical Device

- 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).
- 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.
- Click **Export** to export the logical device and return to the table view.

Interface Maps (Datacenter Blueprint)

IN THIS SECTION

- [Interface Maps Overview \(Blueprint\) | 561](#)
- [Import Interface Map | 561](#)
- [Delete Interface Map \(Blueprint\) | 562](#)

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.

2.

3.

Import Interface Map

Query: All

1-9 of 9

Page Size: 25

| Name | Device Profile | Logical Device | Actions |
|--|---|----------------------------|---------|
| Generic_Server_1RU_1x10G_AOS-1x10-1 | Generic_Server_1RU_1x10G | AOS-1x10-1 | |
| 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 13 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 table 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 table view.

Property Sets (Datacenter Blueprint)

IN THIS SECTION

- [Import / Re-import Property Set \(Datacenter Blueprint\) | 563](#)
- [Delete Property Set \(Datacenter Blueprint\) | 564](#)

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.

Dashboard

Analytics

Staged

Uncommitted

Active

Time Voyager

Physical

Virtual

Policies

Catalog

Tasks

Connectivity Templates

Logical Devices

Interface Maps

Property Sets

Configlets

AAA Servers

Tags

Find by tags

Import Property Set

Query: All

1-1 of 1

Page Size: 25

| Name | Keys | Stale? | Actions |
|------------|------------------|----------------------|------------------------------|
| NTP server | { {NTP_SERVER} } | As in global catalog | <div><div>Delete</div></div> |

Import / Re-import Property Set (Datacenter Blueprint)

IN THIS SECTION

- [Import Property Set | 563](#)
- [Re-import Property Set | 563](#)

Import Property Set

1. Make sure the "property set" on page 52 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 table 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 > Catalog > Property Sets' interface. Red arrows indicate the following steps:

1. Click on the 'Catalog' tab in the top navigation bar.
2. Click on the 'Property Sets' sub-tab.
3. Click on the 'Import Property Set' button (a blue button with a plus icon).
4. Click on the 'Re-import' button in the 'Actions' column of the table.

The table below shows the 'Property Sets' data:

| Name | Keys | Stale? | Actions |
|------------|----------------|-------------------------------|---|
| NTP server | {{NTP_SERVER}} | Different from global catalog |  |

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 table view.

Delete Property Set (Datacenter 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 summary table view.

AAA Servers (Datacenter Blueprint)

IN THIS SECTION

- [AAA Servers Overview | 564](#)
- [Create AAA Server | 566](#)
- [Edit AAA Server | 566](#)
- [Delete AAA Server | 566](#)
- [Configure AAA RADIUS Server | 566](#)
- [Configure Client Supplicant | 567](#)

AAA Servers Overview

AAA servers are used with ["interface policies" on page 535](#). AAA servers include the following details:

| Parameter | Description |
|-----------|----------------------------|
| Label | To identify the AAA server |

(Continued)

| Parameter | Description |
|-----------------|---|
| Server Type | <ul style="list-style-type: none"> • RADIUS 802.1x - 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. • RADIUS COA (Change of Authorization) - 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 assign dynamic VLANs after authentication instead of during auth. All RADIUS COA implementations are hard-coded to auth port 3799. |
| 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.

1. Staged

2. Catalog

3. AAA Servers

Create AAA Server

Query: All

1-1 of 1

Page Size: 25

| Label | Server Type | Hostname | Auth Port | Accounting Port | Actions |
|------------|---------------|--------------|-----------|-----------------|---------|
| freeradius | RADIUS 802.1x | 172.20.191.5 | 1812 | N/A | |

Edit

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 table 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 table 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 table view.

Configure AAA RADIUS Server

Configuring AAA RADIUS servers 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"
```

This example shows a simple credential; when you configure you may use any EAP method that both the client and RADIUS server support.

Configure Client Supplicant

Configuring client supplicant is 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 (Datacenter Blueprint)

IN THIS SECTION

- [Tags Overview \(Blueprint\) | 568](#)
- [Search Tags \(Blueprint\) | 569](#)
- [Find by Tags | 569](#)
- [Create Tag \(Blueprint\) | 570](#)
- [Import Tag | 570](#)
- [Export Tag | 570](#)
- [Edit Tag \(Blueprint\) | 570](#)
- [Delete Tag \(Blueprint\) | 570](#)

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.

1. Click **Staged** in the top navigation bar.

2. Click **Catalog** in the left sidebar.

3. Click **Tags** in the bottom navigation bar.

Buttons: **Create Tag**, **Find by tags**

Query: All

Page Size: 25

| Name | Applied To | Description | Actions |
|-----------------|---------------------------|-------------|--|
| Clients | CONNECTIVITY TEMPLATE : 1 | | ↺ ✎ 🖨 🗑 |
| external router | NODE : 2 | | Export ✎ 🖨 🗑 |

Click for details

Search Tags (Blueprint)

You can filter tagged elements based on tag names and/or element types.

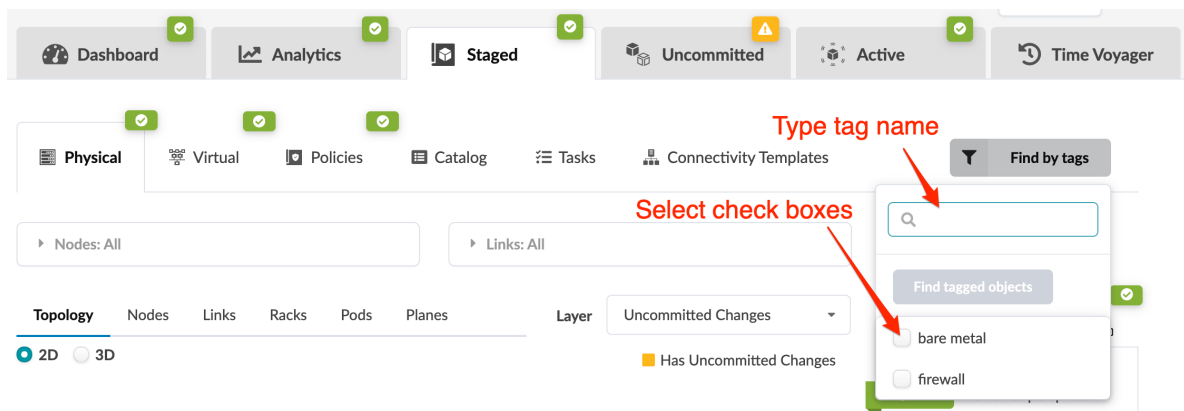
- From the blueprint, navigate to **Staged > Catalog > Tags** and click **Query** to open the dialog.
- Enter search criteria:
 - To see elements associated with tags, enter tag name(s) in the **Name** field.
 - To see tags that elements are associated with, select element type(s) from the drop-down list in the **Applied To** field.
 - To filter both by tag name and element type, enter details in both fields.
- Click **Apply** to see filtered results in the table.
- To go to the table 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.

Find by Tags

With **Find by Tags**, you can search the entire blueprint for nodes, links, and connectivity templates that have associated tags.

- 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.

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 table 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 table 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 table view.

Tasks (Datacenter) Staged

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 | 574](#)
- [Create Connectivity Template for Multiple VNs on Same Interface \(Example\) | 586](#)
- [Create Connectivity Template for Layer 2 Connected External Router \(Example\) | 589](#)
- [Assign Connectivity Template | 592](#)
- [Edit Connectivity Template | 596](#)
- [Delete Connectivity Template | 596](#)

Connectivity templates enable you to apply various network configurations to devices connected to generic systems, as a Day 2 operation. Devices could be leaf devices, spine devices, or in 5-stage Clos topologies, superspine devices. 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-leaf devices and external routers.

Connectivity templates consist of combinations of primitives as described in later sections.

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. To see static routes and protocol sessions, navigate to **Staged > Virtual** in the blueprint.

From the blueprint, navigate to **Staged > Connectivity Templates** to go to the connectivity template table view. You can create, assign, edit, and delete connectivity templates.

1. Click 'Staged' in the top navigation bar.

2. Click 'Connectivity Templates' in the left sidebar.

Click CT name for details

| Name | Description | Tags | Primitives | Status | Actions |
|--|-------------|------|---|---------------------------|--|
| rtr_leaf1_leaf2:l3:ct_bgp_subintf_to_subintf:ipv4_ipv6 | | | <ul style="list-style-type: none"> BGP Peering (Generic System) IP Link | Assigned on 2 endpoint(s) | Link Edit Delete |
| vn_endpoints_blue_300_evpn_mlag_001_l_v4_vlan_tagged | | | <ul style="list-style-type: none"> Virtual Network (Single) | Assigned on 3 endpoint(s) | Link Edit Delete |

With advanced search (new in Apstra version 4.1.0) you can filter based on primitive types, and based on the types, you can show parameters and filter on those parameters. You can take this search to multiples levels. For example, you can search for all the logical links in routing zone green or all the static routes with the same next hop.

In Apstra version 4.1.0 and 4.1.1, the advanced search and the status query are two separate search boxes.

▼ Advanced Search: All

Add item to query:

Main Properties

[CT Properties \(title, tags\)](#)

Primitive Types

[Virtual Network \(Single\)](#)

[Virtual Network \(Multiple\)](#)

[IP Link](#)

[Static Route](#)

[Custom Static Route](#)

[BGP Peering \(IP Endpoint\)](#)

🔍 Apply

↺ Clear

▼ Query: All

Status

🔍 Apply

↺ Clear

In Apstra version 4.1.2, all search fields are combined into one search box. (The Status query moved into CT Properties.)

▼ Advanced Search: All

Add item to query:

Main Properties

[CT Properties \(title, tags, status\)](#)

Primitive Types

[Virtual Network \(Single\)](#)

[Virtual Network \(Multiple\)](#)

[IP Link](#)

[Static Route](#)

[Custom Static Route](#)

[BGP Peering \(IP Endpoint\)](#)

🔍 Apply

↺ Clear

Primitives

IN THIS SECTION

- Virtual Network (Single) Primitive | 575
- Virtual Network (Multiple) Primitive | 576
- IP Link Primitive | 576
- Static Route Primitive | 577
- Custom Static Route Primitive | 578
- BGP Peering (IP Endpoint) Primitive | 579
- BGP Peering (Generic System) Primitive | 580
- Dynamic BGP Peering Primitive | 582
- Routing Policy Primitive | 583
- Routing Zone Constraint Primitive | 584
- User-defined | 585
- Pre-defined | 586

The **Primitives** tab includes the supported configuration functions that can be added to connectivity templates.

Create Connectivity Template

Parameters
Primitives
User-defined
Pre-defined

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

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

Application Point
any

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) Primitive

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

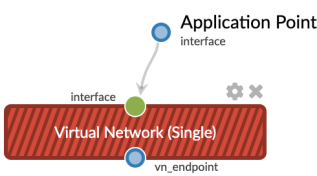
▼ Virtual Network (Single) ⚠

Virtual Network ID *

Value is required

Virtual Network Tag Type *

☒ VLAN Tagged☐ Untagged



Virtual Network (Multiple) Primitive

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 Primitive

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. If you select a tagged interface, the VLAN ID is required.

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

ParametersPrimitivesUser-definedPre-defined

▼ IP Link ⚠

Routing Zone *

Value is required

Interface Type * ⓘ

Tagged

Untagged

VLAN ID * ⓘ

2

Value is required

IPv4 Addressing Type *

None

Numbered

IPv6 Addressing Type *

None

Link local

The diagram illustrates the connection of an IP Link primitive. A red rectangular box labeled 'IP Link' contains a green dot labeled 'interface' and a blue dot labeled 'ip_link'. An arrow points from a blue dot labeled 'Application Point interface' to the 'interface' dot within the IP Link box. A gear icon and a close icon are located to the right of the IP Link box.

To see the IPv4 address that's assigned, navigate to **Staged > Routing Zones**, then select the routing zone name in the table and scroll down to the **Interfaces** section.

Static Route Primitive

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 ⓘ

The diagram illustrates a network configuration. At the top, a blue circle labeled 'Application Point' with the text 'ip_link, vn_endpoint' below it is connected by a grey arrow to a green circle. This green circle is also labeled 'ip_link, vn_endpoint'. Below the green circle is a red rectangular box with diagonal hatching, labeled 'Static Route'. To the right of the 'Static Route' box are a gear icon and a close icon (X).

Custom Static Route Primitive

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 form for a 'Custom Static Route' primitive. The form includes the following fields and errors:

- Routing Zone**: A dropdown menu with a red 'Value is required' error message.
- Network**: A text input field containing '203.0.113.0/24 or 2001:db8::/32' with a red 'Value is required' error message.
- Next Hop IP Address**: A text input field containing '198.51.100.5 or fc00::5' with a red 'Value is required' error message.

On the right side, a diagram illustrates the configuration: a 'system' (green dot) is connected to an 'Application Point system' (blue dot) via a 'Custom Static Route' (red box). The 'Custom Static Route' box also contains a gear icon and a close icon.

BGP Peering (IP Endpoint) Primitive

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 (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

protocol_endpoint

BGP Peering (IP Endpoint)

BGP Peering (Generic System) Primitive

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 (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)
- Local ASN - Configured on a per-peer basis. It allows a router to appear to be a member of a second autonomous system (AS) by prepending a local-as AS number, in addition to its real AS number, announced to its eBGP peer, resulting in an AS path length of two.
- 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 BGP peering configuration. It shows a central 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 Primitive

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 (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 uses the local virtual network (from when you assigned the connectivity template) as the subnet value. In this case, if the virtual network only has a virtual gateway IP address, and it doesn't have any specific IP address per leaf switch, then it also renders an additional IP address on the leaf SVI besides the virtual gateway IP address.
- 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 'Primitives' tab is also visible. To the right, a diagram illustrates the application point configuration.

Parameters Tab:

- Dynamic BGP Peering:**
 - IPv4 AFI:** ON
 - IPv6 AFI:** OFF
 - TTL:** 2
 - Single-hop BFD:** OFF
 - Password:** [Empty field]
 - Keep Alive Timer (sec):** [Empty field]
 - Hold Time Timer (sec):** [Empty field]
 - IPv4:** OFF
 - IPv6:** OFF
 - IPv4 Subnet for BGP Prefix Dynamic Neighbors:** [Empty field]

Primitives Tab:

- Application Point:** ip_link, svi
- Dynamic BGP Peering:** ip_link, svi, protocol_endpoint

Routing Policy Primitive

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

Parameters

PrimitivesUser-definedPre-defined

▼ Summary

Title *

The New CT

Description

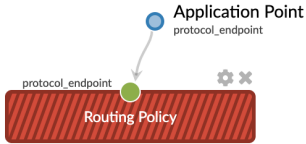
Tags

No tags

▼ Routing Policy ⚠

Routing Policy *

Value is required



Routing Zone Constraint Primitive

When you want to apply the routing zone constraint to an application point, add the Routing Zone Constraint primitive to the connectivity template and specify the routing zone or routing zone group.

Create Connectivity Template

ParametersPrimitivesUser-definedPre-defined

▼ Summary

Title *

The New CT

Description

Tags

No tags

▼ Routing Zone Constraint ⚠

Routing Zone Constraint *

Value is required

Application Point
interface

interface
Routing Zone Constraint

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

Application Point
any

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 primitives that ship with the Apstra software.

Create Connectivity Template

The screenshot shows the 'Create Connectivity Template' interface. At the top, there are tabs for 'Parameters', 'Primitives', 'User-defined', and 'Pre-defined'. The 'Pre-defined' tab is selected. On the left, there is a search bar labeled 'Quick Search' and a list of pre-defined connectivity templates: 'BGP Dynamic over L3 connectivity', 'BGP over L2 connectivity', and 'BGP over L3 connectivity'. On the right, there is a sidebar with three options: '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)), and 'Pre-defined' (Re-create Connectivity Template based on a pre-defined template). The sidebar also includes the 'Application Point' logo.

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 up to 64 primitives in each connectivity template (increased from 18 as of Apstra version 4.0.1). We'll use examples to illustrate the process. First we'll show 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

The screenshot shows the 'Create Connectivity Template' interface with the 'Parameters' tab selected. The form includes fields for 'Title' (labeled 'The New CT'), 'Description', and 'Tags' (labeled 'No tags'). The sidebar on the right is the same as in the previous screenshot, showing options for 'Primitives', 'User-defined', and 'Pre-defined' templates, along with the 'Application Point' logo.

2. In the **Parameters** tab, enter a connectivity name in the **Title** field. You can optionally enter a description, and tags that you can 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

1. The 'Primitives' tab is selected.

2. The 'Virtual Network (Single)' primitive is selected.

3. The selected primitive appears in the staging area.

The staging area diagram shows an 'Application Point interface' connected to a 'Virtual Network (Single)' block, which has an 'interface' and a 'vn_endpoint'.

4. 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

1. The 'Parameters' tab is selected.

2. The 'Virtual Network (Single)' configuration section is highlighted.

The configuration section includes:

- Virtual Network ID ***: A dropdown menu with a red border and a message 'Value is required'.
- Virtual Network Tag Type ***: Radio buttons for 'VLAN Tagged' (selected) and 'Untagged'.

The staging area diagram on the right remains the same as in the previous screenshot.

5. When it's successfully configured, the color of the selected primitive changes from red to gray. Click the **Primitives** tab.

Create Connectivity Template

The screenshot shows the 'Create Connectivity Template' interface. The 'Parameters' tab is active, displaying fields for Title, Description, Tags, and Virtual Network ID. The 'Primitives' tab is highlighted with a red arrow. The 'Virtual Network (Single)' section is expanded, showing the 'Virtual Network ID' field with the value 'blue_300_leaf3_v4 (40000)' and the 'Virtual Network Tag Type' set to 'VLAN Tagged'. To the right, a diagram shows an 'Application Point' connected to an 'interface' on a 'Virtual Network (Single)', which is then connected to a 'vn_endpoint'.

6. From the **Primitives** tab, click **Virtual Network (Multiple)**.

Create Connectivity Template

The screenshot shows the 'Create Connectivity Template' interface with the 'Primitives' tab active. A red arrow labeled '1.' points to the 'Primitives' tab. Below the 'Quick Search' bar, two results are listed: 'Virtual Network (Single)' and 'Virtual Network (Multiple)'. A red arrow labeled '2.' points to 'Virtual Network (Multiple)'. A red arrow labeled '3.' points to the 'Virtual Network (Multiple)' box in the diagram on the right. The diagram shows an 'Application Point' connected to an 'interface' on a 'Virtual Network (Single)', which is then connected to a 'vn_endpoint'. Another 'Virtual Network (Multiple)' box is shown, also connected to an 'interface' and a 'vn_endpoint'.

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 table 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

| <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">BGP Peering (Generic System)IP Link | Assigned on 2 endpoint(s) | Link Edit Delete |
| <input type="checkbox"/> | The New CT | | | <ul style="list-style-type: none">Virtual Network (Multiple)Virtual Network (Single) | Ready | Link Edit Delete |

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)

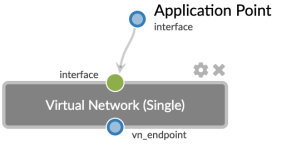
Virtual Network ID *

blue_300_leaf3_v4 (40000)

Virtual Network Tag Type *

☒ VLAN Tagged

☐ Untagged



- 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...

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

2. ...you can see what you can attach to it.

- 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 with the 'Primitives' tab selected. The 'Primitives' tab lists several connectivity templates:

- 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

On the right, a diagram illustrates the staging area. It shows an 'Application Point interface' connected to a 'Virtual Network (Single)' box. The 'Virtual Network (Single)' box has an 'interface' port connected to the 'Application Point interface' and a 'vn_endpoint' port. The 'vn_endpoint' port is connected to a 'BGP Peering (Generic System)' box. The 'BGP Peering (Generic System)' box has an 'ip_link, vn_endpoint' port connected to the 'vn_endpoint' and a 'protocol_endpoint' port. A red arrow points from the 'BGP Peering (Generic System)' box to the text '1. Select a primitive...'. Another red arrow points from the 'BGP Peering (Generic System)' box to the text '2. ...to add it to the staging area'.

- Proceed with configuring the parameters and click **Create** to create the template.

Assign Connectivity Template

IN THIS SECTION

- Assign Connectivity Template Overview | 592
- Method 1 | 593
- Method 2 | 594
- Force Assign VN Templates | 595

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 table 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:i3:ct_bgp_subintf_to_subintf:ipv4 | | | <ul style="list-style-type: none"> BGP Peering (Generic System) IP Link | Assigned on 2 endpoints | |
| <input type="checkbox"/> | vn_endpoints_blue_300_leaf3_v4_vlan_tagged | | | <ul style="list-style-type: none"> Virtual Network (Single) | Assigned on 1 endpoint(s) | |
| <input type="checkbox"/> | vn_endpoints_blue_301_leaf4_v4_vlan_tagged | | | <ul style="list-style-type: none"> Virtual Network (Single) | 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**. 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

☒ 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="radio"/> <input checked="" type="radio"/> <input type="radio"/> |

Assign

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

☒ 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_subintfip4 | + |

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**. 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

ON Table view

Pod

All

Rack

All

Node

All

Applied templates

All

Application point search

Search...

Tags

All

1-8 of 8

<

>

Applicable templates

Filter applicable templates

Bulk assign templates

Page Size: 25

Filter selected by

all

selected only

unselected only

| <input type="checkbox"/> 0 selected | Pod | Rack | Node | Application point | Tags | Applied templates | Actions |
|-------------------------------------|------|------------------|--------------|---|------|---|---------|
| <input type="checkbox"/> | pod1 | evpn-esi_001_001 | leaf1 (Leaf) | xe-0/0/0 -> rtr_leaf1_leaf2 (Interface) | | rtr_leaf1_leaf2:13:ct_bgp_subintf_to_subintf_ipv4 | |

Assign

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 table 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 table 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 table 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 table view.

Active (Datacenter Blueprint)

IN THIS SECTION

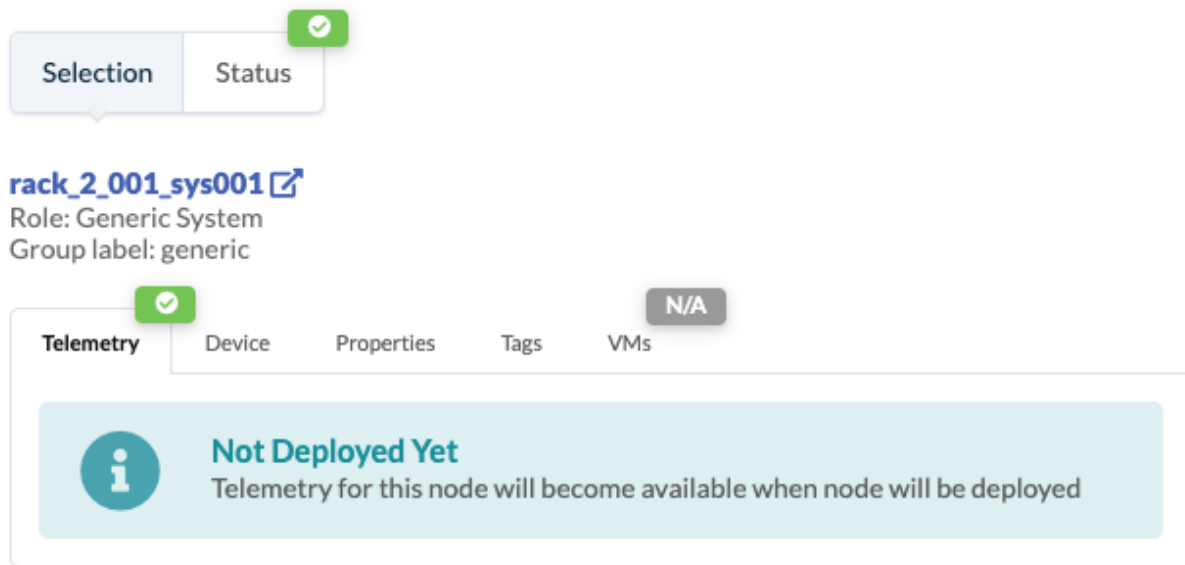
- [Active Blueprint Overview | 597](#)
- [Selection Panel | 597](#)
- [Status Panel | 598](#)
- [Topology \(Active\) | 598](#)
- [Nodes \(Active\) | 607](#)
- [Links \(Active\) | 609](#)
- [Racks \(Active\) | 610](#)
- [Pods \(Active\) | 611](#)
- [Query | 612](#)
- [Anomalies \(Service\) | 613](#)
- [Root Causes | 620](#)

Active Blueprint Overview

When you deploy your network (by committing the staged blueprint), network status and other details are shown in the **Active** view. From here, you can monitor your network and see any anomalies at-a-glance. You can filter alerts and anomalies by different layers to conduct root cause analysis of problems.

Selection Panel

When you select a node in the active **Topology** or **Nodes** view, information about telemetry, device, properties, tags, and VMs for that node are available in the right **Selection** panel.



When you select a link in the active **Topology** or **Links** view, properties and tags information for that link is available in the right **Selection** panel.

Status Panel

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.

Topology (Active)

IN THIS SECTION

- [2D Topology View \(Active\) | 599](#)
- [3D Topology View \(Active\) | 600](#)
- [Neighbors View \(Active\) | 601](#)
- [Links View \(Active Topology\) | 604](#)
- [Virtual Networks Endpoints \(Active\) | 605](#)
- [Headroom \(Topology\) | 605](#)

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.

1.

2.

3.

Make selection from drop-down lists, or...

...click selection directly for more info

Rollover an element to see info

3D Topology View (Active)

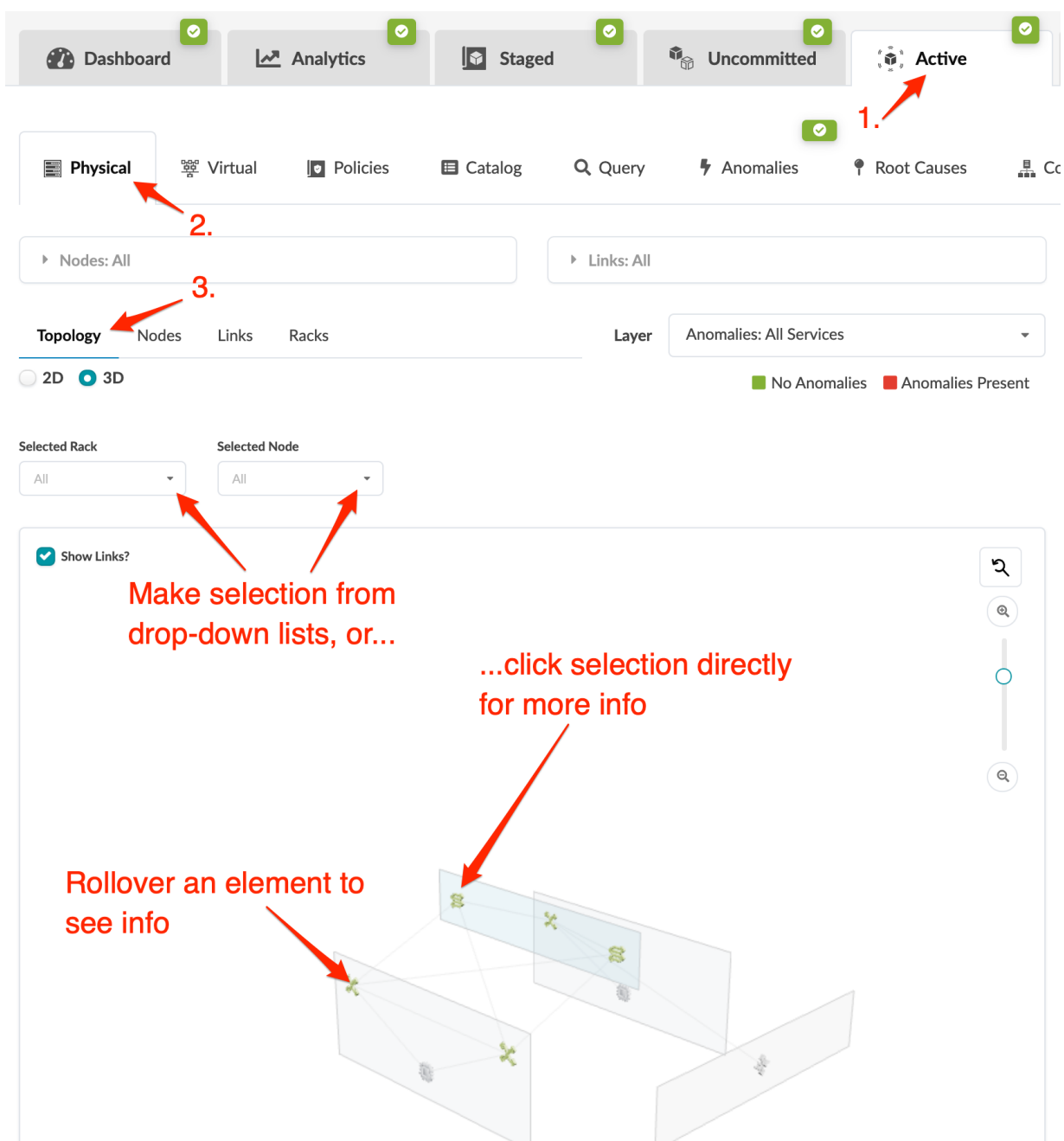


NOTE: This feature has been classified as a "Juniper Apstra Technology Preview" on page 1191. These features are "as is" and voluntary use. "Juniper Technical Support" on page 824 will attempt to resolve any issues that customers experience when using these features and create bug reports on behalf of support cases. However, Juniper may not provide comprehensive support services to Tech Preview features.

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.

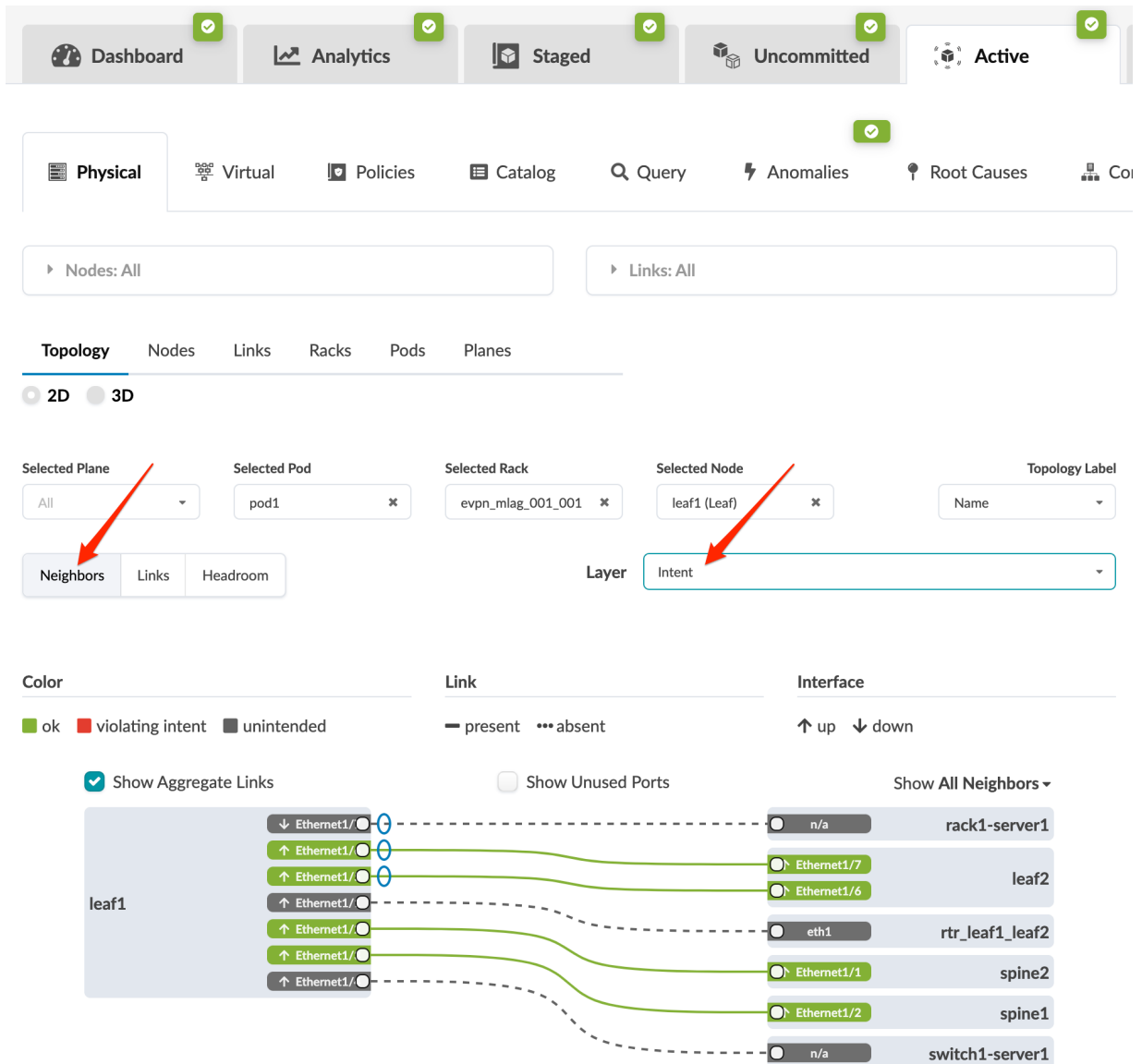


Neighbors View (Active)

- To show aggregate links, click the **Show Aggregate Links** check box.

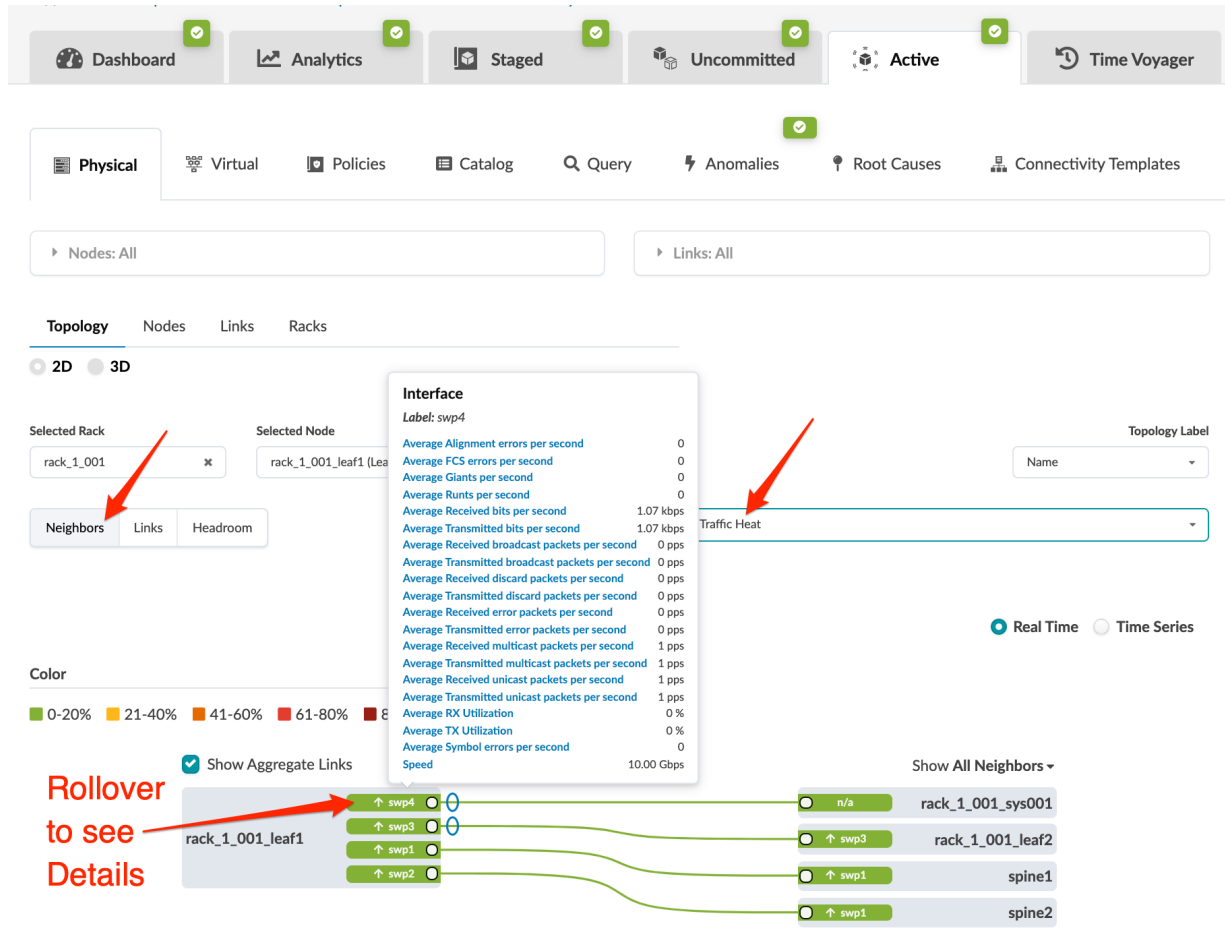
- To show unused ports, click the **Show Unused Ports** check box.
- To show a different label (name, hostname, S/N), select a different label from the **Topology Label** drop-down list (right side).
- To show a different layer, select a different layer from the **Layer** drop-down list.
- Choose to show all neighbors or only specific ones (generic, leaf, spine, and so on).

The intent layer is shown below.

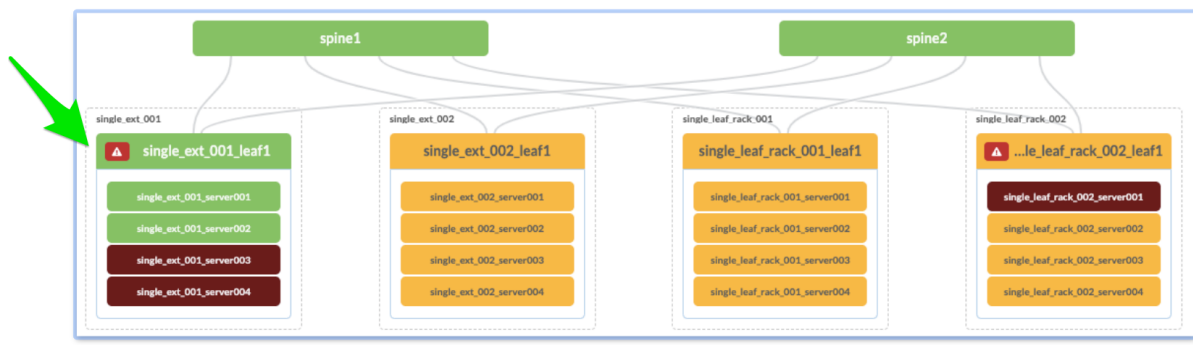


The traffic heat layer is shown below. The colors represent different available/used capacity based on the current system level TX/RX, averaged to 2 minutes, by default. If the aggregated TX or RX across all the device interfaces is < 20% it's **green**. If it's between 21-40%, it's **yellow** and so on. For each 20%

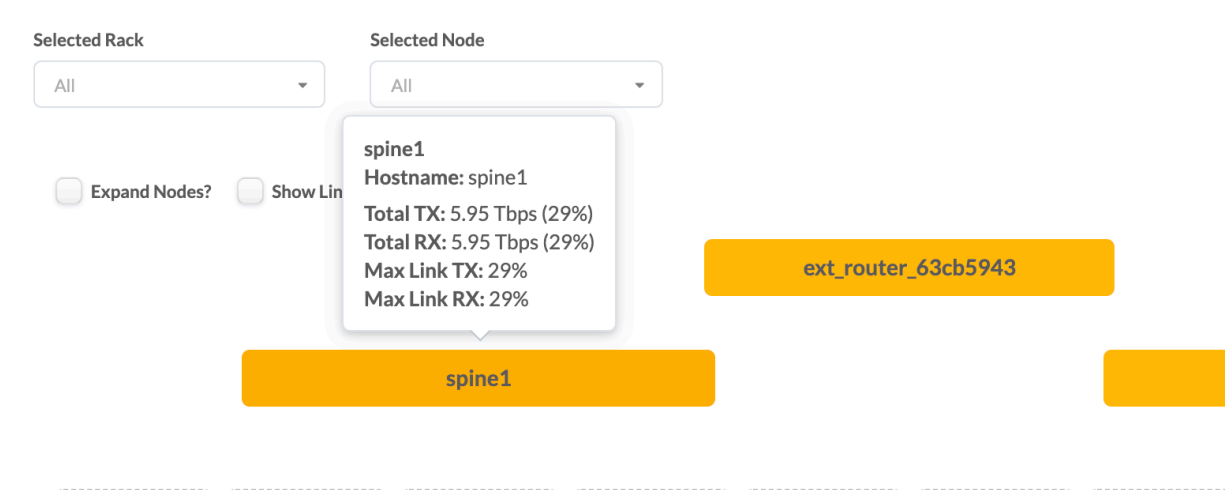
difference, capacity is shown with a different color. (Server color is calculated based on the interface counters of the leaf ports facing that server. To see RX/TX per interface for a single node, click the node.



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.

DashboardAnalyticsStagedUncommittedActive

PhysicalVirtualPoliciesCatalogQueryAnomaliesRoot Causes

Nodes: AllLinks: All

TopologyNodesLinksRacksPodsPlanes

2D3D

Selected PlaneAll

Selected Podpod1

Selected Rackevpn_single_001_001

Selected Nodeswitch3-server1 (Gen eric System)

Topology LabelName

NeighborsLinksVirtual Networks EndpointsHeadroom

Query: All1-7 of 7

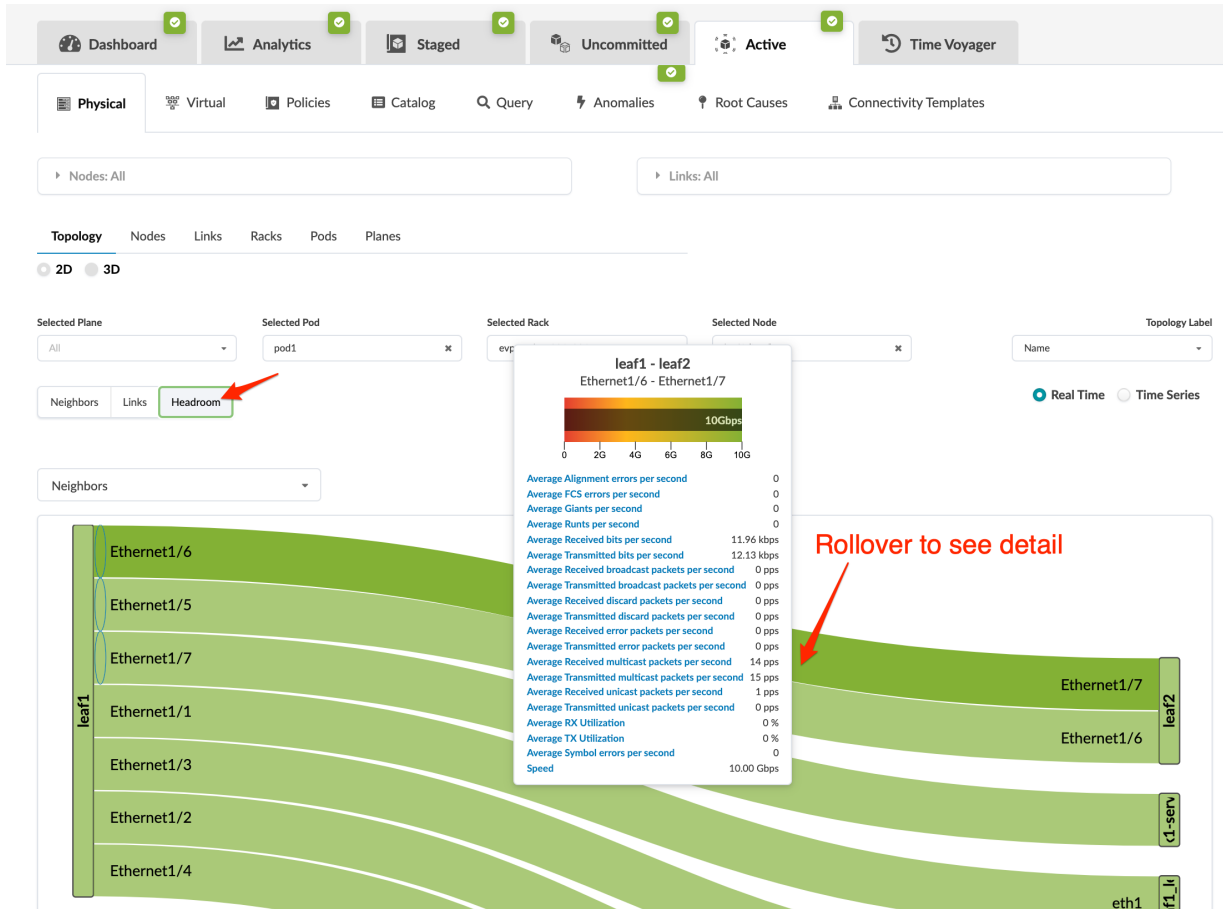
Page Size: 25

| 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 |

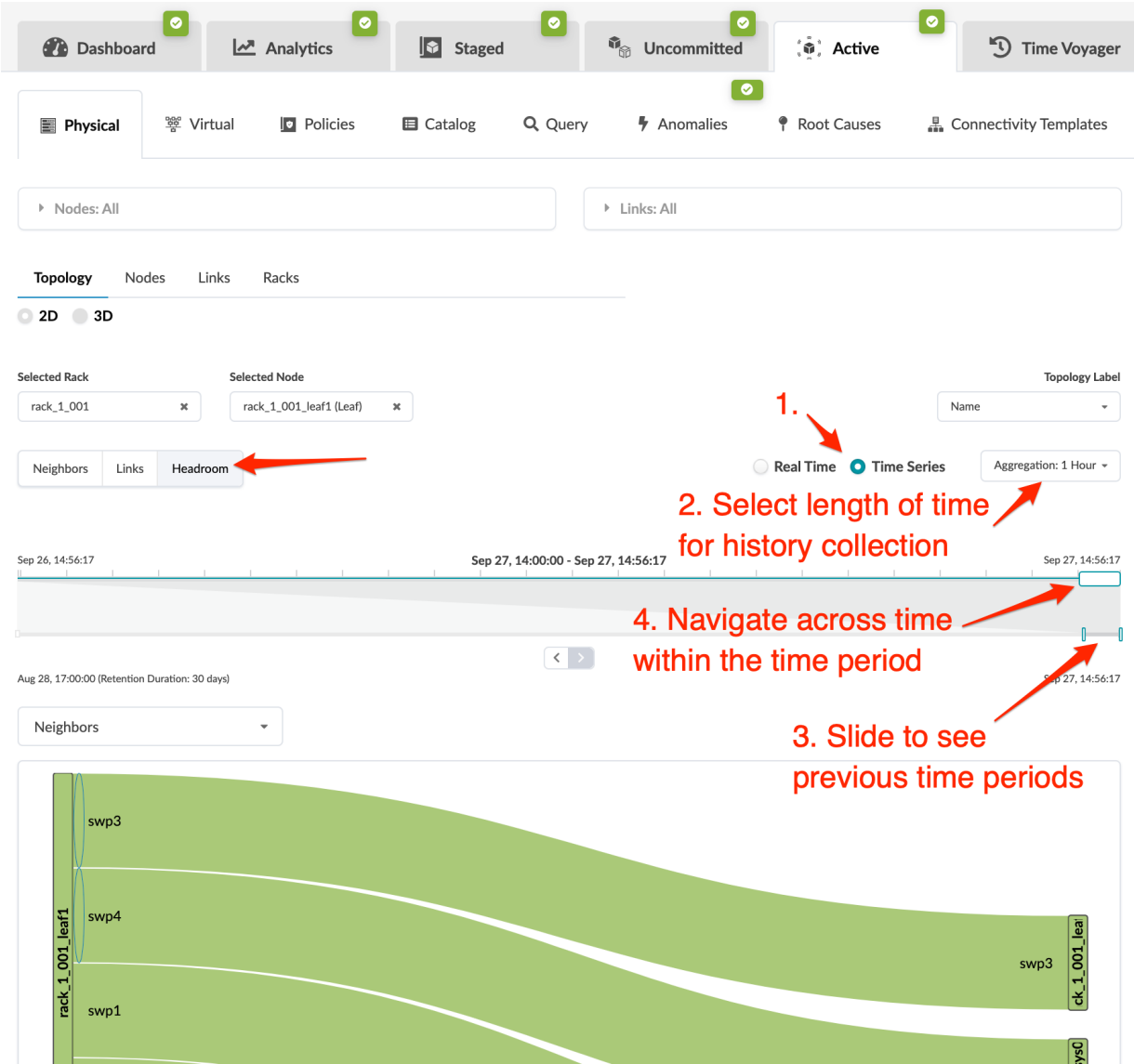
Headroom (Topology)

i

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 1010.



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 | 608](#)
- [Apply Full Config | 608](#)

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 left sidebar. The main area displays a table of nodes. Red arrows and numbers indicate the following steps:

- 1.** Points to the 'Active' tab in the top navigation bar.
- 2.** Points to the 'Physical' tab in the left sidebar.
- 3.** Points to the 'Nodes' tab in the left sidebar.

Callouts include:

- Select a name to see more details in right panel:** Points to the 'sspine1' node in the table.
- Select what to display:** Points to the 'Columns (10/20)' dropdown menu, which is open showing options like Name, Tags, Role, External?, and Pod.
- Table Card:** Points to the table view toggle icon.

The table shows columns: Name, Tags, Role, External?, Deploy Mode, Device Profile, Hostname, and Deploy Status. The right panel shows details for 'sspine1' (Role: Superspine) with tabs for Telemetry, Device, Properties, and Tags. The 'Telemetry' tab is active, showing a list of services like Probes, All Services, Liveness, Config, Interface, and Cabling.

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 613](#).

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 | 609](#)
- [Export Cabling Map | 609](#)

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 left sidebar. The main content area displays a table of links. Annotations include:

- 1.** Points to the 'Active' tab in the top navigation bar.
- 2.** Points to the 'Physical' tab in the left sidebar.
- 3.** Points to the 'Links' tab in the left sidebar.

Below the table, there is a section for 'Export cabling map' and a table of links. The table has columns for Name, Role, Speed, Tags, Endpoint 1, and Endpoint 2. The first two rows of data are shown:

| Name | Role | Speed | Tags | Endpoint 1 | Endpoint 2 |
|----------------------------------|----------------|-------|------|------------|------------|
| leaf001_001_1<->leaf001_001_2[1] | Leaf Peer Link | 10G | | leaf1 | leaf2 |
| leaf001_001_1<->leaf001_001_2[1] | Leaf Peer Link | 10G | | leaf2 | leaf1 |

Annotations for the table include:

- Export cabling map**: Points to the 'Export cabling map' button.
- Click a name...**: Points to the 'Name' column header.
- ... to see properties**: Points to the 'Name' column header.

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](#) | 610

To go to rack details in the active blueprint, navigate to **Active > Physical > Racks**. You can change the default view from a table to a list. You can search for specific racks by name or rack type.

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 'Active' tab selected in the top navigation bar. Below it, the 'Physical' tab is selected in the sub-navigation bar. The 'Racks' tab is also selected in the sub-sub-navigation bar. The interface displays a list of racks, with 'evpn_mlag_001_001' selected. The 'Rack Properties' panel on the right shows the 'Name' field with an edit icon. The 'Generic Systems Capacity' section shows a table of server capacities.

Annotations in the image:

- 1. Points to the 'Active' tab in the top navigation bar.
- 2. Points to the 'Physical' tab in the sub-navigation bar.
- 3. Points to the 'Racks' tab in the sub-sub-navigation bar.
- 4. Points to the 'evpn_mlag_001_001' rack name in the list.
- 5. Points to the edit icon next to the 'evpn_mlag_001_001' rack name in the 'Rack Properties' panel.

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 deployed pods. You can search for specific nodes or links and select a layer to see anomalies, deploy modes, deployment status and more. 3-stage topologies have one pod, while 5-stage topologies have two or more pods. Click a

rack name in a pod to see its rack type preview.

1. Click **Active** in the top navigation bar.

2. Click **Physical** in the left sidebar.

3. Click **Pods** in the top navigation bar.

4. Click a rack type name (e.g., **global**) in the pod table to see a preview.

Select layer to see anomalies, deployment status and more

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 |

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**.

1. Click **Active** in the top navigation bar.

2. Click **Query** in the top navigation bar.

3. Click **MAC**, **ARP**, or **VMs** in the left sidebar.

4. Click **Query: All** in the top navigation bar.

1-25 of 204 < > Page Size: 25

| Node Name | S/N | Hostname | Type | VLAN | VXLAN | MAC | Interface |
|-----------|--------------|----------|--------|------|-------|-------------------|-----------|
| leaf3 | 52540060B540 | leaf3 | Static | 0 | | 00:1c:73:00:00:01 | sup-eth1 |

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

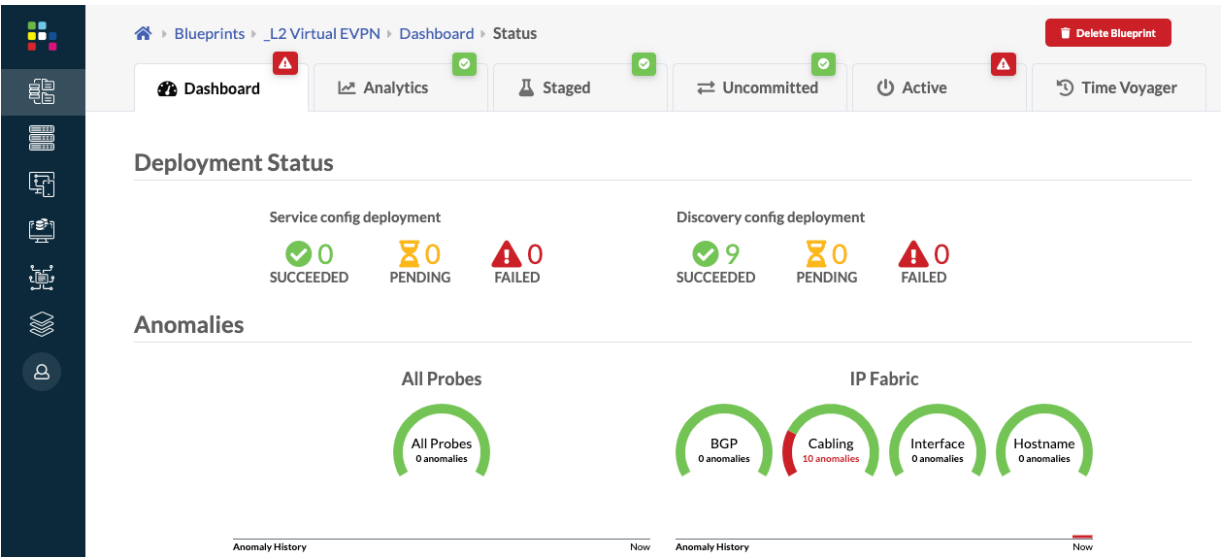
Discovery Anomalies | 613

Configuration Deviation | 617

This section covers service anomalies. For analytics anomalies see ["IBA Anomalies." on page 706](#)

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.

Blueprints > _L2 Virtual EVPN > Active > Anomalies

Dashboard Analytics Staged Uncommitted Active Time Voyager

Physical Virtual Policies Settings Query **Anomalies** Root Causes

Query: Service = IP Fabric and Anomaly Type = cabling 1-10 of 10 Page Size: 25

| Node | Hostname | Service | Anomaly Type | Role | Anomaly Extra Details | Expected | Actual | Time Updated |
|----------------------------|---------------------------|-----------|--------------|---------------|---|---|---|----------------|
| spine1 | spine1 | IP Fabric | cabling | Spine to Leaf | Property Value Interface "evp1" | 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 "evp4" | 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**.

Blueprints > _L2 Virtual EVPN > Active > Physical > Status

Dashboard Analytics Staged Uncommitted Active Time Voyager

Physical Virtual Policies Settings Query **Anomalies** Root Causes

Nodes: All Links: All Selection Status

Topology Nodes Links Racks Layer Anomalies: All Services

No Anomalies Anomalies Present

Selected Rack: All Selected Node: All Topology Label: Name

Expand Nodes? Show Links?

router1

spine1 spine2 spine3

_l2_virtual_evpn_001
_l2_virtual_evpn_001_leaf1
_l2_virtual_evpn_001_leaf2
_l2_virtual_evpn_001_server001

_l2_virtual_evpn_002
_l2_virtual_evpn_002_leaf1
_l2_virtual_evpn_002_leaf2
_l2_virtual_evpn_002_server001

_l2_virtual_evpn_003
_l2_virtual_evpn_003_leaf1
_l2_virtual_evpn_003_leaf2
_l2_virtual_evpn_003_server001

10 Anomalies: All Services
0 Anomalies: BGP
10 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
0/0/0 Deploy Mode
9/0/0 Deployment Status: Discovery
0/0/0 Deployment Status: Service









To see the topology view of the anomalies affecting spine1, click **Spine1** in the topology.

The screenshot shows the network management interface with the following components:

- Top Navigation Bar:** Includes breadcrumbs (Blueprints > _L2 Virtual EVPN > Active > Physical > Selection > Node) and tabs (Dashboard, Analytics, Staged, Uncommitted, Active, Time Voyager).
- Physical Tab:** Contains sub-tabs for Physical, Virtual, Policies, Settings, Query, Anomalies, and Root Causes.
- Filters:** Nodes: All, Links: All, Selection, Status.
- Topology View:** Shows a diagram of spine1 with its interfaces (swp1 to swp6) and their connections to other nodes (Ethernet1/1, _l2_virtual_evpn_002..., _l2_virtual_evpn_001..., _l2_virtual_evpn_003...).
- Right Panel:** Contains a list of services with status indicators (green for OK, red for violating intent, grey for unintended). The services listed are:
 - Probes (green)
 - All Services (red, 5 violations)
 - Liveness (green)
 - Config (green)
 - Interface (green)
 - Cabling (red, 5 violations)
 - BGP (green)
 - Route (green)
 - 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 |
|-----------|--------------|---------------|-----------------------|--------|--------------------|-----------------------------|--------------------|-----------------------------|--------------|
| | | | Property | Value | Property | Value | Property | Value | |
| IP Fabric | cabling | Spine to Leaf | Interface | "svp3" | neighbor interface | "svp1" | neighbor interface | "Ethernet1/1" | 4 hours ago |
| | | | | | neighbor name | "12-virtual-evpn-001-leaf2" | neighbor name | "12-virtual-evpn-002-leaf1" | |
| IP Fabric | cabling | Spine to Leaf | Interface | "svp6" | neighbor interface | "Ethernet1" | neighbor interface | "Ethernet1" | 4 hours ago |
| | | | | | neighbor name | "12-virtual-evpn-003-leaf1" | neighbor name | "12-virtual-evpn-003-leaf2" | |
| IP Fabric | cabling | Spine to Leaf | Interface | "svp5" | neighbor interface | "Ethernet1" | neighbor interface | "Ethernet1" | 4 hours ago |
| | | | | | neighbor name | "12-virtual-evpn-003-leaf2" | neighbor name | "12-virtual-evpn-003-leaf1" | |
| IP Fabric | cabling | Spine to Leaf | Interface | "svp1" | neighbor interface | "Ethernet1/1" | neighbor interface | "svp1" | 4 hours ago |
| | | | | | neighbor name | "12-virtual-evpn-002-leaf1" | neighbor name | "12-virtual-evpn-001-leaf1" | |
| IP Fabric | cabling | Spine to Leaf | Interface | "svp2" | neighbor interface | "svp1" | neighbor interface | "svp1" | 4 hours ago |
| | | | | | neighbor name | "12-virtual-evpn-001-leaf1" | 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 386](#).

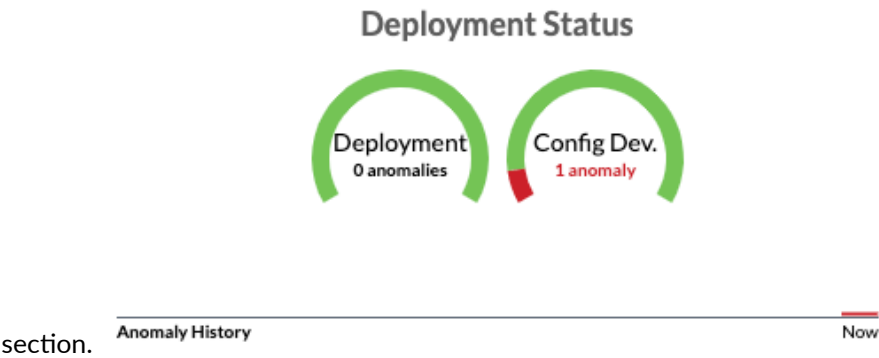
Configuration Deviation

IN THIS SECTION

- Config Deviation and Configlets | 620

Running configurations on devices are continuously compared with the ["Golden Config" on page 61](#). If a config deviation is found, a configuration anomaly is raised. Typically such deviations are seen when changes were made outside of Apstra (from 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 system. 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 row of tabs shows 'Anomalies', 'Config', 'Interface', 'MAC', 'LLDP', 'BGP', 'Route', 'Hostname', 'Counters', 'ARP', 'Transceivers', and 'Utilization'. The 'Config' tab is selected and highlighted with a green box. Below the tabs, there are buttons for 'Apply Full Config' and 'Accept Changes'. A red warning banner states 'Actual config deviated from golden config'. Below the banner, there are 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 45](#).



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.

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.

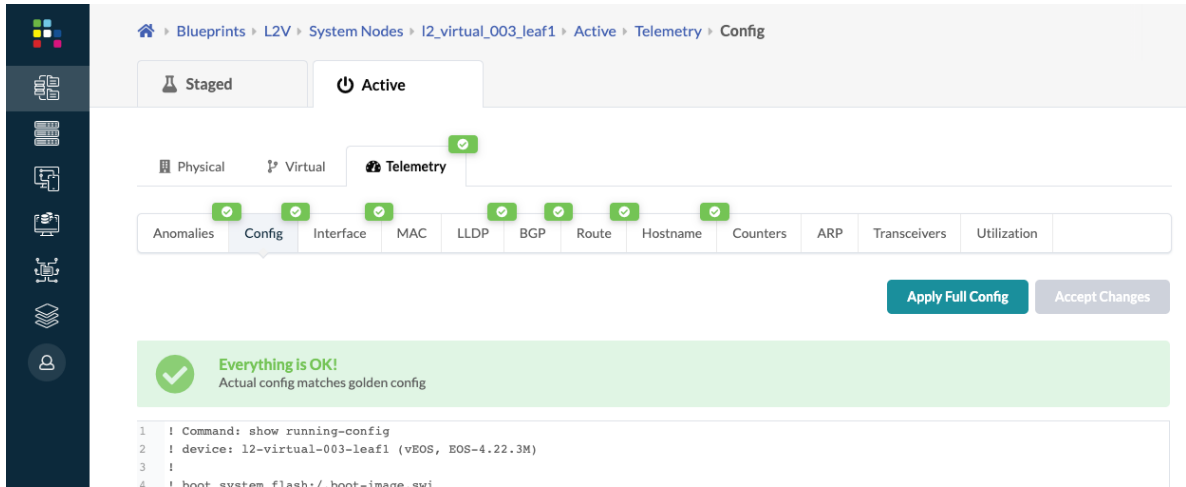


CAUTION: Applying a full config is a disruptive operation and results in a temporary loss of service to the device.



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 824](#).

- After resolving the config deviation anomaly (accept changes or apply 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 device rejects the command), 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 | 621](#)
- [Enable Root Cause Analysis | 621](#)
- [View Root Cause Analysis | 622](#)

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**.

The screenshot shows the Apstra interface with the following elements:

- Top Navigation Bar:** Dashboard, Analytics, Staged, Uncommitted, **Active** (selected), Time Voyager.
- Left Sidebar:** Physical, Virtual, Policies, Catalog, Query, Anomalies, **Root Causes** (selected), Connectivity Templates.
- Main Content Area:** A table with columns: Model Name, State, Top Level Root Causes Count, Trigger Period, Config Updated, Status Updated, Actions. The table is currently empty, showing "No items".
- Buttons:** A blue button labeled "Enable Root Cause Analysis" is located in the top right of the main content area.

Red arrows and numbers indicate the steps to enable Root Cause Analysis:

1. Click **Active** in the top navigation bar.
2. Click **Root Causes** in the left sidebar.
3. Click **Enable Root Cause Analysis** button.

- 2. Enter a **Trigger Period** or leave the default, and click **Create** to enable root cause analysis and return to the table 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 CausesCo

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

BGP Route Tagging

IN THIS SECTION

- [BGP Route Tag Format | 623](#)

Apstra version 4.1.2 introduces a new feature where the following are tagged with BGP communities (RFC1997 - BGP Communities Attribute):

- All routes (IPv4 and IPv6) generated within the data center fabric
- Routes received from external generic systems
- Routes received from remote EVPN gateways

These communities allow you to identify any BGP route within the data center fabric quickly. They'll be used for running more sophisticated route telemetry in future releases.

Introducing this new feature results in new lines of configuration on deployed network devices. These configuration changes won't impact the control or forwarding plane and thus won't be service-impacting.

BGP Route Tag Format

Each route is tagged with two communities (32-bits each) in the following format:

[<system_index>:<function_id>] [<vrf_id>:<peer_id>]

| Field | Description | Possible Range of Values |
|--------------|--|--|
| system_index | Identifies the device where the route is learned (sourced) in Apstra A unique blueprint-wide value is generated for every leaf, spine, and super spine in the data center fabric. | 0 - 19999 <ul style="list-style-type: none">• 0 - don't care• 1 - 19999 usable values // block of 20.000 |

(Continued)

| Field | Description | Possible Range of Values |
|-------------|--|--|
| function_id | <p>Identifies the route source or a function associated with it</p> <p>A unique blueprint-wide value is generated for every leaf, spine, and super spine in the data center fabric.</p> <p>The base for function_id is 20000. The function_id value will be 20000 + function_id. Function_id MUST be set in every tagged BGP update.</p> <p>The following function_id's have been defined:</p> <ul style="list-style-type: none"> • EVPN DCI-GW gateway = 1 • External router generic = 2 • Redistributed from OSPFv2 = 3 • Redistributed from OSPFv3 = 4 • Redistributed from static-v4 = 5 • Redistributed from static-v6 = 6 • Redistributed from connected-v4 = 7 • Redistributed from connected-v6 = 8 • Redistributed from BGP-AFI/SAFI 1/1 = 9 • Redistributed from BGP-AFI/SAFI 2/1 = 10 <p>The following function ids are not supported in Apstra version 4.1.2</p> <ul style="list-style-type: none"> • External router generic = 2 • Redistributed from OSPFv2 = 3 • Redistributed from OSPFv3 = 4 | <p>20000 - 20999</p> <ul style="list-style-type: none"> • 20000 - don't care • 20001 - 20999 usable values // block of 1000 |

(Continued)

| Field | Description | Possible Range of Values |
|---------|---|--|
| vrf_id | <p>Identifies the VRF associated with the route</p> <p>A unique value is generated for every configured VRF in the blueprint. The vrf_id value in the BGP community tag will be 21000 + vrf_id.</p> | <p>21000 - 25999</p> <ul style="list-style-type: none"> • 25000 - don't care • 21001 - 25999 usable values // block of 5000 |
| peer_id | <p>Optional field. Possibly identifying the peer via which the route is learned. This field is not used in Apstra 4.1.2 and is set to a don't care value (26000).</p> <p>The peer_id is not used and is set to the default value of 26000 in Apstra version 4.1.2</p> | <p>26000 - 28999</p> <ul style="list-style-type: none"> • 26000 - don't care • 26001 - 28999 usable values // block of 3000 |

Freeform Reference Design

IN THIS SECTION

- [Freeform Overview | 626](#)
- [Freeform Workflow | 628](#)
- [Create / Delete Freeform Blueprint | 629](#)
- [Freeform Blueprint Summary and Dashboard | 630](#)
- [Topology \(Freeform\) | 631](#)
- [Systems \(Freeform\) | 632](#)
- [Device Context \(Freeform\) | 655](#)
- [Links \(Freeform\) | 657](#)
- [Resource Management | 663](#)
- [Config Templates \(Freeform Blueprint\) | 680](#)

- [Import Device Profile \(Freeform\) | 685](#)
- [Property Sets \(Freeform Blueprints\) | 686](#)
- [Tags \(Freeform Blueprint\) | 688](#)
- [Tasks - Staged \(Freeform\) | 690](#)
- [Active | 691](#)

Freeform Overview

IN THIS SECTION

- [Reference Designs | 626](#)
- [Device Management | 626](#)
- [Freeform Blueprints and Device Profiles | 627](#)
- [Systems and Links | 627](#)
- [Config Templates, Property Sets and Tags | 627](#)

Reference Designs

If your network architecture is comprised of a 3-stage Clos, 5-stage Clos or collapsed fabric, you'll want to take advantage of the abstraction and automation that's included with the **Datacenter** reference design. For all other topologies, you can use the **Freeform** reference design (new in Apstra version 4.1.1) to leverage any feature, protocol, or architecture.

Blueprints created in the Datacenter reference design use a set of design elements to abstract and automate many network activities. Blueprints created in the Freeform reference design consist of *systems* and links that you add and configure yourself, giving you complete control over your architecture. In Freeform we use the term **system** to represent all the types of devices that can be linked in the Apstra environment: switches, routers, Linux hosts and so on.

Device Management

Device management for Freeform blueprints is the same as for Datacenter blueprints. The process of installing agents and acknowledging them to bring them under Apstra management is the same in both

reference designs. In Apstra version 4.1.1 and 4.1.2, only Juniper devices are supported in Freeform blueprints.

Freeform Blueprints and Device Profiles

A newly created Freeform blueprint is just an empty blueprint with a name. (Freeform doesn't use blueprint templates.) You'll start building your blueprint by importing **device profiles** from the design (global) catalog. A device profile represents a device's capabilities without specifying its system ID (serial number). This is what enables you to build your entire network 'offline' before deploying it.

Systems and Links

You'll create **internal systems** and assign device profiles to them. Internal systems are devices that are managed in the Apstra environment. You can bring your devices under Apstra management at any time. If you have them ready, you can assign them as you're creating your internal systems. If they're not ready, that's OK. You can assign them any time before deploying your network.

The other type of system in Freeform blueprints are **external systems**. These are systems that are linked to internal systems, and are not under Apstra management.

When you link your systems, you'll select ports and transformations, as applicable. You can also add IP addresses and *tags* as you're creating those links.

Config Templates, Property Sets and Tags

Config templates are text files used to configure internal systems in Freeform. You'll assign a config template to every internal system. You *could* paste configuration directly from your devices into a config template to create a static config template, but then you wouldn't be using the potential of config templates. With some Jinja2 knowledge (and maybe some Python), you can parametrize config templates to do powerful things.

Property sets provide a valuable capability to fully parameterize config templates. Consisting of key-value pairs, they enable you to separate static portions of config templates from variables. You create property sets in the blueprint catalog. (Property sets used in Freeform blueprints are not related to property sets in the design (global) catalog.) You'll include property set names in your config template and then the values in those property sets will be used when configuration is rendered.

You can also create a property set and assign it directly to one system.

Tags are a way for you to assign metadata to Apstra-managed resources. They can help you identify, organize, search for, and filter Apstra systems and links. With tags, you can categorize resources by purpose, owner, environment, or other criteria. Because tags are metadata, they aren't just used for visual labeling; they are also applied as properties of nodes in the Apstra graph database. This node property (or device property) is then available for you to reference in Jinja config templates for dynamic

variables in config generation and the Apstra real-time analytics via Apstra's Live Query technology and Apstra Intent-Based Analytics.

An example of when you might want to use tags is if you have bare metal servers with SRIOV interfaces, and you need to produce specific configuration for those interfaces. You would add the tag `sriov` to the links, then specify in the config template that links with that tag are to be configured a certain way.

Freeform Workflow

1. Access the ["Apstra GUI" on page 3](#).
2. ["Bring your devices under Apstra management" on page 79](#) (same procedure as for Datacenter blueprints). If you don't have your system IDs (serial numbers) yet, that's OK. You can build your entire network 'offline' in the Apstra environment and bring your devices under Apstra management any time before deploying your network.
3. ["Create Freeform blueprint" on page 629](#).
4. ["Import device profiles" on page 685](#) for the internal systems you'll create.
5. ["Add internal systems" on page 633](#) for the systems that Apstra will manage.
6. ["Add external systems" on page 638](#) for unmanaged systems, as applicable.
7. ["Add links" on page 657](#) to your systems.
8. ["Create config templates" on page 680](#), and ["property sets" on page 686](#) as needed.
9. ["Assign config templates" on page 644](#) to internal systems. In Apstra version 4.1.2, only internal systems with deploy mode set to **Deploy** require an assigned config template. In versions 4.1.0 and 4.1.1, ALL internal systems require an assigned config template regardless of deploy mode.
10. If you haven't brought your ["devices under Apstra management" on page 79](#) yet, it's time to do that now.
11. ["Assign system IDs \(if you haven't already\) and set the deploy mode on your systems to Deploy" on page 649](#).
12. Before deploying your network, you can use the `apstra-cli` utility to validate config template syntax. For more information, see [Juniper Support Knowledge Base article KB69779](#).
13. ["Deploy" on page 692](#) blueprint.

Create / Delete Freeform Blueprint

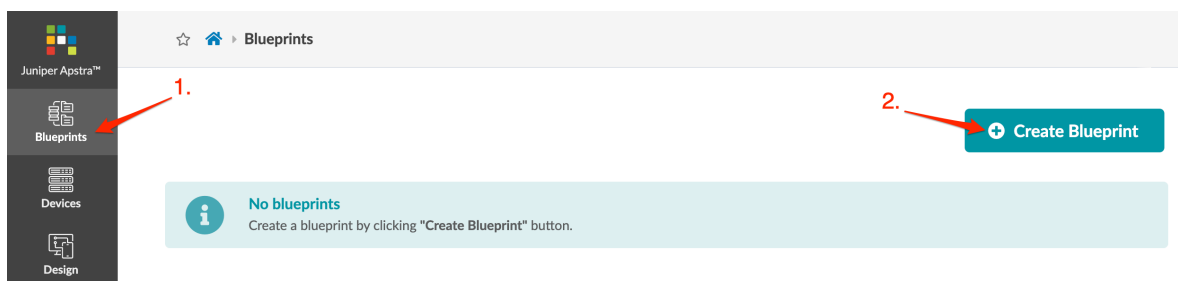
IN THIS SECTION

- [Create Freeform Blueprint | 629](#)
- [Delete Freeform Blueprint | 629](#)

Create Freeform Blueprint

As of Apstra version 4.1.1, you can build any type of architecture in the Apstra environment. You start by creating a "blank" freeform blueprint. For more information, including the Freeform workflow, see ["Freeform Reference Design" on page 625](#).

1. From the left navigation menu of the Apstra GUI, click **Blueprints**, then click **Create Blueprint**.



2. Enter a blueprint name and select **Freeform** reference design.
3. Click **Create** to create the blueprint and return to the blueprint summary view. The newly created blueprint appears in the summary.

Next Steps:

- ["Import device profiles" on page 685](#) into the blueprint catalog.
- You can ["bring your devices under Apstra management" on page 79](#) anytime before deploying your network.

Delete Freeform Blueprint

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.

Freeform Blueprint Summary and Dashboard

IN THIS SECTION

- [Blueprints Summary | 630](#)
- [Blueprint Dashboard | 630](#)

Blueprints Summary

The blueprints summary page shows a summary of all your blueprints. At the top of the page, different status indicators show various statuses across all blueprints (deployment status, anomalies, root causes, build errors and warnings, and uncommitted changes). This is useful to see any issues at a glance when you have many blueprints in your Apstra instance.

From the left navigation menu of the Apstra GUI, click **Blueprints** to go to the blueprints summary page.

Juniper Apstra™

Blueprints

Devices

Design

Resources

External Systems

Platform

☆ Home > Blueprints

Deployment Status

Anomalies

Root Causes

Build Errors

Build Warnings

Uncommitted Changes

+ Create Blueprint

Query: All

1-1 of 1

Table View

Card View

Page Size: 25

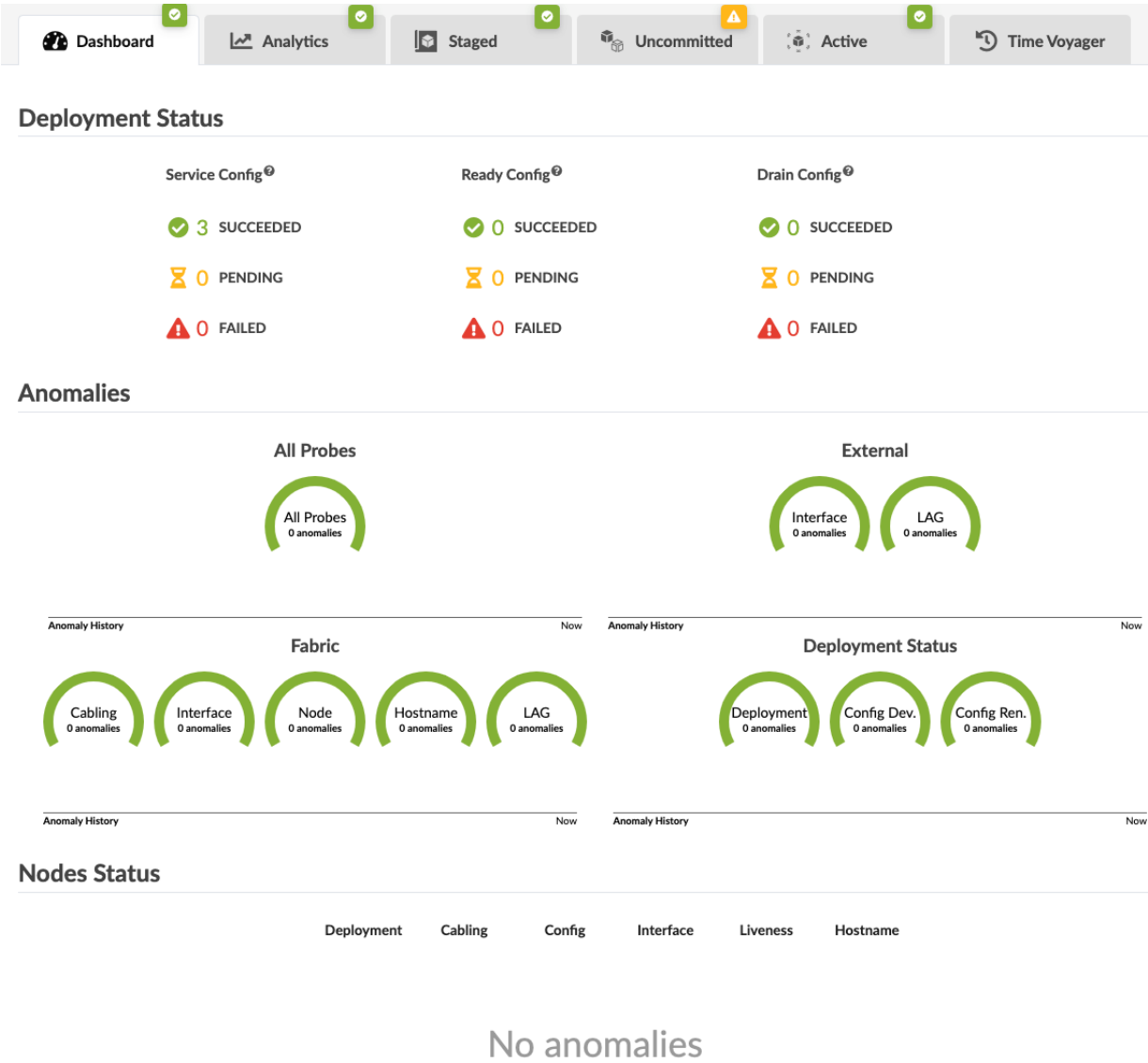
| Name ↕ | Design ↕ | Version ↕ | Structure | Deployment Status | Service Anomalies ↕ | Probe Anomalies ↕ | Root Causes ↕ | Last Modified |
|--|----------|-----------|--|-------------------|---------------------|-------------------|---------------|---------------|
| freeform.crb_virtual_vex5f8a817a | Freeform | 72 | 5 internal systems, 7 external systems | 5 | 0 | 0 | 0 | 9 hours ago |

Click to go to blueprint dashboard

Blueprint Dashboard

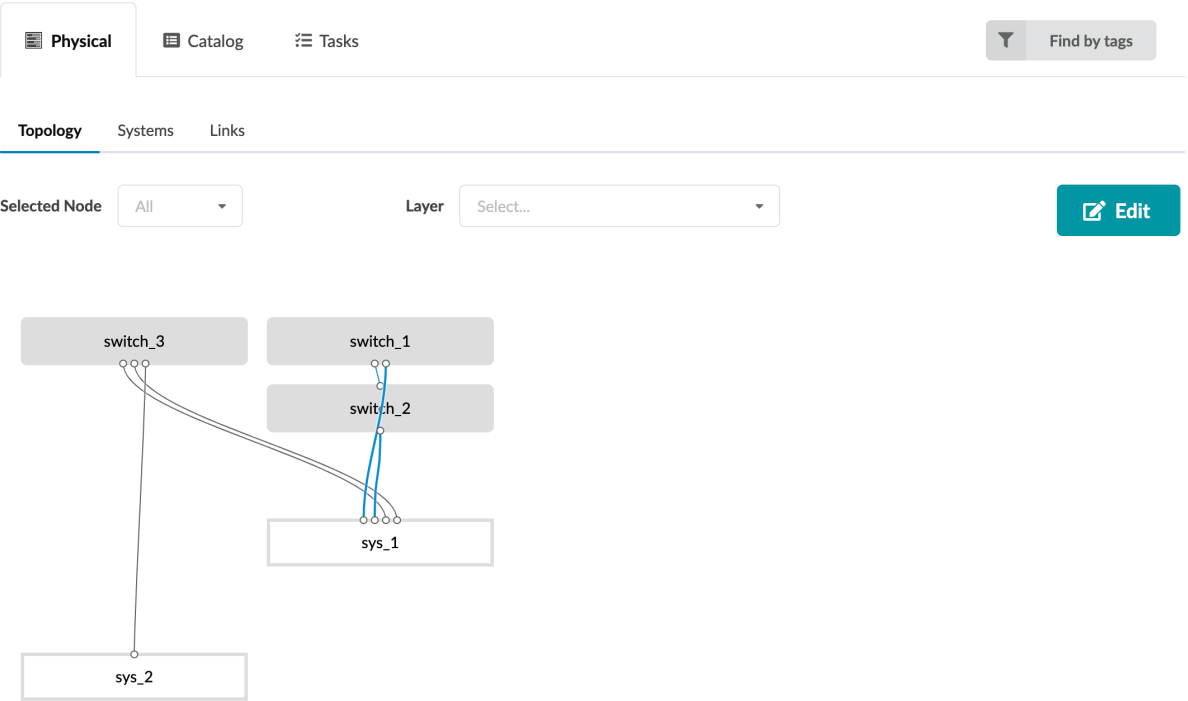
From the left navigation menu of the Apstra GUI, click **Blueprints**, then click the name of the blueprint that you want to go to. The blueprint dashboard is the default view, which shows the blueprint's overall

health and status.



Topology (Freeform)

The **Topology** view (Staged > Physical > Topology) shows in a graphical way the collection of devices/ objects that make up the network and the links that connect devices. It self-documents your intended network state. This is then modeled/created in the Apstra GraphDB for intent based modeling. You can perform various tasks from the **Topology** view, via the topology editor, as described in later sections.



Systems (Freeform)

IN THIS SECTION

- [Create Internal System \(Freeform\) | 633](#)
- [Create External System \(Freeform\) | 638](#)
- [Edit System Properties \(Freeform\) | 642](#)
- [Delete System \(Freeform\) | 643](#)
- [Assign Config Template | 644](#)
- [Remove Config Template Assignment | 645](#)
- [Set Deploy Mode \(Freeform\) | 646](#)
- [Assign System \(Freeform\) | 649](#)
- [Unassign System \(Freeform\) | 651](#)

The **Systems** view (Staged > Physical > Systems) shows in a table format the collection of devices/objects that make up the network (similar to the Nodes view in Datacenter reference designs). The table includes information about internal and external systems in the blueprint, tags, deploy mode, assigned device profile, assigned system ID, hostname, operation mode (full control), assigned config template, and assigned property set. You can see details at a glance and tell if there are any issues with missing requirements. You can customize what appears in the table by selecting/deselecting elements in the columns drop-down list. You can perform various tasks from the **Systems** view as described in later sections.

[Topology](#)
[Systems](#)
[Links](#)

[+ Create System](#)

Query: All
 1-5 of 5

Columns (11/11)
 Page Size: 25

Filter selected by
 ☒ all
 ☐ selected only
 ☐ unselected only

| <input type="checkbox"/> | Name | Type | Tags | Deploy Mode | Device Profile | S/N | Hostname | Operation Mode | Config Template | Property Set | Actions |
|--------------------------|----------|----------|------|-------------|----------------|--------------|----------|----------------|---------------------------|--------------|---------|
| <input type="checkbox"/> | switch_1 | INTERNAL | red | Deploy | Juniper vQFX | 525400519CE8 | switch1 | FULL CONTROL | junos_configuration.jinja | test | |
| <input type="checkbox"/> | switch_2 | INTERNAL | red | Deploy | Juniper vQFX | 525400A19B67 | switch2 | FULL CONTROL | junos_configuration.jinja | Not assigned | |
| <input type="checkbox"/> | switch_3 | INTERNAL | red | Deploy | Juniper vQFX | 5254006252C1 | switch3 | FULL CONTROL | junos_configuration.jinja | Not assigned | |
| <input type="checkbox"/> | sys_1 | EXTERNAL | blue | Deploy | N/A | N/A | | UNMANAGED | N/A | N/A | |

Create Internal System (Freeform)

IN THIS SECTION

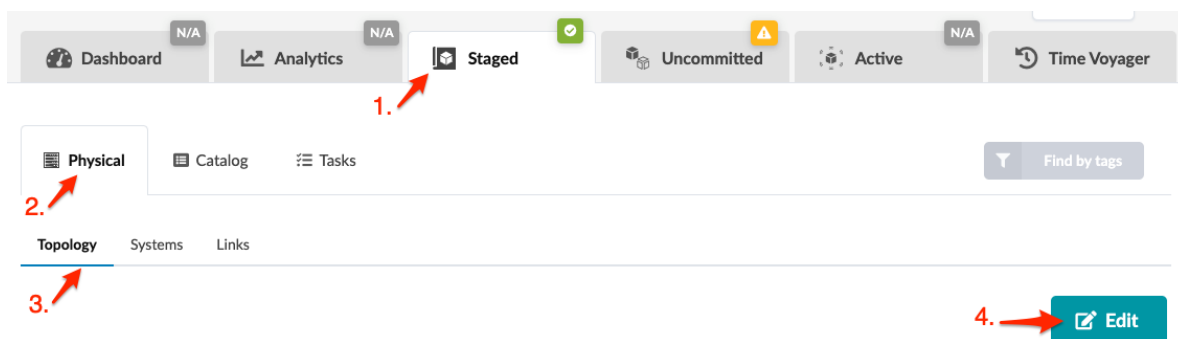
- Create Internal System (from Topology Editor) | [634](#)
- Create Internal System (from Systems View) | [636](#)
- Clone Internal System (from Topology Editor) | [637](#)
- Clone Internal System (from Systems View) | [637](#)

Systems represent switches, routers, Linux hosts and so on. Managed devices that you add to a blueprint are called *internal systems*. You can create systems from scratch, or you can clone systems and customize them to create new ones. You can create (and clone) from the **Topology** view or from the **Systems** view. (Cloning from the **Systems** view is new in Apstra version 4.1.2).

Internal systems must be mapped to device profiles, so before creating systems, ["import" on page 685](#) the relevant device profiles to the blueprint catalog.

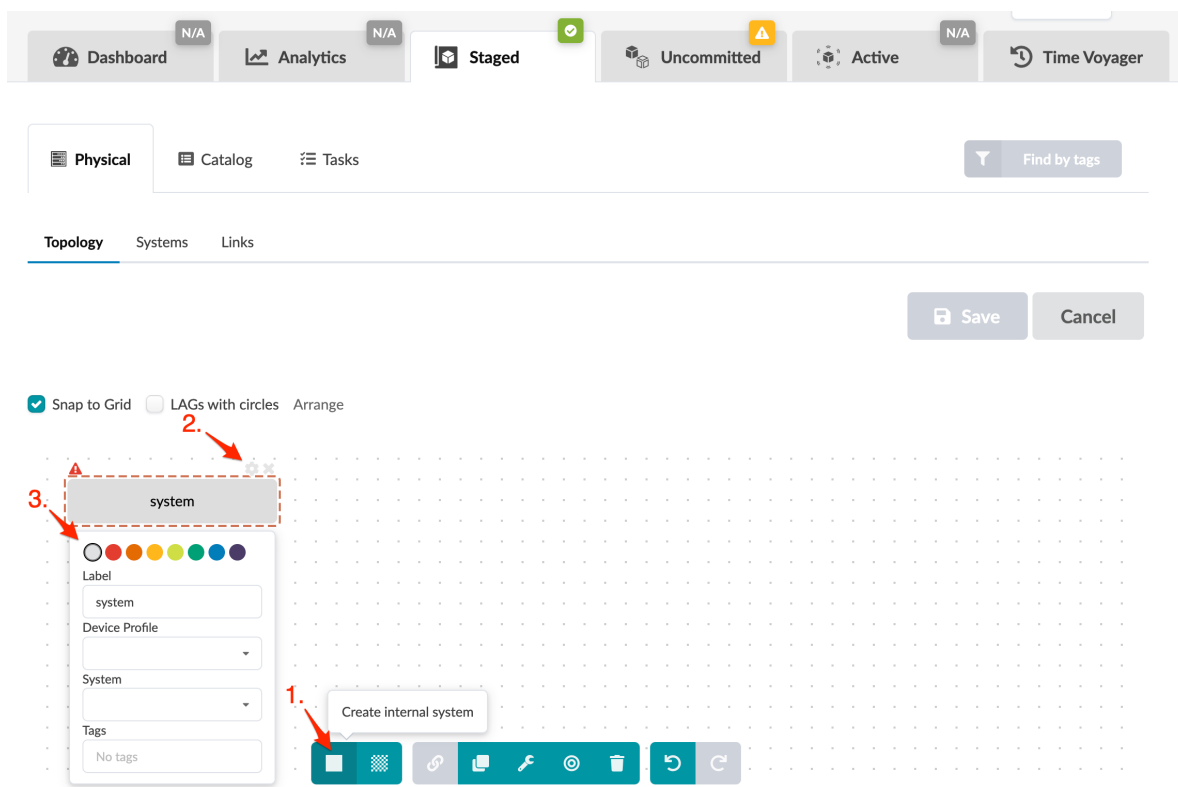
Create Internal System (from Topology Editor)

1. From the blueprint, navigate to **Staged > Physical > Topology** and click **Edit**. (Images in this section are from Apstra version 4.1.1.)



CAUTION: Be careful. If you click away from the topology editor without clicking **Save**, your changes are discarded.

2. In the topology editor, click the **Create internal system** button (bottom-left). The system appears as a gray rectangle with a system-generated name. The red triangle indicates that information is needed for required fields. In this case, it's the device profile. You can move systems around on the canvas and when you save your changes in the editor and then reopen it, your systems will still be where you moved them to.



3. Click the gear to open the parameters dialog.
4. You can change the system color that displays in the topology. This is useful for designating different roles or anything else you'd like to visually differentiate.
5. You can change the system label to customize it for your environment.
6. Select a device profile from the drop-down list. (Device profiles come from the blueprint catalog. If you don't see the one you need, import it into the blueprint catalog.)
7. You can assign the system ID now or "later" on page 649. To assign it now, select it from the **System** drop-down list. (The list includes managed devices that haven't been assigned yet. If you have your devices ready and they're not appearing in this list, you still need to ["bring them under Apstra management"](#) on page 79.)
8. You can add tags, then later when you want to find systems you can use the **Find by Tags** feature (upper-right) to find them. You can also include tags in config templates, then systems with those tags will be rendered as specified in the config template.
9. Click **Save** to stage your new system and return to the **Topology** view. (If you leave the page without saving, your changes are discarded.)

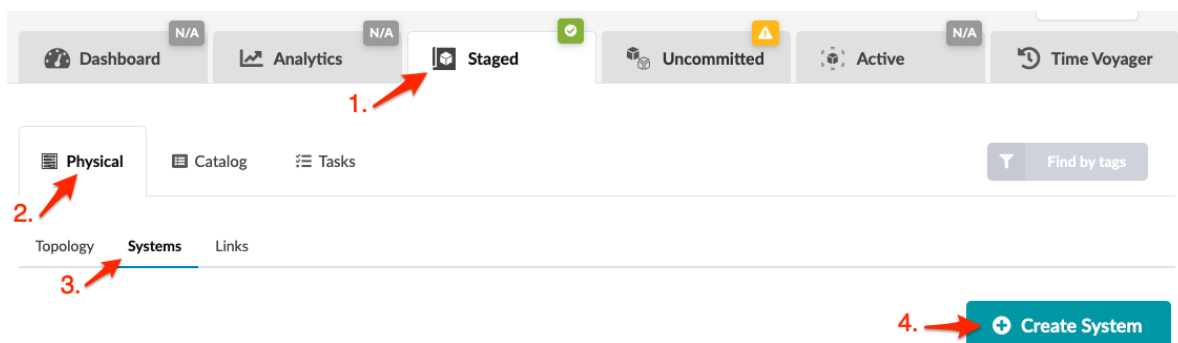
Next Steps:

- Continue to create internal and ["external systems"](#) on page 638 until you've added your devices to the topology.
- ["Add links"](#) on page 657 to systems.

- ["Assign config templates" on page 644](#) to your internal systems. In Apstra version 4.1.2, only internal systems with deploy mode set to **Deploy** require an assigned config template. In versions 4.1.0 and 4.1.1, ALL internal systems require an assigned config template regardless of deploy mode.

Create Internal System (from Systems View)

1. From the blueprint, navigate to **Staged > Physical > Systems** and click **Create System**. (The image below is from Apstra version 4.1.1.)



2. In the **Create System** dialog, enter a name and select **INTERNAL**.
3. Internal systems are associated with device profiles. You can either assign just the device profile now (and ["assign the system ID" on page 649](#) later), or if you've brought your devices under Apstra management, you can select the system ID now.
 - **From Scratch** - select a device profile (that was imported into the blueprint catalog.) (You'll assign the system ID later.)
 - **From Managed Devices** - select a managed device to assign its system ID to the system.
4. Enter a hostname (optional).
5. You can add tags, then later when you want to find systems you can use the **Find by Tags** feature (upper-right) to find them. You can also include tags in config templates, then systems with those tags will be rendered as specified in the config template.
6. Click **Create** to stage your new system and return to the **Systems** view. The newly created system appears in the list.

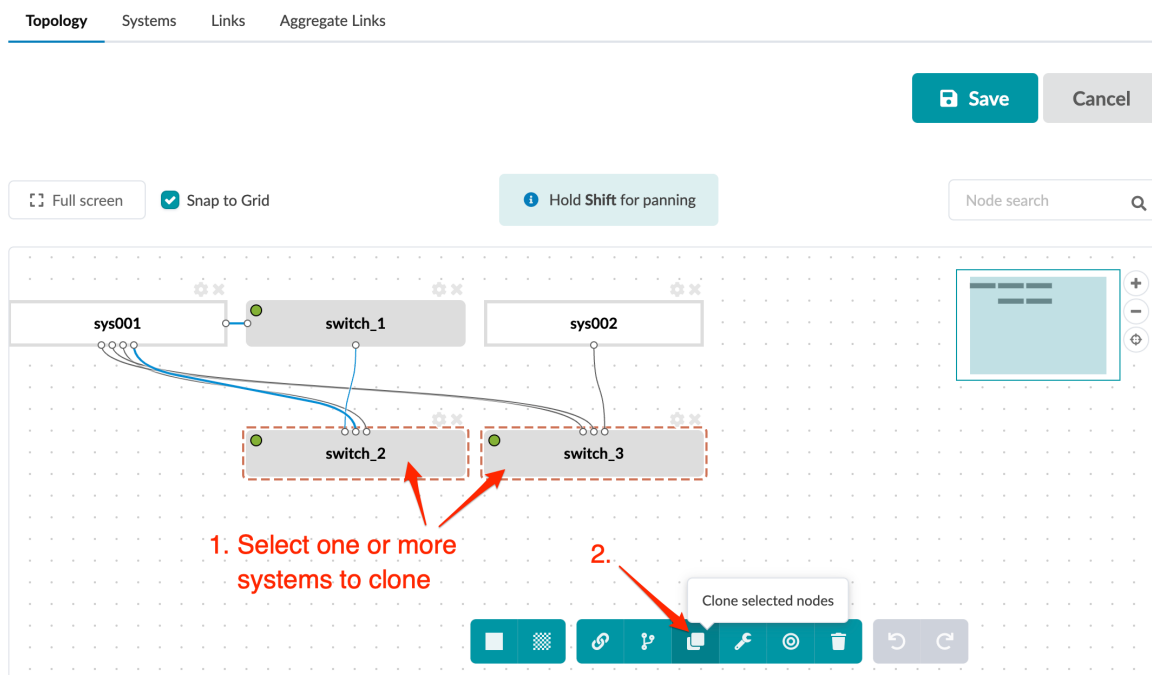
Next Steps:

- Continue to create internal and ["external systems" on page 638](#) until you've added your devices to the topology.
- ["Add links" on page 657](#) to systems.
- ["Assign config templates" on page 644](#) to your internal systems. In Apstra version 4.1.2, only internal systems with deploy mode set to **Deploy** require an assigned config template. In versions 4.1.0 and 4.1.1, ALL internal systems require an assigned config template regardless of deploy mode.

Clone Internal System (from Topology Editor)

You can clone systems and customize them to create new ones from the **Topology** view.

1. From the blueprint, navigate to **Staged > Physical > Topology** and click **Edit**.
2. In the topology editor, select one or more existing internal systems, then click the **Clone selected nodes** button. (The image below is from Apstra version 4.1.2.)



3. The new system(s) appear as gray rectangles with system-generated names. You can move systems around on the canvas and when you save your changes in the editor and then reopen it, your systems will still be where you moved them to.
4. Click the gear to open the parameters dialog, and change details to customize your new system.
5. Click **Save** to stage your new system(s) and return to the **Topology** view. (If you leave the page without saving, your changes are discarded.)

Clone Internal System (from Systems View)

You can clone systems and customize them to create new ones from the **Systems** view, as of Apstra version 4.1.2.

1. From the blueprint, navigate to **Staged > Physical > Systems** and click **Clone System** for the system you want to clone.

Dashboard Analytics Staged Uncommitted Active Time Voyager

Physical Resource Management Catalog Tasks Find by tags

Topology Systems Links Aggregate Links

Create System

Query: All 1-5 of 5 Columns (11/11) Page Size: 25

Filter selected by ☒ all ☐ selected only ☐ unselected only

| | Name | Type | Tags | Deploy Mode | Device Profile | S/N | Hostname | Operation Mode | Config Template | Property Set | Actions |
|--------------------------|----------|----------|---------------|-------------|----------------|--------------|----------|----------------|---------------------------|--------------|---------|
| <input type="checkbox"/> | switch_1 | INTERNAL | Internal red | Deploy | Juniper vQFX | 52540081C441 | switch1 | FULL CONTROL | junos_configuration.jinja | Not assigned | |
| <input type="checkbox"/> | switch_2 | INTERNAL | blue internal | Deploy | Juniper vQFX | 5254001ABE9F | switch2 | FULL CONTROL | junos_configuration.jinja | Not assigned | |

2. Change details to customize your new system.
3. Click **Clone** to stage your new system and return to the **Systems** view.

Create External System (Freeform)

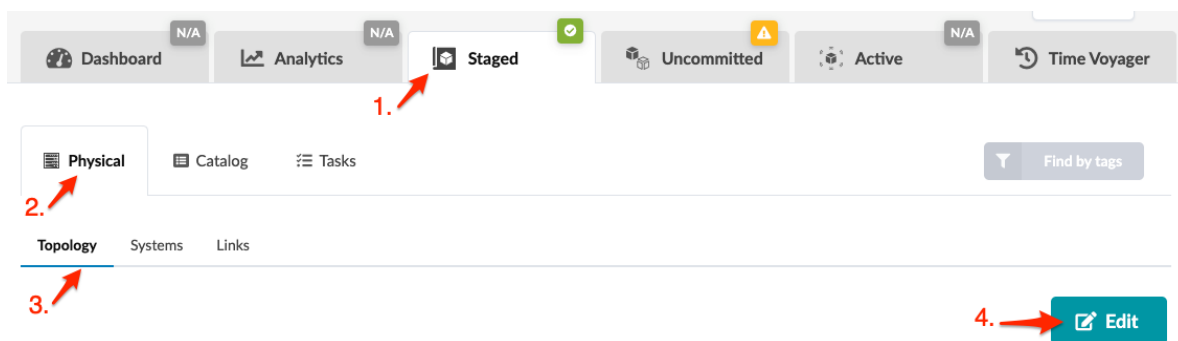
IN THIS SECTION

- Create External System (from Topology Editor) | 638
- Create External System (from Systems View) | 640
- Clone External System (from Topology Editor) | 640
- Clone External System (from Systems View) | 641

Systems represent switches, routers, Linux hosts and so on. Unmanaged devices that you add to a blueprint are called *external systems*. They link to managed (internal) systems. You can create systems from scratch, or you can clone systems and customize them to create new ones. You can create (and clone) from the **Topology** view or from the **Systems** view. (Cloning from the **Systems** view is new in Apstra version 4.1.2).

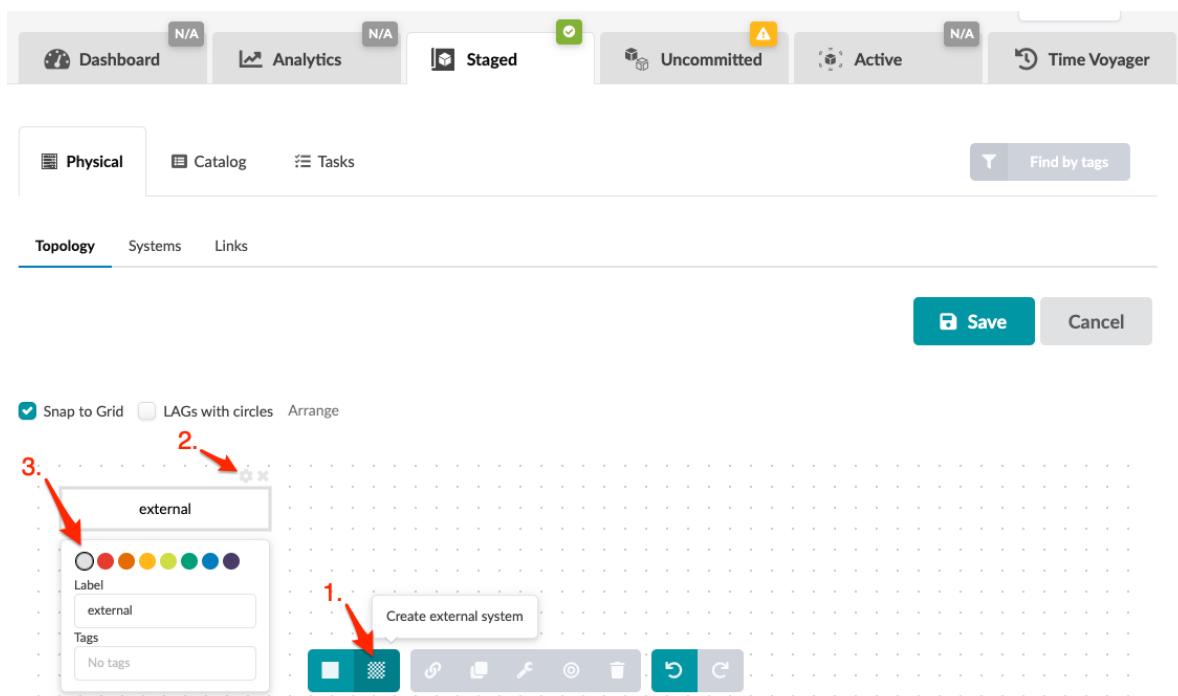
Create External System (from Topology Editor)

1. From the blueprint, navigate to **Staged > Physical > Topology** and click **Edit**. (Images in this section are from Apstra version 4.1.1.)



CAUTION: Be careful. If you click away from the topology editor without clicking **Save**, your changes are discarded.

2. In the topology editor click the **Create external system** button. The system appears as a rectangle with a system-generated name. You can move systems around on the canvas and when you save your changes in the editor and then reopen it, your systems will still be where you moved them to. You can save the system as is since there are no other required fields, or you can open the parameters dialog and configure optional fields.



3. Click the gear to open the parameters dialog.
4. You can change the system color that displays in the topology. This is useful for designating different roles or anything else you'd like to visually differentiate.
5. You can change the system label to customize it to your environment.

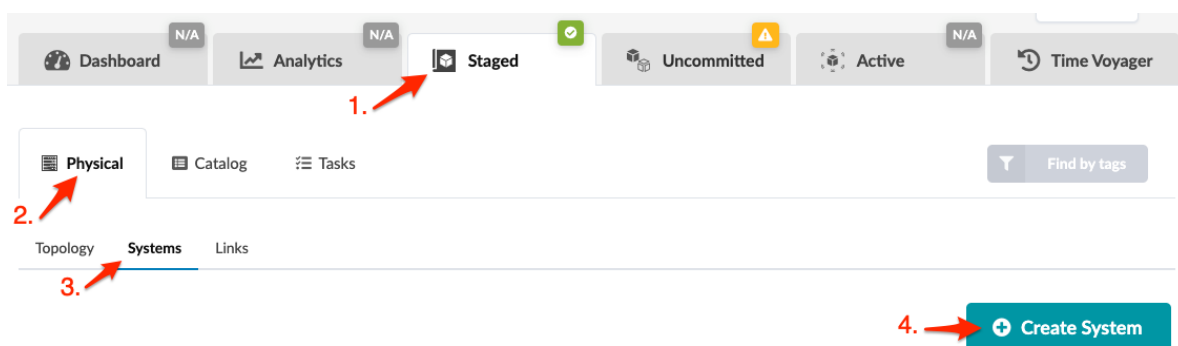
6. You can add tags, then later when you want to find systems you can use the **Find by Tags** feature (upper-right) to find them.
7. Click **Save** to stage your new system and return to the **Topology** view. (If you leave the page without saving, your changes are discarded.)

Next Steps:

Continue to create external systems and ["internal systems" on page 633](#) until you've added your devices to the topology. Then you can ["create links" on page 657](#) for them.

Create External System (from Systems View)

1. From the blueprint, navigate to **Staged > Physical > Systems** and click **Create System**. (The image below is from Apstra version 4.1.1.)



2. Enter a name and select **EXTERNAL**.
3. Enter a hostname (optional) and tags (optional). If you add tags, then later when you want to find systems you can use the **Find by Tags** feature (upper-right) to find them.
4. Click **Create** to stage the change and return to the **Systems** view. The newly created system appears in the list.

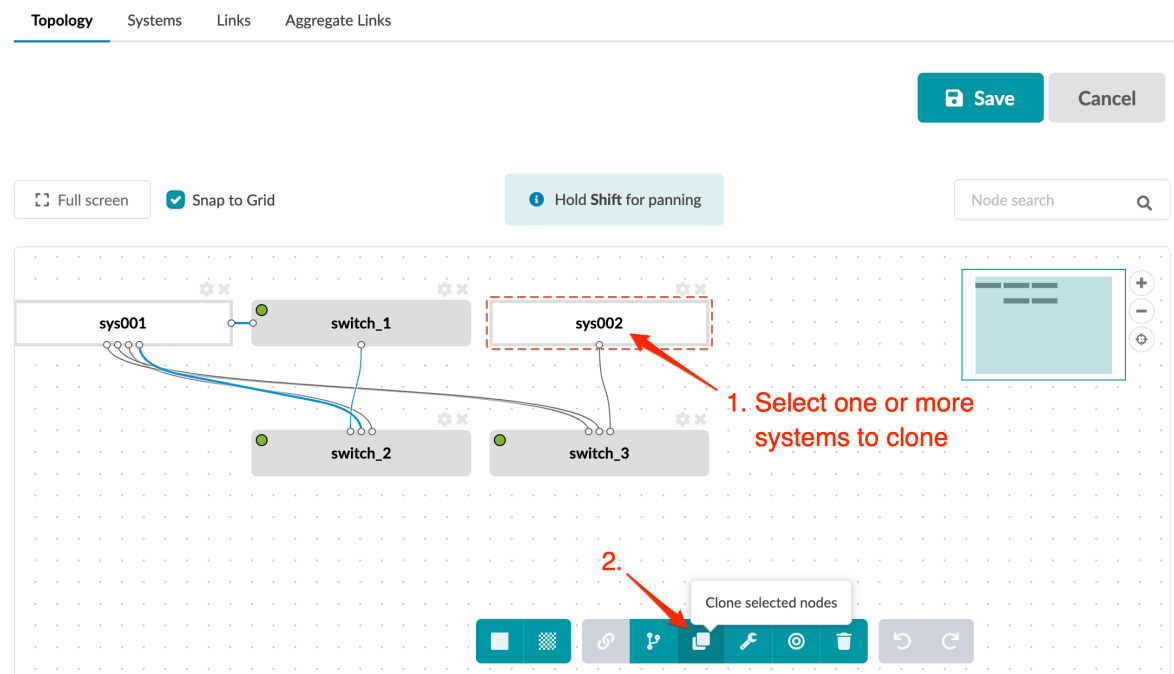
Next Steps:

Continue to create external systems and ["internal systems" on page 633](#) until you've added your devices. Then you can ["create links" on page 657](#) for them.

Clone External System (from Topology Editor)

You can clone systems and customize them to create new ones from the **Topology** view.

1. From the blueprint, navigate to **Staged > Physical > Topology** and click **Edit**.
2. In the topology editor, select one or more existing external systems, then click the **Clone selected nodes** button. (The image below is from Apstra version 4.1.2.)



3. The new system(s) appear as gray rectangles with system-generated names. You can move systems around on the canvas and when you save your changes in the editor and then reopen it, your systems will still be where you moved them to.
4. Click the gear to open the parameters dialog, and change details to customize your new system.
5. Click **Save** to stage your new system(s) and return to the **Topology** view. (If you leave the page without saving, your changes are discarded.)

Clone External System (from Systems View)

You can clone systems and customize them to create new ones from the **Systems** view, as of Apstra version 4.1.2.

1. From the blueprint, navigate to **Staged > Physical > Systems** and click **Clone System** for the system you want to clone.

Dashboard Analytics Staged Uncommitted Active Time Voyager

Physical Resource Management Catalog Tasks

Topology Systems Links Aggregate Links

Create System

Query: All

1-5 of 5

Columns (11/11) Page Size: 25

Filter selected by ☒ all ☐ selected only ☐ unselected only

| | Name | Type | Tags | Deploy Mode | Device Profile | S/N | Hostname | Operation Mode | Config Template | Property Set | Actions |
|--------------------------|--------|----------|----------|-------------|----------------|-----|----------|----------------|-----------------|--------------|---------|
| <input type="checkbox"/> | sys001 | EXTERNAL | external | Undeploy | N/A | N/A | sys001 | UNMANAGED | N/A | N/A | |
| <input type="checkbox"/> | sys002 | EXTERNAL | external | Undeploy | N/A | N/A | sys002 | UNMANAGED | N/A | N/A | |

2. Change details to customize your new system.
3. Click **Clone** to stage your new system and return to the **Systems** view.

Edit System Properties (Freeform)

1. From the blueprint, navigate to **Staged > Physical > Topology** and click **Edit**.



CAUTION: Be careful. If you click away from the topology editor without clicking **Save**, your changes are discarded.

2. In the topology editor, click the system to change, then click the **Manage selected nodes properties** button.

Dashboard Analytics Staged Uncommitted Active Time Voyager

Physical Catalog Tasks

Topology Systems Links

☒ Snap to Grid ☐ LAGs with circles Arrange

1.

2.

Label: switch_3

Device Profile: Juniper vQFX

System: 5254006252C1 (10.28.60.9) - some_location

Tags: red

Manage selected nodes properties

3. Change the label, device profile, system ID and/or tags, as applicable.
4. To close the dialog, click the **Manage selected nodes properties** button again.

- Click **Save** to stage your changes, exit the topology editor and return to the **Topology** view. (If you leave the page without saving, your changes are discarded.)

Next Steps:

When you're ready to activate your changes, commit them from the **Uncommitted** tab.

Delete System (Freeform)

IN THIS SECTION

- Delete System (from Topology Editor) | 643
- Delete System (from Systems View) | 644

You can delete systems from the **Topology** view or the **Systems** view.

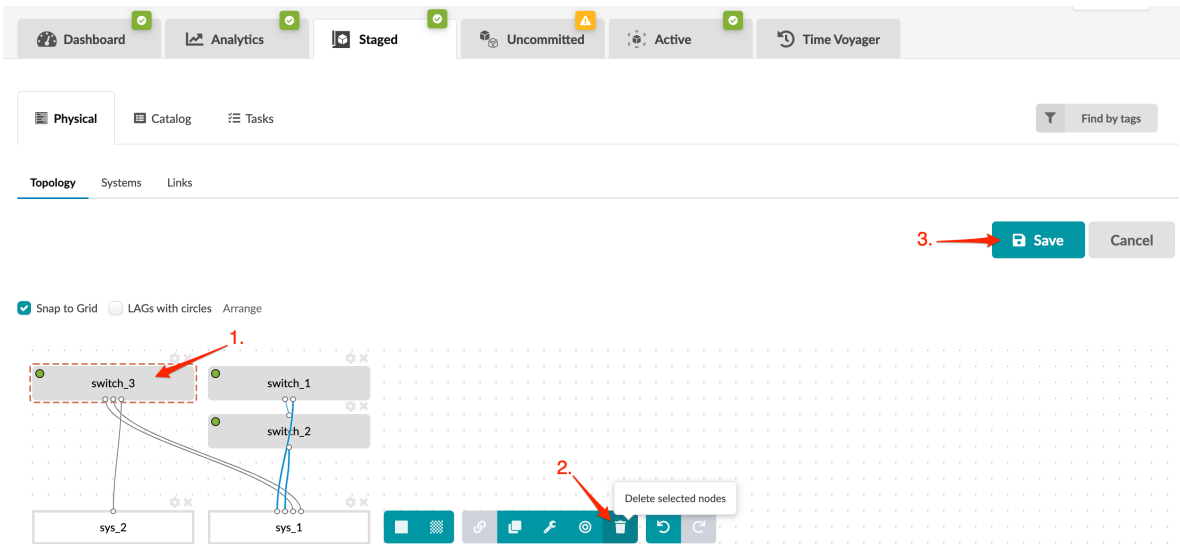
Delete System (from Topology Editor)

- From the blueprint, navigate to **Staged > Physical > Topology** and click **Edit** (right-side).



CAUTION: Be careful. If you click away from the topology editor without clicking **Save**, your changes are discarded.

- In the topology editor select the system to delete and click the **Delete selected nodes** button.



- Click **Save** to stage your changes, exit the topology editor and return to the **Topology** view.

When you're ready to activate your changes, commit them from the **Uncommitted** tab.

Delete System (from Systems View)

1. From the blueprint, navigate to **Staged > Physical > Systems** and click the **Delete** button for the system to delete.

The screenshot shows the 'Systems' view in the Apstra interface. At the top, there are tabs for 'Topology', 'Systems' (which is active), and 'Links'. A 'Create System' button is in the top right. Below the tabs is a search bar with 'Query: All' and pagination controls showing '1-5 of 5'. There are also controls for 'Columns (11/11)' and 'Page Size: 25'. Below these is a filter section with radio buttons for 'all' (selected), 'selected only', and 'unselected only'. The main part of the interface is a table with the following columns: Name, Type, Tags, Deploy Mode, Device Profile, S/N, Hostname, Operation Mode, Config Template, Property Set, and a 'Delete' button. The table contains one row for a system named 'switch_1' with Type 'INTERNAL', Tags 'red', Deploy Mode 'Deploy', Device Profile 'Juniper vQFX', S/N '525400519CE8', Hostname 'switch1', Operation Mode 'FULL CONTROL', Config Template 'junos_configuration.jinja', and Property Set 'Not assigned'. A red arrow points to the 'Delete' button in the last column of the table.

| | Name | Type | Tags | Deploy Mode | Device Profile | S/N | Hostname | Operation Mode | Config Template | Property Set | Delete |
|------------|----------|----------|------|-------------|----------------|--------------|----------|----------------|---------------------------|--------------|--------|
| 0 selected | switch_1 | INTERNAL | red | Deploy | Juniper vQFX | 525400519CE8 | switch1 | FULL CONTROL | junos_configuration.jinja | Not assigned | |

2. Click **Delete** to stage the deletion and return to the **Systems** view.

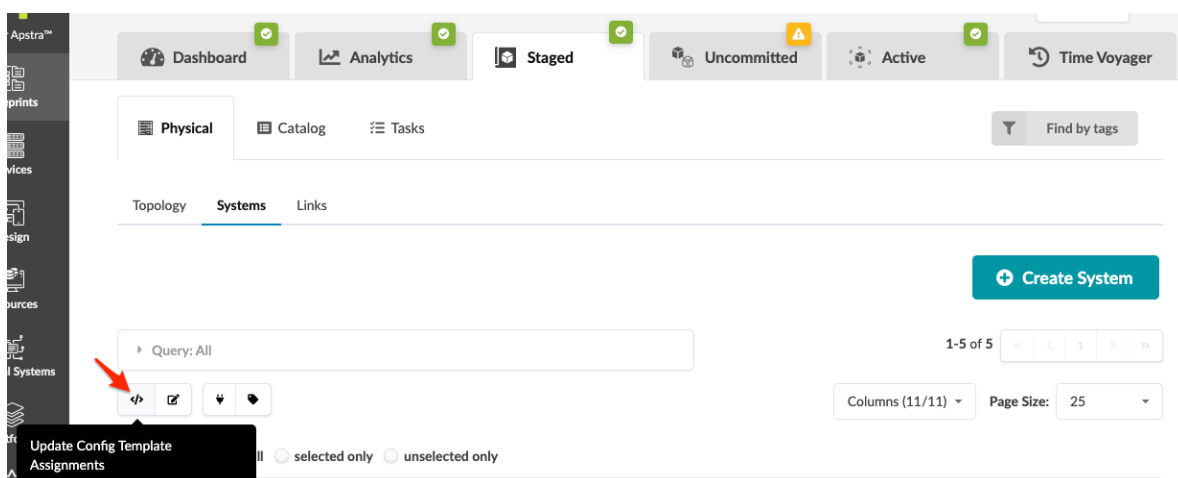
When you're ready to activate your changes, commit them from the **Uncommitted** tab.

Assign Config Template

In Apstra version 4.1.2, only internal systems with deploy mode set to **Deploy** require an assigned config template. In versions 4.1.0 and 4.1.1, ALL internal systems require an assigned config template regardless of deploy mode.

If you haven't created your ["config templates"](#) on [page 680](#) yet, do that now.

1. From the blueprint, navigate to **Staged > Physical > Systems** to go to the **Systems** view.
2. Select the check boxes for the system(s) where you want to add a config template.
3. Click the **Update Config Template Assignments** button that appears above the list after selecting system(s).



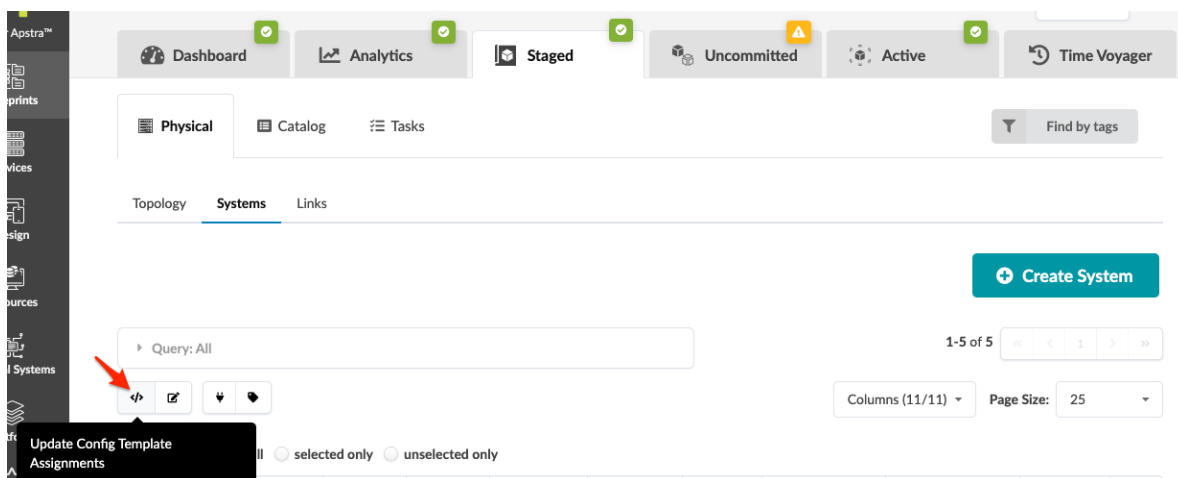
4. In the **Update Config Template Assignments** dialog, leave the default **Override Assignment** selected to add or replace a template.
5. Select a config template from the drop-down list. You can preview the template text. Each internal system is assigned only one config template (but that config template could nest other config templates within it.)
6. Click **Assign Config Template** to stage the changes and return to the **Systems** view.

Next Steps:

When you've assigned all required config templates and all other requirements are met, you can deploy your blueprint from the **Uncommitted** tab.

Remove Config Template Assignment

1. From the blueprint, navigate to **Staged > Physical > Systems** to go to the **Systems** view.
2. Select the check boxes for the systems where you want to remove config templates.
3. Click the **Update Config Template Assignments** button that appears above the list after selecting system(s).



4. In the **Update Config Template Assignments** dialog, select **Remove Assignment**.
5. Click **Remove Config Template Assignments** to stage the changes and return to the **Systems** view.

When you're ready to activate your changes, commit them from the **Uncommitted** tab.

Set Deploy Mode (Freeform)

IN THIS SECTION

- [Set Deploy Mode on One System | 646](#)
- [Set Deploy Modes on Multiple Systems | 647](#)



NOTE: When you set the deploy mode on a system, it appears in its **Device Context**. But if you haven't added `deploy_mode` (as a Jinja variable) to the config template that's assigned to that system, it has no effect on the rendered configuration.

You can set the deploy mode for one system at a time, or for multiple systems at the same time in one dialog.

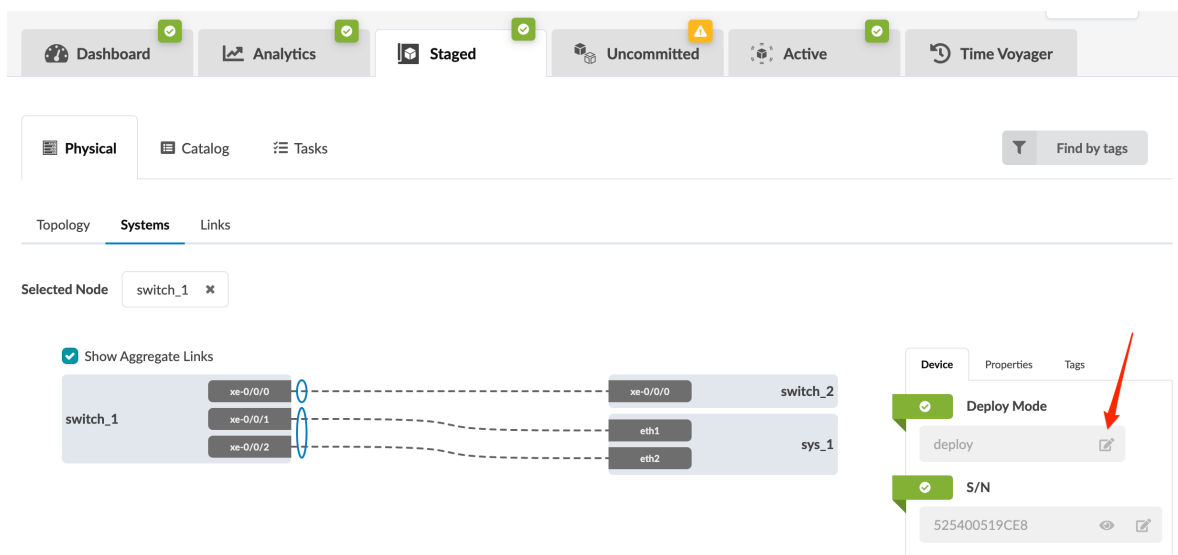
Set Deploy Mode on One System

1. From the blueprint, navigate to **Staged > Physical > Systems** and click the name of the system that needs its deploy mode set. The details page appears.



NOTE: You can also get to the details page from the **Topology** view. From the blueprint, navigate to **Staged > Physical > Topology** and select the system that needs its deploy mode set.

2. Click the **Edit** button for the **Deploy Mode** field.



3. Select the deploy mode (deploy, ready, drain, undeploy), then click the **Save** button to stage your changes.



NOTE: You can also assign the system ID in the same device panel.

In Apstra version 4.1.2, internal systems with deploy mode set to **Deploy** require an assigned config template. In versions 4.1.0 and 4.1.1, ALL internal systems require an assigned config template regardless of deploy mode. Make sure the config template assigned to the device includes `deploy_mode` or your changes will have no effect on configuration.

Set Deploy Modes on Multiple Systems

1. From the blueprint, navigate to **Staged > Physical > Systems** and select the check boxes for one or more systems, then click the **Set Deploy Mode** button.

Dashboard Analytics Staged Uncommitted Active Time Voyager

Physical Catalog Tasks Find by tags

Topology **Systems** Links

Create System

Query: All 1-12 of 12 Columns (11/11) Page Size: 25

Set Deploy Mode selected only unselected only

| | Name | Type | Tags | Deploy Mode | Device Profile | S/N | Hostname | Operation Mode | Config Template | Property Set | Actions |
|------------|--------|----------|---------------|-------------|----------------|--------------|----------|----------------|-----------------|--------------|---------|
| 3 selected | spine2 | INTERNAL | fabric, spine | Deploy | Juniper vEX | 5254007A706E | spine2 | FULL CONTROL | crb_root.jinja | Not assigned | |
| | spine1 | INTERNAL | fabric, spine | Deploy | Juniper vEX | 5254004F349B | spine1 | FULL CONTROL | crb_root.jinja | Not assigned | |
| | leaf1 | INTERNAL | fabric, leaf | Deploy | Juniper vEX | 5254008FBA82 | leaf1 | FULL CONTROL | crb_root.jinja | Not assigned | |
| | leaf2 | INTERNAL | fabric, leaf | Deploy | Juniper vEX | 5254004D4CAD | leaf2 | FULL CONTROL | crb_root.jinja | Not assigned | |



NOTE: You can also set deploy modes (and system IDs) by clicking the **Change System IDs assignments** button (to the left of the **Set Deploy Mode** button).

2. In the dialog, select the deploy mode (deploy, ready, drain, undeploy) for the selected systems.



NOTE: You can also assign system IDs from the same dialog.

3. Click **Set Deploy Mode** to stage the changes and return to the **Systems** view.

In Apstra version 4.1.2, internal systems with deploy mode set to **Deploy** require an assigned config template. In versions 4.1.0 and 4.1.1, ALL internal systems require an assigned config template regardless of deploy mode. Make sure the config template assigned to the device includes `deploy_mode` or your changes will have no effect on configuration.

Assign System (Freeform)

IN THIS SECTION

- [Assign System\(s\) \(from Systems View\) | 649](#)
- [Assign System \(from Topology Editor\) | 650](#)
- [Assign System \(from Device Panel\) | 650](#)

You can assign system IDs (serial numbers) to systems from the **Systems** view, the **Topology** view or from the **Device** panel.

Assign System(s) (from Systems View)

1. From the blueprint, navigate to **Staged > Physical > Systems** and select the check boxes for one or more systems, then click the **Change System IDs assignments** button.

| Name | Type | Tags | Deploy Mode | Device Profile | S/N | Hostname | Operation Mode | Config Template | Property Set | Actions |
|--------|----------|---------------|-------------|----------------|--------------|----------|----------------|-----------------|--------------|---------|
| spine2 | INTERNAL | fabric, spine | Deploy | Juniper vEX | 5254007A706E | spine2 | FULL CONTROL | crb_root.jinja | Not assigned | |
| spine1 | INTERNAL | fabric, spine | Deploy | Juniper vEX | 5254004F349B | spine1 | FULL CONTROL | crb_root.jinja | Not assigned | |
| leaf3 | INTERNAL | fabric, leaf | Deploy | Juniper vEX | 5254008FBA82 | leaf3 | FULL CONTROL | crb_root.jinja | Not assigned | |
| leaf2 | INTERNAL | fabric, leaf | Deploy | Juniper vEX | 5254004D4CAD | leaf2 | FULL CONTROL | crb_root.jinja | Not assigned | |

2. In the dialog, select system IDs from the drop-down lists (available managed devices). (If you don't see your system, you may still need to ["acknowledge" on page 79](#) it.)



NOTE: You can also set the deploy modes from the same dialog.

3. Click **Update Assignments** to stage your changes and return to the **Systems** view.

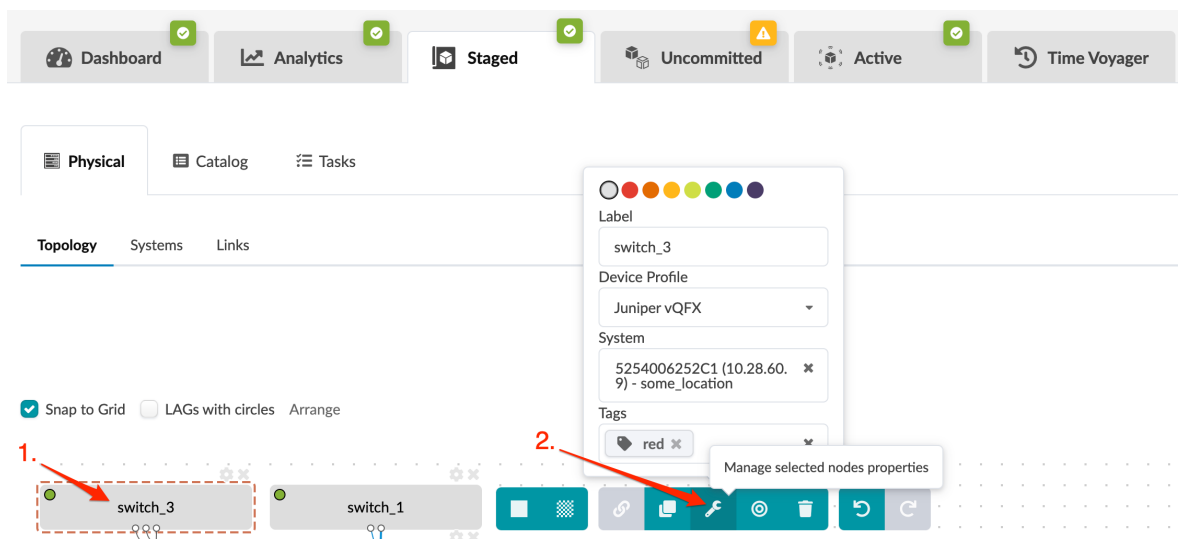
Assign System (from Topology Editor)

1. From the blueprint, navigate to **Staged > Physical > Topology** and click **Edit** to open the topology editor.



CAUTION: Be careful. If you click away from the topology editor without clicking **Save**, your changes are discarded.

2. In the topology editor, click the system that you want to assign a system ID to, then click the **Manage selected nodes properties** button. (The screenshot below is from Apstra version 4.1.1).

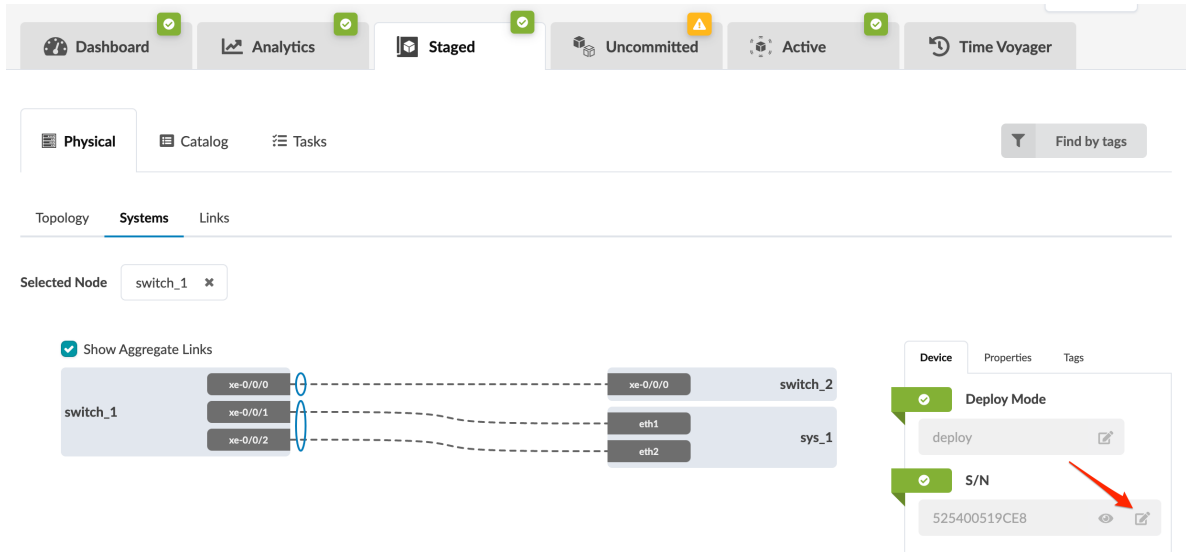


3. Select the system ID (serial number) from the **System** drop-down list (includes available managed devices). (If you don't see your system, you may still need to acknowledge it.)
4. To close the dialog, click the **Manage selected nodes properties** button again.
5. Click **Save** to stage your changes, exit the topology editor and return to the **Topology** view. (If you leave the page without saving, your changes are discarded.)

When you're ready to activate your changes, commit them from the **Uncommitted** tab.

Assign System (from Device Panel)

1. From the blueprint, navigate to **Staged > Physical > Systems** and click the name of the system that needs an ID assigned, either from the **Topology** view or from the **Systems** view. The **Device** panel for that system appears.
2. Click the **Edit** button for the **S/N** field.



3. Select the system ID from the drop-down list (includes available managed devices). (If you don't see your system, you may still need to acknowledge it.)
4. If you're going to deploy the device, make sure the deploy mode is set to **Deploy**, then save it.
5. Click the **Save** button to stage your changes.

When you're ready to activate your changes, commit them from the **Uncommitted** tab.

Unassign System (Freeform)

IN THIS SECTION

- Unassign System(s) (from Systems View) | 651
- Unassign System (from Topology Editor) | 653
- Unassign System (from Device Panel) | 655

You can unassign system IDs (serial numbers) from the **Systems** view, the **Topology** view, or from the **Device** panel.

Unassign System(s) (from Systems View)

1. From the blueprint, navigate to **Staged > Physical > Systems** and select the check box(es) for one or more systems to unassign, then click the **Change System IDs assignments** button.

Dashboard Analytics Staged Uncommitted Active Time Voyage

Physical Resource Management Catalog Tasks Find by tags

Topology **Systems** Links Aggregate Links

Create System

Query: All 1-12 of 12 Columns (11/11) Page Size: 25

Change System IDs assignments selected only unselected only

| | Name | Type | Tags | Deploy Mode | Device Profile | S/N | Hostname | Operation Mode | Config Template | Property Set | Actions |
|-------------------------------------|-------|----------|--------------|-------------|----------------|--------------|----------|----------------|-----------------|--------------|---------|
| <input checked="" type="checkbox"/> | leaf1 | INTERNAL | fabric, leaf | Deploy | Juniper vEX | 5254005E0016 | leaf1 | FULL CONTROL | crb_root.jinja | Not assigned | |
| <input checked="" type="checkbox"/> | leaf2 | INTERNAL | fabric, leaf | Deploy | Juniper vEX | 5254002891AB | leaf2 | FULL CONTROL | crb_root.jinja | Not assigned | |

2. In the dialog that opens, click the **Remove assignment** button (trash can) for the system(s) to unassign and change the deploy mode to **Undeploy**.

Assign Systems ×

Query: All 1-2 of 2

| Name | Hostname | System ID | Deploy Mode |
|-------|----------|--|---|
| leaf1 | leaf1 | 5254005E0016 (10.29.43.13) - some_location ✕ | <input checked="" type="radio"/> Deploy <input type="radio"/> Ready <input type="radio"/> Drain <input type="radio"/> Undeploy |
| leaf2 | leaf2 | 5254002891AB (10.29.43.15) - some_location ✕ | <input checked="" type="radio"/> Deploy <input type="radio"/> Ready <input type="radio"/> Drain <input type="radio"/> Undeploy |

Update Assignments

3. Click **Update Assignments** to stage your changes and return to the **Systems** view.

4. Click **Uncommitted** and ["commit" on page 692](#) changes to the blueprint to remove the system from the fabric.

The device is still under Apstra management. It's ready and available to be assigned to any blueprint.

To remove the device completely from Apstra management, ["remove the device from Managed Devices" on page 80](#).

Unassign System (from Topology Editor)

1. From the blueprint, navigate to **Staged > Physical > Topology** and click **Edit** to open the topology editor.



CAUTION: Be careful. If you click away from the topology editor without clicking **Save**, your changes are discarded.

2. In the topology editor, click the system that you want to unassign, then click the settings button.

The screenshot displays a network management interface with a topology diagram and a settings box for a selected node.

Top Navigation Bar: Dashboard, Analytics, Staged, Uncommitted, Active.

Left Sidebar: Physical, Resource Management, Catalog, Tasks.

Top Tabs: Topology, Systems, Links, Aggregate Links.

Top Right Buttons: Save, Cancel.

Top Left Controls: Full screen, Snap to Grid, Hold Shift for panning, Node search.

Topology Diagram: A network diagram showing a central router1 connected to spine1 and spine2. Spine1 is connected to leaf1 and leaf2. Spine2 is connected to leaf3. Leaf1 is connected to leaf1_server1. Leaf2 is connected to rack1_server1. Leaf3 is connected to leaf3_server1.

Settings Box for leaf2:

- Name: leaf2
- Hostname: leaf2
- Device Profile: Juniper vEX
- System: 5254002891AB (10.29.4 3.15) - some_location
- Deploy Mode: Deploy
- Tags: fabric, leaf

Annotations:

- 1. Arrow pointing to the leaf2 node in the topology.
- 2. Arrow pointing to the leaf2 node in the settings box.
- 3. Arrow pointing to the System field in the settings box, with text: "3. Remove system ID".
- 4. Arrow pointing to the Deploy Mode dropdown in the settings box, with text: "4. Change deploy mode to Undeploy".
- 5. Arrow pointing to the Save button in the top right.

Bottom Bar: Active Tasks: 0

3. Remove the system ID and change deploy mode to **Undeploy**, then click the system (or outside the settings box) to close the settings box.

4. Click **Save** to stage your changes, exit the topology editor and return to the **Topology** view. (If you leave the page without saving, your changes are discarded.)
5. Click **Uncommitted** and ["commit" on page 692](#) changes to the blueprint to remove the system from the fabric.

The device is still under Apstra management. It's ready and available to be assigned to any blueprint.

To remove the device completely from Apstra management, ["remove the device from Managed Devices" on page 80](#).

Unassign System (from Device Panel)

1. From the blueprint, navigate to **Staged > Physical** and click the name of the system to unassign, either from the **Topology** view or from the **Systems** view. The **Device** panel for that system appears.
[image]
2. Click the **Edit** button for deploy mode, and change it to **Undeploy**, then click the **Save** button.
3. In the **S/N** section, click the **Edit** button.
4. Click the red square in the **S/N** section to unassign the system ID.
5. Click **Uncommitted** and ["commit" on page 692](#) changes to the blueprint to remove the system from the fabric.

The device is still under Apstra management. It's ready and available to be assigned to any blueprint.

To remove the device completely from Apstra management, ["remove the device from Managed Devices" on page 80](#).

Device Context (Freeform)

The device context includes all the contextual data that you can use when creating dynamic Jinja config templates. It includes such data as interfaces, IP addresses, prefix lengths, name, and state. It also shows you what the neighbor interface is of other devices. You can search for data in a query box to pinpoint the information you're looking for.

1. From the blueprint, either from the **Topology** view or the **Systems** view, click the name of the system to view. Its details appear in the **Systems** view.

Topology **Systems** Links

Selected Node switch_3 ✕

Device Properties Tags

Deploy Mode

deploy

S/N

5254006252C1

Device Info

| | |
|----------------|-----------------|
| Management IP | 10.28.60.9 |
| OS | Junos 21.4R2.10 |
| Operation Mode | FULL CONTROL |

Hostname

switch3

Config

Rendered
Incremental
Pristine
Device Context

- At the bottom of the device panel on the right, click **Device Context** to go to device context for the device.

Device Context

Search

```

{
  all_resources { ... }
  interfaces { ... }
  system_tags { ... }
  aos_version: "4.1.1"
  chassis_config: {}
  configured_system_type: "internal"
  deploy_mode: "deploy"
  hostname: "switch3"
  id: "switch3"
  management_ip: null
  model: "Juniper_VQFX-10000"
  name: "switch_3"
  os_family: "Junos"
  property_sets: {}
  reference_architecture: "freeform"
  resources: {}
  routing_instance_supported: true
  system_type: "internal"
}

```


Links (Freeform)

IN THIS SECTION

- [Add Link \(Freeform\) | 657](#)
- [Edit Cabling Map \(Freeform\) | 659](#)
- [Fetch Discovered LLDP Data \(Freeform\) | 660](#)
- [Manage Link Tags \(Freeform\) | 661](#)
- [Delete Link \(Freeform\) | 661](#)

The **Links** view (Staged > Physical > Links) shows all the links that connect your devices together. The table includes information about endpoint names, link type, tags, speed, role, interface names and IP addresses. You can customize what appears in the table by selecting/deselecting elements in the columns drop-down list. You can perform various tasks from the **Links** view as described in later sections.

Topology
Systems
Links

Query: All

1-11 of 11

Columns (13/15)

Page Size: 25

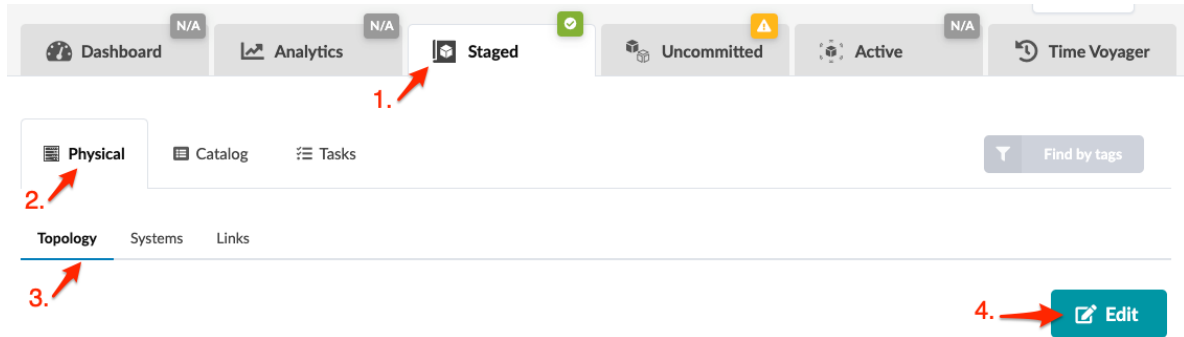
Filter selected by
☒ all
☐ selected only
☐ unselected only

| | | | | | | Endpoint 1 | | | | Endpoint 2 | | | |
|--------------------------|-------------------------|-----------|------|-------|----------|------------|----------------|--------------|--------------|------------|----------------|--------------|--------------|
| <input type="checkbox"/> | Name | Type | Tags | Speed | Role | Name | Interface Name | IPv4 address | IPv6 address | Name | Interface Name | IPv4 address | IPv6 address |
| <input type="checkbox"/> | switch_1<->switch_2 | Physical | | 10G | internal | switch_1 | xe-0/0/0 | Not assigned | Not assigned | switch_2 | xe-0/0/0 | Not assigned | Not assigned |
| <input type="checkbox"/> | switch_1<->switch_2_[1] | Aggregate | | | internal | switch_1 | ae1 | Not assigned | Not assigned | switch_2 | ae1 | Not assigned | Not assigned |
| <input type="checkbox"/> | switch_1<->sys_1[1] | Physical | | 10G | external | switch_1 | xe-0/0/1 | Not assigned | Not assigned | sys_1 | eth1 | Not assigned | Not assigned |

Add Link (Freeform)

After you've created systems you can link them to each other from the **Topology** view.

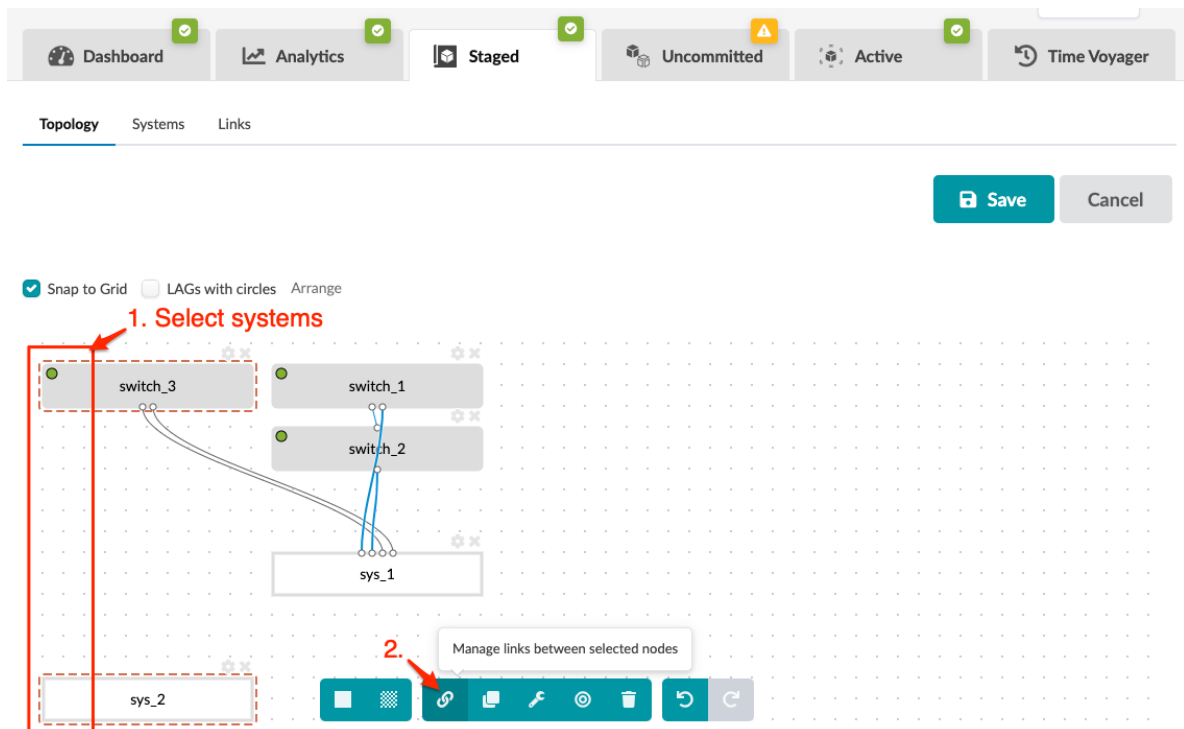
1. From the blueprint, navigate to **Staged > Physical > Topology** and click **Edit**.



CAUTION: Be careful. If you click away from the topology editor without clicking **Save**, your changes are discarded.

2. In the topology editor select the two systems that you want to link. You can select them in a couple of different ways:

- Click and drag across the two systems.
- Hold down the **alt** key (**command** key on a Mac) while clicking the two systems.



When you select two systems additional tasks become available in the context-aware menu at the bottom.

3. Click the **Manage links between selected nodes** button. The **Links Management** dialog opens showing the two node names (and device profiles, as applicable).

4. Click **Create Link**. The port representations appear.

Links Management

Node: switch_3
Device profile: Juniper vQFX

External Node: sys_2

1. ☒ switch_3<->sys_2

Tags: No tags

2.

3. IPv4: IPv6:
Tags: No tags

10G

4. ☒ Port #3 Tr. #1 (10 Gbps, default)

5.

☒ Select All With 1 selected: Add to Aggregate ▾ Delete

5. Select the check box for the first node.
6. Click the gear (upper-right) to show fields for IPv4, IPv6 and tags.
7. You can enter IP addresses and/or tags to add them to the device model which can be used later when creating config templates.
8. Select ports (and transformations as applicable), then click **Save**. (If you're connecting to an external system as in the example screenshot, you won't select ports.) You're still in the topology editor and if you click away without saving, your changes are discarded.
9. Click **Save** in the topology editor to save your changes and leave the topology editor. (Depending on the size of your topology, you may need to scroll to see the **Save** button.)

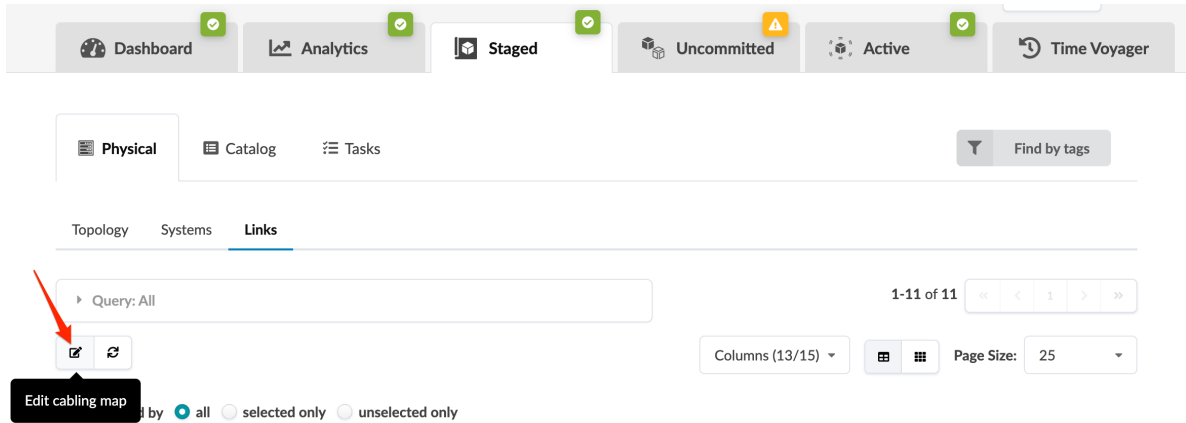
Next Steps:

If you haven't ["created config templates" on page 680](#) yet, create them now. If you have config templates ready for your devices and haven't assigned them yet, ["assign" on page 644](#) them now. When you've assigned all required config templates and all other requirements are met, you can deploy your blueprint from the **Uncommitted** tab.

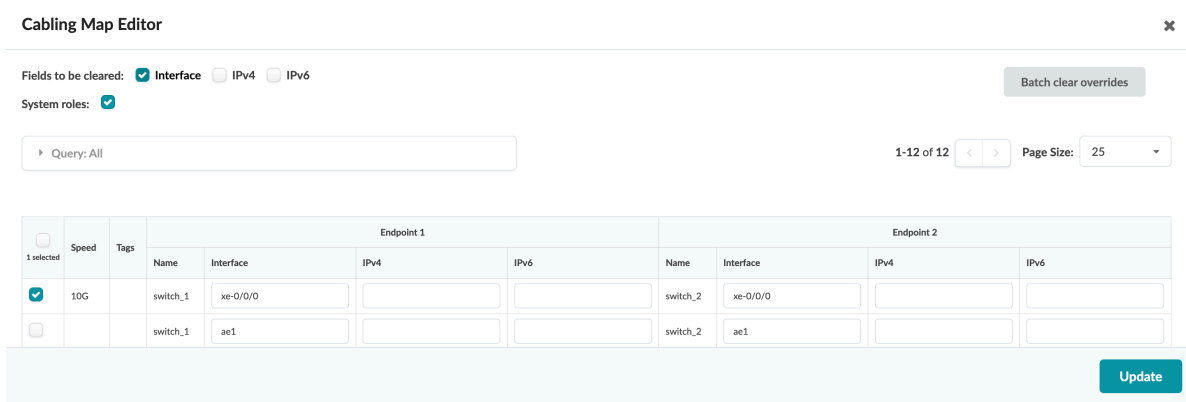
Edit Cabling Map (Freeform)

You can change one or more interfaces and IP addresses in the cabling map editor.

1. From the blueprint, navigate to **Staged > Physical > Links** and click the **Edit cabling map** button.



2. In the cabling map editor, change interface names and/or IP addresses, as applicable.
 - You can use **Batch clear override** to clear all interfaces and IPv4/IPv6 values for selected links.
 - To drop the override for either an interface name or IPv4/IPv6 address, submit an empty value in the corresponding field.



3. Click **Update** to stage your changes and return to the **Links** view.

Next Steps:

When you're ready to activate your changes, commit them from the **Uncommitted** tab.

Fetch Discovered LLDP Data (Freeform)

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 (second of two 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?** check box to see only the differences.
5. To accept the changes and update the map to match LLDP data, click **Update Staged Cabling Map from LLDP**.

Manage Link Tags (Freeform)

1. From the blueprint, navigate to **Staged > Physical > Links** and select one or more check boxes for the links to manage.

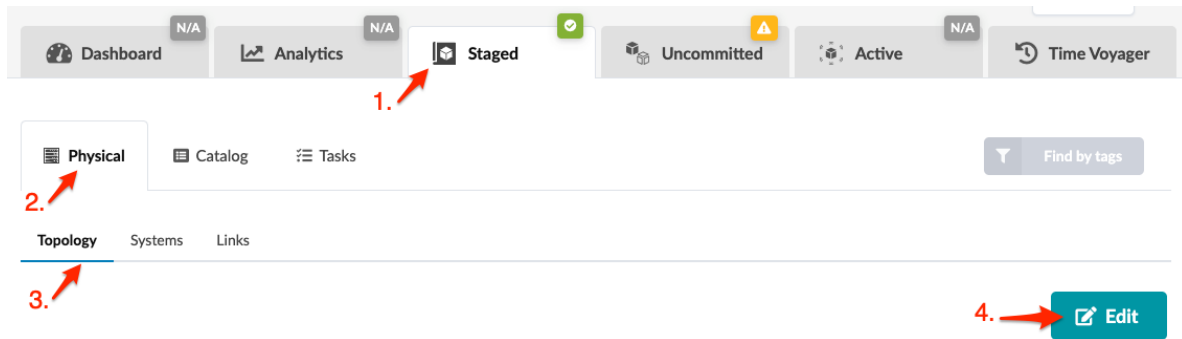
The screenshot shows the Apstra interface with the 'Links' view under the 'Physical' tab. The top navigation bar includes 'Dashboard', 'Analytics', 'Staged', 'Uncommitted', 'Active', and 'Time Voyager'. Below this is a sub-navigation bar with 'Physical', 'Catalog', and 'Tasks'. The 'Links' tab is selected. A search bar with 'Query: All' and a 'Find by tags' button are present. A 'Filter select' dropdown shows 'all' selected. A table lists links with columns for Name, Type, Tags, Speed, Role, and Endpoint 1/2 details. Two links are selected, indicated by red arrows and the number '1.'. A 'Tag' button is highlighted with a red arrow and the number '2.'.

| | Name | Type | Tags | Speed | Role | Endpoint 1 | | | | Endpoint 2 | | | |
|-------------------------------------|------------------------|-----------|------|-------|----------|------------|----------------|--------------|--------------|------------|----------------|--------------|--------------|
| | | | | | | Name | Interface Name | IPv4 address | IPv6 address | Name | Interface Name | IPv4 address | IPv6 address |
| <input checked="" type="checkbox"/> | switch_1<->switch_2 | Physical | | 10G | Internal | switch_1 | xe-0/0/0 | Not assigned | Not assigned | switch_2 | xe-0/0/0 | Not assigned | Not assigned |
| <input checked="" type="checkbox"/> | switch_1<->switch_2[1] | Aggregate | | | Internal | switch_1 | ae1 | Not assigned | Not assigned | switch_2 | ae1 | Not assigned | Not assigned |
| <input type="checkbox"/> | switch_1<->sys_1[1] | Physical | | 10G | External | switch_1 | xe-0/0/1 | Not assigned | Not assigned | sys_1 | eth1 | Not assigned | Not assigned |

2. Click the **Tag** button that appears above the list after selecting link(s).
3. In the dialog, add and/or remove tags, as needed.
4. Click **Add/Remove Tags** to stage the changes and return to the **Links** view.

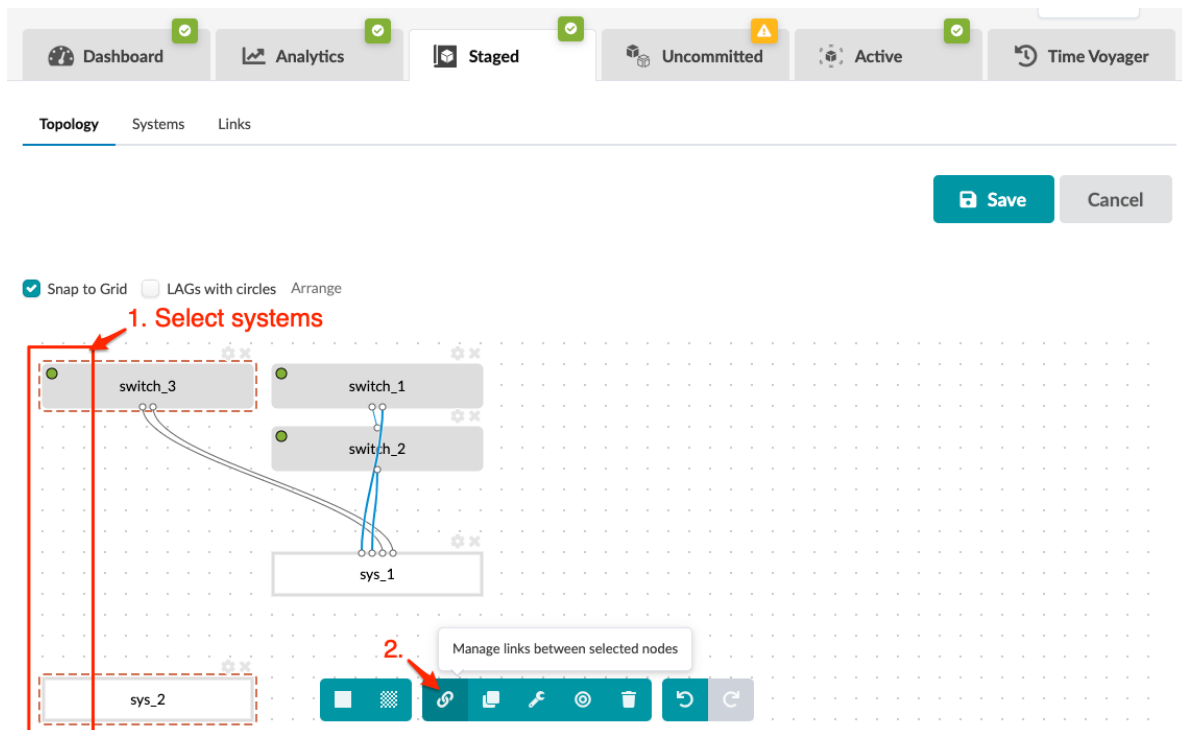
Delete Link (Freeform)

1. From the blueprint, navigate to **Staged > Physical > Topology** and click **Edit**.



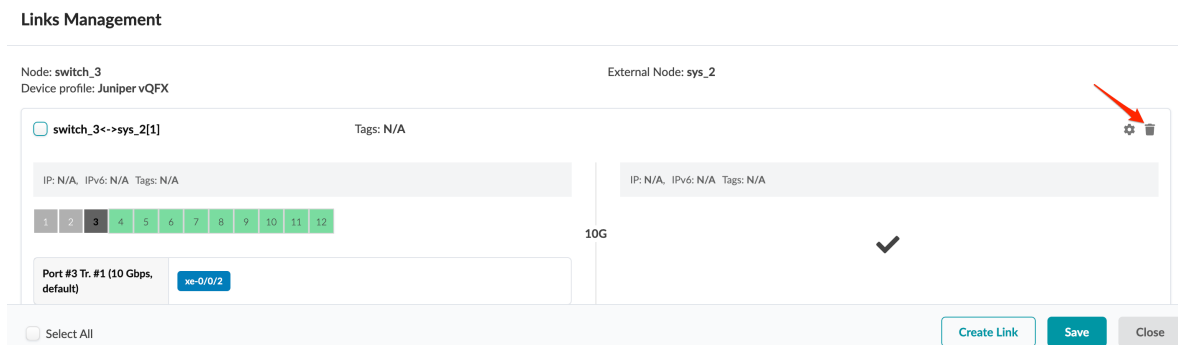
2. In the topology editor select the two systems where the link is that you want to delete. You can select them in a couple of different ways:

- Click and drag across the two systems.
- Hold down the **alt** key (**cmd** key on a Mac) while clicking the two systems.



When you select two systems additional tasks become available in the context-aware menu at the bottom.

3. Click the **Manage links between selected nodes** button. The **Links Management** dialog opens showing the two node names (and device profiles, as applicable).



4. Click the **Delete** button for the link to delete, then click **Save**. You're still in the topology editor and if you click away without saving, your changes are discarded.
5. Click **Save** (right-side) in the topology editor to stage your changes and return to the **Topology** view.

When you're ready to activate your changes, commit them from the **Uncommitted** tab.

Resource Management

IN THIS SECTION

- [Resource Management Introduction \(Freeform\) | 664](#)
- [Create Allocation Group \(Freeform\) | 669](#)
- [Create Group \(Freeform\) | 671](#)
- [Create Group Generator \(Freeform\) | 672](#)
- [Create Resource \(Freeform\) | 675](#)
- [Create Resource Generator \(Freeform\) | 676](#)
- [Create Local Pool \(Freeform\) | 677](#)
- [Create Local Pool Generator \(Freeform\) | 678](#)

Resource Management Introduction (Freeform)

SUMMARY

You can manage resources in Freeform blueprints from the **Resource Management** tab. Resources include IPv4 addresses, IPv6 addresses, ASNs, VNIs and integers that are used in VLANs.

IN THIS SECTION

- [Resource Types | 664](#)
- [Resource Groupings | 664](#)
- [Generators | 666](#)
- [Config Templates | 668](#)
- [Resource Management Workflow | 668](#)

In Apstra version 4.1.1, you assign resources in Freeform blueprints using property sets. Property sets specify values for variables, in this case for the different resource types. For more information, see ["Property Sets \(Freeform Blueprints\)" on page 686](#).

Apstra version 4.1.2 introduces a resource management feature for Freeform similar to the one provided for Datacenter. With Datacenter the mechanism is set up for you, and with Freeform you're responsible for setting it up yourself. You can set it up so resources are assigned and unassigned automatically as needed, just like in the Datacenter reference design.

Resource Types

In Apstra, resources are values that are assigned to various elements of the network. Resources include the following types:

- IPv4 (including Host IPv4)
- IPv6 (including Host IPv6)
- ASN - (autonomous system number)
- VNI (virtual network identifier)
- VLAN (virtual local area network)
- Integer - used for pool type VLAN in local pools (new in Apstra version 4.1.2)

Resource Groupings

Resources for Freeform blueprints are grouped and organized in the following ways:

Resource Pools

- consist of one or more ranges of resource values.
- contain one resource type (ASN, VNI, Integer, IPv4, or IPv6).
- are created in the global **Resources** catalog.
- can be used in one or more blueprints.
- are associated with allocation groups.

Allocation Groups

- consist of mappings to one or more resource pools.
- contain one resource type (ASN, VNI, Integer, IPv4, or IPv6).
- are created in the blueprint.
- are specific to one blueprint.
- provide the mechanism for pulling resources from pools and assigning them.

In the Datacenter reference design, templates determine the initial resource requirements. When you create a Datacenter blueprint (from a template) allocation groups are created automatically. Freeform reference design doesn't use templates, so resource requirements can't be determined when you create a Freeform blueprint. You'll create them yourself in Freeform blueprints.

Groups (Folders)

- are folders that are organized into a directory.
- contain assigned resources (and resource generators, described below).
- are used to arrange resources in any combination you like.
- can be nested inside other groups.
- can contain more than one resource type per group.
- are created in the blueprint.
- are specific to one blueprint.
- can be created and deleted automatically as needed, using group generators (described below).

- All resources must reside within a group (or group generator) that you create (not directly in the built-in **Root** group).

Local Pools

- consist of one or more ranges of resource values.
- contain only resource type Integer.
- contain only pool type VLAN.
- are created in the blueprint.
- are specific to one blueprint.
- can be created and deleted automatically as needed, with local pool generators (described below).

Generators

A generator in Apstra is a mechanism that automatically creates or deletes something based on your requirements so you don't have to do it manually. It can generate resources, local pools or groups for organizing resources and local pools. The graph database returns a set of objects based on a set of conditions that you specify. These conditions define the scope of what is added and/or removed.

Group Generator

You can put all of your resources in one group (folder), but if your design is complex, it's easier to manage resources in multiple groups. You can organize resources in any group combination that makes sense for you. You probably want to have nested groups, and you might want to have a group for every system in your network. Creating groups manually is simple enough; just click the group that you want to put your new group in and give it a name. Then you'd populate the group with your resources, either manually, or automatically with resource generators (described later). But, if you have many systems and you want a group for every system, creating each group manually is a lot of unnecessary work. You can automate this process with group generators.

To create a group generator, give it a name, then specify a scope based on how you want your groups to be created and managed. Our example of creating one group for every internal system uses the following scope:

```
node('system', system_type='internal', name='target')
```

This scope tells the graph database to find all internal systems and create a group for each one; and assign the applicable system name to each group. The state of the groups keeps in synch with the graph database as the fabric changes. If you subsequently delete a system, the group created for that system is

also deleted. All resources in that group are released back to the pool they came from, ready to be re-used. Conversely, if you create a system after this group generator is created, a group for that system is automatically created (and if you created resource generators inside the group generator, resources are also allocated accordingly).

Resource Generator

When it matters what the value is, you can allocate a resource manually, but in most cases you'll want to automate the process with resource generators. Resource generators don't actually generate resources; they pull existing resources from resource pools via allocation groups, based on a specified scope. Before creating a resource generator create any resource pools and allocation groups that you'll need. Creating an allocation group is straightforward; give it a name and select one or more resource pools to include in the group.

Resource Generator in a Group

Resources must be inside a group (or group generator as described below) that you create. To put all resources generated from a resource generator in one group, select the group and create your resource generator from there.

To create a resource generator, give it a name, then specify a resource type, an allocation group, a subnet prefix length for IPv4 only, and a scope. For example, you might want a group to contain link IPs (/31 addresses) for the links between all internal systems (switches) . First, create any resource pools and allocation groups that you'll need. In the resource generator, specify resource type IPv4, an applicable allocation group, the subnet prefix length, and the following scope:

```
node('link', role='internal', name='target')
```

This scope tells the graph database to find all fabric-facing links. The generator specifies to create link IPs for them, and add them to the group. Resources are automatically generated or released as links are added or removed.

Resource Generator in a Group Generator

To put every generated resource in its own group automatically, you can put your resource generator inside a group generator. The resource generator inherits the scope of the group generator.

For example, to create a group for every system and put an ASN in each group, you'd select the group generator already created and create the resource generator from there. The resource generator inherits the scope from the group generator. In our example, the scope is:

```
node('system', system_type='internal', name='target')
```

The graph database finds every internal system, allocates an ASN to each one, then puts each ASN in the applicable group based on internal systems.

Multiple Resource Generators in a Group Generator

You can put multiple resource generators inside a group generator (or group). Let's continue our example that already has a group for every internal system and an ASN in every group. You might also want your internal system groups to include loopback IP addresses. You can create a resource generator for loopback IP addresses in the same group generator as for the ASNs; you'd just select resource type IPv4.

The process is the same as when you added the ASNs. From the same group generator as before create the resource generator.....

Select a group to put the resource in, give it a name, specify the resource type and select an allocation group to pull the resource from. Then you'll have a resource in the specified folder. **You can see the resource in the table and the allocation group** it was pulled from. You can see if it's been assigned yet. Initially, it won't be. (put this in the task doc)

Local Pool Generator

You can create and assign a specific VLAN ID to a specific system (node) in your blueprint. If it doesn't matter what the specific value is, you can create a generator that will dynamically create and delete VLAN IDs based on the conditions you set. Values will be pulled from these pools as needed. These pools are specific to each blueprint.

Config Templates

(Add resources to config templates.)

Resource Management Workflow

1. Create resource pools ("[ASNs](#)" on page 262, "[VNIs](#)" on page 264, Integers, "[IPv4 addresses](#)" on page 267, "[IPv6 addresses](#)" on page 270) in the global **Resources** catalog. This is where you specify ranges of resource values.
2. "[Create allocation groups](#)" on page 669 in the blueprint.

This is where you specify one or more resource pools to be included in an allocation group. When you're ready to assign resources, you'll select resource pools from one of these allocation groups.
3. Plan how you'd like to organize your resources, then create "[groups](#)" on page 671 and "[group generators](#)" on page 672 in the blueprint, as applicable.
4. Create "[resources](#)" on page 671 and "[resource generators](#)" on page 676 in the blueprint, as applicable.

5. Create ["local pools" on page 677](#) and ["local pool generators" on page 678](#) in the blueprint, as applicable.
6. To render configuration, apply resources to Jinja2 config templates. For details, see [Resource Allocation in Apstra Freeform](#) lab guide.

Create Allocation Group (Freeform)

SUMMARY

Allocation groups consist of one or more resource pools that you use to assign resources (IPv4, IPv6, ASN, VNI, Integers).

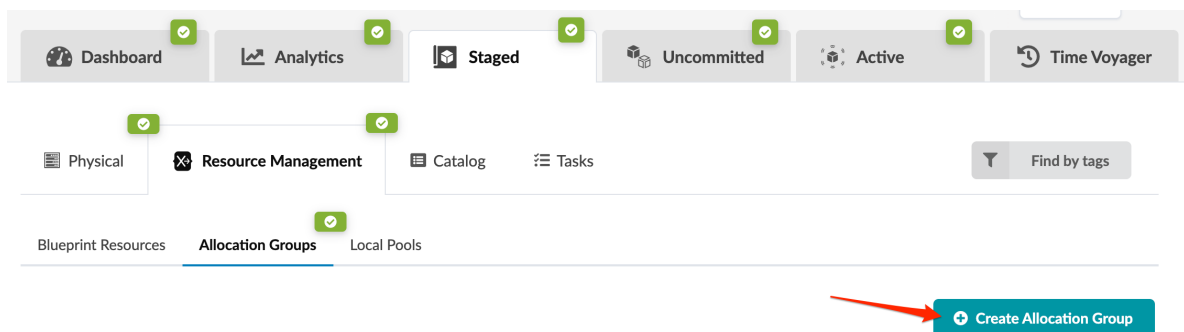
IN THIS SECTION

- [Create Allocation Group \(from Resource Management Tab\) | 669](#)
- [Create Allocation Group \(from Topology View\) | 670](#)

An allocation group consists of one or more ["global resource pools" on page 261](#). You'll assign resources later from one of these allocation groups. If you haven't created the resource pools you need, go do that before proceeding here.

Create Allocation Group (from Resource Management Tab)

1. From the blueprint, navigate to **Staged > Resource Management > Allocation Groups > Create Allocation Group**.



2. Enter an allocation group name and select the resource type (IPv4, IPv6, ASN, VNI, Integer).

Create Allocation Group ✕

Group Name *

Type
☒ IPv4 ☐ IPv6 ☐ ASN ☐ VNI ☐ Integer

Resource Pools
 Query: All 1-7 of 7

Filter selected by ☒ all ☐ selected only ☐ unselected only

| <input type="checkbox"/> | Pool Name | Total Usage | Per Subnet Usage | Status | |
|--------------------------|---|-------------|------------------|----------------|--------------|
| <input type="checkbox"/> | TESTNET-203.0.113.0/24 | 0% | 0% | 203.0.113.0/24 | ● NOT IN USE |
| <input type="checkbox"/> | Private-10.0.0.0/8 | 0% | 0% | 10.0.0.0/8 | ● NOT IN USE |
| <input type="checkbox"/> | Private-172.16.0.0/12 | 0% | 0% | 172.16.0.0/12 | ● NOT IN USE |
| <input type="checkbox"/> | f4d2a37f-9807-4537-b380-7040b1b58fbf-ra_link_ipv4 | 4.69% | 4.69% | 10.0.1.0/24 | ♥ IN USE |
| <input type="checkbox"/> | f4d2a37f-9807-4537-b380-7040b1b58fbf-ra_svis_ipv4 | 2.34% | 2.34% | 192.168.0.0/16 | ♥ IN USE |
| <input type="checkbox"/> | Private-192.168.0.0/16 | 0% | 0% | 192.168.0.0/16 | ● NOT IN USE |

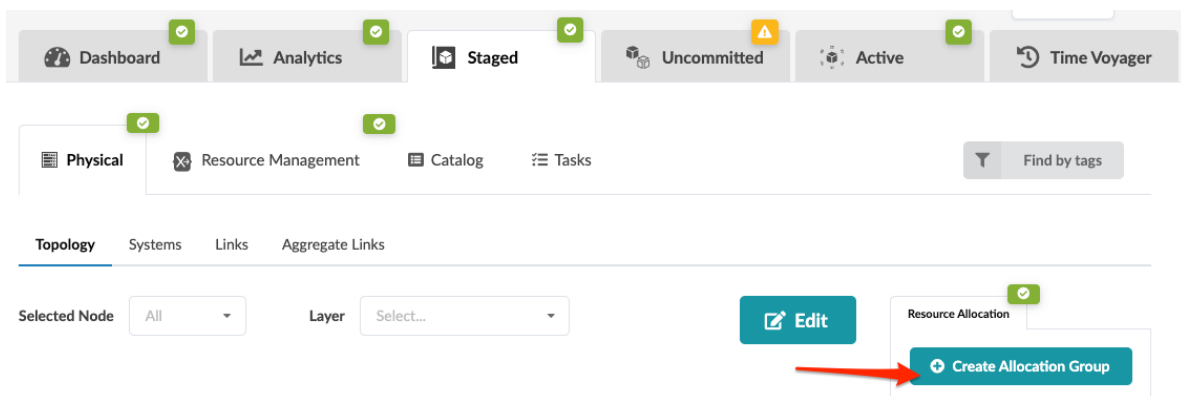
Create

- Select one or more check boxes for the resource pools to include in the allocation group. (These resource pools are from the global **Resources** catalog in the left navigation menu. You can add resource pools at any time if you need more resources available for your allocation groups.) You can create the group without selecting any resource pools, but of course, you'll need to add at least one before you can assign resources from it.
- Click **Create** to create the allocation group and return to the table view.

Next Steps: When you assign resources, you'll select an allocation group that you've created; then Apstra will pull resources from the group and assign them, as needed.

Create Allocation Group (from Topology View)

- From the blueprint, navigate to **Staged > Physical > Topology > Create Allocation Group**.



- Enter an allocation group name and select a resource type (IPv4, IPv6, ASN, VNI, Integer).

Create Allocation Group

Group Name *

Type

IPv4

IPv6

ASN

VNI

Integer

Resource Pools

Query: All

1-7 of 7

Filter selected by

all

selected only

unselected only

| <div><div>0</div><div>selected</div></div> | Pool Name | Total Usage | Per Subnet Usage | Status |
|--|---|------------------|---------------------------------|-------------------------|
| <input type="checkbox"/> | TESTNET-203.0.113.0/24 | <div>0%</div> | <div>0%</div> 203.0.113.0/24 | <div>●</div> NOT IN USE |
| <input type="checkbox"/> | Private-10.0.0.0/8 | <div>0%</div> | <div>0%</div> 10.0.0.0/8 | <div>●</div> NOT IN USE |
| <input type="checkbox"/> | Private-172.16.0.0/12 | <div>0%</div> | <div>0%</div> 172.16.0.0/12 | <div>●</div> NOT IN USE |
| <input type="checkbox"/> | f4d2a37f-9807-4537-b380-7040b1b58fbf-ra_link_ipv4 | <div>4.69%</div> | <div>4.69%</div> 10.0.1.0/24 | <div>♥</div> IN USE |
| <input type="checkbox"/> | f4d2a37f-9807-4537-b380-7040b1b58fbf-ra_svis_ipv4 | <div>2.34%</div> | <div>2.34%</div> 192.168.0.0/16 | <div>♥</div> IN USE |
| <input type="checkbox"/> | Private-192.168.0.0/16 | <div>0%</div> | <div>0%</div> 192.168.0.0/16 | <div>●</div> NOT IN USE |

Create

3. Select one or more check boxes for the resource pools to include in the allocation group. (These resource pools are from the global **Resources** catalog in the left navigation menu. You can add resource pools at any time if you need more resources available for your allocation groups.)
4. Click **Create** to create the allocation group and return to the **Topology** view.

When you assign resources, you'll select an allocation group that you've created; then Apstra will pull resources from the group and assign them, as needed.

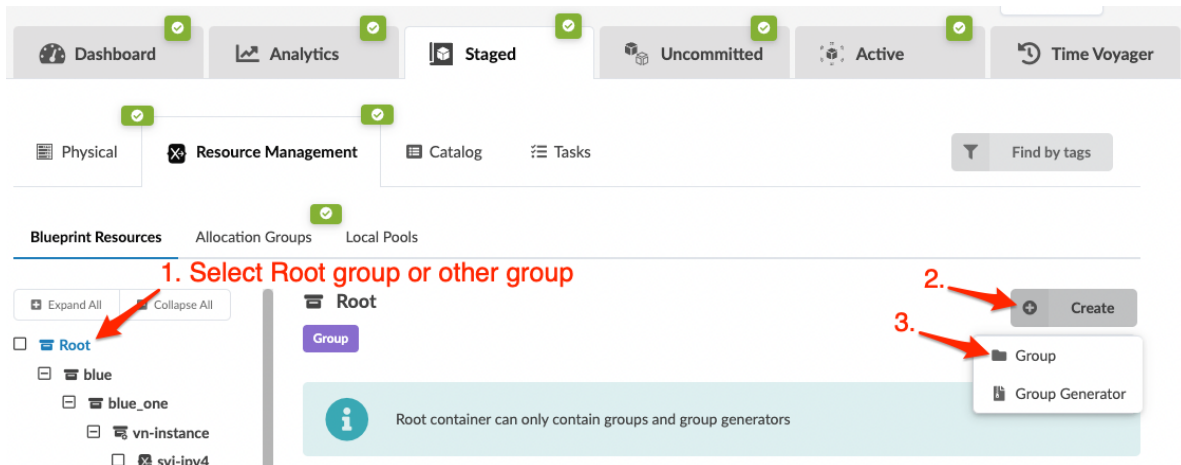
Create Group (Freeform)

SUMMARY

Organize resources in Freeform blueprints with groups (folders).

Groups are folders used to organize resources in Freeform blueprints. You can nest groups inside other groups in as many levels needed to organize your resources. You can add new groups to any existing group. If you haven't created any groups yet, you'll put your new group in the built-in **Root** group. (Instead of, or in addition to, creating groups manually as described here, you can ["create group generators"](#) on page 672 that create groups automatically and dynamically based on conditions that you set.)

1. From the blueprint, navigate to **Staged > Resource Management > Blueprint Resources**.



2. Click the group where you want to put the new group, then click **Create** (right-side) and select **Group**. The group you selected appears in the immutable **Parent** field.
3. Enter a group name.
4. (The Data field holds metadata that you can associate with the group. It's used to impart information to the object you've created. It may be used for things like a description or perhaps to indicate to others what the object represents.) If you'd like to add context to the group, enter applicable key-value pairs in the **Data** field.
Example: `{"group_type": "vn"}`
5. Click **Create** to create the group and return to the **Blueprint Resources** view.

When you've created one or more groups you can start putting resources and resource generators into them.

Create Group Generator (Freeform)

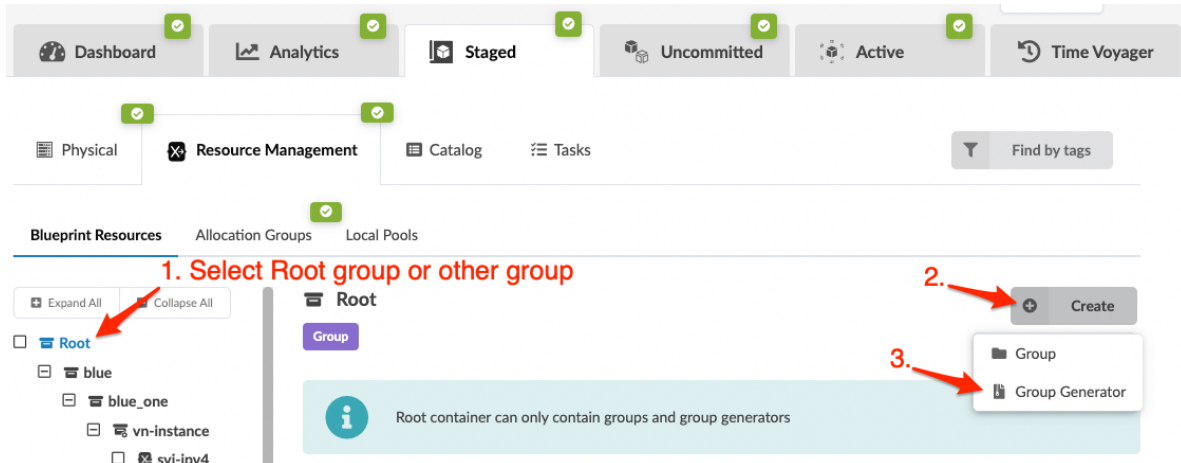
SUMMARY

Groups (folders) in Freeform blueprints organize resources. Group generators (folders with properties) automatically create and delete groups based on specified conditions.

For more explanation, see the ["Freeform Resource Management Introduction" on page 664](#).

1. From the blueprint, navigate to **Staged > Resource Management > Blueprint Resources**.

The group directory appears on the left. You can nest group generators inside any group.



2. Click the group where you want to put the new group generator, then click **Create** (right-side) and select **Group Generator**.

The name of the group you selected appears in the immutable **Parent** field.

3. Enter a group generator name, then specify a scope based on how you want your groups to be created and managed.

For example, to create one group for every internal system, use the following scope:

```
node('system', system_type='internal', name='target')
```

This scope tells the graph database to find all internal systems and create a group for each one of them; then assign the name of the system to each group. If you subsequently delete a system, the group created for that system is also deleted. Conversely, if you create a system after this group generator is created, a group for that system is automatically created.

Create Group Generator

Group Generator Name *

system

Parent

Root

Scope *

1 node('system', system_type='internal', name='target')

Open in Graph Explorer

Create

You can click the **Open in Graph Explorer** button to open a new tab that shows the groups that will be created based on the current topology. In our example, the topology includes 3 internal systems, and 3 groups will be created, as expected.

☆
🏠
Platform
Developers
Graph Explorer

Apstra Graph Explorer
Type: staging
Save Changes

Query Editor
Query Builder

1 = node('system', system_type='internal', name='target')

Show reference design schema
Show full blueprint
Fetch contextual data
Prettify
Refresh
Execute

Code
Graph

```

{
  "count": 3,
  "items": [
    {
      "target": {
        "id": "z4cI0b0V7JYfhVL9gVY",
        "type": "system",
        "label": "switch_3",
        "deploy_mode": "deploy",
        "hostname": "switch3",
        "management_level": "full_control",
        "property_set": null,
        "system_id": "525400E6F894",
        "system_type": "internal",
        "tags": null
      }
    }
  ],
  {
    "target": {

```

- Back in the **Create Group Generator** dialog, click **Create** to create the group generator and return to the **Blueprint Resources** view.

Groups will be created and deleted dynamically based on your specified conditions.

In our example, the group generator named **system** was created inside the **Root** folder, and it automatically created 3 groups, one for each of the systems in the topology. To see the resources in a group, click the name of the group. We haven't put any resources into the group we just created, so the resource table is empty.

Blueprint Resources

Allocation Groups

Local Pools

Expand All

Collapse All

Root

system

system (switch_2)

system (switch_3)

system (switch_1)

Root / system

Group (generated from system)

Create

Details

Data

{ }

Assignments

switch_2

Resources

Query: All

Page Size: 25

| Name | Type | Value | Generated By | Allocated From | Assigned To | Actions |
|----------|------|-------|--------------|----------------|-------------|---------|
| No items | | | | | | |

The group is empty.

Next Steps: ["Set up resource generators" on page 676](#) to automatically add and delete resources in your groups, as needed.

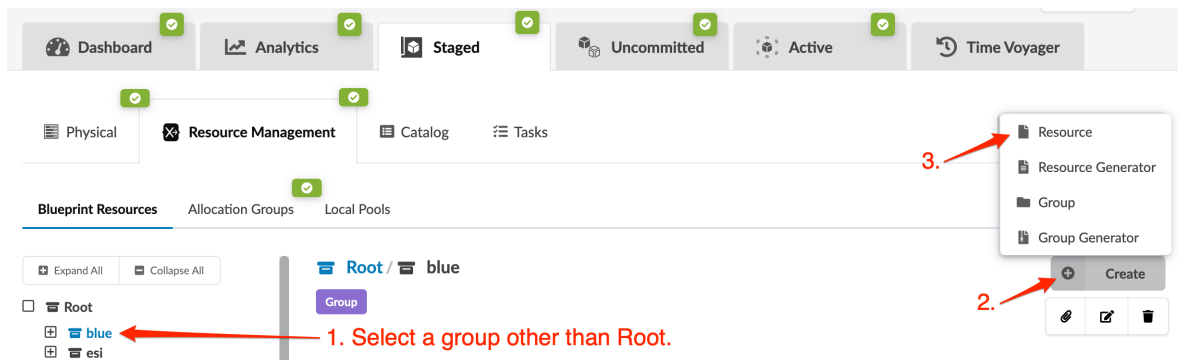
Create Resource (Freeform)

SUMMARY

Resources are values that you assign to systems and links. Resources include IPv4 addresses, IPv6 addresses, ASNs, VNIs, VLANs, and integers.

Resources are located inside groups (folders) that you create. (Resources can't be put directly in the predefined **Root** group). If you haven't ["created groups" on page 671](#) yet, create them before proceeding here.

1. From the blueprint, navigate to **Staged > Resource Management > Blueprint Resources**.



2. Click the group where you want to put the new resource, then click **Create** (right-side) and select **Resource**.

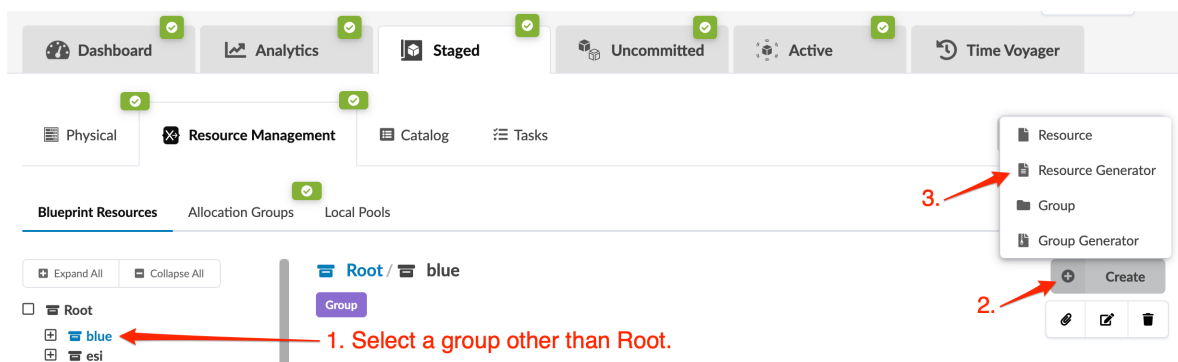
The group you selected appears in the immutable **Parent** field.

3. Enter a resource name and select a resource type (IPv4, Host IPv4, IPv6, Host IPv6, ASN, VNI, VLAN, Integer).
4. To have Apstra automatically pull resources from pools, select the applicable allocation group from the drop-down list.
5. To manually allocate a resource, enter the value. in the **Value (override)** field.
6. Enter a subnet prefix length, as applicable.
7. Click **Create** to create the resource and return to the **Blueprint Resources** view.

Create Resource Generator (Freeform)

Resource generators are located inside groups (folders) that you create. If you haven't ["created groups"](#) on [page 671](#) yet, create them before proceeding. To automate resource allocation you'll also need to confirm that you've created allocation groups and that they map to a sufficient number of resources.

1. From the blueprint, navigate to **Staged > Resource Management > Blueprint Resources**.



2. Select the group where you want to put the new resource generator, then click **Create** (right-side) and select **Resource Generator**.

The type and name of the container (group) appear in the immutable **Container Type** and **Container** fields, respectively.

3. Enter a resource generator name, then enter the scope for your generator.

To assist with determining scope, you can use the **Graph Explorer**.

Apstra Graph Explorer

Type: staging Save Changes

Query Editor Query Builder

1 = node('system', system_type='internal', name='target')

Show reference design schema Show full blueprint Fetch contextual data Prettify Refresh Execute

```

{
  "count": 3,
  "items": [
    {
      "target": {
        "id": "z4cI0b0V7JYfhVL9gVY",
        "type": "system",
        "label": "switch_3",
        "deploy_mode": "deploy",
        "hostname": "switch3",
        "management_level": "full_control",
        "property_set": null,
        "system_id": "525400E6F894",
        "system_type": "internal",
        "tags": null
      }
    }
  ],
  {
    "target": {

```

4. Click **Create** to create the resource generator and return to the **Blueprint Resources** view.

Resources will be generated and deleted dynamically based on the scope.

Create Local Pool (Freeform)

1. From the blueprint, navigate to **Staged > Resource Management > Local Pools > Create Local Pool**.

Dashboard Analytics Staged Uncommitted Active Time Voyager

Physical Resource Management Catalog Tasks Find by tags

Blueprint Resources Allocation Groups Local Pools

Create Local Pool

Query: All

1-5 of 5

Local Pools Local Pool Generators

2. Enter a local pool name.

Create Local Pool

Name *

Owner *

Select...

Pool Type

VLAN

Ranges *

1

4094

Add a range

Create

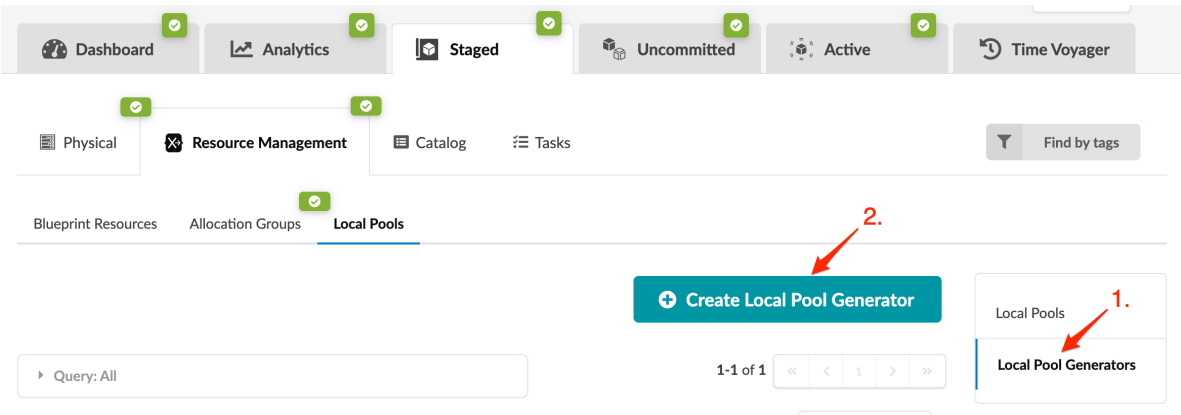
3. You'll be applying the integers to a system. Select the system from the **Owner** drop-down list.
4. Enter the range of integers for the pool.
5. If you want to add another range, click **Add a range** and enter the range.
6. Click **Create** to create the pool and return to the table view.

Create Local Pool Generator (Freeform)

SUMMARY

Create local pool generators to automatically create and delete local pools based on your criteria.

1. From the blueprint, navigate to **Staged > Resource Management > Local Pools > Local Pool Generator > Create Local Pool Generator**.




2. Enter a local pool generator name, then enter the scope for your generator.

Create Local Pool Generator

Name *

Scope *

1


Open in Graph Explorer 

Pool Type

VLAN

Ranges *

1 4094



 Add a range

Create

To assist with determining scope, you can use the **Graph Explorer**.






☆ [Home](#) > [Platform](#) > [Developers](#) > [Graph Explorer](#)



Apstra Graph Explorer

Type: staging   Save Changes

Query Editor Query Builder

1 = node('system', system_type='internal', name='target')

 Show reference design schema  Show full blueprint ☐ Fetch contextual data  Prettify  Refresh  Execute

 Code  Graph

```
{
  "count": 3,
  "items": [
    {
      "target": {
        "id": "z4cI0b0V7JYfhVL9gVY",
        "type": "system",
        "label": "switch_3",
        "deploy_mode": "deploy",
        "hostname": "switch3",
        "management_level": "full_control",
        "property_set": null,
        "system_id": "525400E6F894",
        "system_type": "internal",
        "tags": null
      }
    }
  ],
  {
    "target": {
```

3. Click **Create** to create the local pool generator and return to the **Local Pools** table view.

Config Templates (Freeform Blueprint)

IN THIS SECTION

- [A Simple Config Template | 680](#)
- [Config Template With Variable | 681](#)
- [Config Template and Property Sets | 681](#)
- [Create Config Template \(Freeform Blueprint\) | 681](#)
- [Edit / Delete Config Template \(Freeform\) | 683](#)
- [Import / Export Config Template \(Freeform\) | 684](#)

We recommend that you familiarize yourself with the Jinja [Template Designer](#) before working with config templates.

Several predefined config templates are included with the Apstra product. To get familiar with the syntax and how config Jinja is used in config templates, check out the sections below.

A Simple Config Template

Let's take a look at the config template `junos_protocols.jinja`, which ships with Apstra software.

```
protocols {  
  lldp {  
    port-id-subtype interface-name;  
    port-description-type interface-description;  
    neighbour-port-info-display port-id;  
    interface all;  
  }  
}
```

This straightforward template doesn't include any variables or other conditions. It's nested inside the config template `junos_configuration.jinja`, one of the other predefined config templates. You could create your own config template and nest this basic one in it as well.

Config Template With Variable

Let's look at `junos_system.jinja`, another predefined config template.

```
{% if hostname %}
system {
    host-name {{hostname}};
}
{% endif %}
```

This template includes an if-then statement and the variable `hostname`. When configuration is rendered, if the system device context includes a value for `hostname`, then the rendered configuration includes that value.

Config Template and Property Sets

An example of using property sets is with NTP servers. Configuration for NTP might be consistent across all devices in the enterprise except for time sources or strata per geography. You can build a config template with a variable, named `ntp` for example, in place of the actual IP address. The configuration will be generated with the value of the `ntp` property in a property set. You'd import the same config template into all blueprints, but for blueprints running in the east region you'd import the "EAST" property set, and for blueprint running in the west region you'd import the "WEST" property set. Property sets are global, that is they are blueprint-wide.

The config template could look like this.

```
{% if property_sets.get('ntp') %}
system {
    ntp {
        server {{property_sets['ntp']['ntp_server']}};
    }
}
{% endif %}
```

The example below shows the syntax for the property set `ntp` that contains the IP address.

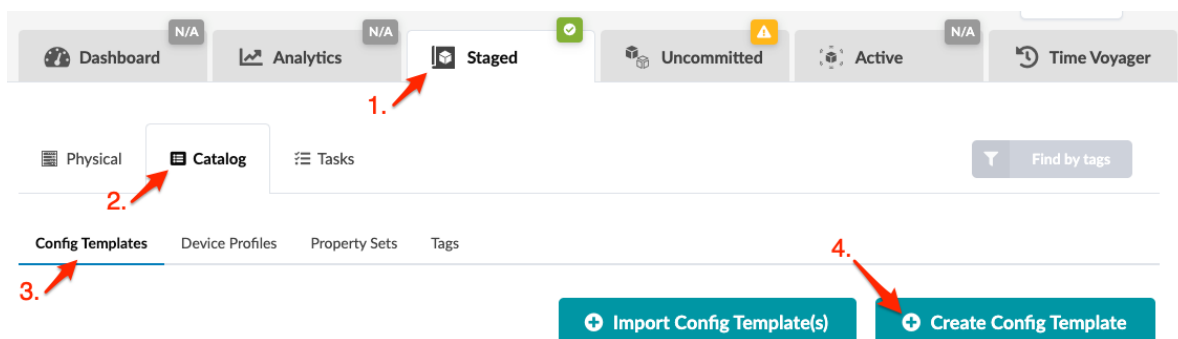
```
ntp_server = '1.2.3.4'
```

Create Config Template (Freeform Blueprint)

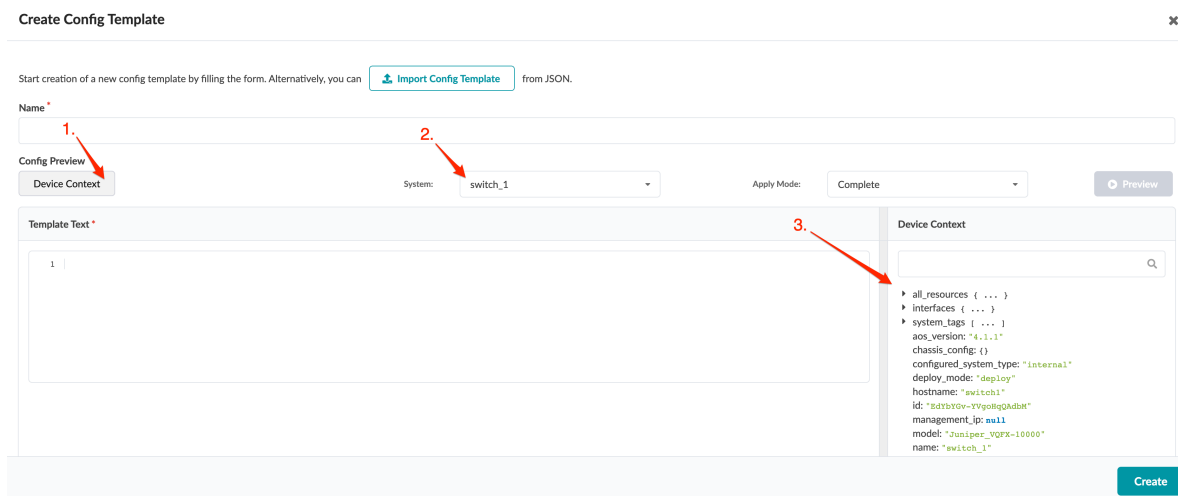
Creating config templates in the blueprint catalog (instead of the design catalog), gives you access to device context for systems that you've already added to your blueprint. Device context groups relevant

information into one place, making it easier to get the information you need while creating config templates.

1. From the blueprint, navigate to **Staged > Catalog > Config Templates** and click **Create Config Template**.



2. In the dialog, enter a name for the config template including the `.jinja` extension. (The `.jinja` extension is required even if you're not using Jinja.)
3. Enter or paste your content into the **Template Text** field. You can also import a config template that you created in the design (global) catalog.
 - To see device context, click **Device Context**.
 - To see device context for a specific system, select it from the **System** drop-down list.
 - Preview and Preview Mode are available only when you're editing a config template. (Preview Mode was called Apply Mode before Apstra version 4.1.2.)



4. Click **Create** to create the config template and return to the config template catalog view.

When you're ready you can ["assign config templates" on page 644](#) to internal systems.

Edit / Delete Config Template (Freeform)

IN THIS SECTION

- [Edit Config Template | 683](#)
- [Delete Config Template | 684](#)

Edit Config Template

1. From the blueprint, navigate to **Staged > Catalog > Config Templates** to go to the table view.
2. Either from the table view or the details view click the **Edit** button for the config template to edit.

Update Config Template

Name: junos_configuration.jinja

Click Device Context to see device context in right panel

Click Preview to see a preview of your changes

Config Preview

Device Context

System: switch_1

Apply Mode: Complete

Preview

Template Text

```

1 (# junos_system.jinja handles the system hostname #)
2 ({% include "junos_system.jinja" %})
3
4 (# junos_chassis.jinja handles chassis options, such as fpc config for
5  channelized port break-outs on certain device platforms. This also handles
6  aggregate-devices ethernet device-count for port-channel (aeX) interfaces. #)
7 ({% include "junos_chassis.jinja" %})
8
9 (# junos_interfaces.jinja handles front-panel interface configuration, including
10 interface description, ipv4/ipv6 address assignment, and physical link properties
11 derived from device profiles. #)
12 ({% include "junos_interfaces.jinja" %})
13
14 (# junos_protocols.jinja initiates LLDP collection on all ports for telemetry
15 purposes #)
16 ({% include "junos_protocols.jinja" %})
17

```

Preview (as rendered for device switch_1)

```

1 system {
2   host-name switch1;
3 }
4 chassis {
5   aggregated-devices {
6     ethernet {
7       device-count 2;
8     }
9   }
10 }
11 interfaces {
12   replaces xe-0/0/0 {
13     description "facing_switch-2:xe-0/0/0";
14     ether-options {
15       ...
16     }
17   }
18 }

```

Device Context

all_resources (...)

interfaces (...)

system_tags (...)

aos_version: "4.1.1"

chassis_config: { }

configured_system_type: "internal"

deploy_mode: "deploy"

hostname: "switch1"

id: "6d5b50v-Vyq0tqQadht"

management_ip: null

model: "Juniper_VQFX-10000"

name: "switch_1"

Save Changes

Update

3. In the dialog, make your changes.
 - To see device context, click **Device Context**.
 - To see device context for a specific system, select it from the **System** drop-down list.
 - To see a preview of your changes, click **Preview**.
 - To see the full configuration, including the changes you're making, select **Complete** from the **Preview Mode** drop-down list. (Preview Mode was called Apply Mode before Apstra version 4.1.2.)
 - To see only the configuration that you've changed, select **Incremental** from the **Apply Mode** drop-down list.
4. Click **Update** (bottom-right) to update the config template and return to the table view.

Delete Config Template

1. From the blueprint, navigate to **Staged > Catalog > Config Templates** to go to the table view.
2. Either from the table view or the details view click the **Delete** button for the config template to delete.
3. Click **Delete** to stage the deletion and return to the table view.

Import / Export Config Template (Freeform)

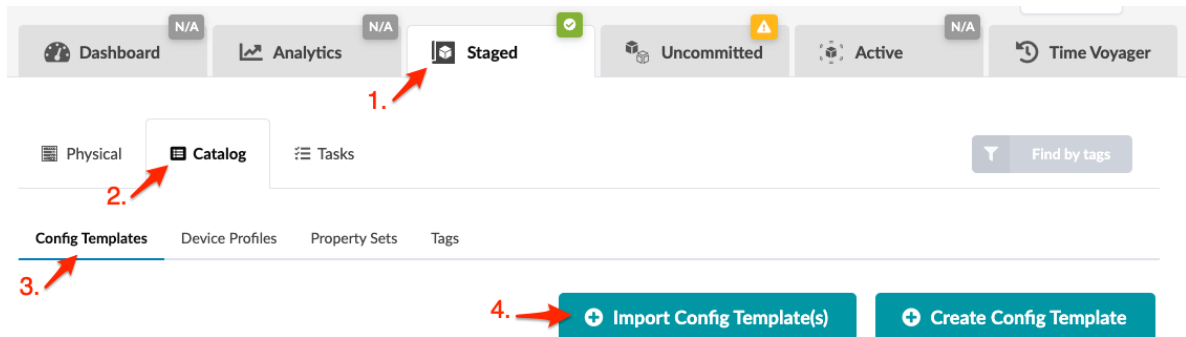
IN THIS SECTION

- [Import Config Template | 684](#)
- [Export Config Template | 684](#)

Import Config Template

You can create config templates in the design (global) catalog, then import them into as many blueprints as you want. (You can also create config templates directly in your blueprint, which gives you access to device context making it easier to write config template.)

1. From the blueprint, navigate to **Staged > Catalog > Config Templates** and click **Import Config Template(s)**.



2. Select the check boxes for the config templates to import from the design (global) catalog.
3. Click **Import** to stage the import and return to the table view.

Export Config Template

If you create a config template directly in a blueprint, and you want to make it available to other blueprints, you can export it to the design (global) catalog.

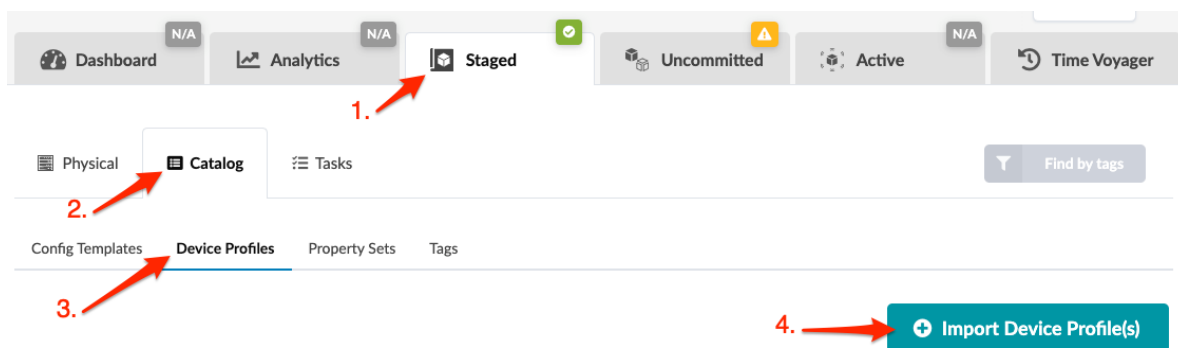
1. From the blueprint, navigate to **Staged > Catalog > Config Templates** to go to the table view.

2. Either from the table view or the details view click the **Export config template** button for the config template to export.
3. Click **Copy** to copy the contents, **Export to Global** to export the config template to the design (global) catalog, or click **Save As File** to download the file.
4. When you've copied, exported or downloaded the config template, close the dialog to return to the table view.

Import Device Profile (Freeform)

"[Device Profiles](#)" on [page 95](#) define the capabilities of supported hardware devices. They interact with devices via system agents. They don't include system IDs (serial numbers) which enables you to build your network in the Apstra environment 'offline' before you have your devices ready. In Freeform blueprints you import device profiles to provide context for configuring systems with config templates.

1. From the blueprint, navigate to **Staged > Catalog > Device Profiles** and click **Import Device profile(s)** (right-side).



2. Select one or more check boxes for the device profile(s) to import into the blueprint. Only device profiles that are supported in Freeform appear in the list (Juniper devices only in Apstra release 4.1.1).
3. Click **Import** to stage the change and return to the table view. The newly imported device profile(s) appear in the list.

Next Steps:

You're ready to "[create internal systems](#)" on [page 633](#) and assign your imported device profiles to them.

Property Sets (Freeform Blueprints)

IN THIS SECTION

- [Create Property Set \(Freeform Blueprint\) | 686](#)
- [Edit / Delete Property Set \(Freeform Blueprint\) | 687](#)

Property sets provide a valuable capability to fully parameterize config templates. Consisting of key-value pairs, they enable you to separate static portions of config templates from variables. You create property sets in the blueprint catalog. (Property sets used in Freeform blueprints are not related to property sets in the design (global) catalog.) You'll include property set names in your config template and then the values in those property sets will be used when configuration is rendered.

You can also create a property set and assign it directly to one system.

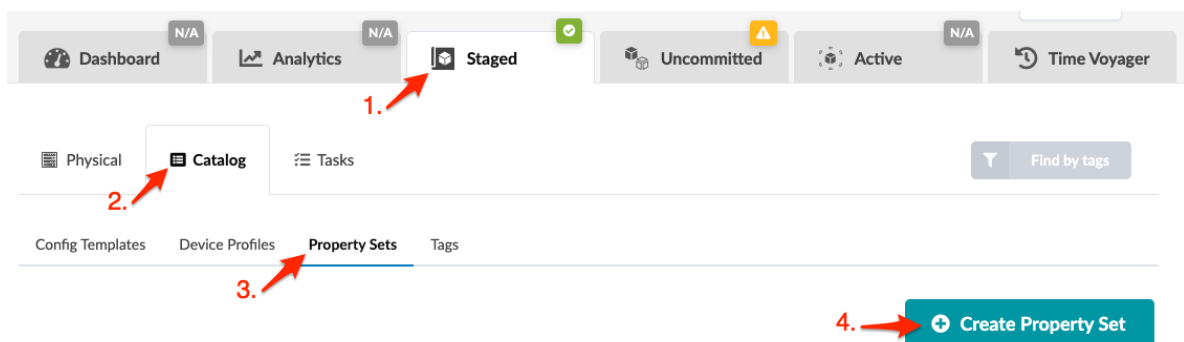
Create Property Set (Freeform Blueprint)

IN THIS SECTION

- [Create Property Set with Builder | 686](#)
- [Create Property Set with Editor | 687](#)

Create Property Set with Builder

1. From the blueprint, navigate to **Staged > Catalog > Property Sets** and click **Create Property Set**.

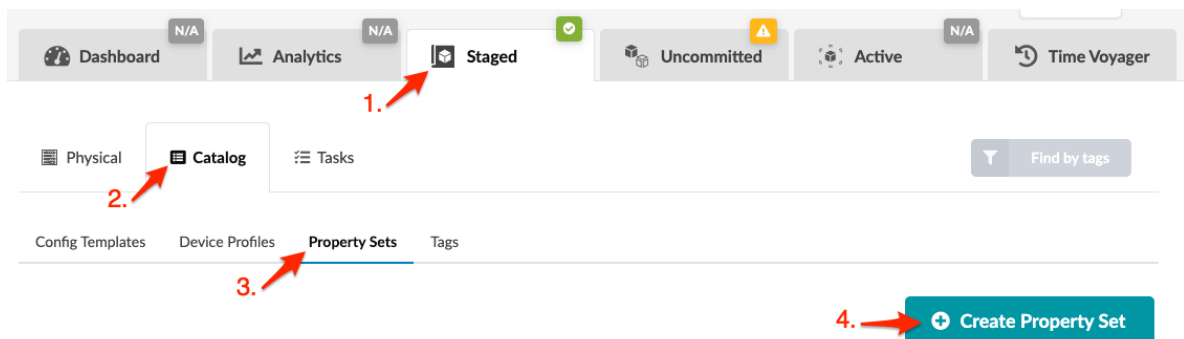


2. Enter a name for the property set.

3. If you want to assign the property set to a specific system, select it from the **System** drop-down list.
4. The **Builder** input type is selected, by default.
5. Use the interactive builder to help you create the content for your property set.
6. Click **Create** to stage the new property set and return to the property set catalog. The newly created property set is in the list.

Create Property Set with Editor

1. From the blueprint, navigate to **Staged > Catalog > Property Sets** and click **Create Property Set**.



2. Enter a name for the property set.
3. If you want to assign the property set to a specific system, select it from the **System** drop-down list.
4. Select **Editor** input type.
5. Copy and paste your content in the editor or type it in.
6. Click **Create** to stage the new property set and return to the property set catalog. The newly created property set is in the list.

Edit / Delete Property Set (Freeform Blueprint)

IN THIS SECTION

- [Edit Property Sets | 687](#)
- [Delete Property Sets | 688](#)

Edit Property Sets

1. From the blueprint, navigate to **Staged > Catalog > Property Sets** to go to the table view.
2. Either from the table view or the details view, click the **Edit** button for the property set to edit.
3. Make your changes.

4. Click **Update** to stage your changes and return to the table view.

Delete Property Sets

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 table view.

Tags (Freeform Blueprint)

IN THIS SECTION

- [Create Tag \(Freeform Blueprint\) | 689](#)
- [Edit / Delete Tag \(Freeform Blueprint\) | 689](#)

You can add **tags** to systems, then later when you want to find systems you can use the **Find by Tags** feature to find them.

You can include **Tags** in config templates. Systems/links with those tags will be rendered as specified in the config template. For example, if you have bare metal servers with SRIOV interfaces, and you need to produce specific configuration for those interfaces, you can add the tag `sriov`, then specify that links with that tag to be configured per the config template.

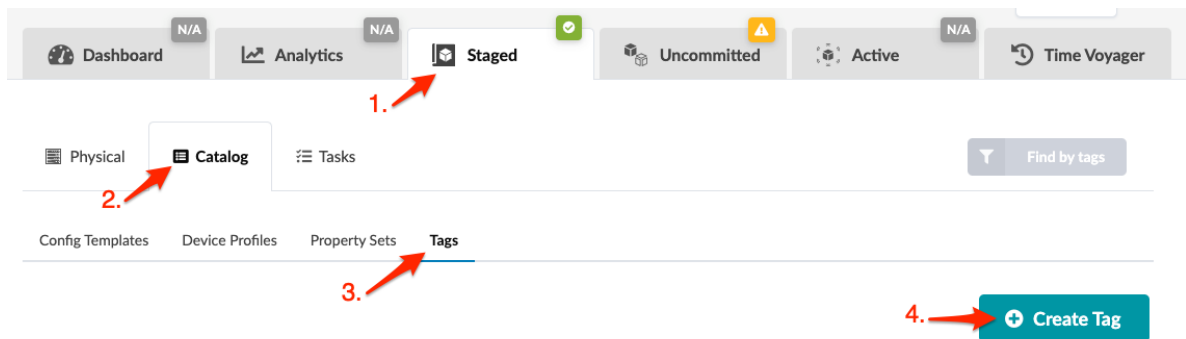
Tags are a way for you to assign metadata to Apstra-managed resources. Tags can help you identify, organize, search for, and filter Apstra systems and links. With tags, you can categorize resources by purpose, owner, environment, or other criteria. Because tags are metadata, they are not just used for visual labeling; they are also applied as properties of nodes in the Apstra graph database. This node property (or device property) is then available for you to reference in Jinja for dynamic variables in config generation and the Apstra real-time analytics via Apstra's Live Qurey technology and Apstra Intent-Based Analytics.

Here is an example of using the tag firewall in a ["config template" on page 680](#) to render a specific description.

```
{% if has_tag(interface.link.neighbor_system.id, 'firewall') %}
  description "this is a firewall facing interface";
{% endif %}
```

Create Tag (Freeform Blueprint)

1. From the blueprint, navigate to **Staged > Catalog > Tags** and click **Create Tag**.



2. Enter a tag name. Names are case-insensitive.
3. Enter a description (optional).
4. Click **Create** to stage the tag addition and return to the list view. The newly created tag appears in the summary table.

Edit / Delete Tag (Freeform Blueprint)

IN THIS SECTION

- [Edit Tags | 689](#)
- [Delete Tag | 690](#)

Edit Tags

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 table view.

Delete Tag

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 table view.

Tasks - Staged (Freeform)

Tasks that haven been performed in blueprints appear in the Tasks tab. Blueprint task details include task type, task status (succeeded, failed, in progress), user who performed the task, date/time created/started/updated, and the duration of the task. For any failed tasks, you can click to see error messages. From the blueprint, navigate to **Staged > Tasks** to go to task history.

Dashboard

Analytics

Staged

Uncommitted

Active

Time Voyager

Physical

Catalog

Tasks

Find by tags

Query: All

1-13 of 13

Page Size: 25

| ID | Type | Status | User | Created At | Started At | Last Updated At | Duration, s |
|--------------------------------------|------------------------------------|-----------|-------|----------------------|----------------------|----------------------|-------------|
| e0ad5d9a-084f-4880-9ff7-d0b924eb4798 | Deploy Blueprint | Succeeded | admin | 2022-07-19, 10:08:25 | 2022-07-19, 10:08:25 | 2022-07-19, 10:08:25 | 0.114 |
| 5aada114-8258-4489-b6f1-3f9843ea6cb3 | Update Graph Node Property (Batch) | Succeeded | admin | 2022-07-19, 10:08:24 | 2022-07-19, 10:08:24 | 2022-07-19, 10:08:24 | 0.164 |
| 8d6d8e50-eb81-45fc-939a-49ee11162579 | Deploy Blueprint | Succeeded | admin | 2022-07-19, 10:07:07 | 2022-07-19, 10:07:07 | 2022-07-19, 10:07:07 | 0.082 |
| 23ba7728-4d29-40c9-b385-600c66d3a5f | Update Graph Node Property (Batch) | Succeeded | admin | 2022-07-19, 10:07:07 | 2022-07-19, 10:07:07 | 2022-07-19, 10:07:07 | 0.167 |
| 508642e7-679d-432f-9c79-b3d1677c7421 | Deploy Blueprint | Succeeded | admin | 2022-07-19, 10:05:50 | 2022-07-19, 10:05:50 | 2022-07-19, 10:05:50 | 0.092 |
| ecfc1459-d99b-4814-bd5f-ebdb71e4cb90 | Update Graph Node Property (Batch) | Succeeded | admin | 2022-07-19, 10:05:49 | 2022-07-19, 10:05:49 | 2022-07-19, 10:05:50 | 0.202 |
| bc832ffe-718c-4ee7-bd7f-113d69bc2fb5 | Deploy Blueprint | Succeeded | admin | 2022-07-19, 10:03:15 | 2022-07-19, 10:03:16 | 2022-07-19, 10:03:16 | 0.303 |
| ca1a2112-313f-4af0-8186-33adfad8decf | Update Graph Node Property (Batch) | Succeeded | admin | 2022-07-19, 10:03:15 | 2022-07-19, 10:03:15 | 2022-07-19, 10:03:15 | 0.379 |
| 730a443b-0dc0-454e-bb0e-c6e5765e9ede | Update Graph Node Property (Batch) | Succeeded | admin | 2022-07-19, 10:02:19 | 2022-07-19, 10:02:19 | 2022-07-19, 10:02:19 | 0.155 |
| 55289b82-02cd-4676-ba78-e54f25c0c7ac | Batch Update Systems/Cabling Map | Succeeded | admin | 2022-07-19, 10:02:18 | 2022-07-19, 10:02:18 | 2022-07-19, 10:02:18 | 0.55 |
| 49255455-5f44-44dc-83b0-5bd6ee958a87 | Import Config Template(s) | Succeeded | admin | 2022-07-19, 10:02:17 | 2022-07-19, 10:02:17 | 2022-07-19, 10:02:17 | 0.244 |
| 7537c577-c88b-40ee-add8-7be29d558844 | Import Device Profile(s) | Succeeded | admin | 2022-07-19, 10:02:16 | 2022-07-19, 10:02:16 | 2022-07-19, 10:02:17 | 0.207 |
| ca031541-5045-4625-abd4-bb6904caf795 | Create Blueprint | Succeeded | admin | 2022-07-19, 10:02:08 | 2022-07-19, 10:02:08 | 2022-07-19, 10:02:12 | 4.494 |

Active

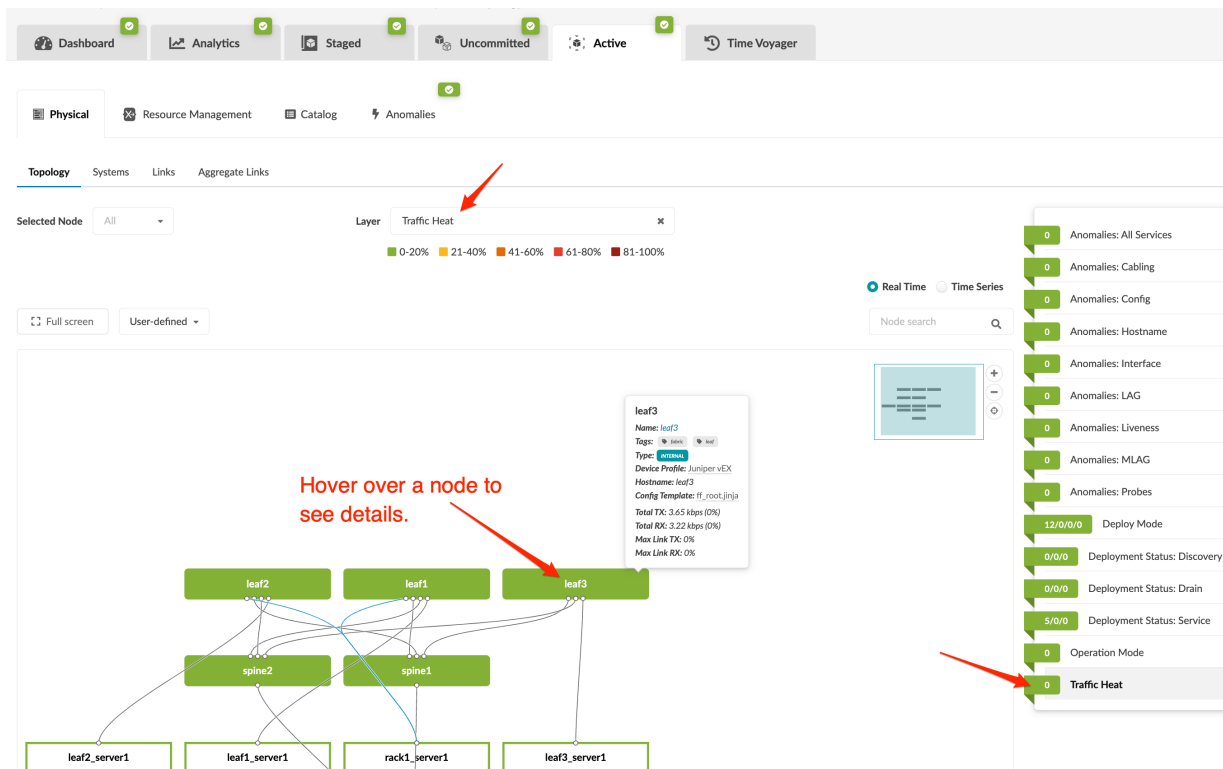
IN THIS SECTION

- [Traffic Heat \(Freeform\) | 691](#)

Traffic Heat (Freeform)

Traffic and resource utilization information is available in Freeform blueprints as of Apstra version 4.1.2.

From the blueprint, navigate to **Active > Physical > Topology**. In the status panel on the right you can quickly see if your network is having any issues. In this section, we're looking at Traffic Heat, whose status indicator is at the bottom of the panel. You can see more details by selecting **Traffic Heat** from the **Layer** drop-down list.



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, hover over the node.

Commit Blueprint

IN THIS SECTION

- [Uncommitted Overview | 692](#)
- [Review Staged Changes | 694](#)
- [Commit Staged Changes | 696](#)
- [Revert Staged Changes | 697](#)

Uncommitted Overview

While you're 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 18: Uncommitted Status Indicators

| Status Indicator Color | Description |
|------------------------|---|
| Red | The blueprint needs staging or has Build Errors that must be resolved before you can commit. |
| Orange | The blueprint has Warnings 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'. 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.

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.

Click to see changes

| 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 |

2. From **Logical Diff**, click a name from the **Name** column to see detailed changes, additions or deletions for that element.

Connectivity Template Preview

Properties

Title: rtr_leaf1_leaf2:l3:ct_bgp_subintf_to_subintf:ipv4 → rtr_leaf1_leaf2:l3:ct_bgp_subintf_to_subintf:ipv4

Description: ... → ...

Tags: ... → ...

Parameters

- logical_link_blue_0
- bgp_blue_0
- logical_link_red_0
- bgp_red_0
- logical_link_default_0

Staged

Active

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 | Unassigned ↓ VLAN Tagged |
| leaf2 | Ethernet1/4 | switch2-server1 | single-link | single-server-2 | Unassigned ↓ VLAN Tagged |

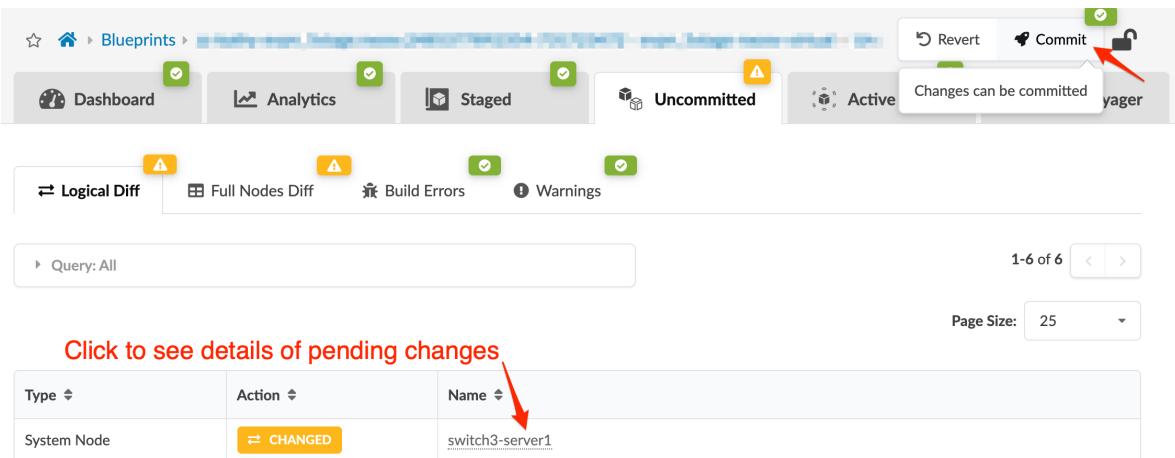
In Apstra version 4.1.2, the preview for config template changes is color-coded to easily see the content that has been added (in green) and the content that has been removed (in red).



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.



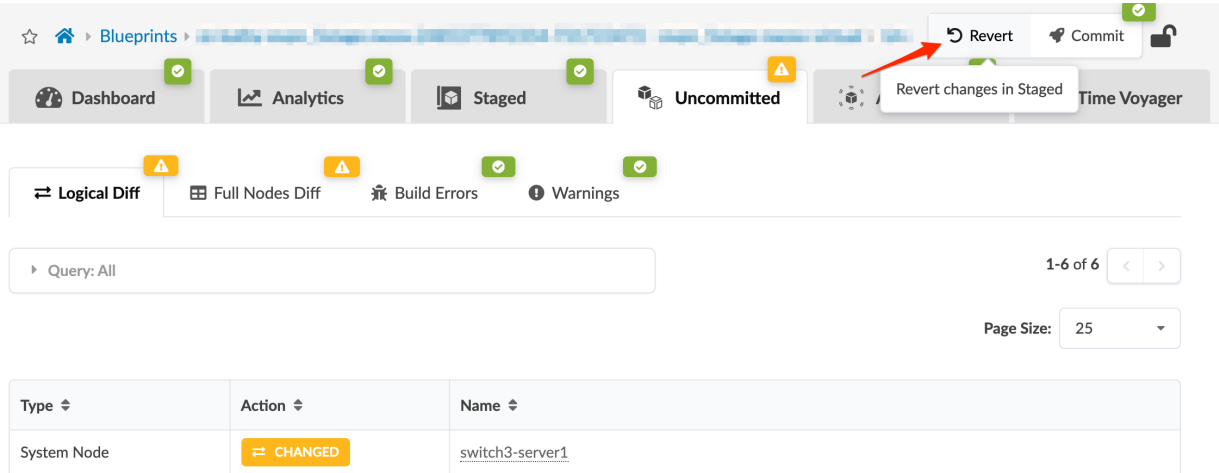
2. Click **Commit** and add a description of the changes. We recommend that you enter the optional revision description to identify changes. These descriptions are displayed in the **Revisions** section of "Time Voyager" on page 697. If you don't add a description now you can always add one later. If 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. The Apstra engine validates all commits and makes sure everything works as it pushes configuration. Cabling anomalies may appear until validation is complete.
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 279](#).



Time Voyager

IN THIS SECTION

- [Time Voyager Overview | 698](#)
- [Jump to Previous Blueprint Revision | 700](#)
- [Keep Saved Blueprint Revision | 701](#)
- [Update Blueprint Revision Description | 701](#)
- [Delete Kept Blueprint Revision | 701](#)

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 844](#). 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 'Dashboard', 'Analytics', 'Staged', 'Uncommitted', 'Active', and 'Time Voyager'. Below the navigation bar is a 'Revisions' section with a search bar and a 'Query: All' filter. A table lists revisions with columns for Description, Created At, User, and Actions. The first revision is marked as 'current'. Red arrows point to the 'Time Voyager' tab, the 'Jump to this revision' button, the 'Update description' button, and the 'Delete' button.

| Description | Created At | User | Actions |
|----------------------------|---|-------|---|
| | 2021-10-13, 16:09:30 current | admin | Jump to this revision Update description Delete |
| | 2021-10-13, 13:51:56 | admin | Jump to this revision Update description Delete |
| snmp trap option configlet | 2021-10-13, 11:52:38 | admin | Jump to this revision Update description Delete |
| | 2021-10-12, 13:17:49 | admin | Jump to this revision Update description Delete |
| | 2021-10-12, 13:16:58 | admin | Jump to this revision Update description Delete |
| with NTP error | 2021-10-11, 18:37:38 | admin | Jump to this revision Update description Delete |
| with NTP | 2021-10-11, 17:58:39 | admin | Jump to this revision Update description Delete |
| base | 2021-10-11, 17:45:59 | admin | Jump to this revision Update description Delete |

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 277](#).
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 table 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 table view.

Analytics

IN THIS SECTION

- [Analytics Overview | 702](#)
- [Analytics Dashboard | 703](#)
- [Configure Auto-Enabled Dashboards | 704](#)
- [Instantiate Predefined Dashboard | 704](#)
- [Create Analytics Dashboard | 705](#)
- [Edit / Delete Dashboard | 705](#)
- [Anomalies \(Analytics\) | 706](#)
- [Widgets Overview | 706](#)
- [Create Anomaly Heat Map Widget | 707](#)
- [Create Stage Widget | 707](#)
- [Edit / Delete Widget | 708](#)
- [Probes | 709](#)
- [Instantiate Predefined Probe | 715](#)
- [Create Probe | 716](#)
- [Import / Export Probe | 716](#)
- [Edit / Delete Probe | 717](#)

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 1175](#) with current and historic data from devices to reason about the network at-large.

Agents ingest data that devices generate and send them to the Apstra server. With IBA ["probes" on page 709](#), you can aggregate data across devices based on how they are configured. Combining probes with intent from the blueprint graph generates a reduced set of data. You can directly inspect advanced data from the Apstra GUI or from ["REST API" on page 802](#) to gain real-time insight about the network. You can stream data out with our existing streaming infrastructure. Also, based on the state of this advanced data, probes can raise ["anomalies" on page 706](#).

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.

System snapshots and old images from in-place Apstra server upgrades may use additional space. You can delete them or move them 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 \(operational\) blueprint" on page 596](#). 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 706](#) 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 274](#) 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. Select a **Display mode**

(summary, preview, expanded) to view dashboards in various levels of detail.

1.

2.

Click dashboard name for details

Default analytics dashboard will be shown on the blueprint's dashboard.

| 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 |

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

You can instantiate several predefined dashboards and modify them 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 995](#) 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 706](#) to include in the dashboard.
5. Click **Create Dashboard** to create the dashboard and return to the table 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 | 705](#)
- [Delete Dashboard | 706](#)

Edit Dashboard

You can modify auto-enabled dashboards, 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 table 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. If you want to delete all widgets and probes that are exclusively used this dashboard, check the check box. Deleting unnecessary widgets and probes frees up disk space.
3. Click **Delete Dashboard** to delete the dashboard and return to the table view.

Anomalies (Analytics)

From the blueprint, navigate to **Analytics > Anomalies** to go to the list of anomalies that the IBA probes have detected. 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** check box. 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.

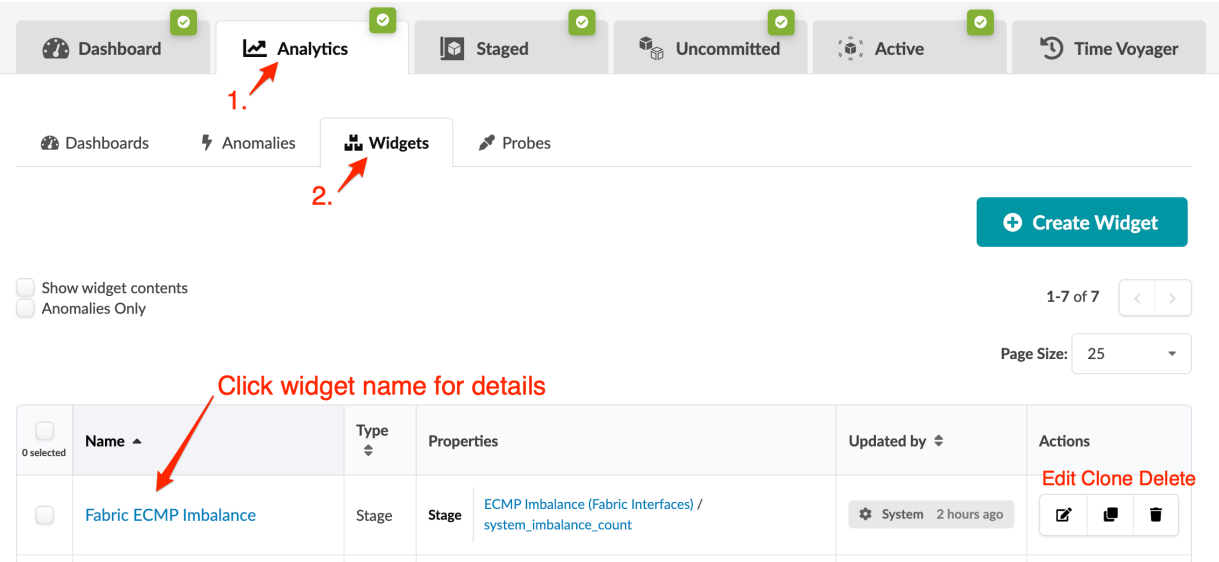


NOTE: The "blueprint dashboard" on page 274 shows a summary of all anomalies including those that IBA probes generated. Clicking the **All Probes** gauge on the dashboard takes you to a list of anomalies (Analytics > Anomalies).

Widgets Overview

Widgets generate data that are based on IBA "probes" on page 709. The widget type 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). You can view widgets by themselves or you can add them to "analytics dashboards" on page 702. You can create widgets before you create the dashboard or while you're creating it.

From the blueprint, navigate to **Analytics > Widgets** to go to the widgets table view. You can create, clone, edit and delete widgets.



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 table 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 | 708](#)
- [Create Stage Widget from Probes View | 708](#)

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 table 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 table 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 | 708](#)
- [Delete Widget | 709](#)

Edit Widget

You can modify auto-created widgets, although defaults should work in most cases. Modifying widgets affects any dashboards that they're used in.

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 table view.

Delete Widget

You can't delete a widget if it's used in a dashboard.

1. From the table 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 table view.

Probes

IN THIS SECTION

- [IBA Probes Overview | 709](#)

IBA Probes Overview

IN THIS SECTION

- [Processors | 710](#)
- [Ingestion Filters | 711](#)
- [IBA Collection Filter | 711](#)
- [IBA Filter Format | 711](#)

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 1119](#) 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 1077](#) 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 table view. To go to a probe's details, click its name. You can instantiate, create, clone, edit, delete, import, and export probes.

1.

2.

Click for options

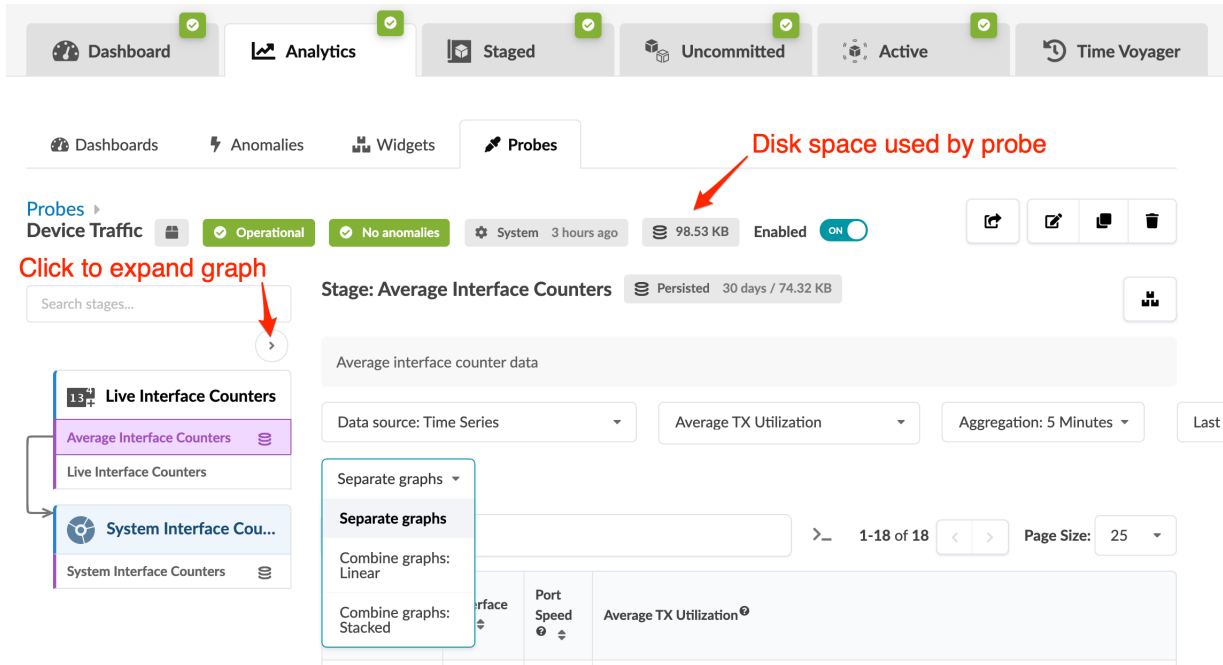
Click probe name to see details

| 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.

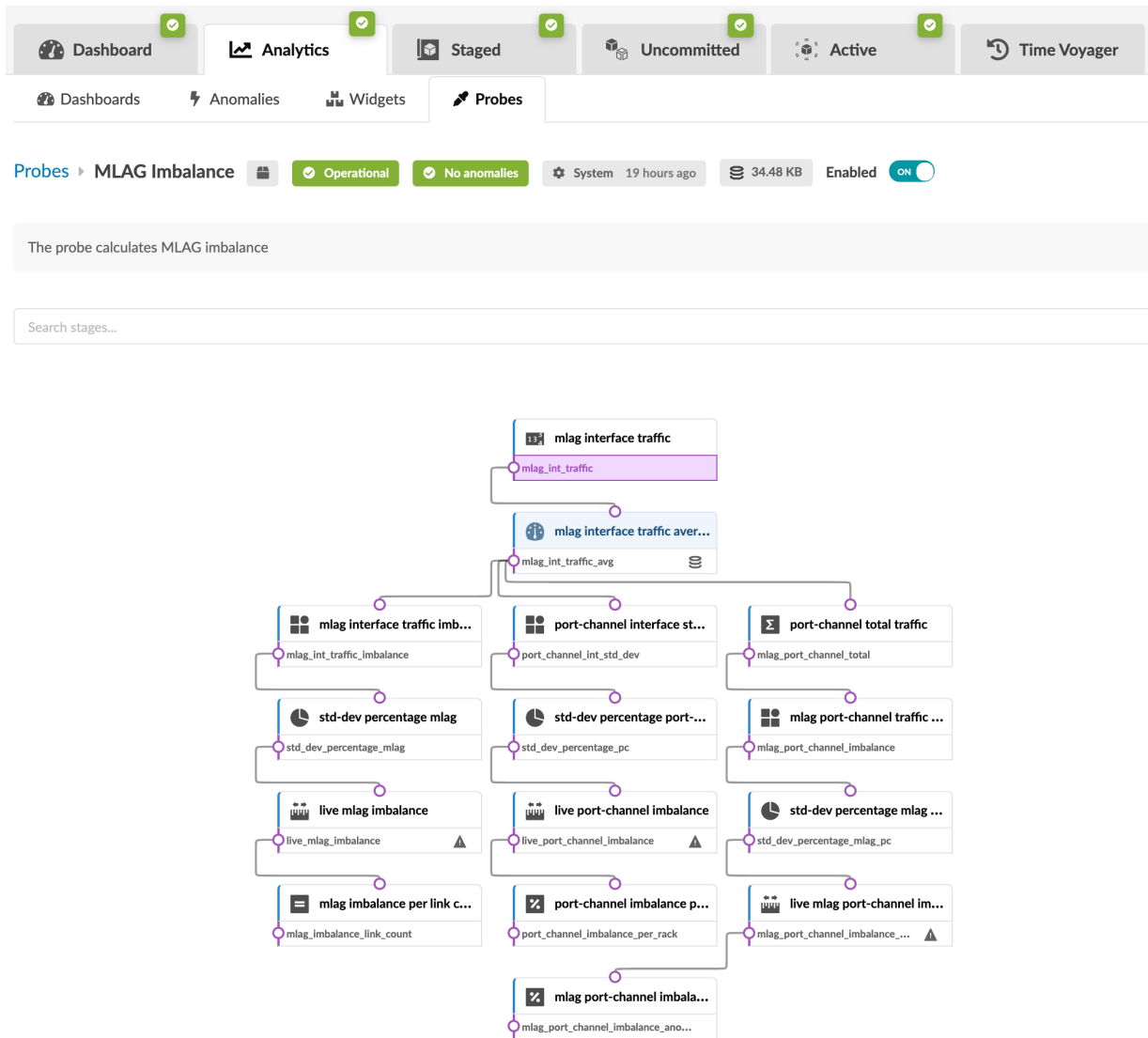


CAUTION: If the Apstra controller has insufficient disk space, older telemetry data files are deleted. To retain older telemetry data, you can increase capacity with ["Apstra VM Clusters"](#) on page 767.



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 999 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 1077](#) 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 table view.

Import / Export Probe

IN THIS SECTION

- [Import Probe | 716](#)
- [Export Probe | 716](#)

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 JSON 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 table 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 | 717](#)
- [Delete Probe | 717](#)

Edit Probe

If a widget is using a probe, editing the probe affects those widget(s) and related dashboard(s).

1. From the table 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 table 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 table 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 table view.

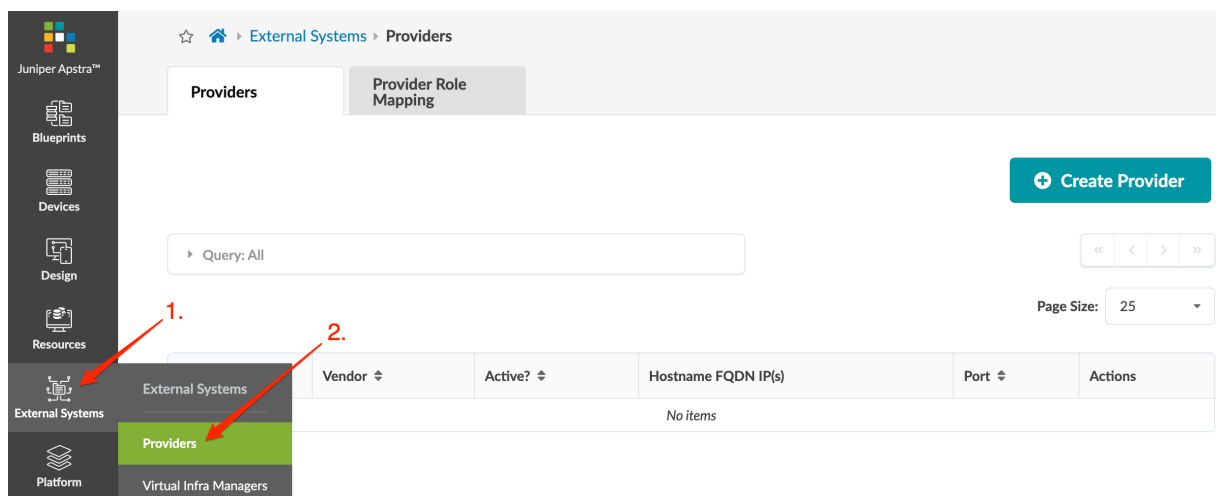
Providers (External Systems)

IN THIS SECTION

- [LDAP Provider | 719](#)
- [Active Directory Provider | 722](#)
- [TACACS+ Provider | 723](#)
- [RADIUS Provider | 725](#)
- [Edit / Delete Provider | 727](#)
- [Provider Role Map Overview | 728](#)
- [Create Provider Role Map | 729](#)
- [Edit / Delete Role Map | 729](#)

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 | 719](#)
- [Configure LDAP Provider | 721](#)

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 table 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 728](#) 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 | 722](#)

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 table view.

After configuring and activating a provider, you must ["map" on page 728](#) 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 | 724](#)
- [Configure TACACS+ Provider | 724](#)

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 table 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 728](#) that provider to one or more user roles to give access permissions to users with those roles.

RADIUS Provider

IN THIS SECTION

- [RADIUS Limitations | 725](#)
- [Create RADIUS Provider | 726](#)

Remote Authentication Dial-In User Service (RADIUS). See below for limitations.

RADIUS Limitations

- All password changes should be done on the RADIUS server. You cannot change the user's password from Apstra once you enable the External Provider.
- RADIUS authentication does not control Linux user login via SSH.
- If the user group is changed on the RADIUS server, you will need to change role-mapping in Apstra accordingly.
- 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 728](#) that provider to one or more user roles to give permissions to users with those roles.

Edit / Delete Provider

IN THIS SECTION

- [Edit Provider | 727](#)
- [Delete Provider | 728](#)

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 table 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 table view.

Delete Provider

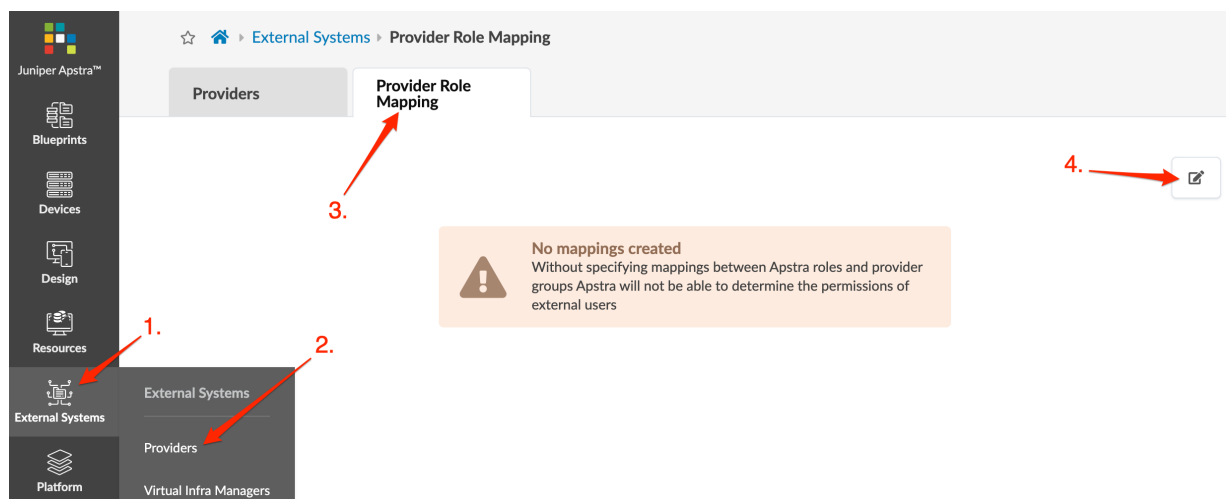
1. Either from the table view (External Systems > Providers) or the details view, click the **Delete** button for the provider to delete.
2. Click **Delete** to delete the provider and return to the table view.

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 | 730](#)
- [Delete Role Map | 730](#)

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 \(Platform\) | 731](#)
- [Security \(Platform\) | 746](#)
- [Syslog Configuration \(Platform\) | 754](#)
- [Receivers \(Platform\) | 761](#)
- [Global Statistics \(Platform\) | 764](#)
- [Event Log \(Platform\) | 765](#)
- [Apstra VM Clusters | 767](#)
- [Developers \(Platform\) | 778](#)
- [Juniper Technical Support | 824](#)

User/Role Management (Platform)

IN THIS SECTION

- [User Profile Management | 731](#)
- [User Role Management | 732](#)
- [User Profile Use Cases | 735](#)
- [Create User Profile | 739](#)
- [Change Apstra GUI User Password | 739](#)
- [Log Out User | 739](#)
- [Edit / Delete User Profile | 739](#)
- [User Role Use Cases | 740](#)
- [Create User Role | 745](#)
- [Edit / Delete User Role | 745](#)

User Profile Management

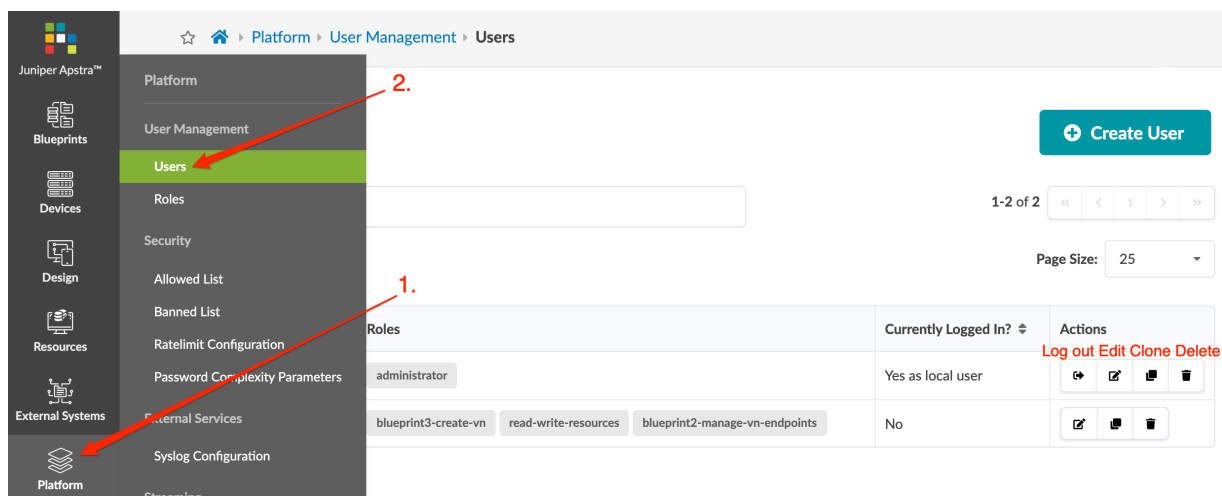
User profiles include the following details and options:

- Username
- First Name (optional)
- Last Name (optional)
- Email (optional)
- Password
- Roles



NOTE: Creating a user in the Apstra GUI does not provide that user access to the Apstra platform via SSH. To access the Apstra platform via SSH, you must create a local Linux system user.

From the left navigation menu in the Apstra GUI, navigate to **Platform > User Management > Users** to go to user profiles.



User Role Management

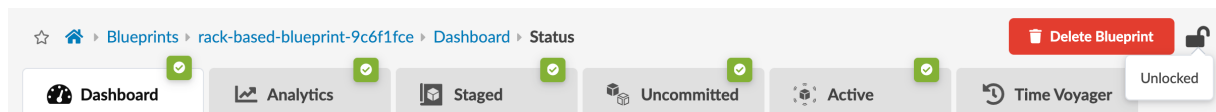
Users with the **administrator** role can create, clone, edit and delete user roles (which are assigned to user profiles). These roles can also be **"mapped"** on page 728 to external groups used by authentication providers such as LDAP, Active Directory, TACACS+, and RADIUS.

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.

For example, a user assigned the role **Manage generic systems** can add generic systems, copy existing generics, add links to generic systems, add links to leaf devices, and update node tags. A user assigned the role **Manage racks and links** can perform all those operations plus they can change rack speeds and delete links. A user with the **Manage racks and links** role essentially has permissions for all FE/FFE operations. If you want to restrict a user to physical server operations only, assign them the **Manage generic systems** role, and not the **Manage racks and links** role.

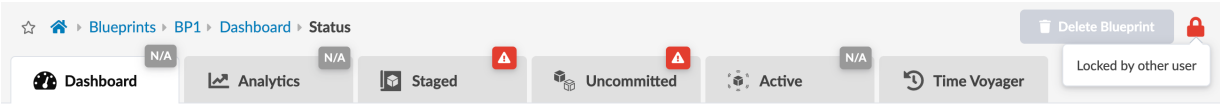
The blueprint locking feature prevents restricted users (based on their roles) from making changes that effectively are not permitted. In particular, a restricted user should not be able to commit changes made by another user.

If a blueprint has no changes to commit, it is unlocked.

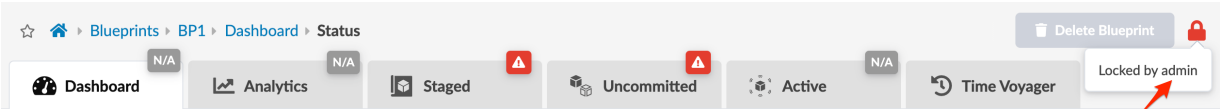


If you have permission (based on the your assigned roles) to create/update/delete virtual networks, for example, and another user has made uncommitted changes to the blueprint, the blueprint is locked. You can't create/update/delete virtual networks until the changes are committed or reverted by the locking

user who made the uncommitted changes, unless you are the locking user.



If you have permission (based on your assigned roles) to see the name of the user who created the pending changes, the name is displayed.



An admin user who has "Write/Commit Blueprints" permissions can make any changes to, apply changes for, revert changes for any blueprint.



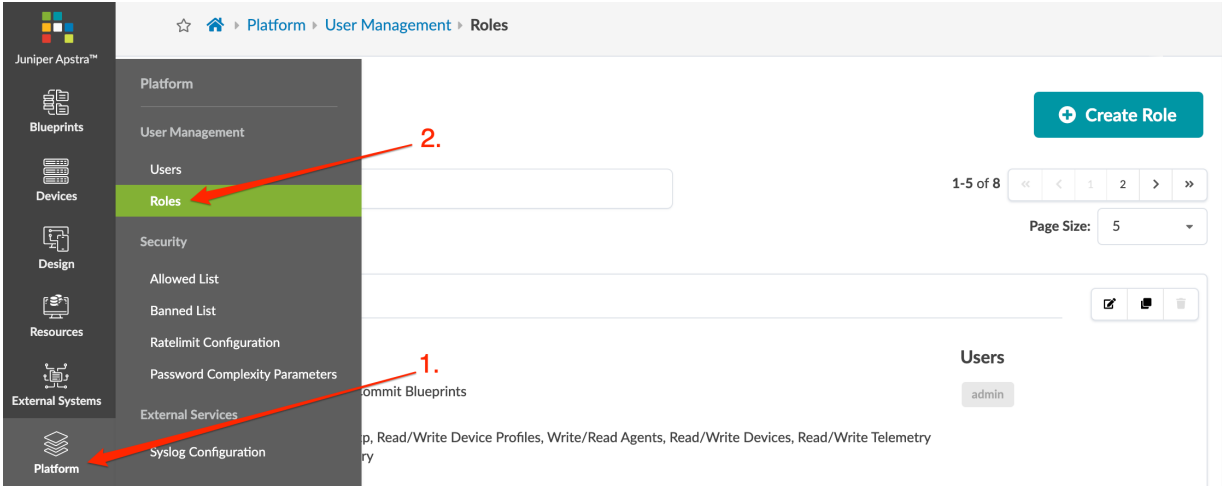
User roles include the following details and options:

| Parameter | Description |
|---|--|
| Name | role name |
| Type | global permission or per-blueprint permissions |
| Global Permissions (read, write, commit, delete, as applicable) | blueprints, connectivity templates, agents, chassis profiles, device profiles, devices, linecard profiles, telemetry service registry, ztp, config templates, configlets, interface maps, logical devices, port aliases, property sets, rack types, tags templates, ASN pools, Integer pools (new in Apstra version 4.1.2), IP pools, IPv6 pools, VNI pools, audit config, audit events, roles, security config, users, AAA providers, virtual infra manager, exempt Juniper Apstra cluster management read-only mode, Juniper Apstra cluster management, Juniper Apstra metric logs, streaming, SysDB data, port setting schema |

(Continued)

| Parameter | Description |
|---------------------------|--|
| Per-Blueprint Permissions | <ul style="list-style-type: none"> • Scope <ul style="list-style-type: none"> • All blueprints • Selected blueprints • Permissions <ul style="list-style-type: none"> • Read blueprint • Make any changes to staging blueprint (includes managing VNs and their endpoints) • Commit changes • Read information about user who locked blueprint • Datacenter-specific: Manage racks and links • Datacenter-specific: Manage generic systems • Datacenter-specific: Manage virtual networks (includes managing VN endpoints) • Datacenter-specific: Manage virtual network endpoints • Freeform-specific: Manage property sets (new in Apstra version 4.1.2) • Freeform-specific: Manage resources (new in Apstra version 4.1.2) |

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 can't be modified.



User Profile Use Cases

IN THIS SECTION

- [Use Case Overview | 735](#)

Use Case Overview

IN THIS SECTION

- [Use Case 1: Create Virtual Networks Only \(not Including Allocating Resources\) | 737](#)
- [Use Case 2: Create Virtual Networks and Allocate Resources | 738](#)

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 740](#).

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 740](#).

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

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 752.](#))
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 Apstra GUI 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 752.](#))
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 | 739](#)
- [Delete User Profile | 740](#)

Edit User Profile

1. Either from the table 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 table 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 table 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 | 740](#)

Use Cases Overview

IN THIS SECTION

- [Use Case 1: Read, Write and Commit Specific Blueprints | 741](#)
- [Use Case 2: Manage VN Endpoints on Specific Blueprints | 742](#)
- [Use Case 3: Create Virtual Networks \(not Including Allocating Resources\) | 743](#)
- [Use Case 3A: Create Virtual Networks and Allocate Resources | 744](#)
- [Use Case 4: Read and Write Resources on All Blueprints | 745](#)

The following use cases are described below. (Screenshots are from Apstra version 4.1.1 which look slightly different from version 4.1.2.)

☆
🏠
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 | 746

Delete User Role | 746

Edit User Role

The four predefined user roles (administrator, device_ztp, user, viewer) cannot be modified.

1. Either from the table 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 table view.

Delete User Role

The four predefined user roles (administrator, device_ztp, user, viewer) cannot be deleted.

1. Either from the table 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 table 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 | 747](#)
- [Banned List | 748](#)
- [ACL Rules | 749](#)
- [Rate Limit Configuration | 751](#)
- [Edit Password Complexity Requirements | 752](#)

Allowed List

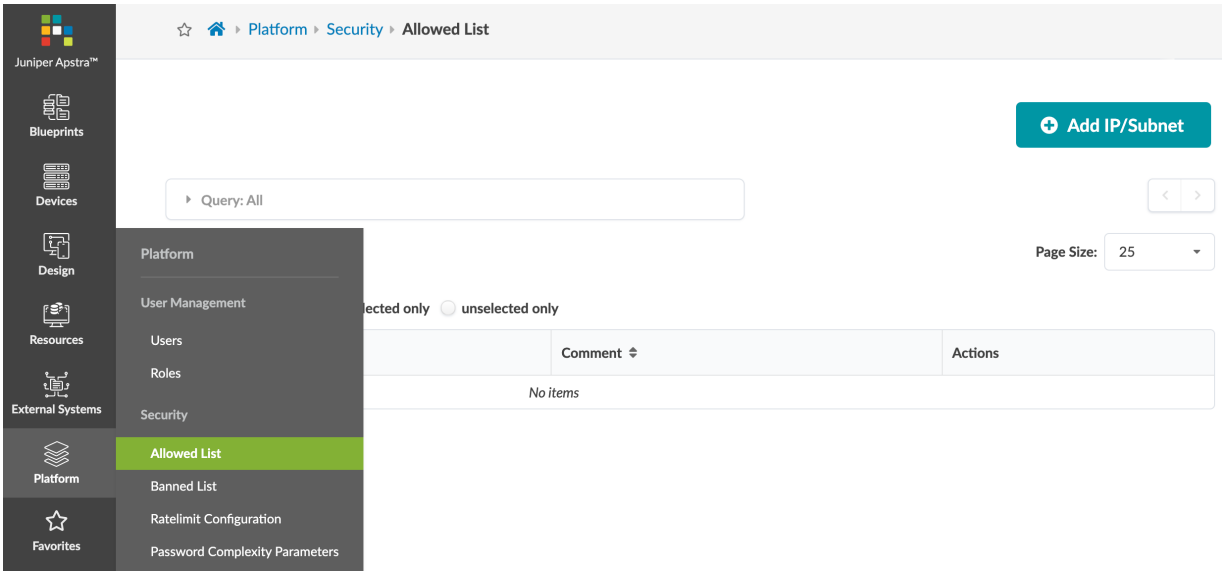
IN THIS SECTION

- Allowed List Overview | 747
- Add IP/Subnet to Allowed List | 747
- Edit IP/Subnet to Allowed List | 748
- Delete IP/Subnet from Allowed List | 748

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. Changes to the allowed list are recorded in the event log (Platform > Event Log).

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

- From the left navigation menu, navigate to **Platform > Security > Allowed List** and click **Add IP/Subnet**.
- Enter an IP address or subnet, and a comment.
- 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 table 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 table 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 table view.

Banned List

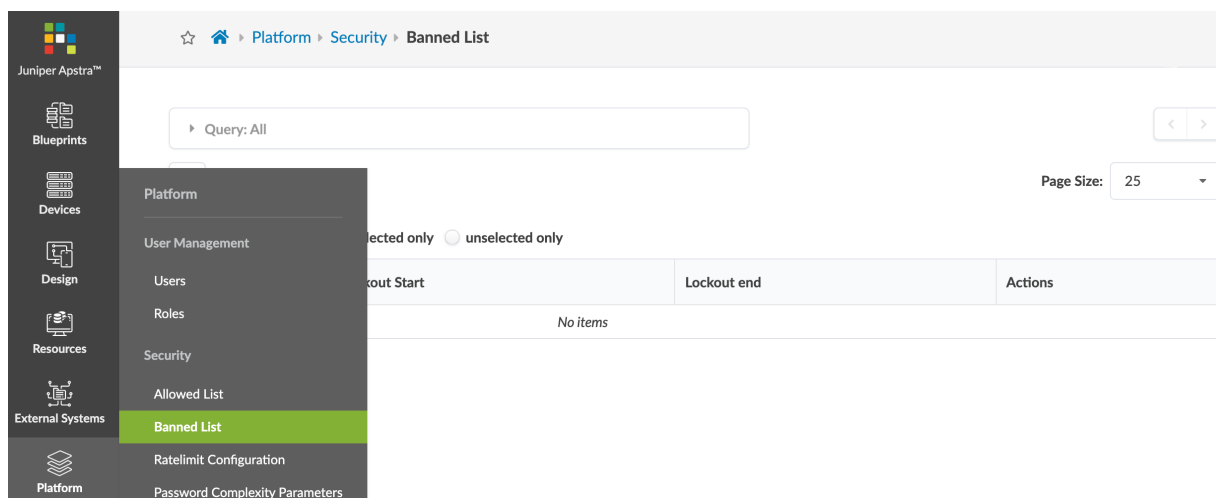
IN THIS SECTION

- [Banned List Overview | 748](#)
- [Delete IP/Subnet from Banned List | 749](#)

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. Changes to the banned list are recorded in the event log (Platform > Event Log).

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.

ACL Rules

IN THIS SECTION

- [Overview | 749](#)
- [Enable / Disable ACL Rules | 750](#)
- [Add ACL Rule | 750](#)
- [Edit ACL Rule | 750](#)
- [Delete ACL Rule | 751](#)

Overview

Subnet-based access control for Apstra GUI access (whitelisting) is introduced in Apstra version 4.1.2 as part of a platform security enhancement. You can configure Access Control List (ACL) rules for IPv4 networks. (IPv6 is not supported on the Apstra web framework.) When you create and enable rules, the rules are automatically sorted from more specific to less specific, and IP addresses are checked against

them in that order. If the rule allows access to a subnet, any IP address within that subnet is allowed access. If the rule denies access to a subnet, any IP address within that subnet is denied access.

Juniper Apstra™

☆ 🏠 > Platform > Security > ACL

Blueprints

Devices

Design

Resources

External Systems

Platform

Favorites

Enable / Disable

Query: All

1-1 of 1

Page Size: 25

Enabled?

Access control rules are disabled. You can enable them with the toggle above.

Overlapping subnets are automatically ordered and applied from most specific to least specific.

| IPv4 subnet | Policy | Comment | Actions |
|-------------|--------|-----------|-------------|
| 0.0.0.0/0 | Allow | Allow all | Edit Delete |

Enable / Disable ACL Rules

Access control list rules are disabled by default.

If you enable ACL rules, make sure you always add a rule to allow access to a subnet that your IP address is a part of, so you don't lock yourself out.

If you enable ACL rules, and the default rule (0.0.0.0/0) is set to deny, the Apstra UI and system agents can't make necessary REST API calls to the Apstra controller unless you add a rule to allow access from loopback (127.0.0.0/8) and docker (172.17.0.0/16) networks.

1. From the left navigation menu, navigate to **Platform > Security > ACL** to go to the table view.
2. Click the toggle to enable or disable the rules, as applicable.

Add ACL Rule

1. From the left navigation menu, navigate to **Platform > Security > ACL** and click **Add ACL rule**.
2. Enter an IP subnet and select whether to allow or deny access to IP addresses within that subnet. You also have the option of adding a comment.
3. Click **Create** to create the rule and return to the table view.

Edit ACL Rule

1. From the left navigation menu, navigate to **Platform > Security > ACL** and click the **Edit** button for the rule to edit.
2. Change the policy, as applicable. You also have the option of adding/editing/deleting a comment.

- 3. Click **Update** to change the rule and return to the table view.

Delete ACL Rule

So that an IP address eventually matches to a subnet, 0.0.0.0/0 can't be deleted..

- 1. From the left navigation menu, navigate to **Platform > Security > ACL** and click the **Delete** button for the rule to delete.
- 2. Click **Delete** to delete the rule and return to the table view.

Rate Limit Configuration

IN THIS SECTION

- [Rate Limit Configuration Overview | 751](#)
- [Edit Rate Limit Configuration | 751](#)

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™

Blueprints

Devices

Design

Resources

External Systems

Platform

User Management

Users

Roles

Security

Allowed List

Banned List

Ratelimit Configuration

Password Complexity Parameters

☆ 🏠 > Platform > Security > Ratelimit Configuration

Edit

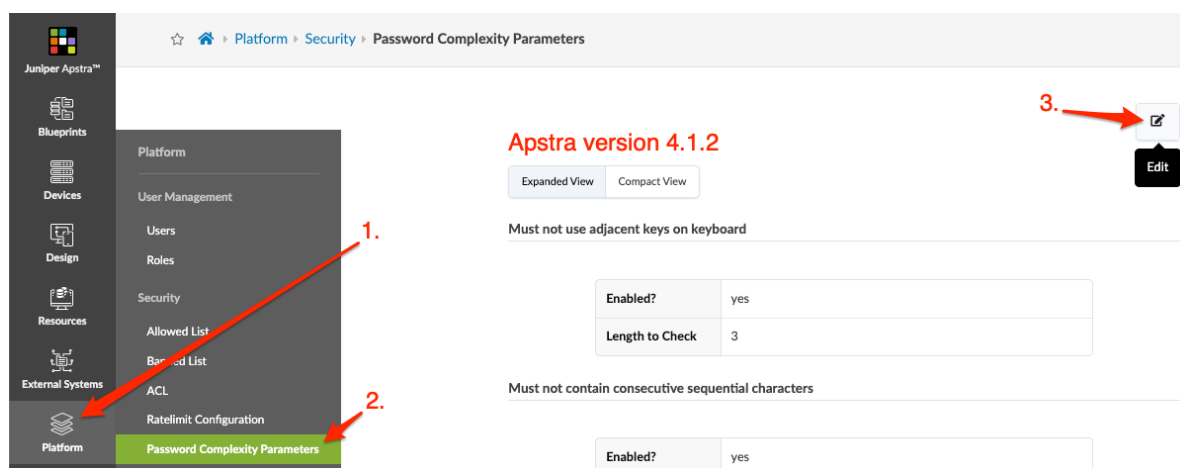
| | |
|--------------------------|-----|
| Lockout period (sec) | 180 |
| Observation window (sec) | 60 |
| Number of observations | 5 |

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

When you update password complexity requirements, the requirements are applied when you subsequently create or edit passwords. Existing passwords are not affected until you change them.

1. From the left navigation menu, navigate to **Platform > Security > Password Complexity Parameters** and click the **Edit** button (top-right). The screenshot below is for Apstra version 4.1.2. Previous versions have fewer complexity options.



2. Add, change and/or delete requirements, as applicable. Different Apstra versions have different options as shown in the list and screenshots below:
 - Password History Length (4.1.2 only) - User is not allowed to re-use a certain number of previous passwords (including the current one). For example, if you don't want the user to use their previous two passwords, you would enter 3 in this field.
 - Must not use adjacent keys on keyboard (4.1.2 only)
 - Must not contain consecutive sequential characters (4.1.2 only)
 - Must not contain repeat of the same character (4.1.2 only)
 - Must not be the same as username (4.1.2 only)
 - Length should be at least 9 (default)
 - Must contain uppercase letter
 - Must contain lowercase letter
 - Must contain digit
 - Must contain special character

For regular expressions:

- 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.

Apstra version 4.1.2 Password Complexity Parameters

Edit Password Complexity Parameters

Changes will be applied to the newly created passwords only. Existing passwords won't be affected.

Password History Length

3

Must not use adjacent keys on keyboard

Allowed length

3

Must not contain consecutive sequential characters

Allowed length

3

Must not contain repeat of the same character

Allowed length

3

Must not be the same as username

Regular Expression

^[9,]

Error Message

Length should be at least 9

X

Regular Expression

[A-Z]

Error Message

Must contain uppercase letter

X

Regular Expression

[a-z]

Error Message

Must contain lowercase letter

X

Regular Expression

[0-9]

Error Message

Must contain digit

X

Regular Expression

[^A-Za-z0-9]

Error Message

Must contain special character


X

Add

Update

Apstra versions 4.1.1 and 4.1.0 Password Complexity Parameters

Edit Password Complexity Parameters

 The changes will be applied to the newly created passwords only, the existing passwords won't be affected

| Regular Expression | Error Message | |
|--|---|---|
| <input type="text" value="^(?=.{8,}) "/> | <input type="text" value="Length should be at least 8"/> | ✖ |
| <input type="text" value="^(?=.*[A-Z])"/> | <input type="text" value="Must contain uppercase letter"/> | ✖ |
| <input type="text" value="^(?=.*[a-z])"/> | <input type="text" value="Must contain lowercase letter"/> | ✖ |
| <input type="text" value="^(?=.*[0-9])"/> | <input type="text" value="Must contain digit"/> | ✖ |
| <input ,.="" \`>?])"="" type="text" value="^(?=.*[!@#\$%^&*~.-_+{} ;':"/> | <input type="text" value="Must contain special character"/> | ✖ |

- Click **Update** to complete the change and close the dialog. When you create or update passwords, the new requirements will take effect.

Syslog Configuration (Platform)

IN THIS SECTION

- [Syslog Overview | 754](#)
- [Create Syslog Config | 760](#)
- [Edit Syslog Config | 760](#)
- [Delete Syslog Config | 760](#)

Syslog Overview

System Log (syslog) is a running list of everything that's going on in your system. You can use these logs to audit events or review anomalies. You can configure syslog to send messages for specific types of systems (facilities) to external syslog servers. (You can also ["export event logs to a CSV file" on page 767.](#))

Syslog configuration includes the following details:

| Name | Description |
|------------|---|
| IP Address | The remote syslog server IP address or hostname |
| Port | The remote syslog server port |
| Protocol | UDP or TCP |

(Continued)

| Name | Description |
|----------|---|
| Facility | <p>The type of system that's logging the messages</p> <p>Facilities are mapped to Apstra syslogs as follows:</p> <ul style="list-style-type: none"> • 0 - kern - kernal messages • 1 - user - user-level messages • 2 - mail - mail system • 3 - daemon - system daemons • 4 - auth - security/authentication messages • 5 - syslog - messages generated internally by syslogd • 6 - lpr - line printer subsystem • 7 - news - network news subsystem • 8 - uucp - UUCP subsystem • 10 - authpriv - security/authentication messages • 11 - ftp - FTP daemon • 15 - cron - Cron subsystem • 16 - local0 - locally used facilities • 17 - local1 - locally used facilities • 18 - local2 - locally used facilities • 19 - local3 - locally used facilities • 20 - local4 - locally used facilities • 21 - local5 - locally used facilities • 22 - local6 - locally used facilities • 23 - local7 - locally used facilities |

(Continued)

| Name | Description |
|-----------|---|
| Time Zone | The syslog message time zone. If you have proper time zone translation, you won't need to synch the system time zone (or Docker time zone) with your external syslog server. Rather than assuming the message time is in Zulu/UTC-0, the time zone translation needs to append the correct time zone information to the timestamp. Then, you can better correlate Apstra events in your external message systems. |

Syslog messages follow Common Event Format (CEF) conventions as shown below:



NOTE: {host} is the the Apstra server hostname. If you want to change the hostname, you must use the procedure on the ["Change Apstra Server Hostname" on page 858](#) page. If you change the hostname with any other method, the new hostname won't be included in syslog entries.

AOS Log Format:

```
'{timestamp} {host}'
'CEF:{version}|{device_vendor}|{device_product}|{device_version}|'
'{device_event_class_id}|{name}|{severity}|{extension}'
```

Where:

```
{version}      : CEF version, currently 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}      : "5" for audit logs, "10" 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

blueprint_label : Name of the blueprint the anomaly was raised in.
 timestamp : Unix timestamp when the Anomaly was raised.
 origin_name : Serial Number of the device the anomaly affects.
 alert : The value is a JSON Payload with the actual anomaly (see Alert JSON Payload below)
 origin_hostname : Hostname of the device the anomaly affects. It can be AOSHOST, an empty string if the hostname could not be determined or a valid value.
 device_hostname : Hostname of the device the anomaly affects or <device hostname unknown> if a hostname could not be determined
 origin_role : Role of the device the anomaly affects.

Alert JSON Payload:

<ALERT TYPE>_alert: Contains a JSON payload with key-value pair of information pertaining to the alert. Here <ALERT TYPE>_alert can be valid anomaly/alert names such as hostname_alert, probe_alert, liveness_alert etc.

id : UUID of the anomaly.
 first_seen : Unix timestamp when the Anomaly was raised for the first time.
 raised : True when anomaly is present, False when it is cleared.
 severity : The severity level of the anomaly. Set to 3 for critical, 2 for high, 1 for medium and 0 for low.

Audit Log Format:

cat : Activity performed. Valid values: "Login",
 "Logout", "BlueprintCommit", "BlueprintRevert", "BlueprintRollback",
 "BlueprintDelete", "DeviceConfigChange",
 "OperationModeChangeToMaintenance", "OperationModeChangeToNormal", "OperationModeChangeToReadOnly",
 "RatelimitExceptionAdd", "RatelimitExceptionDelete",
 "RatelimitClear", "SystemChangeApiOperationModeToMaintenance", "SystemChangeApiOperationModeToNormal",
 "UserCreate", "UserUpdate", "UserDelete",
 "SyslogCreate", "SyslogUpdate", "SyslogDelete", "AuthAclEnable", "AuthAclDisable", "AuthAclRuleAdd", "AuthAclRuleUpdate" and "AuthAclRuleDelete".
 src : Source IP of the client making HTTP requests to perform the activity.
 suser : Who performed the activity.
 act : Outcome of the activity - free-form string. In the case when the activity was performed successfully, the value stored is "Success". In case of error, include error string.
 Ex: Unauthorized

cs1Label : The string “Blueprint Name”. Only exists if activity is associated with a blueprint (optional)
 cs1 : Name of the blueprint on which action was taken. Only exists if activity is associated with a blueprint (optional)
 cs2Label : The string “Blueprint ID”. Only exists if activity is associated with a blueprint (optional)
 cs2 : Id of the blueprint on which action was taken. Only exists if activity is associated with a blueprint (optional)
 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. Only exists if activity is associated with a device such as for “DeviceConfigChange” (optional)
 deviceConfig : Config that is pushed and applied on the device where “#012” is used to indicate a line break to log collectors and parsers. Only exists if activity is associated with a device such as for “DeviceConfigChange” (optional)

Example of Audit Syslog Message:

```

Jan 31 03:11:01 aos-server - 2023-01-31T03:11:01.699190+0000 aos-server
CEF:0|Apstra|AOS|4.1.2-269|100|Audit event|5|cat=Logout src=172.24.212.62 suser=admin act=Success

Jan 31 03:11:01 aos-server - 2023-01-31T03:11:01.699190+0000 aos-server
CEF:0|Apstra|AOS|4.1.2-269|100|Audit event|5|cat=BlueprintCommit src=172.24.212.62 suser=admin
act=Success cs1Label=Blueprint Name
cs1=rack-based-blueprint-33ded50f cs2Label=Blueprint ID cs2=rack-based-blueprint-33ded50f
  
```

Example of Anomaly Syslog Message:

```

Jan 31 03:11:01 aos-server - 2023-01-31T03:11:01.699190+0000 aos-server
CEF:0|Apstra|AOS|4.1.2-269|101|Alert|10|msg={u'blueprint_label': u'rack-based-
blueprint-33ded50f', u'timestamp': 1679002758562407, u'origin_name':
u'time_series', u'alert': {u'probe_alert': {u'expected_int_max': 99, u'stage_name':
u'leaf_match_perc_range', u'probe_label': u'leaf_to_spine_interface_statuses',
u'actual_int': 83, u'probe_id': u'60b03bb0-0e22-4a6d-b32d-e15085149b7b', u'key_value_pairs': [],
u'item_id': u'1', u'expected_int': -9223372036854775808},
u'first_seen': 1679002758562121, u'raised': False, u'severity': 3, u'id': u'02a17b60-cc3e-4afb-
baba-733a8c654df6'}, u'origin_hostname': u'AOSHOST',
'device_hostname': '<device hostname unknown>', u'origin_role': u''}
  
```

```
Jan 31 03:11:01 aos-server - 2023-01-31T03:11:01.699190+0000 aos-server
CEF:0|Apstra|AOS|4.1.2-269|101|Alert|10|msg={u'blueprint_label': u'rack-based-
blueprint-33ded50f', u'timestamp': 1679002754682990, u'origin_name':
u'50540015FA9D', u'alert': {u'first_seen': 1679002749600167, u'raised': False, u'severity': 3,
u'hostname_alert': {u'expected_hostname': u'leaf-3',
u'actual_hostname': u''}, u'id': u'0457a759-7d3a-4bf8-97e8-e13e518cf267'}, u'origin_hostname':
u'', 'device_hostname': '<device hostname unknown>', u'origin_role': u'leaf'}
```

From the left navigation menu, navigate to **Platform > External Services > Syslog Configuration** to see configurations. You can create, clone, edit and delete syslog configurations.

Create Syslog Config

1. From the left navigation menu, navigate to **Platform > External Services > Syslog Configuration** and click **Create Syslog Config** (top-right).
2. Configure the Syslog server. (See overview above for details.)
3. Click **Create** to save the configuration and return to the table 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.

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 table 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 table view.

Receivers (Platform)

IN THIS SECTION

- [Streaming Receivers Overview | 761](#)
- [Create Receiver | 762](#)
- [Delete Receiver | 762](#)
- [Configure Receivers Using Telegraf Plugin | 762](#)

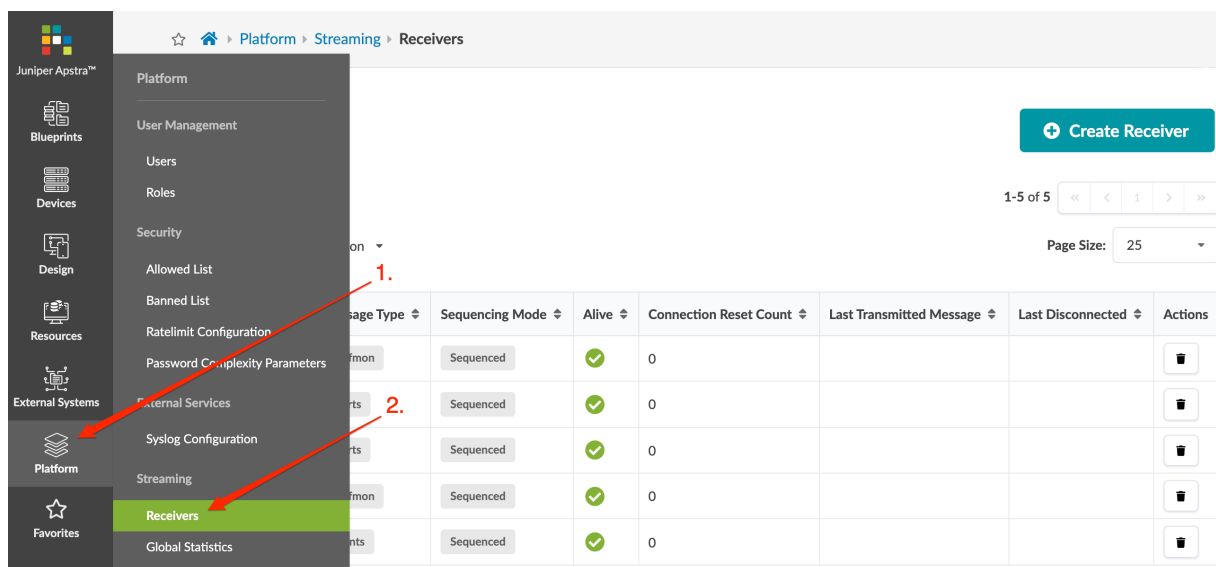
Streaming Receivers Overview

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.



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 table 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 table 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 902](#). 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.

The screenshot shows the Juniper Apstra web interface. The left navigation menu has the 'Platform' item highlighted. The main content area shows the 'Global Statistics' page. The breadcrumb navigation at the top indicates the path: Platform > Streaming > Global Statistics. The table below displays statistics for alerts, events, and perfmon.

| | 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 | 765](#)
- [Export Event Log to CSV File | 767](#)
- [Send Event Log to External System | 767](#)

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) (Login)
- User logout (Logout)
- User creation, by creating or cloning (UserCreate) (new in Apstra version 4.1.2)
- User Edit (UserUpdate) (new in Apstra version 4.1.2)
- User Delete (UserDelete) (new in Apstra version 4.1.2)
- Blueprint commit (applies changes from staged to active blueprint)
- Blueprint revert (discards changes in staged blueprint)
- Blueprint rollback (rolls back the staged blueprint to a previous version)
- Blueprint deletion
- Device Config change
- Operation Mode changed by user (maintenance, normal, read-only)
- Operation Mode changed by system (new in Apstra version 4.1.2)
 - Normal - when disk usage and memory is under the utilization threshold, the operation mode is in read/write mode
 - Maintenance - when utilization threshold is surpassed, the system moves API layer to read-only mode
- Changes to login banned/allowed list (new in Apstra version 4.1.1)

- Syslog Configuration creation, by creating or cloning (SyslogCreate) (new in Apstra version 4.1.2)
- Syslog Edit (SyslogUpdate) (new in Apstra version 4.1.2)
- Syslog Delete (SyslogDelete) (new in Apstra version 4.1.2)

Each event includes the following information which is searchable and sortable:

- Time - when the event occurred (hover 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.

Juniper Apstra™

Platform

User Management

Users

Roles

Security

Allowed List

Banned List

Ratelimit Configuration

Password Complexity Parameters

External Services

Syslog Configuration

Streaming

Receivers

Global Statistics

Event Log

Export to CSV

1-25 of 300

Page Size: 25

| Source 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 754.

Apstra VM Clusters

IN THIS SECTION

- [Apstra Cluster Nodes | 768](#)
- [Apstra Cluster Management | 775](#)
- [Change Cluster Application Memory Usage \(API\) | 777](#)

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 offbox agents, or if you are 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.

Apstra Cluster Nodes

IN THIS SECTION

- [Nodes Overview | 768](#)
- [Create Apstra Node | 773](#)
- [Edit Apstra Node | 774](#)
- [Delete Apstra Node | 774](#)

Nodes Overview

The Apstra controller acts as the cluster manager. When you add a worker VM to the main Apstra controller VM, it registers with the Apstra server VM through sysDB. It collects facts about the VM (such as core/memory/disk configuration and usage), and launches a local VM container. 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 19: Apstra VM Nodes Parameters

| Name | Description |
|---------|--|
| Address | IP address or Fully-Qualified Domain Name (FQDN) of the VM |
| Name | Apstra VM name, such as controller (the main Apstra controller node) or worker - iba (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 offbox 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. |

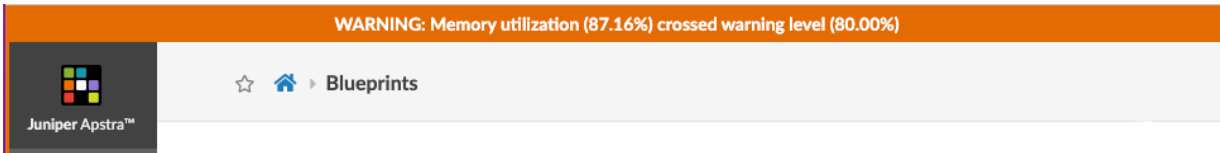
Table 19: Apstra VM Nodes Parameters (Continued)

| Name | Description |
|------------------|---|
| Capacity Score | <p>Apstra uses the capacity score for load balancing new containers across the cluster of available nodes. It's calculated in relation to the configured application weight of each container based on allocated memory.</p> <p>Example calculation - 64GB of memory allocated for the VM and an application weight of 250MB configured for offbox agents:</p> <ul style="list-style-type: none"> • Each offbox agent has a capacity score cost of 5 • $(64\text{GB} / 250\text{MB}) * 5$ capacity score of each offbox agent = 1280 total capacity score • Controller nodes have half the capacity score available due to overhead (1280 / 2 = 640 in above example) but worker nodes have the full capacity score available (1280 in above example) <p>The capacity score changes only if the memory allocated to the VM is changed, or if the application weight is changed.</p> |
| Containers Count | Number of containers |
| 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"> • Memory Usage (percentage) • CPU Usage (percentage) • Disk Usage - Current VM disk usage per logical volume (GB and percentage) • Container Service Usage - derived from the required resources and the size of the container. For example, if an offbox agent that needs 250 MB is running in a 500MB <i>worker</i> node, the container service usage is 50%. (An IBA container may require 1GB.) A <i>controller</i> node begins at 50% usage because it includes its own processing agents that perform controller-specific processing logic. |
| Containers | The containers running on the node and the resources that each container uses |

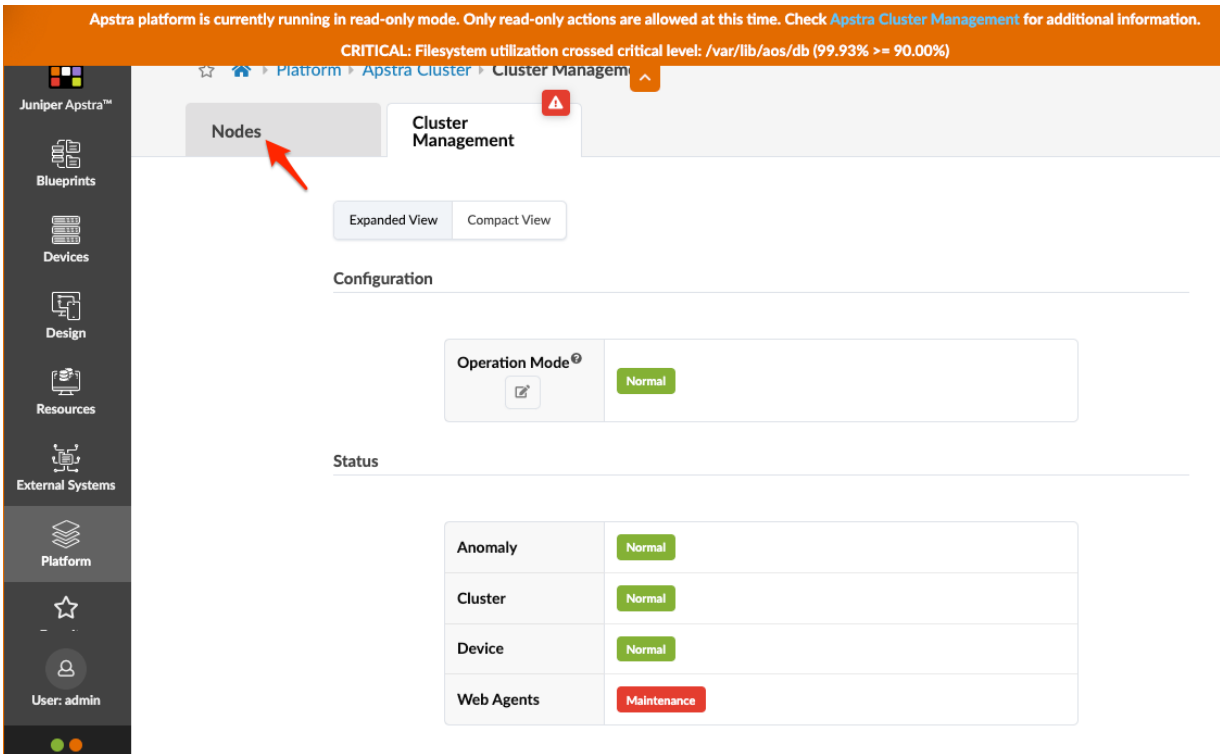
Table 19: Apstra VM Nodes Parameters *(Continued)*

| Name | Description |
|-----------------------|--|
| Username/ Password | Apstra Server VM SSH username/password login credentials |

* As of Apstra version 4.1.2, if memory utilization exceeds 80%, a warning message appears at the top of all GUI pages. This lets you know that you need to free up or add disk space and/or memory soon, to avoid a critical resource shortage.



If memory utilization exceeds 90%, a critical message appears at the top of all GUI pages. Before you can make any more changes to the fabric, you must address the shortage by adding disk space to the problematic filesystem(s) or by adding memory, as needed. You can click the link to go to **Apstra Cluster Management** for more information.



Click the **Nodes** tab, then click the IP address of the controller for details.

Juniper Apstra™

Blueprints

Devices

Design

Resources

External Systems

Platform

Favorites

Platform

Apstra Cluster

Nodes

controller

Nodes

Cluster Management

← back to list

Expanded View

Compact View

Static Configuration

| | |
|----------------|-----------------------------------|
| Address | 10.28.32.3 |
| Name | controller |
| Roles | controller |
| Tags | <div>liba</div> <div>offbox</div> |
| Capacity Score | 156 |
| CPU | 4 |

Errors

!

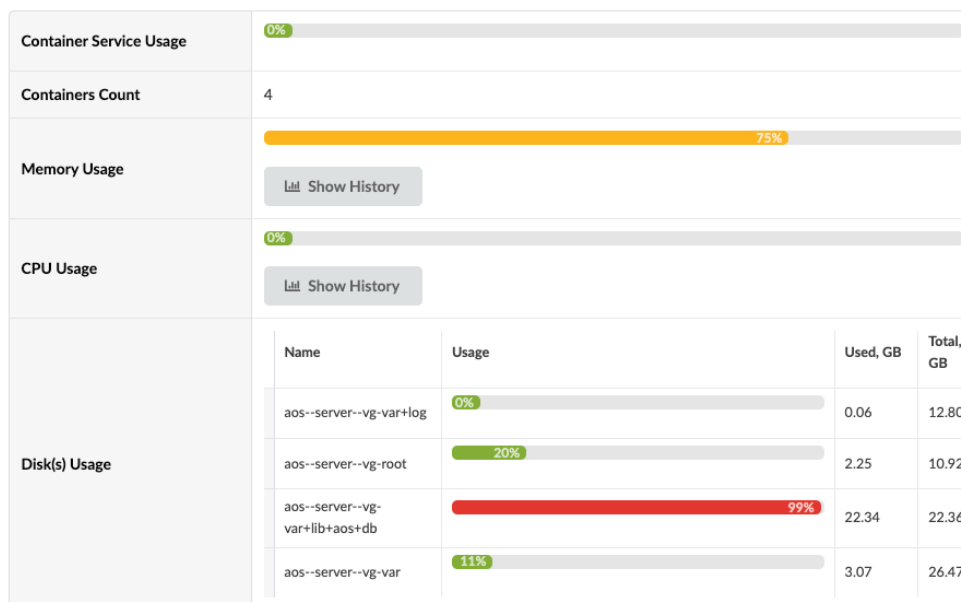
Configuration Error

!

Some partitions are almost full

Scroll down to see usage.

Usage



Some suggestions for recovering resources are as follows:

- Remove the **iba** tag from the controller VM so that IBA units are rescheduled to worker nodes, thus reducing both memory and disk space usage.
- Create worker nodes to spread out the load for IBA units and/or offbox device agents.

You can change the default thresholds that trigger warnings and critical messages. In the "[Apstra server configuration file](#)" on [page 1159](#) (/etc/aos/aos.conf) change the options for `system_operation_filesystem_thresholds` and/or `system_operation_memory_thresholds`. Then, send SIGHUP to the ClusterManager Agent. You can set disk space utilization thresholds on a per-filesystem basis. For example, you might want to be more conservative with `/var/lib/aos/db` which contains MainSysdb's persistence files and Time Voyager revisions, so crossing a lower usage threshold (such as 85%) triggers the read-only mode.

To access Apstra VMs, from the left navigation menu, navigate to **Platform > Apstra Cluster**. Click a node address to see its details. You can create, clone, edit and delete Apstra nodes.

1. Click on 'Apstra Cluster' in the left navigation menu.

2. Click on 'Nodes' in the left navigation menu.

| 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 | iba | 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 |

At the bottom left section of every page, you have continuous visibility of platform health. Green indicates the active state. Red indicates an issue, such as missing agent, the disk being in read only mode, or an agent rebooting (after the agent has rebooted, the status returns to active). If **IBA Services** or **Offbox Agents** is green, all containers are launched. If one of them is red, at least one container has failed. From any page, click one of the dots, then click a section for details. Clicking **Controller**, **IBA Services**, and **Offbox Agents** all take you to **Nodes** details.

1. Click on 'Configuration' in the left navigation menu.

2. Click for details

| Address | Name | State | Roles | Tags | Capacity Score | Containers Count | CPU | Memory Usage, Gb | CPU Usage | Disk Usage | Container Service |
|---------|------------|--------|------------|-------------|----------------|------------------|-----|------------------|-----------|------------|-------------------|
| | controller | ACTIVE | controller | iba, offbox | 120 | 7 | 2 | 7.56 (64%) | 15% | 7% | 25% |

Create Apstra Node

1. Follow the steps in the [Juniper Apstra Installation and Upgrade guide](#) to install Apstra software on the node to become an Apstra worker. The same Apstra VM image is used for both the Apstra controller and Apstra workers. Make sure to install the same Apstra version as the controller node.

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**.

When you've configured the new worker node in the Apstra controller, Apstra automatically converts the new VM to a worker node. Once the VM becomes a worker node, it can't be used as a controller node.

Edit Apstra Node

1. Either from the table 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 offbox 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 offbox 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.

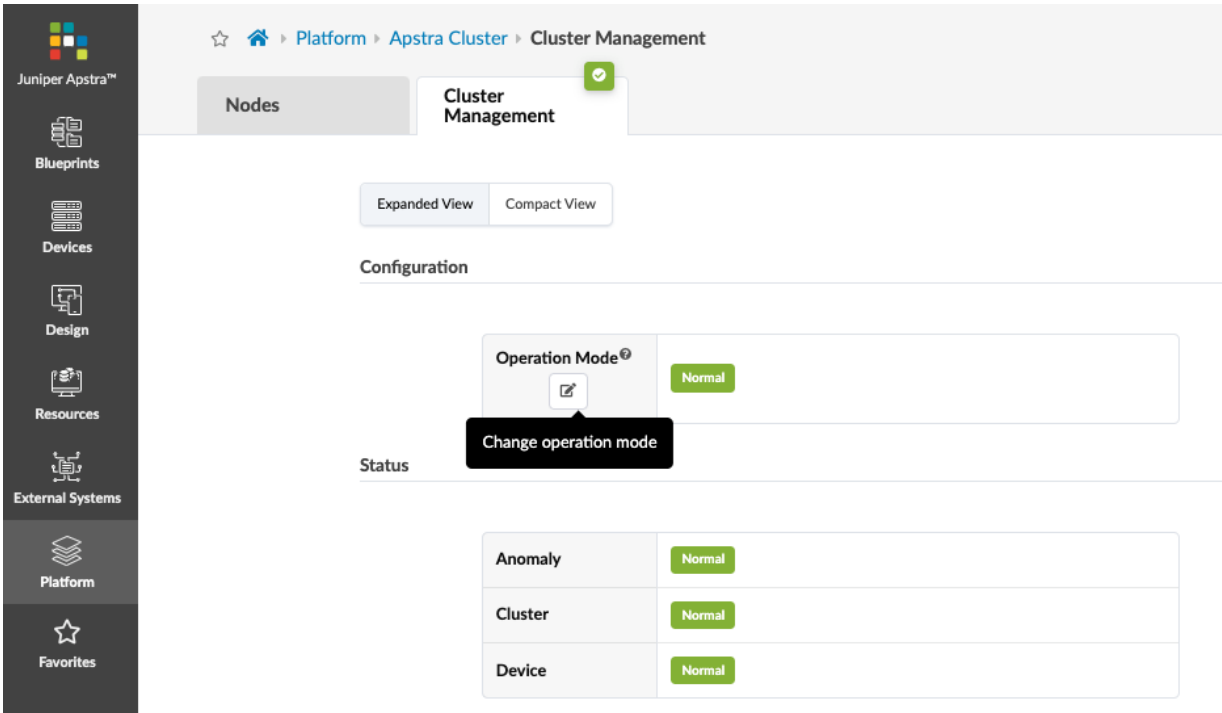


CAUTION: To prevent containers from being deleted, don't delete nodes with iba and/or offbox tags unless another node in the cluster has the same tags.

1. Either from the table 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.

Apstra Cluster Management

From the left navigation menu, navigate to **Platform > Apstra Cluster > Cluster Management** to go to Apstra cluster configuration and status.



Apstra admins may want to temporarily block all users (including themselves) from performing design and blueprint changes in the Apstra environment because they're troubleshooting something, or want to perform some maintenance operations on the Apstra server (backups, VM migration, VM OS updates and so on). Admins can change the **operation mode** (new in Apstra version 4.1.0) from **Normal** to **Read-only** to block users from API and WebUI (PUT/POST). By default, only admins have permission to enable/disable the read-only mode.

Apstra platform is currently running in read-only mode. Only read-only actions are allowed at this time. Check [Apstra Cluster Management](#) for additional information.

Platform > Apstra Cluster > Cluster Management

Nodes Cluster Management

Expanded View Compact View

Configuration

Operation Mode[®] Read-only

Status

| | |
|------------|-------------|
| Anomaly | Normal |
| Cluster | Normal |
| Device | Normal |
| Web Agents | Maintenance |

At the bottom left section of every page, you have continuous visibility of platform health. Green indicates the active state. Red indicates some kind of issue, such as a missing agent, the disk being in read only mode, or an agent rebooting (after the agent has rebooted, the status returns to active). From any page, click one of the dots, then click the section that you want details for. Clicking **Operation Mode** takes you to cluster management details.

Platform > Apstra Cluster > Nodes

Nodes Cluster Management

Query: All

| Address | Name | State | Roles | Tags | Capacity Score | Containers Count | CPU | Memory Usage, Gb | CPU Usage | Disk Usage | Container Service |
|---------|------|--------|------------|------------|----------------|------------------|-----|------------------|-----------|------------|-------------------|
| | | ACTIVE | controller | iba offbox | 120 | 7 | 2 | 7.56 (64%) | 15% | 7% | 25% |

Configuration

Operation Mode **Normal**

Controller Node

Controller **Active**

Application containers

IBA Services **Launched**

Offbox Agents **Launched**

1. Click for details

2. Click for details

Change Cluster Application Memory Usage (API)

You can change cluster application memory usage for offbox agents and Intent Based Analytics (IBA) via API. Make sure you've ["authenticated" on page 779](#) before attempting to make API calls.

If you're using Juniper offbox agents, increase memory allocation to 500 MB (from the 250 MB default). A single API call applies to all offbox agents.

1. From the left navigation menu in the Apstra GUI, navigate to **Platform > Developers** and click **REST API Documentation**.

The Swagger API developer tool for the Apstra environment appears.

2. Click **cluster**, click **GET /api/cluster/application-weight**, then click **Execute**.

The current values for **offbox** and **iba** appear in the response body.

3. Click **PUT / api/cluster/application-weight**, then click **Try it out**.

The parameters become editable.

cluster ▾

PUT /api/cluster/application-weight Update cluster scheduling parameters

Update the memory usage of different AOS applications that is used by AOS cluster to schedule containers.

Parameters Cancel

| Name | Description |
|----------------------------------|---|
| body * required (body) | <p>Example Value Model</p> <pre>{ "offbox": 0, "iba": 0 }</pre> <p>Cancel</p> <p>Parameter content type application/json ▾</p> <p>Execute</p> |

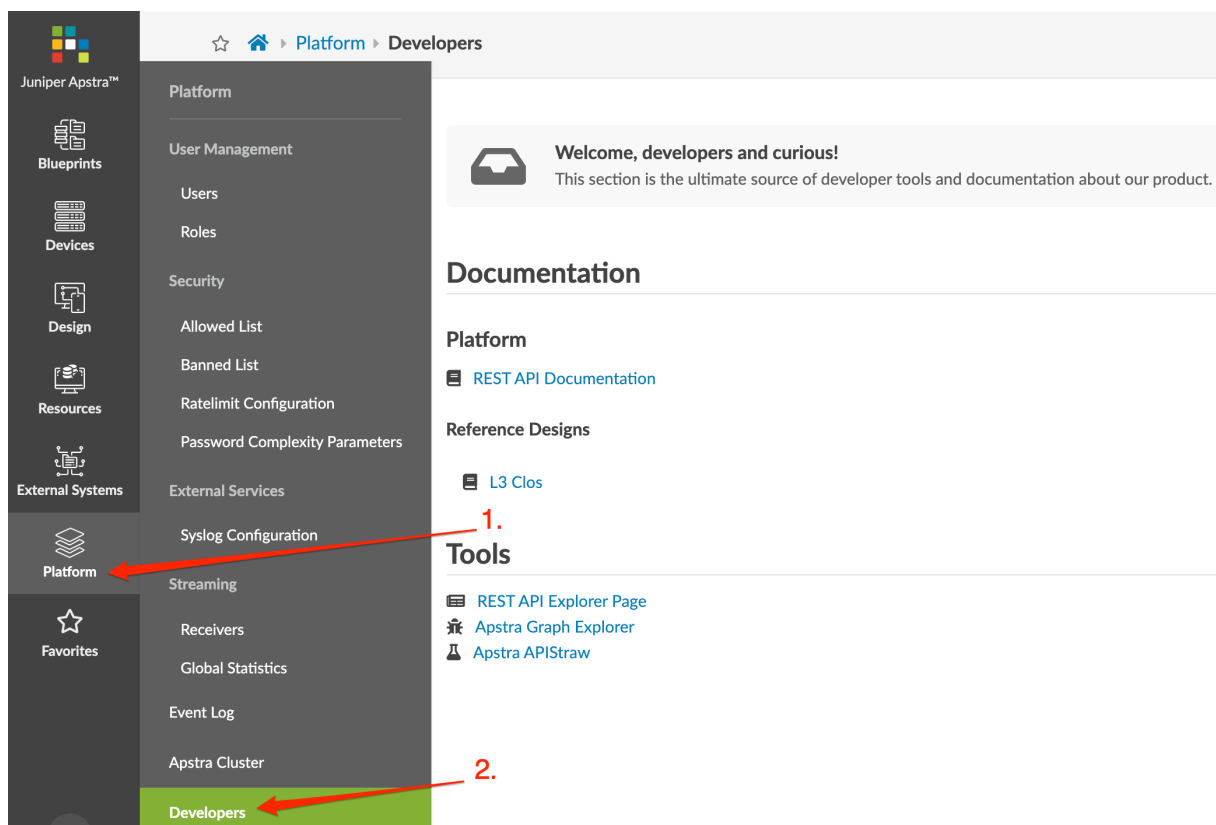
4. Enter values for both **offbox** and **iba**, then click **Execute**. (The values must be positive and multiples of 50.) Juniper offbox agents require 500 MB.
5. To confirm your changes, click **cluster**, click **GET /api/cluster/application-weight**, then click **Execute**.
6. You can close the window at any time to leave the tool.

Developers (Platform)

IN THIS SECTION

- [Authenticate User \(API\) | 779](#)
- [Resource Pools \(API\) | 782](#)
- [Configlets \(API\) | 793](#)
- [Property Sets \(API\) | 796](#)
- [Interface Descriptions \(API\) | 798](#)
- [Probes \(API\) | 802](#)
- [RCI Fault Model \(API\) | 816](#)
- [Health Check Apstra VMs \(API\) | 820](#)
- [API From Python | 820](#)
- [REST API Explorer | 823](#)

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).

Authenticate User (API)

Let's get you authenticated so you can make API calls.

1. From the left navigation menu, navigate to **Platform > Developers**, then click **Rest API Documentation**.

The Swagger API developer tool for the Apstra environment appears.



- POST** **/api/aaa/login** Authenticate given user

After posting a user's valid credentials to this endpoint, a token is returned that can be used for future API requests by putting it in the header.
DEPRECATED: /api/user/login

Parameters

| Name | Description |
|---|--|
| body <small>required</small> (body) | <div>Example Value Model</div> <pre>{ "username": "string", "password": "string"}</pre> <div>1. Replace string with your username and password</div> <div>Cancel</div> <div>Parameter content type application/json</div> <div>2.</div> <div>Execute</div> |

- Responses

Response content typeapplication/json

Curlcurl -X POST "https://10.28.112.3/api/saa/login" -H "accept: application/json" -H "content-type: application/json" -d '{"username": "admin", "password": "admin"}'

Server response

CodeDetails

Copy token

201Response body

```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1cm9udGVzZXkiOnsiYm9keSI6ImVzZXkiLCJpcyI6ImFkbWwifQ==",
  "id": "6c47ebe8-de23-4556-acd1-30c136af3b9d"
```

- Copy the token from the response body, scroll to the top, then click **Authorize** (top-right, shown in the first step).

The **Authorize** dialog appears.

Available authorizations X

Api key authorization

Name: AUTHTOKEN
In: header
Value:

eyJhbGciOiJIUzUxMiIsInR5cCI6Ikp

1. Paste token

2. Click

Authorize

- Paste the token and click **Authorize**.

The dialog shows that you've been authorized.

Available authorizations X

Api key authorization

Authorized

Name: AUTHTOKEN
In: header
Value: *****

Logout

- Click the **X** in the dialog (top-right) to close the dialog.

The lock in the **Authorize** button changes to a closed lock, indicating that you're authenticated.

swagger

AOS 4.1.2
[API Docs](#)

AOS REST API

Schemes
HTTPS

Authorize

Closed lock means you're authenticated

You're ready to make API calls.

Resource Pools (API)

IN THIS SECTION

- [API - ASN Pools | 782](#)
- [API - IP Pools | 787](#)

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 GUI.

To list resource group slots in a blueprint, perform an authenticated HTTP GET to 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.eyJ1c2VybmFtZSI6ImFkbWluIiwia3JlYXRlZGF9hdCI6IjIwMTctMDU0MzFUMDA6MjI6MDcuNTIwMTgzWiIsInNlc3Npb24iOiJjOTliOGVlOS05Y2NjLTRjZTA0ZDYifQ.FnJMR3crPoD0-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.eyJ1c2VybmFtZSI6ImFkbWluIiwia3JlYXRlZGF9hdCI6IjIwMTctMDU0MzFUMDA6MjI6MDcuNTIwMTgzWiIsInNlc3Npb24iOiJjOTliOGVlOS05Y2NjLTRjZTA0ZDYifQ.FnJMR3crPoD0-lQRXnpPOJ8TCsRG9Wr-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}

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
FtZSI6ImFkbWluIiwiaXNlc3Npb24iOiJjOTliOGVlOS05Y2NjLTRjZTAtYTY5NS0wODI3N2ZkYjA0ZDYifQ.FnJ
iIsInNlc3Npb24iOiJjOTliOGVlOS05Y2NjLTRjZTAtYTY5NS0wODI3N2ZkYjA0ZDYifQ.FnJ
MR3crPoD0-lQRXnpPOJ8TCsRG9Wr-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>

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
mFtZSI6ImFkbWluIiwiaXNlc3Npb24iOiJjOTliOGVlOS05Y2NjLTRjZTAtYTY5NS0wODI3N2ZkYj
wMTgzWiIsInNlc3Npb24iOiJjOTliOGVlOS05Y2NjLTRjZTAtYTY5NS0wODI3N2ZkYj
A0ZDYifQ.FnJMR3crPoD0-lQRXnpPOJ8TCsRG9Wr-DaddnAIj6ko' --data-binary
'{"pool_ids":["c23ea447-8f37-419a-9b1c-c48cc55d5b9c"]}' --compressed --insecure
```

A successful ASSIGNMENT returns HTTP 200 OK.


```
{
  "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"
    }
]
}

```


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}


```

        "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 | 794](#)
- [API - Delete Configlet | 794](#)
- [API - Assign Configlet | 794](#)
- [CURL Example - HTTP PUT | 795](#)
- [API - Unassign Configlet | 796](#)

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. You can assign this configlet from the Apstra GUI. This method is not required 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 assignation 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

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 is an empty json set once the configlet is deleted: {"items": []}

Property Sets (API)

IN THIS SECTION

- [API - Create Property Set | 797](#)
- [API - Delete Property Set | 797](#)
- [API - Assign Property Set | 797](#)
- [CURL Example - API HTTP PUT | 798](#)
- [API - Unassign Property Set | 798](#)

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

```
{
  "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}

```
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 | 799](#)

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 | 802](#)
- [Create Probe | 802](#)
- [Inspect Probe | 808](#)
- [Query Probe Anomalies | 811](#)
- [Introspect Processors | 812](#)
- [Stream Data | 815](#)

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:

| | |
|--------------------|---|
| label | human-readable probe label; required, |
| description | optional description of the probe, |
| tags | list of strings with the probe tags; optional, |
| disabled | 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:

| | |
|--------------------|---|
| description | string with a stage description, |
| tags | list of strings that make a set of tags for stage, |
| units | 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: Stage Types

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 | 816](#)
- [Update Root Cause Identification Instance | 817](#)
- [Delete Root Cause Identification Instance | 818](#)
- [List Root Cause Identification Instances | 819](#)

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.

Health Check Apstra VMs (API)



NOTE: You can also check the health of Apstra VMs from the Apstra GUI.

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 | 821](#)

- [API - Blueprints | 821](#)
- [API - Blueprint Racks | 822](#)
- [API - Blueprint Routing Zones \(Security Zones\) | 822](#)
- [API - Blueprint Virtual Networks | 822](#)
- [Run Python | 823](#)

Following are examples of Python 3 code using the Apstra API. Python 3 is supported as of Apstra version 4.1.2.

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)
```


example below shows the model for checking provider settings by login with username and password.

Juniper Apstra™

Platform > Developers > REST API Explorer

Quick Search

POST /api/aaa/check-login

Check provider settings by login with username and password.

Parameters

Name: Description:

body (body)

Example Value Model

```
{
  "auth_mode": "string",
  "vendor": "string",
  "group_dn_attribute_name": "string",
  "group_search_attribute_name": "string",
  "username_attribute_name": "string",
  "password": "string",
  "group_member_mapping_attribute_name": "string",
  "group_role_mappings": "string"
}
```

Juniper Technical Support

IN THIS SECTION

- [Show Tech: Apstra Controller and Device Agents \(GUI\) | 825](#)
- [Show Tech: Offbox Agents \(CLI\) | 827](#)
- [Show Tech: Infra Offbox Agents \(CLI\) | 829](#)
- [Show Tech: Apstra Controller \(CLI\) | 829](#)
- [Show Tech: Onbox Agents \(CLI\) | 830](#)

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 show tech files, from the GUI (recommended) or the CLI, as described in the next sections. You may also be asked for a ["backup" on page 843](#) of your Apstra database.

Show Tech: Apstra Controller and Device Agents (GUI)

You can collect show tech files for the Apstra controller and connected device agents (onbox and offbox) from the Apstra 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 the **Apstra Controller** check box selected.



NOTE: For Apstra server controllers with large databases, the operation may timeout. If this happens, you must ["collect show tech using the CLI" on page 829](#).

3. Starting in Apstra version 4.1.0, you can collect a show_tech from the Apstra GUI that includes a copy of the backup. If Juniper Support requests a backup, check the **Include Backup** check box. This backup provides information for Support and Engineering. It doesn't include credentials, so it's not suitable for restoring your production environment. (Use backups from the ["Back Up Apstra Database" on page 843](#) procedure instead.)
4. Check the box for **Managed Devices** to see the list of managed devices (devices with agents that have been acknowledged).
5. Select the devices that need show tech collected.



NOTE: When device show tech is collected, the configured device system agent username and password authentication are used. If you've configured the device to use a different authentication (AAA) method with a different username and password (such as RADIUS and TACACS) you can't collect show tech from the Apstra GUI. You must ["collect show tech with CLI" on page 830](#).

6. Click **Collect** to start the collection process.


Collect Show Tech ✕

Choose log source: *

☒ AOS Controller

☐ Include Backup

☒ Managed Devices



Show tech collection is supported for up to 100 devices at a time.


▸ Query: All


1-7 of 7 < >

Page Size: 25 ▾

Filter selected by ☒ all ☐ selected only ☐ unselected only

| <input type="checkbox"/> | Address ↕ | Platform ↕ | Platform Version ↕ | Hostname ↕ |
|-------------------------------------|-------------|------------------|--------------------|------------|
| 2 selected <input type="checkbox"/> | 10.28.44.6 | Ubuntu GNU/Linux | 18.04 LTS | localhost |
| <input type="checkbox"/> | 10.28.44.7 | Ubuntu GNU/Linux | 18.04 LTS | localhost |
| <input checked="" type="checkbox"/> | 10.28.44.9 | EOS | 4.24.5M | leaf-1-1 |
| <input checked="" type="checkbox"/> | 10.28.44.11 | EOS | 4.24.5M | leaf-1-2 |
| <input type="checkbox"/> | 10.28.44.13 | Cumulus | 4.2.1 | spine-1 |
| <input type="checkbox"/> | 10.28.44.12 | Cumulus | 4.2.1 | spine-2 |
| <input type="checkbox"/> | 10.28.44.10 | Cumulus | 4.2.1 | leaf-2 |


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

| 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 AOS Controller |

7. 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

| Device Address | State | Started | Finished | Logs |
|----------------|---------|----------------------|----------------------|----------|
| AOS Controller | SUCCESS | 2021-11-12, 15:55:26 | 2021-11-12, 15:58:35 | (118 MB) |



TIP: After the files have been downloaded, you can free up disk space by deleting jobs.

8. From a computer with the ability to upload, [upload the show tech files to your customer case](#).

Show Tech: Offbox Agents (CLI)

We recommend that you use the Apstra GUI to obtain show tech files, but you have the option of using CLI instead, as described below. You'll need the device management IP address(es) and a valid device SSH username and password.



NOTE: If your offbox agents are for infra, you'll collect show tech with a different method. Refer to ["Show-Tech: Infra Offbox Agents \(CLI\)"](#) on page 829 for details.

1. SSH into the Apstra server that the offbox agent is running on. (ssh admin@<apstra-server-ip> where <apstra-server-ip> is the IP address of the Apstra server.)
2. To copy the show tech file(s) to your user directory, run the `aos_offbox_show_tech_collector` command with the following arguments:
 - `--ips <ip address of one or more devices>` (for example: 11.29.53.7 11.29.53.8 11.29.53.9)
 - `--aos-ip <ip address of the Apstra server>` (for example: 11.29.53.3)
 - `--os-type <vendor OS type>` (for example: junos)
 - `--user <admin user name>` (for example: admin)
 - `--password <admin password>` (for example: xu8&j3d'j1=dHnr)

Example for 3 Devices:

```
admin@aos-server:~$ sudo aos_offbox_show_tech_collector --ips 11.29.53.7 11.29.53.8
11.29.53.9 --aos-ip 11.29.53.3 --os-type junos --user admin
[sudo] password for admin:
SSH password for remote device:
2022-11-15 22:24:09,947 invoking DI container to collect 11.29.53.9 show tech
2022-11-15 22:25:32,778 AOS offbox show tech generated at /home/admin
2022-11-15 22:25:32,805 invoking DI container to collect 11.29.53.8 show tech
2022-11-15 22:26:45,773 AOS offbox show tech generated at /home/admin
2022-11-15 22:26:45,799 invoking DI container to collect 11.29.53.7 show tech
2022-11-15 22:27:55,811 AOS offbox show tech generated at /home/admin

admin@aos-server:~$ ls -l
total 217440
-rw-r--r-- 1 root root      75958 Nov 15 22:27 11.29.53.7-5254009E6B20-junos-show-tech.tar.gz
-rw-r--r-- 1 root root      76180 Nov 15 22:26 11.29.53.8-52540039A6F3-junos-show-tech.tar.gz
-rw-r--r-- 1 root root    107620 Nov 15 22:25 11.29.53.9-5254001A5CEB-junos-show-tech.tar.gz
-rw----- 1 root root      8737 Nov 15 22:27 aos_di_11.29.53.7_show_tech_run.log
-rw----- 1 root root      8614 Nov 15 22:26 aos_di_11.29.53.8_show_tech_run.log
-rw----- 1 root root      8491 Nov 15 22:25 aos_di_11.29.53.9_show_tech_run.log

admin@aos-server:~$
```

3. Copy the show tech file(s) to a local computer with the ability to upload.
4. [Upload the show tech file to your customer case.](#)

Show Tech: Infra Offbox Agents (CLI)

The instructions below are for collecting show tech files for infra offbox agents. If your offbox agents are not for infra, refer to ["Show Tech: Apstra Controller and Device Agents \(GUI\)" on page 825](#) or ["Show Tech: Apstra Offbox Agents \(CLI\)" on page 827](#).

1. SSH into the Apstra server that the offbox 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 offbox 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 Controller (CLI)

We recommend using the ["Apstra GUI" on page 825](#) 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.YmjuJDhatJ
```

```

--- 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: Onbox Agents (CLI)

We recommend using the ["Apstra GUI" on page 825](#) to obtain onbox 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`, 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 | 832](#)
- [Change Your User Password | 833](#)
- [Change Your User Name/Email | 833](#)
- [Log Out | 834](#)

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 Apstra 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 user profile page. The left navigation menu includes options like Blueprints, Devices, Design, Resources, External Systems, Platform, Favorites, and User: admin. The main content area is titled 'Profile' and contains a 'User profile' section with fields for Username, First Name, Last Name, Email, and Roles. Below this is a 'Favorites' section with a search bar, pagination (1-3 of 3), and a table of saved pages. Red arrows and text annotations provide instructions: 'Click to add any page to favorites' points to a 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 list; and 'Click to see all saved pages on your user profile' points to the Favorites table.

User profile

| | |
|------------|---------|
| Username | admin |
| First Name | admin |
| Last Name | admin |
| Email | not_set |
| Roles | |

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

- From any page, click your username in the left navigation menu (bottom) and click **Profile** to see your profile page.
- Click the **Change Password** button (top-right), enter your current password, then enter your new password that meets password complexity requirements, twice.
- Click **Change Password** to update your password and return to your profile.

Change Your User Name/Email

- From any page, click your username in the left navigation menu (bottom) and click **Profile** to go to your profile page.
- Click the **Edit** button (top-right), then change your name and/or email, as applicable.
- 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**. Your viewing preferences are saved so when you log in again, you'll have the same customized views. Preferences can include visible fields shown in tables, your preferred topology view (2D, 3D), and whether to show links.

Juniper Apstra™

☆ Home > Profile

Blueprints

Devices

Design

Resources

External Systems

Platform

Favorites

User: admin

User: admin

Profile

Log Out

Change password

Edit

User profile

| | |
|------------|--------------------------|
| Username | admin |
| First Name | admin |
| Last Name | admin |
| Email | not_set |
| Roles | <input type="checkbox"/> |

Favorites

> Query: All

Page Size: 25

| Label | URL | Actions |
|----------|-----|---------|
| No items | | |

Apstra Server Management

IN THIS SECTION

- [Monitor Apstra Server via CLI | 835](#)
- [Restart Apstra Server | 836](#)
- [Reset Apstra Server VM Password | 836](#)
- [Reinstall Apstra Server | 841](#)
- [Apstra Database Overview | 842](#)
- [Back up Apstra Database | 843](#)
- [Restore Apstra Database | 844](#)
- [Reset Apstra Database | 849](#)

- [Migrate Apstra Database | 850](#)
- [Replace SSL Certificate on Apstra Server with Signed One | 854](#)
- [Replace SSL Certificate on Apstra Server with Self-Signed One | 857](#)
- [Change Apstra Server Hostname | 858](#)

The following sections include information about *managing* the Apstra server.

For information about *installing and upgrading* the Apstra server, see the [Juniper Apstra Installation and Upgrade Guide](#).

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:~$
```

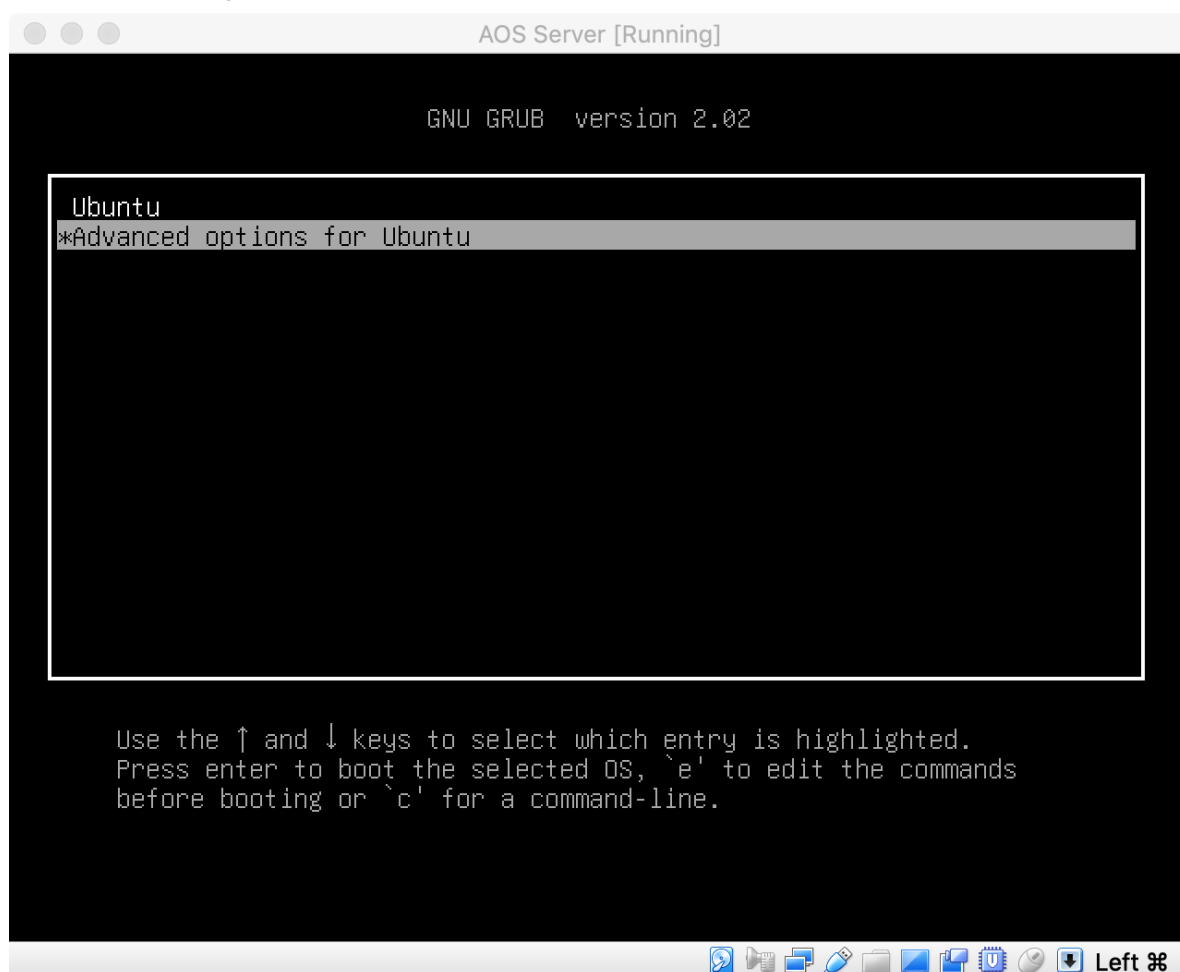
After services are restored (in a minute or two) the "liveness" telemetry alarm resets.

Reset Apstra Server VM Password

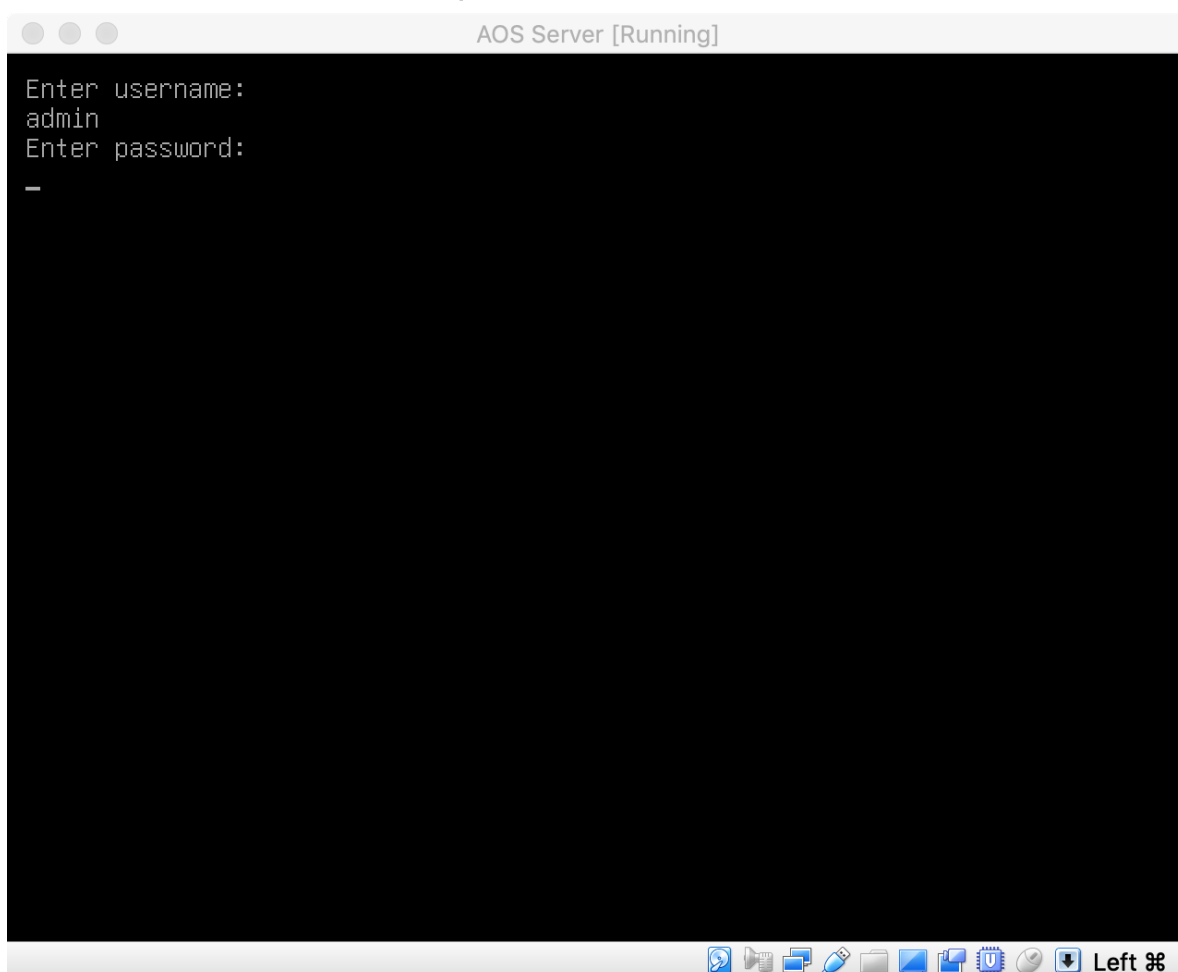
If you lose your **admin** password for the Apstra server VM, and *you still have console access to the Apstra server VM*, you can reset your password.

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.

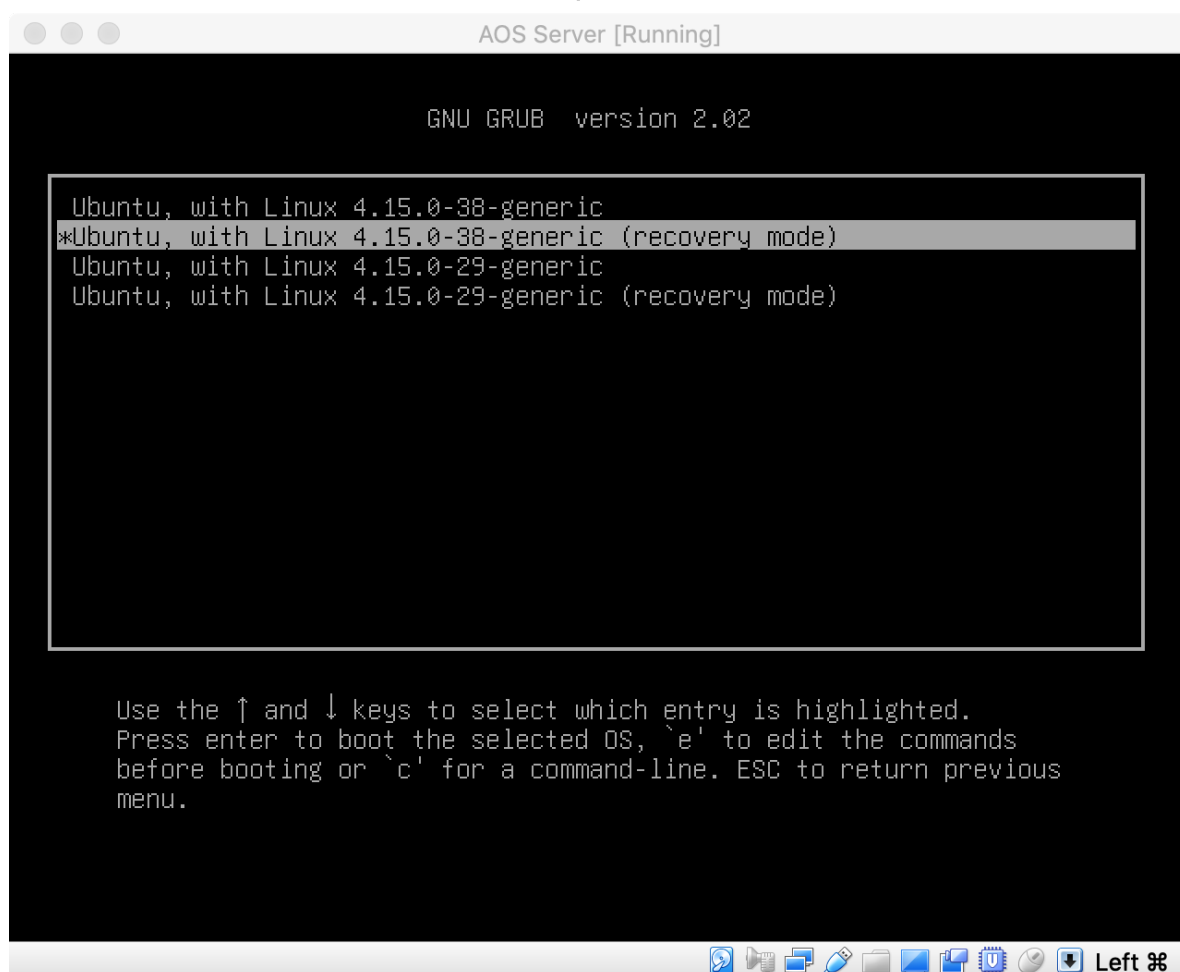
2. Select **Advanced options for Ubuntu**.



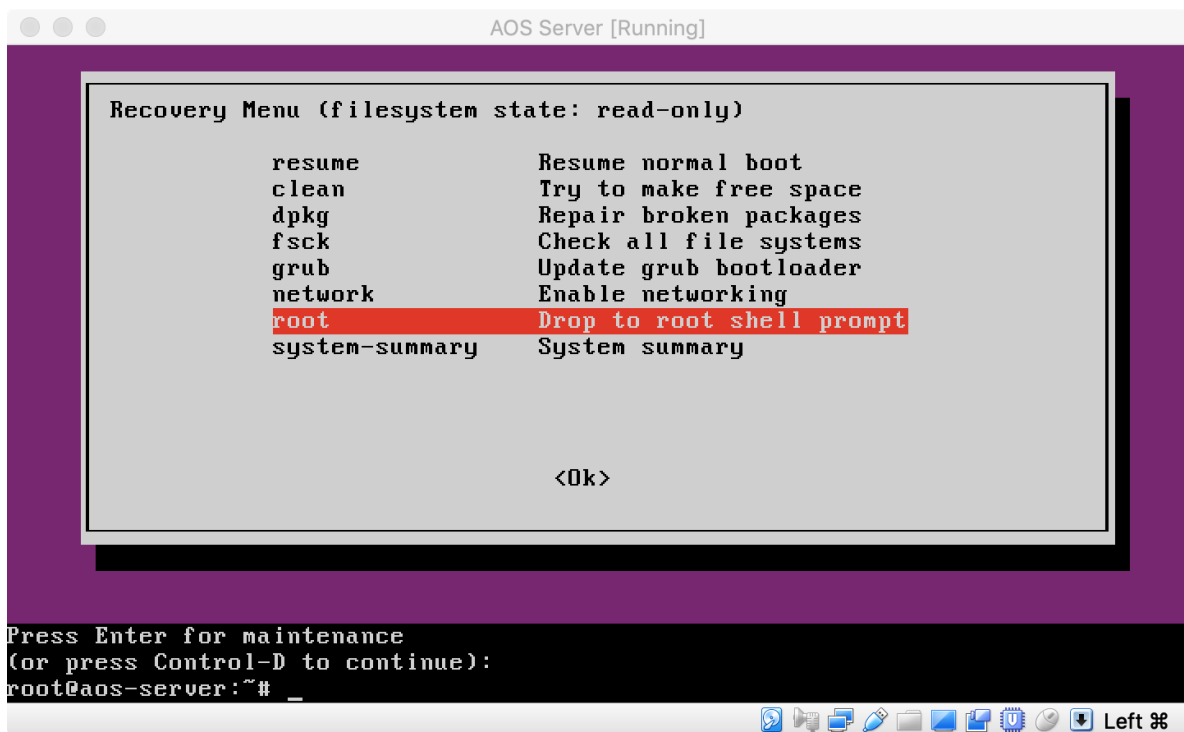
3. Enter username **admin** and password **apstra**.



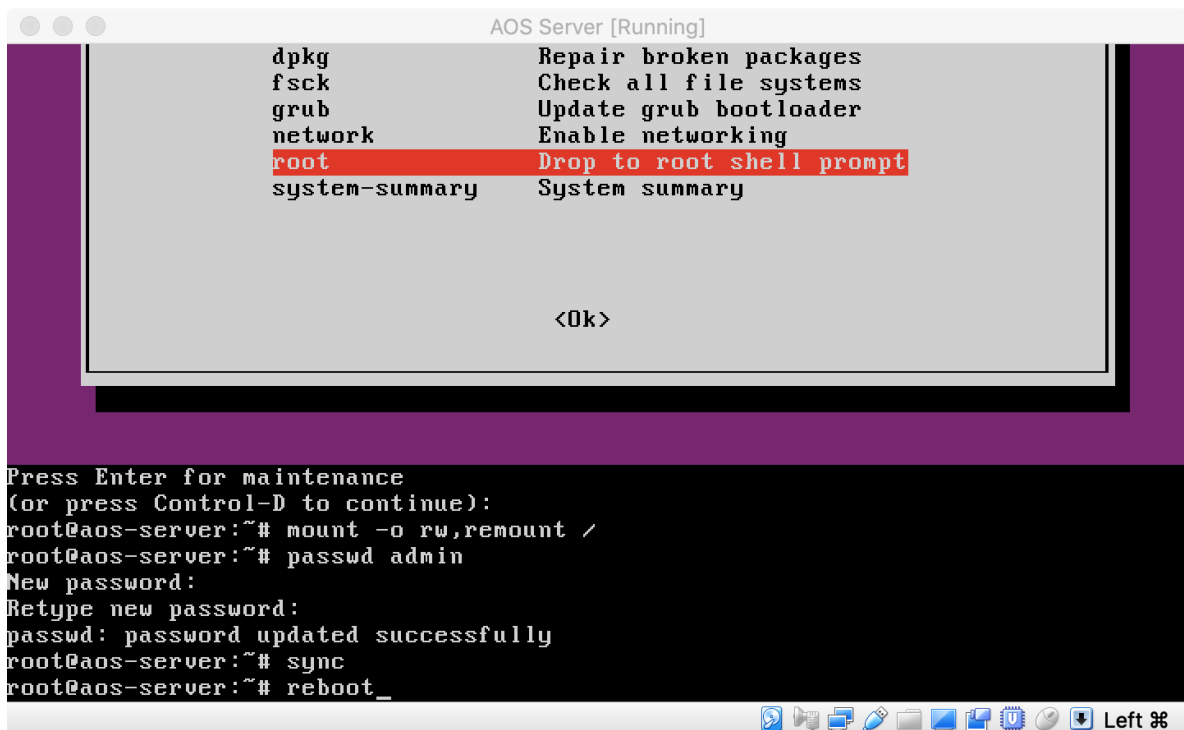
4. At the next GRUB menu, select the first **(recovery mode)** option.



- 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 in to 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 Technical Support](#)" on page 824.

1. If you want to retain the Apstra database, "[back it up](#)" on page 843 now.
2. Download the "Installer" .run file from [Juniper Support Downloads](#).

The screenshot shows the Juniper Support Downloads page. The navigation bar includes links for Support, Downloads, Knowledge Base, Juniper Support Portal, and Community. The 'Downloads' section is active. Below the navigation bar, there is a search bar with the text 'Find a Product' and a prompt 'Start typing a product name to find Software Downloads for that product.' The search results show 'Apstra Fabric Conductor' selected. Below the search bar, there is a table of download results for 'Apstra Fabric Conductor'. The table has columns for Description, Release, File Date, and Downloads. The first row shows 'Apstra Installer for Upgrade' with Release '4.0.1', File Date '28 Oct 2021', and Downloads 'gz (803.93MB) Checksums'.

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

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 | gz (803.93MB) Checksums |

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

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

4. Delete the Apstra server database.

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

5. 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:~$
```

6. Reinstall the Apstra 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 844](#) a database backup or build a new blueprint.

Apstra Database Overview

The Apstra server and related databases run in Docker containers. 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 Technical Support" on page 824](#) 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 Apstra 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 database "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: 2023-06-29_20-56-26
admin@aos-server:~$
```

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

```
admin@aos-server:~$ sudo aos_backup
Including secret keys from the backup
Include all sysdb files
=====
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: 2023-06-29_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 Jun 29 20:58 .
drwxr-xr-x 7 root root 4.0K Jun 29 02:43 ..
drwx----- 2 root root 4.0K Jun 29 02:43 2023-06-29_02-43-12
drwx----- 2 root root 4.0K Jun 29 20:56 2023-06-29_20-56-26
drwx----- 2 root root 4.0K Jun 29 20:58 2023-06-29_20-58-54
admin@aos-server:~$
```

Restore Apstra Database



CAUTION: Always restore a database from a new ["backup" on page 843](#), never from older backups or from the backup included in a `show_tech`.

When you restore a database, the worker VMs will go into a failed state. This problem also occurs when you restore a backup to another worker VM with the same IP address. To fix this issue, add the worker VMs again.

If you make changes after you back up the database, those changes aren't included in the restore. This could create differences between device configs and the Apstra

environment. If this happens, you must perform a full config push, which is service-impacting.

Don't restore a database using the backup included in a `show_tech`. Juniper Support and Engineering use it for analysis. It doesn't include credentials, so it's not suitable for restoring your production environment.



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. Backups are saved as dated snapshots (`/var/lib/aos/snapshot/<date>/aos.data.tar.gz`). The file must be named `aos.data.tar.gz`.

```
admin@aos-server:~$ sudo ls -lah /var/lib/aos/snapshot/2023-06-29_18-40-03
[sudo] password for admin:
total 49M
drwx----- 3 root root 4.0K Jun 29 18:40 .
drwx----- 7 root root 4.0K Jun 29 18:40 ..
-rw----- 1 root root 49M Jun 29 18:40 aos.data.tar.gz
-rwxr-xr-x 1 root root 2.6K Jun 29 18:40 aos_restore
-rw----- 1 root root 1 Jun 29 18:40 comment.txt
drwx----- 2 root root 4.0K Jun 29 18:40 metadata
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/2023-06-29_18-40-03/aos_restore
Including secret keys from the backup
Include all sysdb files
New AOS snapshot: 2023-06-29_19-31-43
[+] Stopping 5/5
  ✓ Container aos_nginx_1
Stopped
                                0.2s
  ✓ Container aos_metadb_1
Stopped
                                10.6s
```

```

✓ Container aos_sysdb_1
Stopped
10.4s

✓ Container aos_auth_1
Stopped
10.5s

✓ Container aos_controller_1
Stopped
10.7s

(Reading database ... 83485 files and directories currently installed.)
Removing aos-compose (99.0.0-5949) ...
tar: Removing leading '/' from member names
/var/lib/aos/db/
/var/lib/aos/db/_AosAuth-0000000064947a9d-000b0094-log
/var/lib/aos/db/_Main-0000000064947aa0-000a1865-log-valid
/var/lib/aos/db/_AosAuth-0000000064947a9d-000b0094-checkpoint-valid
/var/lib/aos/db/_Main-0000000064947aa0-000a1865-log
/var/lib/aos/db/_Main-0000000064947aa0-000a1865-checkpoint-valid
/var/lib/aos/db/_Central-0000000064947a9e-000b9681-log
/var/lib/aos/db/_AosSysdb-0000000064947a9d-000c83d2-checkpoint-valid
/var/lib/aos/db/_AosAuth-0000000064947a9d-000b0094-log-valid
/var/lib/aos/db/_AosAuth-0000000064947a9d-000b0094-checkpoint
/var/lib/aos/db/_AosSysdb-0000000064947a9d-000c83d2-log-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/.sqlite
/var/lib/aos/db/.devpi/server/.serverversion
/var/lib/aos/db/.devpi/server/.event_serial
/var/lib/aos/db/_Main-0000000064947aa0-000a1865-checkpoint
/var/lib/aos/db/_Metadb-0000000064947a9d-000b82ea-log
/var/lib/aos/db/_Metadb-0000000064947a9d-000b82ea-log-valid
/var/lib/aos/db/_AosSysdb-0000000064947a9d-000c83d2-log
/var/lib/aos/db/blueprint_backups/
/var/lib/aos/db/blueprint_backups/configlets/
/var/lib/aos/db/blueprint_backups/configlets/167/
/var/lib/aos/db/blueprint_backups/configlets/167/graph.md5sum
/var/lib/aos/db/blueprint_backups/configlets/167/graph.json.zip
/var/lib/aos/db/blueprint_backups/configlets/161/
/var/lib/aos/db/blueprint_backups/configlets/161/graph.md5sum
/var/lib/aos/db/blueprint_backups/configlets/161/graph.json.zip
/var/lib/aos/db/blueprint_backups/configlets/166/
/var/lib/aos/db/blueprint_backups/configlets/166/graph.md5sum

```

```

/var/lib/aos/db/blueprint_backups/configlets/166/graph.json.zip
/var/lib/aos/db/blueprint_backups/configlets/164/
/var/lib/aos/db/blueprint_backups/configlets/164/graph.md5sum
/var/lib/aos/db/blueprint_backups/configlets/164/graph.json.zip
/var/lib/aos/db/blueprint_backups/configlets/163/
/var/lib/aos/db/blueprint_backups/configlets/163/graph.md5sum
/var/lib/aos/db/blueprint_backups/configlets/163/graph.json.zip
/var/lib/aos/db/_Central-0000000064947a9e-000b9681-log-valid
/var/lib/aos/db/_AosSysdb-0000000064947a9d-000c83d2-checkpoint
/var/lib/aos/db/_Metadb-0000000064947a9d-000b82ea-checkpoint-valid
/var/lib/aos/db/_AosController-0000000064947aa0-000d40b6-log
/var/lib/aos/db/_AosController-0000000064947aa0-000d40b6-checkpoint
/var/lib/aos/db/_Auth-0000000064947a9e-000a44d7-checkpoint
/var/lib/aos/db/_Central-0000000064947a9e-000b9681-checkpoint
/var/lib/aos/db/_Central-0000000064947a9e-000b9681-checkpoint-valid
/var/lib/aos/db/_AosController-0000000064947aa0-000d40b6-log-valid
/var/lib/aos/db/_Auth-0000000064947a9e-000a44d7-checkpoint-valid
/var/lib/aos/db/_Auth-0000000064947a9e-000a44d7-log
/var/lib/aos/db/_Auth-0000000064947a9e-000a44d7-log-valid
/var/lib/aos/db/_Metadb-0000000064947a9d-000b82ea-checkpoint
/var/lib/aos/db/_AosController-0000000064947aa0-000d40b6-checkpoint-valid
/var/lib/aos/anomaly/
/var/lib/aos/anomaly/_Anomaly-0000000064947a9e-000c9d0a-checkpoint-valid
/var/lib/aos/anomaly/_Anomaly-00000000649452ff-00034e81-checkpoint
/var/lib/aos/anomaly/_Anomaly-00000000649452ff-00034e81-checkpoint-valid
/var/lib/aos/anomaly/_Anomaly-0000000064947a9e-000c9d0a-checkpoint
/var/lib/aos/anomaly/_Anomaly-00000000649452ff-00034e81-log-valid
/var/lib/aos/anomaly/_Anomaly-0000000064947a9e-000c9d0a-log
/var/lib/aos/anomaly/_Anomaly-00000000649452ff-00034e81-log
/var/lib/aos/anomaly/_Anomaly-0000000064947a9e-000c9d0a-log-valid
/etc/aos/aos.conf
/etc/aos-img-chksum/
/etc/aos-img-chksum/checksums.signed
/etc/aos-img-chksum/checksums
/etc/aos-img-chksum/key.pub
/opt/aos/aos-compose.deb
/opt/aos/frontend_images/
/opt/aos/frontend_images/jinja_docs.zip
/opt/aos/frontend_images/sdt_docs.zip
/opt/aos/frontend_images/aos-web-ui.zip
/etc/aos/version
/etc/aos-auth/secret_key
/etc/aos-credential/secret_key

```

```

Selecting previously unselected package aos-compose.
(Reading database ... 83454 files and directories currently installed.)
Preparing to unpack /opt/aos/aos-compose.deb ...
Unpacking aos-compose (99.0.0-5949) ...
Setting up aos-compose (99.0.0-5949) ...
Verifying checksums for docker images...
Signature Verified Successfully
Verified.
[+] Building 0.0s
(0/0)

[+] Running 5/5
  ✓ Container aos_nginx_1
Started
                                0.3s
  ✓ Container aos_metadb_1
Started
                                0.2s
  ✓ Container aos_sysdb_1
Started
                                0.2s
  ✓ Container aos_auth_1
Started
                                0.2s
  ✓ Container aos_controller_1
Started
                                0.3s

admin@aos-server:~$

```

3. 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: active (exited) since Thu 2023-06-22 16:45:14 UTC; 1 week 0 days ago
  Docs: man:systemd-sysv-generator(8)
  CPU: 433ms

Jun 22 16:45:14 aos-server aos[1512]: Container aos_sysdb_1 Starting
Jun 22 16:45:14 aos-server aos[1512]: Container aos_controller_1 Starting

```



```

Jun 22 16:45:14 aos-server aos[1512]: Container aos_nginx_1 Starting
Jun 22 16:45:14 aos-server aos[1512]: Container aos_metadb_1 Starting
Jun 22 16:45:14 aos-server aos[1512]: Container aos_nginx_1 Started
Jun 22 16:45:14 aos-server aos[1512]: Container aos_metadb_1 Started
Jun 22 16:45:14 aos-server aos[1512]: Container aos_controller_1 Started
Jun 22 16:45:14 aos-server aos[1512]: Container aos_sysdb_1 Started
Jun 22 16:45:14 aos-server aos[1512]: Container aos_auth_1 Started
Jun 22 16:45:14 aos-server systemd[1]: Started LSB: Start AOS management system.
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 32K
drwx----- 8 root root 4.0K Jun 29 19:31 .
drwxr-xr-x 13 root root 4.0K Jun 29 19:32 ..
drwx----- 3 root root 4.0K Jun 29 15:44 2023-06-29_15-44-51
drwx----- 3 root root 4.0K Jun 29 15:45 2023-06-29_15-45-37
drwx----- 3 root root 4.0K Jun 29 16:21 2023-06-29_16-21-36
drwx----- 3 root root 4.0K Jun 29 18:11 2023-06-29_18-11-34
drwx----- 3 root root 4.0K Jun 29 18:40 2023-06-29_18-40-03
drwx----- 3 root root 4.0K Jun 29 19:31 2023-06-29_19-31-43
admin@aos-server:~$

```

Reset Apstra 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 Apstra 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 onbox 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 onbox device system agent. This is not required for offbox 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, stop service (or completely shut down the Apstra server VM) to disconnect the old Apstra server.

```

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 the Configuration section of the Juniper Apstra Installation and Upgrade guide.
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.

1. Query: All

2. Managed Devices

3. Confirm devices are in active state

| 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 |
| B6FC3A | 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 |
| B35370 | Cisco NXOSv | FULL CONTROL | 10.29.29.16 | AOS_4.0.1_OB.1045 | spine1 | NXOS 9.3(7) | ✓ | IS-ACTIVE |
| B5D4E6 | 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 |

Replace SSL Certificate on Apstra Server 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. It's not for any internal device-server connectivity. Since the HTTPS certificate is not retained when you back up the system, you must manually back up the `etc/aos` folder. 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: Don't modify `nginx.crt` or `nginx.key` filenames. They're referred to in `nginx.conf`. As part of subsequent service upgrades, these files could be replaced, so the filenames must be predictable.

Also, don't change configuration in `nginx.conf`, as this file may be replaced during Apstra server upgrade, and any changes you make would be discarded.

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, don't leave them in the Nginx configuration folder. On startup, Nginx will attempt to load them (*.conf), 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 for 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 on Apstra Server 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. It's not for any internal device-server connectivity. Since the HTTPS certificate is not retained when you back up the system, you must manually back up the `etc/aos` folder. 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

```

Change Apstra Server Hostname

You have the option of changing the default Apstra server hostname (aos-server).

1. SSH into the Apstra server as user **admin** (ssh admin@<apstra-server-ip> where <apstra-server-ip> is the IP address of the Apstra server.)
2. As root user, run the command `aos_hostname <hostname>` where <hostname> is the new hostname.

```

admin@aos-server:~$ sudo aos_hostname new-aos-server
[sudo] password for admin:
admin@aos-server:~$

```

The new hostname will display the next time you log in.



NOTE: Do not use `/etc/hostname` to change the Apstra server hostname. With this method, if you configure syslog to be forwarded to an external server, the default hostname will be entered into the log instead of the new one.

Apstra CLI Utility

IN THIS SECTION

- [Install Apstra-CLI | 859](#)
- [Access Apstra-cli | 860](#)

You can augment Juniper Apstra GUI functionality with Apstra's command-line interface utility called Apstra CLI (previously known as AOS CLI). Apstra CLI is delivered in a Docker container. You can use it on any system that's running a compatible version of Docker.

As of Apstra version 4.1.1, Apstra-CLI is GA. Prior versions are considered experimental. For more information on experimental versions, see [Juniper Support Knowledge Base article KB36747](#).

For steps on installing and accessing Apstra CLI, see sections below.

Install Apstra-CLI

1. Download the **Apstra CLI Utility** for your Apstra version from the **Application Tools** section of [Juniper Support Downloads](#).
2. Copy the Apstra CLI Docker container `tar.gz` file to the Apstra server. (The file name is something like, `apstracli-release_4.1.2.5.tar.gz`.) For example:

```
scp apstracli-release_4.1.2.5.tar.gz admin@10.28.65.3/home/admin
```

3. Load the provided Docker image into Docker with the `docker image load` command. For example:

```
admin@aos-server:~$ docker image load -i apstracli-release_4.1.2.5.tar.gz
beee9f30bc1f: Loading layer [=====>] 5.862MB/
5.862MB
3fc750b41be7: Loading layer [=====>] 821.8kB/
821.8kB
20a7b70bdf2f: Loading layer [=====>] 59.53MB/
59.53MB
879c0d8666e3: Loading layer [=====>] 6.749MB/
6.749MB
ba4121fa2557: Loading layer [=====>] 3.451MB/
3.451MB
ee87976d1e1f: Loading layer [=====>] 470.4MB/
470.4MB
Loaded image: apstracli:release_4.1.2.5
admin@aos-server:~$
```

Access Apstra-cli

1. Start the Juniper Apstra CLI Docker container with the `docker run` command. In the example below replace 4.1.2.5 with your Apstra CLI version, and 10.28.65.3 with the IP address of your Apstra server. The password is your Apstra GUI password (not the VM password).

```
admin@aos-server:~$ docker run --rm -ti -v ~/.mytmp apstracli:release_4.1.2.5 -s 10.28.65.3
Password [admin]:
-----
-----
*****
*****

NOTE: This is a Limited Availability tool

Use it ONLY under the strict supervision of Juniper Apstra personnel

*****
*****
-----
-----

Welcome to Juniper Apstra CLI! Press TAB for suggestions
Juniper Apstra CLI version: release-4.1.2.5
```

```
Juniper Apstra Server URL: https://10.28.65.3:443, Version: 4.1.2-269
apstra-cli>
```

2. Apstra-CLI comes with a built-in feature that auto-completes commands. Press the TAB key, then the up and down arrow keys to explore this tool and its functionality. You can also type `--help` for descriptions of each function.

For examples of how to use apstra-cli, see ["Apstra-CLI Commands" on page 1149](#) in the References section.

Guides

IN THIS SECTION

- [Extensible Telemetry Guide | 861](#)
- [5-Stage Clos Architecture | 877](#)
- [Juniper EVPN Support | 880](#)
- [Intent-Based Analytics with apstra-cli Utility | 889](#)
- [AOSOM-Streaming Guide | 902](#)
- [Mixed Uplink Speeds between Leaf Devices and Spine Devices | 915](#)

Extensible Telemetry Guide

IN THIS SECTION

- [Extensible Telemetry Overview | 862](#)
- [Set Up Development Environment | 862](#)
- [Develop Collector | 863](#)
- [Write Collector | 866](#)
- [Unit Test Collector | 873](#)
- [Package Collector | 874](#)

- [Upload Packages | 875](#)
- [Use Telemetry Collector | 875](#)

Extensible Telemetry Overview

Install Apstra device drivers and telemetry collectors to collect additional telemetry that can be used in ["IBA probes" on page 709](#). The device drivers enable Apstra to connect to a NOS and collect telemetry. Apstra ships with drivers for EOS, NX-OS, Ubuntu, and CentOS. To add a driver for an operating system not listed here, contact ["Juniper Support" on page 824](#).

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 824](#). 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 type of key and value in each item determines the storage schema path. 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 20: 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 |
| 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 21: 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 22: Sample Application Schema

| Sample Item | Application Schema |
|---|--|
| <pre>{ "identity": "eth0", "value": "up", }</pre> | <pre>{ "type": "object", "properties": { "identity": { "type": "string", }, "value": { "type": "string", } } }</pre> |

Table 22: Sample Application Schema (Continued)

| Sample Item | Application Schema |
|---|--|
| <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 218](#) via the GUI.

Write Collector

IN THIS SECTION

- [Collect Data from Device | 867](#)
- [Parse Data | 868](#)

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,
 'memTotal': 4011060,
 'modelName': u'vEOS',
 'serialNumber': u'',
 'systemMacAddress': u'52:54:00:ce:87:37',
 'uptime': 62620.55,
 '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 that Apstra software uses. 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. Cumulus is no longer supported as of Apstra version 4.1.0, although this example remains for illustrative purposes. 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'].iteritems()
        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_<NOS>-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 offbox solution and your NOS is not EOS, 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 | 876](#)
- [Start Collector | 876](#)
- [Delete Collector | 876](#)
- [Get Collected Data | 876](#)
- [List Running Collector Services | 876](#)

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 `apstra-cli`, you have the option of importing all the schemas in the `.tgz` file.

Start Collector

To start a service, use the POST API `/api/systems/<system_id>/services` with the following three arguments:

| Arguments | |
|------------|--|
| 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 `apstra-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 | 877](#)
- [Create 5-Stage Clos Network | 879](#)
- [Modify 5-stage Clos Network | 880](#)

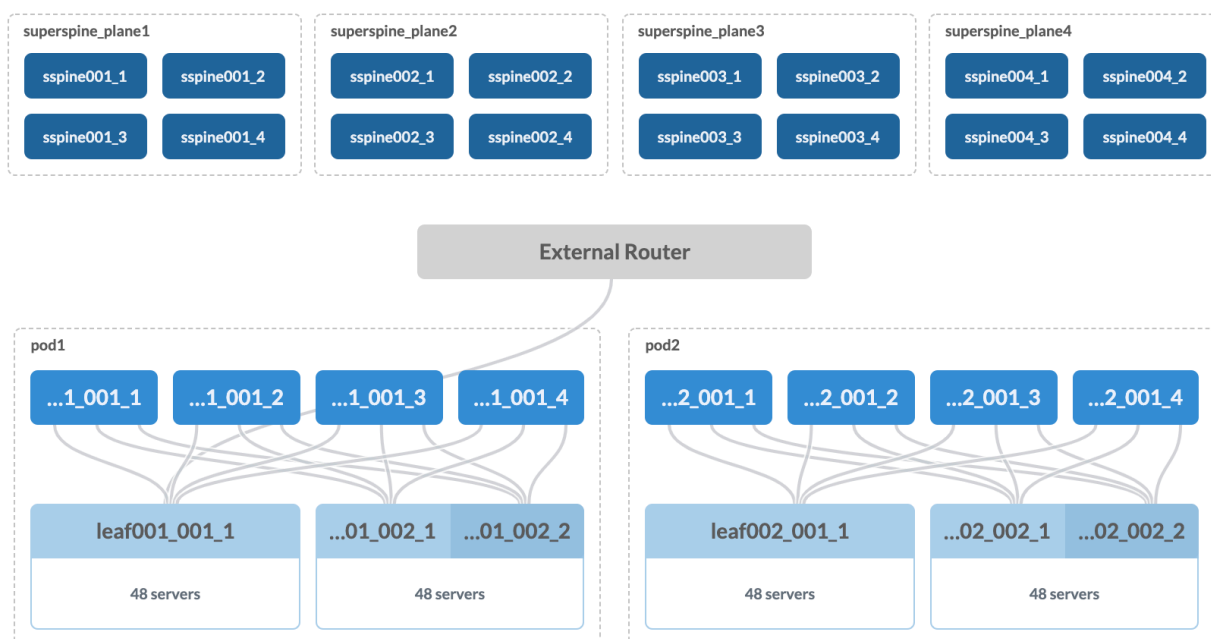
5-Stage Clos Overview

IN THIS SECTION

- [5-Stage Clos Limitations | 878](#)
- [5-Stage Clos and EVPN | 879](#)

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 are groups of superspine devices. Each 5-stage topology consists of

one or more planes. Each plane consists of one or more superspine devices. 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 824](#).

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 918](#).
 - IPv6 applications are supported as of Apstra version 4.1.0.
 - IPv6 virtual networks are supported on EVPN blueprints as of Apstra version 4.1.1.
 - 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 superspine devices

- External generic systems on spine devices and leaf devices in the same pod
- Unsupported blueprint modifications:
 - Add or remove superspine planes

5-Stage Clos and EVPN

Extending EVPN networks across multiple pods within the same blueprint adds the following value:

- **Scaling:** provide any-to-any connectivity for applications distributed across multiple pods.
- **Redistributing Workloads:** To load-balance applications, you can migrate a group of applications from one pod to another pod while preserving application IP and MAC addresses.
- **Performing pod maintenance:** Migrate 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 880](#).

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.

The method for creating cross-pod ["virtual networks" on page 409](#) is the same method as for 3-stage networks.

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 7](#) (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 **Superspine** port role in that port group.
 - Superspine logical devices require a **Spine** port role in the port group.
2. Confirm that the global catalog includes ["interface maps" on page 13](#) (Design > Interface Maps) that map the logical devices to the correct ["device profiles" on page 95](#); 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:
 - Superspine logical device to device profile
 - Spine logical device to device profile
 3. Create one or more rack-based ["templates" on page 35](#), each including at least one link for **Superspine 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 262](#), ["IPv4 addresses" on page 267](#), ["IPv6 addresses" on page 270](#)) needed in the network.
 6. Create a ["blueprint" on page 274](#) 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 391](#), and for information about adding superspine devices to planes see [Planes](#). ["Racks \(Datacenter\)" on page 387](#)

Juniper EVPN Support

IN THIS SECTION

- [Overview | 881](#)
- [EVPN multi-homing Terminology and Concepts | 881](#)
- [Topology Specification | 883](#)

- [EVPN Services | 884](#)
- [Configuration Rendering | 886](#)

Overview

The Junos EVPN ESI multi-homing feature enables you to directly connect end servers to leaf devices and provide redundant connectivity via multi-homing. This feature is supported only on LAGs that span two leaf devices 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 leaf devices 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 leaf devices 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

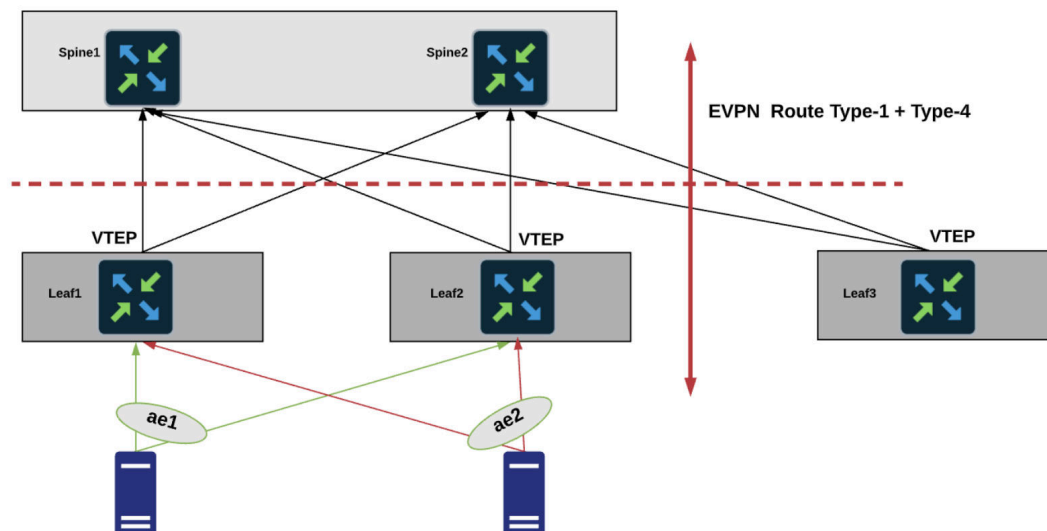
enabled for each aggregated Ethernet interface of the Juniper devices. Multi-homed interfaces into the ES are identified using the ESI.

ESI has certain restrictions and requirements as listed below:

- ESI based ToR leaf devices 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 leaf devices.
- 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 leaf devices 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 leaf devices 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

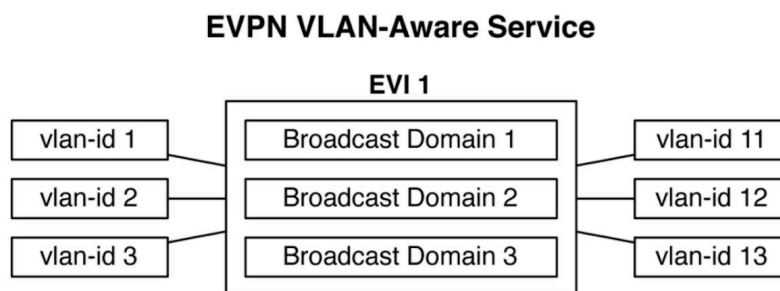
IN THIS SECTION

- [EVPN VLAN-Aware | 884](#)
- [Create EVPN Network | 885](#)

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 ["offbox device agents" on page 154](#) for all switches. (Onbox agents are not supported on Junos.)
2. Confirm that the global catalog includes ["logical devices" on page 7](#) (Design > Logical Devices) that meet Juniper device requirements; create them if necessary:
3. Confirm that the global catalog includes ["interface maps" on page 13](#) (Design > Interface Maps) that map the logical devices to the correct ["device profiles" on page 95](#) for the Juniper devices; create them if necessary.
4. Create a ["rack type" on page 23](#).
 - 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 leaf devices. EVPNs using ESs have a link aggregation option. Select the LAG mode **LACP (Active)**

5. Create a ["rack-based template" on page 35](#).
6. Create a generic system for an external router.
7. Create resource pools for ["ASNs" on page 262](#), ["IP addresses" on page 267](#), and ["VNI" on page 264](#).
8. Create a ["blueprint" on page 274](#) based on the ESI-based template, then build the EVPN-based network topology for the Juniper devices by assigning ["resources" on page 277](#), ["device profiles" on page 280](#), and ["device IDs" on page 295](#).

Configuration Rendering

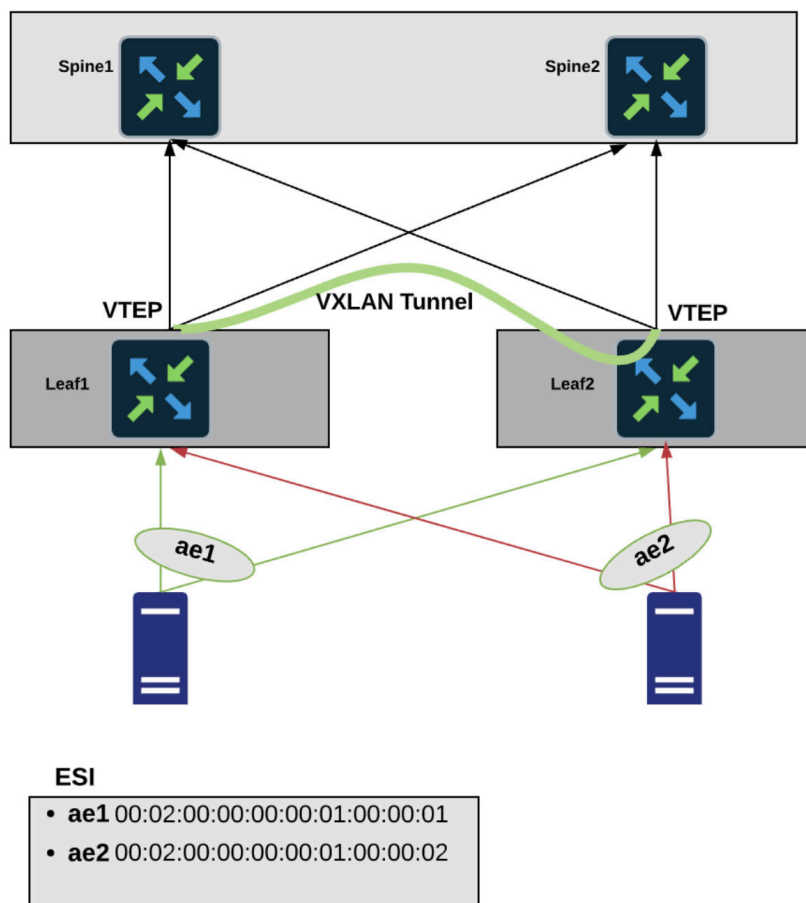
IN THIS SECTION

- [Reference Design | 886](#)
- [Limitations | 888](#)

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 leaf devices. For the inter-VN traffic VXLAN routing is done in the fabric using L3 VNIs on the border leaf devices as per standard design.
- **VXLAN VTEPs** - On Juniper leaf devices 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 leaf devices, they are configured with the same LACP parameter for a given ESI

so that they appear as a single system.



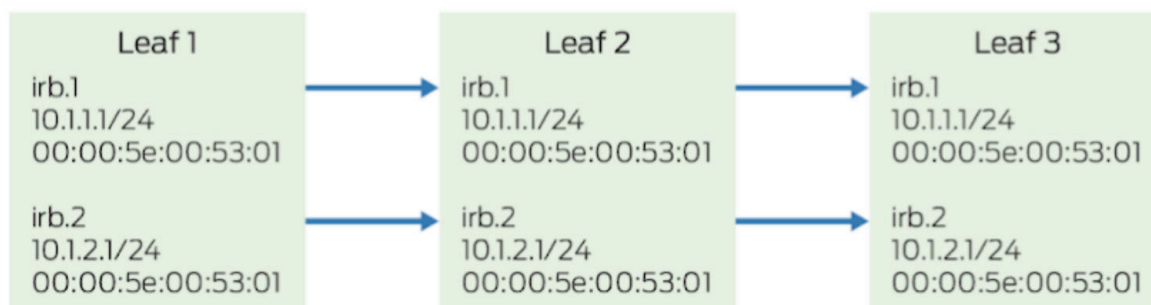
ESI MAC addresses are auto-generated internally. You can ["configure the value of the most significant byte"](#) on [page 551](#) used in the generated MAC. A new facade API is added to update the MSB value. A new node is added to the rack based template that contains the MAC MSB value. The default value 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 leaf devices 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 leaf devices 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 leaf devices to Apstra-managed Layer 3 servers is not supported.

Intent-Based Analytics with apstra-cli Utility

IN THIS SECTION

- [IBA with apstra-cli Overview | 889](#)
- [Install apstra-cli | 890](#)
- [Install Packages | 890](#)
- [Create Agent Profiles | 892](#)
- [Create Agents | 893](#)
- [Update Agents from apstra-cli | 895](#)
- [Install IBA Probes | 896](#)
- [Apstra IBA Probes Examples | 898](#)

IBA with apstra-cli Overview

You can work with Intent-based analytics (IBA) from the Apstra GUI, or for non-production environments you can use the experimental apstra-cli utility (formerly called aos-cli). For information about how to use IBA probes from the GUI, see ["Probes" on page 709](#) in the Analytics section. This guide shows you how to use apstra-cli.



NOTE: The apstra-cli utility is an experimental tool and has limited support. Do not use it in production environments unless advised by Juniper Support. Some versions of apstra-cli are not intended for certain Apstra releases. Some apstra-cli commands may or may not work between different Apstra releases. It's always best to test a version of apstra-cli

with a specific Apstra release in a non-production environment, or contact ["Juniper Support" on page 824](#) for assistance.

The apstra-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 apstra-cli.
2. Install packages.
3. Create device agent profiles.
4. Install device agents.
5. Install IBA probes.

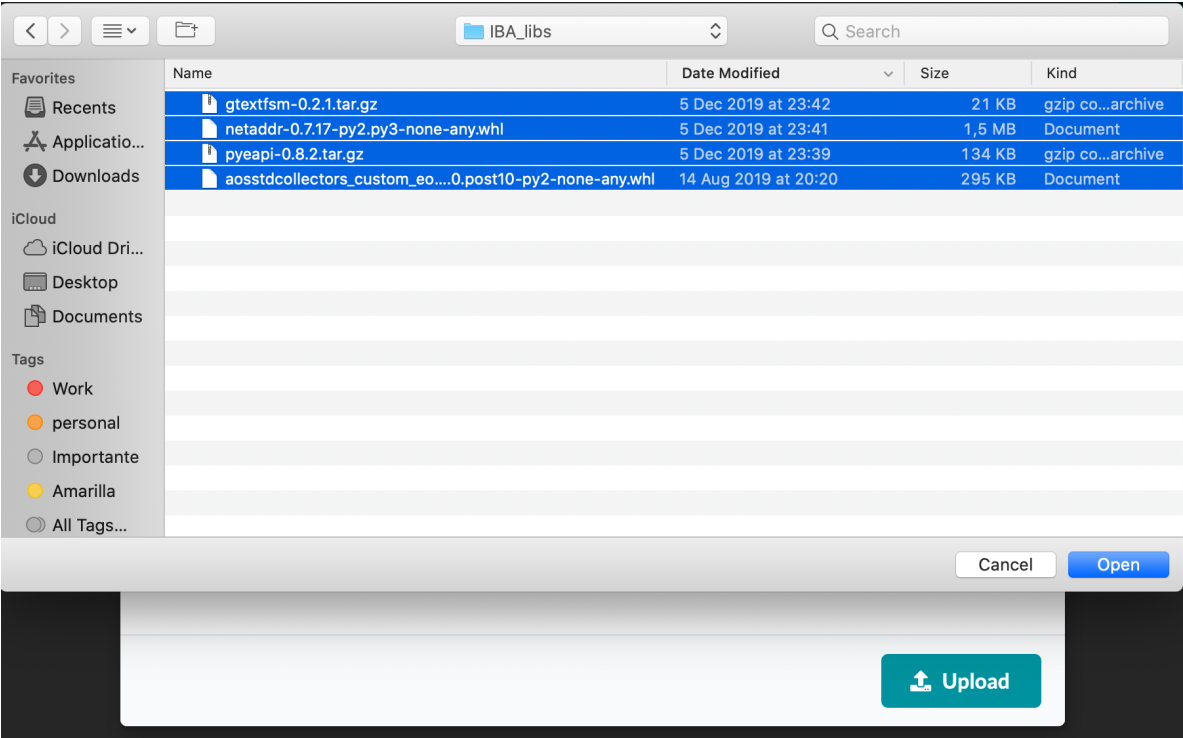
After probes are instantiated you can use ["Syslog" on page 754](#) to send messages to Syslog servers.

Install apstra-cli

["Install the apstra-cli utility" on page 859](#).

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 summary table 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.
(If your environment has Internet access, you only need to include the custom collector package.)

Create Agent Profile

| Key | Value |
|--|-------|
| No options | |
| <input type="button" value="Add an option"/> | |

Packages 4

1-4 of 4 < >

Page Size: 25 ▾

| 4 selected | 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?

4. Click **Create** to create the agent profile and return to the summary table view.

For more information about agent profiles, see ["Agent Profiles" on page 211](#).

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 > Managed Devices** and click the management IP of the device. Click the **Agent** tab. The **Config** section lists any 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 in to

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

Update Agents from apstra-cli

As of apstra-cli 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, apstra-cli shows available agent profiles. To select, use the up and down arrow keys.

```
apstra-cli> system-agents update-profile --ip 172.20.120.6-11 --profile
EOS-IBA  EOS
```

For example.

```
apstra-cli> 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
apstra-cli>
```

Install IBA Probes

You can install IBA probes using the Apstra GUI, or for non-production environments you can use `apstra-cli`. For information about how to create or instantiate predefined probes from the GUI, see ["Probes" on page 709](#) in the Analytics section. This section shows you how to use the `apstra-cli` utility.

All probes described in this document are included in `apstra-cli` build 412 and later. Probe `.j2` files may be made available if the probe file is not built into the `apstra-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 `apstra-cli /mytmp` directory.

```
apstra-cli> 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
apstra-cli>

```

To create probes, use the `probe create apstra-cli` command. You'll be prompted for additional options.

```

apstra-cli> 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.

```

apstra-cli> probe create --blueprint 67cd936d-c2de-49f8-8708-df465f0cdc68
                                L2 Virtual  two_stage_l3clos

```

To list available probes supplied with `apstra-cli`, use `--file` and tab-completion. Scroll through the list with the up and down arrow keys.

```

apstra-cli> 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.

```

apstra-cli> 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 | 898](#)
- [Switch Memory Leak \(Arista EOS only\) | 899](#)
- [Fault Tolerance | 901](#)

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 device interfaces that the Apstra software manages, based on interface telemetry that device agents collect.

| Filename | Description |
|--------------------------|---|
| pkt_discard_anomalies.j2 | Detect Fabric interfaces having sustained packet discards |

To install the `pkt_discard_anomalies.j2` IBA Probe:

```
apstra-cli> 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
apstra-cli>
```

Switch Memory Leak (Arista EOS only)

Switch Memory Leak IBA probes detect abnormal memory leaks in specified processes on devices that the Apstra software manages, based on system telemetry that device agents collect. 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`.

```
apstra-cli> 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 alreadypresent, 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`.

```
apstra-cli> 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
apstra-cli>
```

```
apstra-cli> 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
apstra-cli>
```



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:

```
apstra-cli> 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
apstra-cli>
```

Fault Tolerance

These (2) probes require apstra-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. |
| 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:

```
apstra-cli> 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
apstra-cli>
```



NOTE: number_of_faulty_spines_to_be_tolerated must be specified.

To install the lag_link_fault_tolerance.j2 IBA Probe:

```
apstra-cli> 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
apstra-cli>
```

AOSOM-Streaming Guide

IN THIS SECTION

- [AOSOM-Streaming Overview | 902](#)
- [Configure Aiosom-Streaming | 907](#)
- [Reconfigure Aiosom-streaming after Apstra Server Upgrade | 909](#)
- [Build Aiosom-Streaming VM \(Optional\) | 910](#)
- [Troubleshooting | 914](#)

AOSOM-Streaming Overview

IN THIS SECTION

- [Grafana | 903](#)
- [Prometheus | 904](#)
- [InfluxDB | 906](#)



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. Apstra software fully supports this plug-in for 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>.

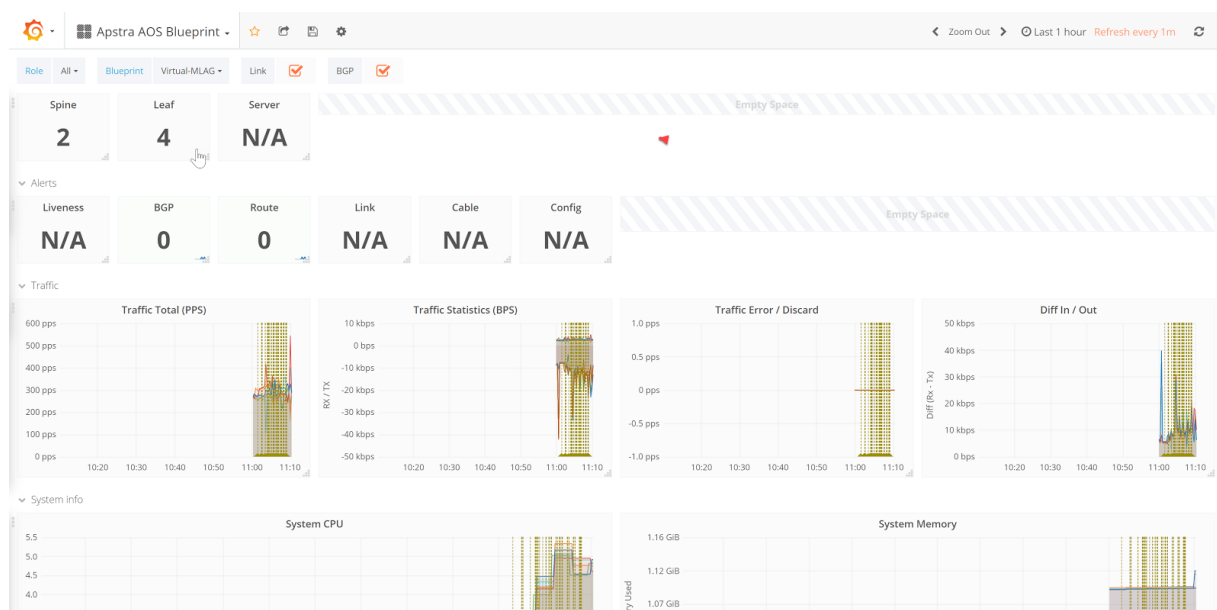
The packaged solution includes:

- 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).

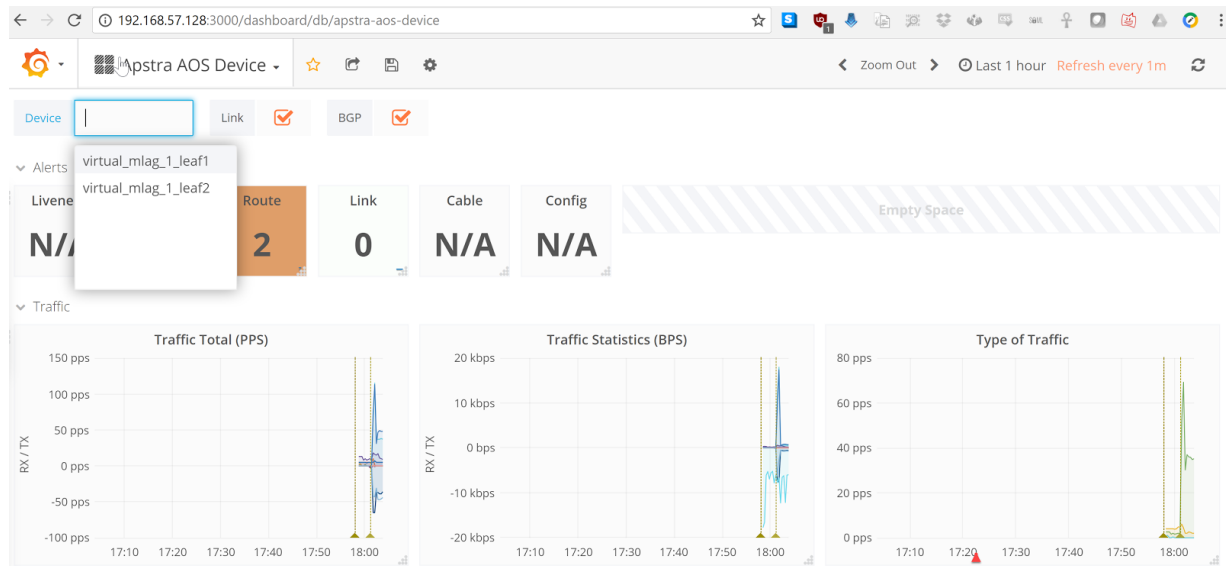
The grafana GUI includes two main sections (top left). **Apstra AOS Blueprint** describes overall telemetry alerts and traffic throughput, as well as individual devices for interface telemetry. Blueprints are learned automatically using the Apstra 'telegraf' Docker container; no further configuration is necessary.



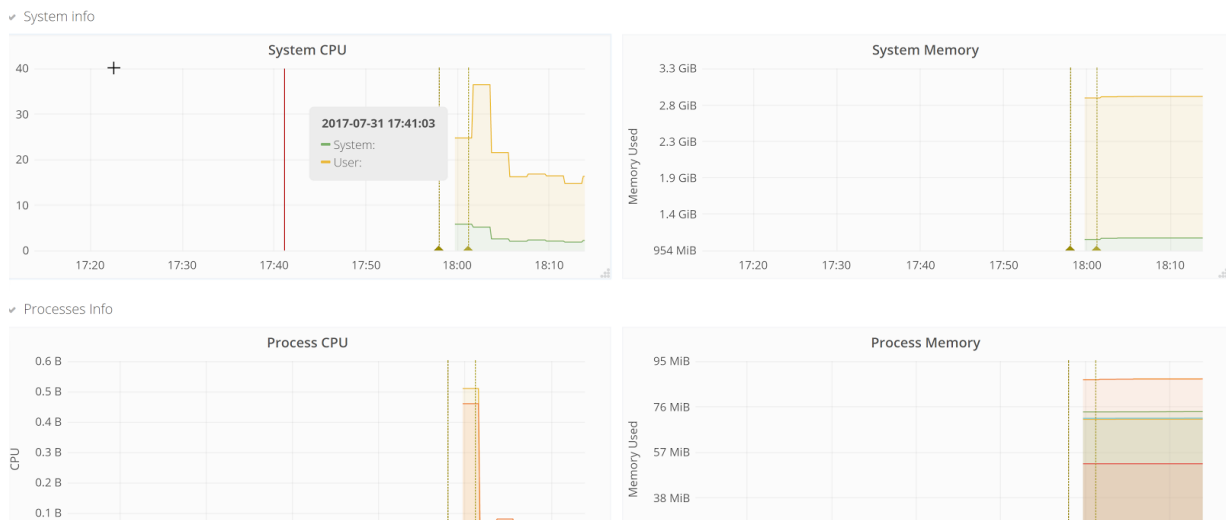
In the screenshot above, we can observe traffic in the demo Apstra environment, and aggregate CPU, traffic, and errors.

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:

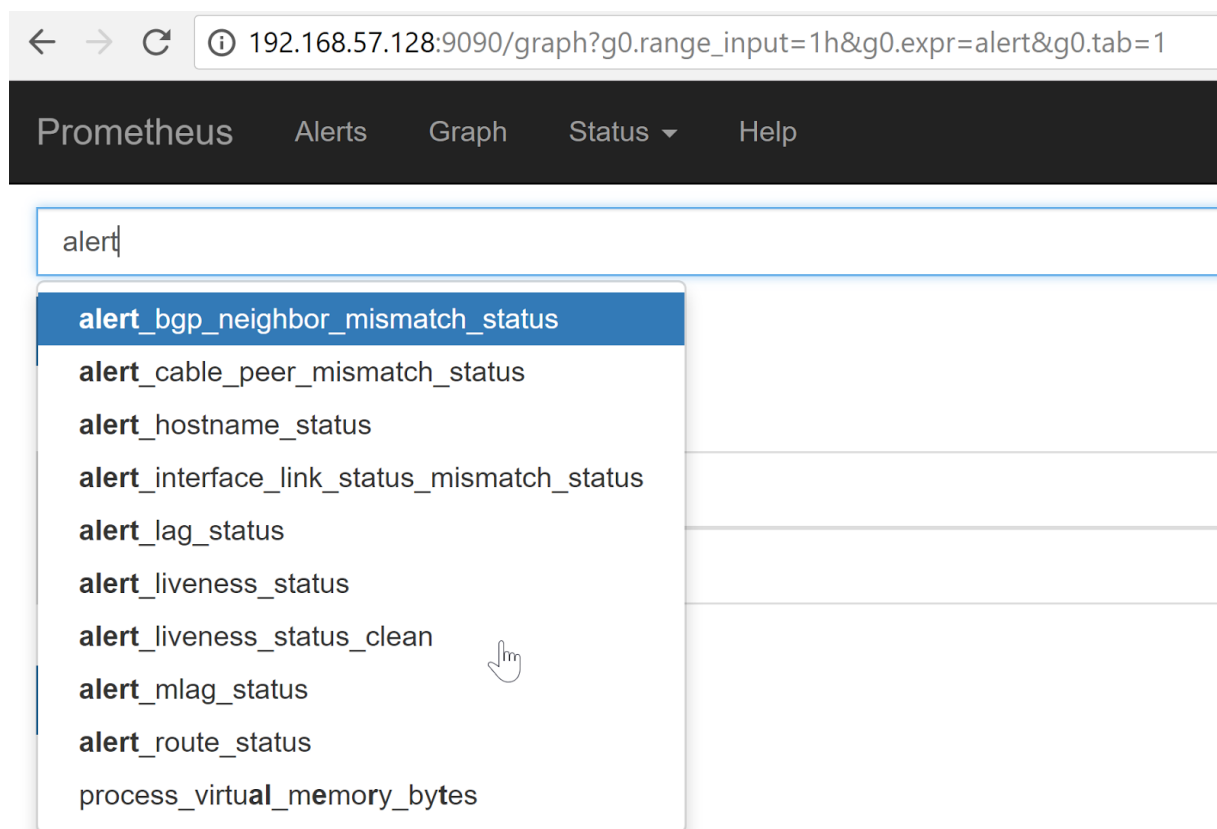


Prometheus

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 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. The main content area shows a search bar with the text 'alert'. A dropdown menu is open, listing the following metrics:

- alert_bgp_neighbor_mismatch_status** (highlighted)
- alert_cable_peer_mismatch_status
- alert_hostname_status
- alert_interface_link_status_mismatch_status
- alert_lag_status
- alert_liveness_status
- alert_liveness_status_clean (with a mouse cursor pointing to it)
- alert_mlag_status
- alert_route_status
- process_virtual_memory_bytes

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")

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 Data

Documentation

Database: aos⌵⚙

Query: show field keys

Generate Query URL

Query Templates⌵

event_arp_state

| fieldKey | fieldType |
|----------|-----------|
| event | "integer" |

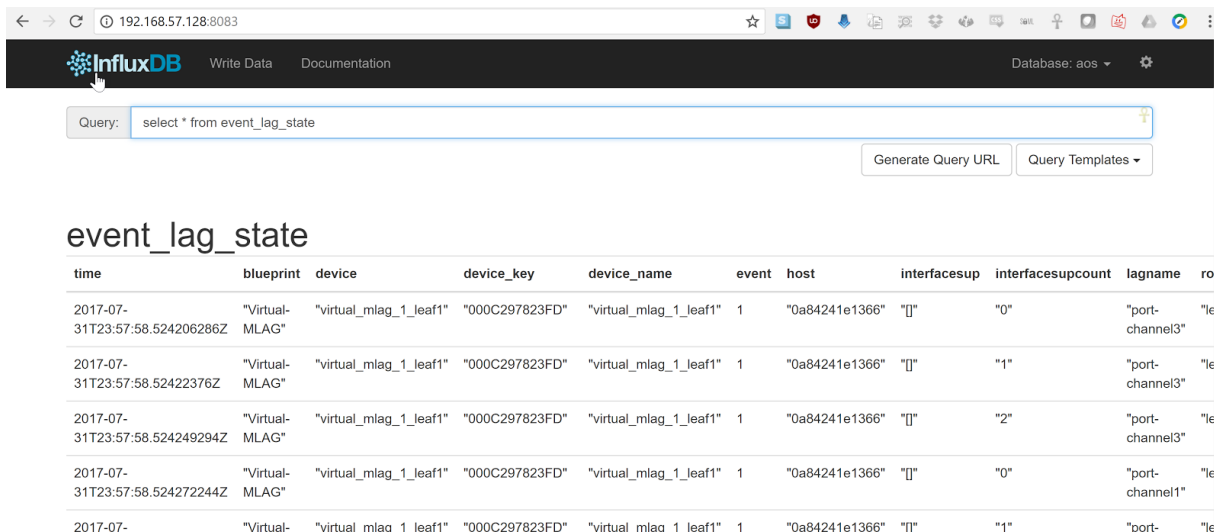
event_bgp_neighbor

| 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 for the `event_lag_state` measurement are displayed in a table.

| time | blueprint | device | device_key | device_name | event | host | interfacesup | interfacesupcount | lagname | ro |
|--------------------------------|----------------|------------------------|----------------|------------------------|-------|----------------|--------------|-------------------|-----------------|-----|
| 2017-07-31T23:57:58.524206286Z | "Virtual-MLAG" | "virtual_mlag_1_leaf1" | "000C297823FD" | "virtual_mlag_1_leaf1" | 1 | "0a84241e1366" | "[]" | "0" | "port-channel3" | "le |
| 2017-07-31T23:57:58.52422376Z | "Virtual-MLAG" | "virtual_mlag_1_leaf1" | "000C297823FD" | "virtual_mlag_1_leaf1" | 1 | "0a84241e1366" | "[]" | "1" | "port-channel3" | "le |
| 2017-07-31T23:57:58.524249294Z | "Virtual-MLAG" | "virtual_mlag_1_leaf1" | "000C297823FD" | "virtual_mlag_1_leaf1" | 1 | "0a84241e1366" | "[]" | "2" | "port-channel3" | "le |
| 2017-07-31T23:57:58.524272244Z | "Virtual-MLAG" | "virtual_mlag_1_leaf1" | "000C297823FD" | "virtual_mlag_1_leaf1" | 1 | "0a84241e1366" | "[]" | "0" | "port-channel1" | "le |
| 2017-07- | "Virtual- | "virtual mlag 1 leaf1" | "000C297823FD" | "virtual mlag 1 leaf1" | 1 | "0a84241e1366" | "[]" | "1" | "port- | "le |



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. Run the command `make start` to set up the project, 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. Run the `docker ps` command to verify that the current **Telegraf** container image matches the proper version for the new Apstra release.

```

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. Run the command `docker-compose up -d` to restart the service.
5. Run the `docker ps` command to verify that the container is running with the new image.



NOTE: For assistance regarding which version to install or if you have any questions about the procedure, contact "[Juniper Support](#)" on page 824.

Build Aosom-Streaming VM (Optional)

IN THIS SECTION

- [Install Ubuntu 16.04.2 | 910](#)
- [Install Packages | 910](#)
- [Set Container Restart Policy | 912](#)
- [Change System Hostname | 913](#)

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 Aiosom-streaming | 914](#)
- [Ensure Containers are Running | 914](#)

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 Aiosom-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 Leaf Devices and Spine Devices

The leaf devices 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 spine logical devices if they're used in a blueprint. You could possibly use the AOS-CLI utility for manual patching. AOS-CLI is an experimental tool and it may not be able to provide a solution. For assistance, contact ["Juniper Support" on page 824](#).

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 spine devices

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

2. Create a **Rack Based** template based on the mixed speed rack type.

Create Template

Structure

Rack TypesMixed line speed rack type

mixed (2x10 Gbps links to spines) ✕

1

Add racks

Spines

Spine Logical Device*

Select... ▼

Count*

1

Superspine Connectivity

Link per Superspine Count

0

Link to Superspine Speed

Select... ▼

Tags

Preview

TopologyRacksSpine Logical Device

☐ Expand Nodes?

☐ Show Links?

spine1

mixed

10gig_1

40gig_1

3. You can create a **Pod Based** template based on the above rack based template.

Create Template

Structure

Pods

mixed template ✕

1

Add pods

Superspines

Superspine Logical Device *

Select...

Plane Count *

1

Per Plane Count *

1

Tags

Select...

Mixed line speed template

Preview

Topology

Pods

Superspine Logical Device

☐ Expand Nodes?
 ☐ Show Links?

⚠ Superspine links are hidden

superspine_plane1

superspine1

mixed template

spine1

mixed

10gig_1

40gig_1

4. As a Day 0 operation you can create a ["blueprint" on page 274](#) 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 389](#) to an existing blueprint.

References

IN THIS SECTION

- [Apstra Feature Matrix | 918](#)
- [Qualified Devices and NOS Versions | 979](#)
- [NOS Upgrade Paths \(Devices\) | 988](#)
- [Predefined Dashboards \(Analytics\) | 995](#)
- [Predefined Probes \(Analytics\) | 999](#)
- [Probe Processors \(Analytics\) | 1077](#)
- [Configlet Examples \(Design\) | 1143](#)
- [Apstra-CLI Commands | 1149](#)
- [Apstra EVPN Support Addendum | 1151](#)
- [Apstra Server Configuration File | 1159](#)
- [Agent Configuration File \(Devices\) | 1170](#)
- [Graph | 1175](#)
- [Juniper Apstra Technology Preview | 1191](#)

Apstra Feature Matrix

IN THIS SECTION

- [Apstra 4.1.2 Feature Matrix | 919](#)
- [Apstra 4.1.1 Feature Matrix | 939](#)
- [Apstra 4.1.0 Feature Matrix | 959](#)

Apstra 4.1.2 Feature Matrix

IN THIS SECTION

- [Fabric Roles | 919](#)
- [Fabric Connectivity | 920](#)
- [Device Management | 921](#)
- [Connectivity \(from Leaf Layer\) | 921](#)
- [Connectivity \(from Access Layer\) | 923](#)
- [Routing Policies | 923](#)
- [Miscellaneous | 924](#)
- [Virtual Network CT Type | 924](#)
- [IP Link CT Type | 925](#)
- [Static Route CT Type | 926](#)
- [Custom Static Route CT Type | 927](#)
- [BGP to Generic CT Type | 928](#)
- [BGP to IP Endpoint CT Type | 933](#)
- [Dynamic BGP Peering CT Type | 935](#)
- [Routing Policy CT Type | 937](#)
- [BGP Attributes \(common to all BGP CTs\) | 938](#)
- [DCI Features | 939](#)

Fabric Roles

| Fabric Roles | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|---|-----|-------|-------|----------|------------------|
| Access Switch | No | No | No | Yes | No |
| Non-EVPN-VXLAN Leaf (IP forwarder only) | Yes | Yes | Yes | Yes | Yes |

(Continued)

| Fabric Roles | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|---------------------|-----|-------|-------|----------|------------------|
| EVPN-VXLAN Leaf | Yes | Yes | Yes | Yes | Yes |
| Spine or Superspine | Yes | Yes | Yes | Yes | Yes |

Fabric Connectivity

| Fabric Connectivity | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|---------------------------------------|-----|-------|-------|----------|------------------|
| 3-stage Clos | Yes | Yes | Yes | Yes | Yes |
| 5-stage Clos | Yes | Yes | Yes | Yes | Yes |
| IP fabric (non-EVPN-VXLAN) | Yes | Yes | Yes | Yes | Yes |
| EVPN-VXLAN fabric | Yes | Yes | Yes | Yes | Yes |
| IPv6 fabric RFC-5549 (non-EVPN) | Yes | Yes | Yes | No | No |
| 3-stage Clos with access switch layer | No | No | No | Yes | Limited |
| Collapsed fabric | No | No | No | Yes | Limited |
| Freeform | No | No | No | Yes | Yes |

Device Management

| Device Management | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|---|-----|-------|-------|--------------|------------------|
| On-box agent | Yes | Yes | Yes | Not Possible | No |
| Off-box agent | Yes | Yes | No | Yes | Yes |
| Telemetry extensibility | Yes | Yes | Yes | Yes | Yes |
| Apstra ZTP | Yes | Yes | Yes | Yes | Yes |
| Device OS upgrade | Yes | Yes | Yes | Yes | Yes |
| Traffic draining (spine devices/ superspine devices - maintenance mode) | Yes | Yes | Yes | Yes | Yes |
| Traffic draining (leaf devices) | Yes | Yes | Yes | Yes | Yes |

Connectivity (from Leaf Layer)

| Connectivity (from Leaf Layer) | EOS | NX-OS | SONic | Junos OS | Junos OS Evolved |
|--------------------------------|-----|-------|--------------|--------------|------------------|
| LAG | Yes | Yes | Yes | Yes | Yes |
| MLAG/vPC | Yes | Yes | Yes | Not possible | Not possible |
| EVPN ESI (with LACP) | No | No | Not possible | Yes | Yes |

(Continued)

| Connectivity (from Leaf Layer) | EOS | NX-OS | SONic | Junos OS | Junos OS Evolved |
|--|-----|-------|--------------|----------|---------------------|
| 802.1x | Yes | No | Not possible | Limited | No |
| VLANs | Yes | Yes | Yes | Yes | Yes |
| Overlay protocol: static VXLAN | Yes | Yes | Not possible | No | No |
| Overlay protocol: EVPN (3-stage and 5- stage) | Yes | Yes | Yes | Yes | Yes |
| IPv4 DHCP relay | Yes | Yes | Yes | Yes | Yes |
| IPv6 DHCP relay | Yes | Yes | Yes | Yes | Yes |
| EVPN DCI | Yes | Yes | Yes | Yes | Yes |
| IPv6 for applications (with EVPN and IPv4 fabric) | Yes | Yes | Yes | Yes | Yes |
| Policy Assurance (L3 ACLs) | Yes | Yes | No | Yes | Yes |

Connectivity (from Access Layer)

| Connectivity (from Access Layer) | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|--|-----|-------|-------|----------|---------------------|
| LAG | N/A | N/A | N/A | Yes | Yes |
| ESI LAG | N/A | N/A | N/A | Yes | Limited |

Routing Policies

| Routing Policies | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|--|-----|-------|-------|----------|---------------------|
| Import all routes or default route or extra routes only | Yes | Yes | Yes | Yes | Yes |
| Export loopback, link and VN IP. Export extra routes | Yes | Yes | Yes | Yes | Yes |
| Export aggregate prefixes | Yes | Yes | Yes | Yes | Yes |
| Export L3 server link subnets | Yes | Yes | Yes | Yes | Yes |
| Route target import/export policies | Yes | Yes | Yes | Yes | Yes |

Miscellaneous

| Miscellaneous | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|---------------------------------------|-----|-------|-------|----------|------------------|
| Configlets | Yes | Yes | Yes | Yes | Yes |
| FFE: add racks/add links/change speed | Yes | Yes | Yes | Yes | Yes |
| Mixed leaf/spine link speed | Yes | Yes | Yes | Yes | Yes |

Virtual Network CT Type

| Virtual Network CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|--|-----|-------|-------|----------|------------------|
| Single Virtual Network (VXLAN-based) | Yes | Yes | Yes | Yes | Yes |
| Multiple Virtual Network (VXLAN-based) | Yes | Yes | Yes | Yes | Yes |
| VLAN (default VRF, non-VXLAN) | Yes | Yes | Yes | Yes | Yes |

IP Link CT Type

| IP Link CT Type | EOS | NX-OS | 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 |
| L3 Sub-interface on non-LAG physical interface (untagged/vlan tagged, default/non-default RZ, IPv6) | Yes | Yes | Yes | Yes | Yes |
| L3 Sub-interface on LAG interface (untagged/vlan tagged, default/non-default RZ, IPv4) | Yes | Yes | Yes | Yes | Yes |
| L3 Sub-interface on LAG interface (untagged/vlan tagged, default/non-default RZ, IPv6) | Yes | Yes | Yes | Yes | Yes |

(Continued)

| IP Link CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|---|-----|-------|-------|----------|------------------|
| L3 Sub-interface on LAG interface (untagged/vlan tagged, default RZ, IPv4) - spine/sspine | Yes | Yes | Yes | Yes | Yes |
| L3 Sub-interface on LAG interface (untagged/vlan tagged, default RZ, IPv6) - spine/sspine | Yes | Yes | Yes | Yes | Yes |

Static Route CT Type

| Static Route CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|---|-----|-------|-------|----------|------------------|
| Static Route (IPv4) applied on L3 Sub-interface | Yes | Yes | Yes | Yes | Yes |
| Static Route (IPv6) applied on L3 Sub-interface | Yes | Yes | Yes | Yes | Yes |
| Static Route (IPv4) applied on SVI | Yes | Yes | Yes | Yes | Yes |

(Continued)

| Static Route CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|--|-----|-------|-------|----------|------------------|
| Static Route (IPv6) applied on SVI | Yes | Yes | Yes | Yes | Yes |
| Static Route with Share IP Endpoint Enabled (IPv4) | Yes | Yes | Yes | Yes | Yes |
| Static Route with Share IP Endpoint Enabled (IPv6) | Yes | Yes | Yes | Yes | Yes |

Custom Static Route CT Type

| Custom Static Route CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|--|-----|-------|-------|----------|------------------|
| Custom Static Route (IPv4, default/non-default RZ) | Yes | Yes | Yes | Yes | Yes |
| Custom Static Route (IPv6, default/non-default RZ) | Yes | Yes | Yes | Yes | Yes |

BGP to Generic CT Type

| BGP to Generic CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|--|------------|--------------|--------------|-----------------|-------------------------|
| BGP session on L3 Sub-interface towards generic (IPv4, default/non-default RZ) | Yes | Yes | Yes | Yes | Yes |
| BGP session on L3 Sub-interface towards generic (IPv6, default/non-default RZ) | Yes | Yes | Yes | Yes | Yes |
| BGP session on SVI towards generic (IPv4, default RZ) | Yes | Yes | Yes | Yes | Yes |
| BGP session on SVI towards generic (IPv4, non-default RZ) | Yes | Yes | Yes | Yes | Yes |
| BGP session on SVI towards generic (IPv6, non-default RZ) | Yes | Yes | Yes | Yes | Yes |
| BGP session on SVI towards generic (IPv6, default RZ) | Yes | Yes | Yes | Yes | Yes |

(Continued)

| BGP to Generic CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|--|-----|-------|--------------|--------------|------------------|
| BGP session on SVI (mlag) towards dual-homed generic using secondary IPs (IPv4, default VRF) | Yes | Yes | Not possible | Not possible | Not possible |
| BGP session on SVI (mlag) towards dual-homed generic using secondary IPs (IPv4, non-default VRF) | Yes | Yes | Not possible | Yes | Yes |
| BGP session on SVI (mlag) towards dual-homed generic using secondary IPs (IPv6, default VRF) | Yes | Yes | Not possible | Not possible | Not possible |
| BGP session on SVI (mlag) towards dual-homed generic using secondary IPs (IPv6, non-default VRF) | Yes | Yes | Not possible | Yes | Yes |
| BGP session to generic with Share IP Endpoint Enabled (IPv4) | Yes | Yes | Yes | Yes | Yes |

(Continued)

| BGP to Generic CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|--|-----|-------|-------|----------|------------------|
| BGP session to generic with Share IP Endpoint Enabled (IPv6) | Yes | Yes | Yes | Yes | Yes |
| BGP session to generic with dynamic ASN (IPv4) | No | No | No | No | No |
| BGP session to generic with Static ASN (IPv4) | Yes | Yes | Yes | Yes | Yes |
| BGP session to generic with dynamic ASN (IPv6) | No | No | No | No | No |
| BGP session to generic with static ASN (IPv6) | Yes | Yes | Yes | Yes | Yes |
| BGP Unnumbered session (link-local peering) on L3 Sub-interface (BP has IPv6 app enabled, default VRF) | Yes | Yes | Yes | No | No |

(Continued)

| BGP to Generic CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|--|-----|-------|-------|----------|------------------|
| BGP Unnumbered session (link-local peering) on L3 Sub-interface (BP has IPv6 app enabled, non-default VRF) | No | Yes | Yes | No | No |
| BGP Unnumbered session (link-local peering) on SVI (BP has IPv6 app enabled, default VRF) | Yes | Yes | Yes | No | No |
| BGP Unnumbered session (link-local peering) on SVI (BP has IPv6 app enabled, non-default VRF) | No | Yes | Yes | No | No |
| BGP Unnumbered session (link-local peering) on L3 Sub-interface (default VRF, BP has IPv6 app disabled) | Yes | Yes | Yes | No | No |

(Continued)

| BGP to Generic CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|---|-----|-------|-------|----------|------------------|
| BGP Unnumbered session (link-local peering) on L3 Sub-interface (non-default VRF, BP has IPv6 app disabled) | No | Yes | Yes | No | No |
| BGP Unnumbered session (link-local peering) on SVI (BP has IPv6 app disabled, default VRF only) | No | No | No | No | No |
| BGP Peering combinations (Int to Int, Lo to Int, Int to Lo, Lo to Lo) | Yes | Yes | Yes | Yes | Yes |
| BGP session (IPv6 addressed) with IPv4 SAFI (rfc5549) with static ASN (BP has IPv6 app enabled) | 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 |

BGP to IP Endpoint CT Type

| BGP to IP Endpoint CT Type | EOS | NX-OS | 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 |
| BGP session from L3 sub-interface to any IP endpoint in the network (IPv6, default/non-default RZ) | Yes | Yes | Yes | Yes | Yes |
| BGP session from SVI to any IP endpoint in the network (IPv4, default/non-default RZ) | Yes | Yes | Yes | Yes | Yes |
| BGP session from SVI to any IP endpoint in the network (IPv6, non-default RZ) | Yes | Yes | Yes | Yes | Yes |
| BGP session from SVI to any IP endpoint in the network (IPv6, default RZ) | Yes | Yes | Yes | Yes | Yes |

(Continued)

| BGP to IP Endpoint CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|--|-----|-------|-------|----------|------------------|
| BGP session from Loopback to any IP endpoint in the network (IPv4, default/non-default RZ) | 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 |
| BGP session with specific peer IP and Static ASN (IPv4) | Yes | Yes | Yes | Yes | Yes |
| BGP session with specific peer IP and Static ASN (IPv6) | Yes | Yes | Yes | Yes | Yes |
| BGP session with specific peer IP and dynamic ASN (IPv4) | No | No | No | No | No |

(Continued)

| BGP to IP Endpoint CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|--|-----|-------|-------|----------|------------------|
| BGP session with specific peer IP and dynamic ASN (IPv6) | No | No | No | No | No |
| BGP session (IPv6 addressed) with IPv4 SAFI (rfc5549) with static ASN (BP has IPv6 app enabled) | 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 |

Dynamic BGP Peering CT Type

| Dynamic BGP Peering CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|---|-----|-------|-------|----------|------------------|
| Dynamic BGP prefix peering on SVI (IPv4), default VRF | Yes | Yes | Yes | Yes | No |

(Continued)

| Dynamic BGP Peering CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|--|-----|-------|-------|----------|------------------|
| Dynamic BGP prefix peering on SVI (IPv4), non-default VRF | Yes | Yes | Yes | Yes | No |
| Dynamic BGP prefix peering on SVI (IPv6), default VRF | Yes | Yes | Yes | Yes | No |
| Dynamic BGP prefix peering on SVI (IPv6), non-default VRF | Yes | Yes | Yes | Yes | No |
| Dynamic BGP prefix peering on L3 sub-interface (IPv4), default VRF | Yes | Yes | Yes | Yes | No |
| Dynamic BGP prefix peering on L3 sub-interface (IPv4), non-default VRF | Yes | Yes | Yes | Yes | No |
| Dynamic BGP prefix peering on L3 sub-interface (IPv6), default VRF | Yes | Yes | Yes | Yes | No |

(Continued)

| Dynamic BGP Peering CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|---|-----|-------|-------|----------|------------------|
| Dynamic BGP prefix peering on L3 sub-interface (IPv6), non-default VRF | Yes | Yes | Yes | Yes | No |
| Dynamic prefix peering (link-local prefix peering, rfc5549), (BP has IPv6 app disabled) | Yes | No | No | No | No |
| Dynamic prefix peering (IPv6 peering, IPv4 AFI, rfc5549), (BP has IPv6 app enabled) | No | No | No | No | No |

Routing Policy CT Type

| Routing Policy CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|--|-----|-------|-------|----------|------------------|
| Routing Policy on a BGP session with import/export IPv4 prefixes | Yes | Yes | Yes | Yes | Yes |

(Continued)

| Routing Policy CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|--|-----|-------|-------|----------|------------------|
| Routing Policy on a BGP session with import/export IPv6 prefixes | Yes | Yes | Yes | Yes | Yes |
| Routing Policy on a BGP session with IPv4 aggregate prefixes | Yes | Yes | Yes | Yes | Yes |
| Routing Policy on a BGP session with IPv6 aggregate prefixes | Yes | Yes | Yes | Yes | Yes |

BGP Attributes (common to all BGP CTs)

| BGP Attributes (common to all BGP CTs) | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|---|-----|-------|-------|----------|------------------|
| BGP: enable Password/MD5 based authentication | Yes | Yes | Yes | Yes | Yes |
| BGP: Custom BGP timers (Keep Alive timer, Hold timer) | Yes | Yes | Yes | Yes | Yes |
| BGP: Custom TTL | Yes | Yes | Yes | Yes | Yes |

(Continued)

| BGP Attributes (common to all BGP CTs) | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|--|-----|-------|-------|----------|---------------------|
| BGP: Enable Single-hop BFD | Yes | Yes | Yes | Yes | Yes |

DCI Features

| DCI Features | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|---------------------------|-----|-------|-------|----------|---------------------|
| Type 5 route filtering | No | Yes | No | Yes | Yes |

Apstra 4.1.1 Feature Matrix**IN THIS SECTION**

- [Fabric Roles | 940](#)
- [Fabric Connectivity | 940](#)
- [Device Management | 941](#)
- [Connectivity \(from Leaf Layer\) | 942](#)
- [Connectivity \(from Access Layer\) | 943](#)
- [Routing Policies | 943](#)
- [Miscellaneous | 944](#)
- [Virtual Network CT Type | 944](#)
- [IP Link CT Type | 945](#)
- [Static Route CT Type | 947](#)
- [Custom Static Route CT Type | 948](#)
- [BGP to Generic CT Type | 948](#)
- [BGP to IP Endpoint CT Type | 953](#)
- [Dynamic BGP Peering CT Type | 956](#)

- [Routing Policy CT Type | 958](#)
- [BGP Attributes \(common to all BGP CTs\) | 959](#)
- [DCI Features | 959](#)

Fabric Roles

| Fabric Roles | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|---|-----|-------|-------|----------|------------------|
| Access Switch | No | No | No | Yes | No |
| Non-EVPN-VXLAN Leaf (IP forwarder only) | Yes | Yes | Yes | Yes | Yes |
| EVPN-VXLAN Leaf | Yes | Yes | Yes | Yes | No |
| Spine or Superspine | Yes | Yes | Yes | Yes | Yes |

Fabric Connectivity

| Fabric Connectivity | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|----------------------------|-----|-------|-------|----------|------------------|
| 3-stage Clos | Yes | Yes | Yes | Yes | Yes |
| 5-stage Clos | Yes | Yes | Yes | Yes | Yes |
| IP fabric (non-EVPN-VXLAN) | Yes | Yes | Yes | Yes | Yes |
| EVPN-VXLAN fabric | Yes | Yes | Yes | Yes | Limited |

(Continued)

| Fabric Connectivity | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|---------------------------------------|------------|--------------|--------------|-----------------|-------------------------|
| IPv6 fabric RFC-5549 (non-EVPN) | Yes | Yes | Yes | No | No |
| 3-stage Clos with access switch layer | No | No | No | Yes | Limited |
| Collapsed fabric | No | No | No | Yes | No |
| Freeform | No | No | No | Yes | Yes |

Device Management

| Device Management | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|--|------------|--------------|-------------------------|-----------------|-------------------------|
| On-box agent | Yes | Yes | Yes | Not possible | No |
| Off-box agent | Yes | Yes | No | Yes | Yes |
| Telemetry extensibility | Yes | Yes | Yes | Yes | No |
| Apstra ZTP | Yes | Yes | Yes | Yes | Yes |
| Device OS upgrade | Yes | Yes | Contact Juniper Support | Yes | Yes |
| Traffic draining (spine devices/superspine devices - maintenance mode) | Yes | Yes | Yes | Yes | Yes |

(Continued)

| Device Management | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|---------------------------------|---------|---------|---------|----------|------------------|
| Traffic draining (leaf devices) | Limited | Limited | Limited | Limited | Limited |

Connectivity (from Leaf Layer)

| Connectivity (from Leaf Layer) | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|--|-----|-------|--------------|--------------|------------------|
| LAG | Yes | Yes | Yes | Yes | Yes |
| MLAG/vPC | Yes | Yes | Yes | Not possible | Not possible |
| EVPN ESI (with LACP) | No | No | Not possible | Yes | No |
| 802.1x | Yes | No | Not possible | Limited | No |
| VLANs | Yes | Yes | Yes | Yes | Yes |
| Overlay protocol: static VXLAN | Yes | Yes | Not possible | No | No |
| Overlay protocol: EVPN (3-stage and 5-stage) | Yes | Yes | Yes | Yes | No |
| IPv4 DHCP relay | Yes | Yes | Yes | Yes | Yes |
| IPv6 DHCP relay | Yes | Yes | Yes | Yes | Yes |
| EVPN DCI | Yes | Yes | Yes | Yes | No |

(Continued)

| Connectivity (from Leaf Layer) | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|--|-----|-------|-------|----------|------------------|
| IPv6 for applications (with EVPN and IPv4 fabric) | Yes | Yes | Yes | Yes | No |
| Policy Assurance (L3 ACLs) | Yes | Yes | No | Yes | Yes |

Connectivity (from Access Layer)

| Connectivity (from Access Layer) | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|-------------------------------------|-----|-------|-------|----------|------------------|
| LAG | N/A | N/A | N/A | Yes | N/A |
| ESI LAG | N/A | N/A | N/A | Limited | N/A |

Routing Policies

| Routing Policies | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|---|-----|-------|-------|----------|------------------|
| Import all routes or default route or extra routes only | Yes | Yes | Yes | Yes | Yes |
| Export loopback, link and VN IP. Export extra routes | Yes | Yes | Yes | Yes | Yes |

(Continued)

| Routing Policies | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|-------------------------------------|-----|-------|-------|----------|------------------|
| Export aggregate prefixes | Yes | Yes | Yes | Yes | No |
| Export L3 server link subnets | Yes | Yes | Yes | Yes | Yes |
| Route target import/export policies | Yes | Yes | Yes | Yes | No |

Miscellaneous

| Miscellaneous | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|---------------------------------------|-----|-------|-------|----------|------------------|
| Configlets | Yes | Yes | Yes | Yes | Yes |
| FFE: add racks/add links/change speed | Yes | Yes | Yes | Yes | Yes |
| Mixed leaf/spine link speed | Yes | Yes | Yes | Yes | Yes |

Virtual Network CT Type

| Virtual Network CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|--------------------------------------|-----|-------|-------|----------|------------------|
| Single Virtual Network (VXLAN-based) | Yes | Yes | Yes | Yes | No |

(Continued)

| Virtual Network CT Type | EOS | NX-OS | SONiC | Juno OS | Junos OS Evolved |
|--|-----|-------|-------|---------|------------------|
| Multiple Virtual Network (VXLAN-based) | Yes | Yes | Yes | Yes | No |
| VLAN (default VRF, non-VXLAN) | Yes | Yes | Yes | Yes | Yes |

IP Link CT Type

| IP Link CT Type | EOS | NX-OS | 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 |
| L3 Sub-interface on non-LAG physical interface (untagged/vlan tagged, default/non-default RZ, IPv6) | Yes | Yes | Yes | Yes | No |

(Continued)

| IP Link CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|---|-----|-------|-------|----------|------------------|
| L3 Sub-interface on LAG interface (untagged/vlan tagged, default/non-default RZ, IPv4) | Yes | Yes | Yes | Yes | Yes |
| L3 Sub-interface on LAG interface (untagged/vlan tagged, default/non-default RZ, IPv6) | Yes | Yes | Yes | Yes | Yes |
| L3 Sub-interface on LAG interface (untagged/vlan tagged, default RZ, IPv4) - spine/sspine | Yes | Yes | Yes | Yes | Yes |
| L3 Sub-interface on LAG interface (untagged/vlan tagged, default RZ, IPv6) - spine/sspine | Yes | Yes | Yes | Yes | Yes |

Static Route CT Type

| Static Route CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|--|-----|-------|-------|----------|------------------|
| Static Route (IPv4) applied on L3 Sub-interface | Yes | Yes | Yes | Yes | Yes |
| Static Route (IPv6) applied on L3 Sub-interface | Yes | Yes | Yes | Yes | Yes |
| Static Route (IPv4) applied on SVI | Yes | Yes | Yes | Yes | No |
| Static Route (IPv6) applied on SVI | Yes | Yes | Yes | Yes | No |
| Static Route with Share IP Endpoint Enabled (IPv4) | Yes | Yes | Yes | Yes | No |
| Static Route with Share IP Endpoint Enabled (IPv6) | Yes | Yes | Yes | Yes | No |

Custom Static Route CT Type

| Custom Static Route CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|--|-----|-------|-------|----------|------------------|
| Custom Static Route (IPv4, default/non-default RZ) | Yes | Yes | Yes | Yes | Yes |
| Custom Static Route (IPv6, default/non-default RZ) | Yes | Yes | Yes | Yes | Yes |

BGP to Generic CT Type

| BGP to Generic CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|--|-----|-------|-------|----------|------------------|
| BGP session on L3 Sub-interface towards generic (IPv4, default/non-default RZ) | Yes | Yes | Yes | Yes | Yes |
| BGP session on L3 Sub-interface towards generic (IPv6, default/non-default RZ) | Yes | Yes | Yes | Yes | No |
| BGP session on SVI towards generic (IPv4, default RZ) | Yes | Yes | Yes | Yes | No |

(Continued)

| BGP to Generic CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|--|-----|-------|--------------|--------------|------------------|
| BGP session on SVI towards generic (IPv4, non-default RZ) | Yes | Yes | Yes | Yes | No |
| BGP session on SVI towards generic (IPv6, non-default RZ) | Yes | Yes | Yes | Yes | No |
| BGP session on SVI towards generic (IPv6, default RZ) | Yes | Yes | Yes | Yes | No |
| BGP session on SVI (mlag) towards dual-homed generic using secondary IPs (IPv4, default VRF) | 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 | Not possible | Yes | No |
| BGP session on SVI (mlag) towards dual-homed generic using secondary IPs (IPv6, default VRF) | Yes | Yes | Not possible | Not possible | No |

(Continued)

| BGP to Generic CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|--|-----|-------|--------------|----------|------------------|
| BGP session on SVI (mlag) towards dual-homed generic using secondary IPs (IPv6, non-default VRF) | Yes | Yes | Not possible | Yes | No |
| BGP session to generic with Share IP Endpoint Enabled (IPv4) | Yes | Yes | Yes | Yes | No |
| BGP session to generic with Share IP Endpoint Enabled (IPv6) | Yes | Yes | Yes | Yes | No |
| BGP session to generic with dynamic ASN (IPv4) | No | No | No | No | No |
| BGP session to generic with Static ASN (IPv4) | Yes | Yes | Yes | Yes | No |
| BGP session to generic with dynamic ASN (IPv6) | No | No | No | No | No |
| BGP session to generic with static ASN (IPv6) | Yes | Yes | Yes | Yes | No |

(Continued)

| BGP to Generic CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|--|-----|-------|-------|----------|------------------|
| BGP Unnumbered session (link-local peering) on L3 Sub-interface (BP has IPv6 app enabled, default VRF) | 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 | No | No |
| BGP Unnumbered session (link-local peering) on SVI (BP has IPv6 app enabled, default VRF) | Yes | Yes | Yes | No | No |
| BGP Unnumbered session (link-local peering) on SVI (BP has IPv6 app enabled, non-default VRF) | No | Yes | Yes | No | No |

(Continued)

| BGP to Generic CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|---|-----|-------|-------|----------|------------------|
| BGP Unnumbered session (link-local peering) on L3 Sub-interface (default VRF, BP has IPv6 app disabled) | 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 | No | No |
| BGP Unnumbered session (link-local peering) on SVI (BP has IPv6 app disabled, default VRF only) | No | No | No | No | No |
| BGP Peering combinations (Int to Int, Lo to Int, Int to Lo, Lo to Lo) | Yes | Yes | Yes | Yes | Yes |

(Continued)

| BGP to Generic CT Type | EOS | NX-OS | 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 |
| BGP session (IPv6 addressed) with IPv4 SAFI (rfc5549) with dynamic ASN (BP has IPv6 app enabled) | No | No | No | No | No |

BGP to IP Endpoint CT Type

| BGP to IP Endpoint CT Type | EOS | NX-OS | 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 |
| BGP session from L3 sub-interface to any IP endpoint in the network (IPv6, default/non-default RZ) | Yes | Yes | Yes | Yes | No |

(Continued)

| BGP to IP Endpoint CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|--|-----|-------|-------|----------|------------------|
| BGP session from SVI to any IP endpoint in the network (IPv4, default/non-default RZ) | Yes | Yes | Yes | Yes | No |
| BGP session from SVI to any IP endpoint in the network (IPv6, non-default RZ) | Yes | Yes | Yes | Yes | No |
| BGP session from SVI to any IP endpoint in the network (IPv6, default RZ) | 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 |
| BGP session from Loopback to any IP endpoint in the network (IPv6, default/non-default RZ) | Yes | Yes | Yes | Yes | No |

(Continued)

| BGP to IP Endpoint CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|---|-----|-------|-------|----------|------------------|
| BGP session with specific peer IP and and Static ASN (IPv4) | Yes | Yes | Yes | Yes | Yes |
| BGP session with specific peer IP and and Static ASN (IPv6) | Yes | Yes | Yes | Yes | No |
| BGP session with specific peer IP and and dynamic ASN (IPv4) | No | No | No | No | No |
| BGP session with specific peer IP and and dynamic ASN (IPv6) | No | No | No | No | No |
| BGP session (IPv6 addressed) with IPv4 SAFI (rfc5549) with static ASN (BP has IPv6 app enabled) | No | No | No | No | No |

(Continued)

| BGP to IP Endpoint CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|--|-----|-------|-------|----------|------------------|
| BGP session (IPv6 addressed) with IPv4 SAFI (rfc5549) with dynamic ASN (BP has IPv6 app enabled) | No | No | No | No | No |

Dynamic BGP Peering CT Type

| Dynamic BGP Peering CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|---|-----|-------|-------|----------|------------------|
| Dynamic BGP prefix peering on SVI (IPv4), default VRF | Yes | Yes | Yes | Yes | No |
| Dynamic BGP prefix peering on SVI (IPv4), non-default VRF | Yes | Yes | Yes | Yes | No |
| Dynamic BGP prefix peering on SVI (IPv6), default VRF | Yes | Yes | Yes | Yes | No |
| Dynamic BGP prefix peering on SVI (IPv6), non-default VRF | Yes | Yes | Yes | Yes | No |

(Continued)

| Dynamic BGP Peering CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|---|-----|-------|-------|----------|------------------|
| Dynamic BGP prefix peering on L3 sub-interface (IPv4), default VRF | Yes | Yes | Yes | Yes | No |
| Dynamic BGP prefix peering on L3 sub-interface (IPv4), non-default VRF | Yes | Yes | Yes | Yes | No |
| Dynamic BGP prefix peering on L3 sub-interface (IPv6), default VRF | Yes | Yes | Yes | Yes | No |
| Dynamic BGP prefix peering on L3 sub-interface (IPv6), non-default VRF | Yes | Yes | Yes | Yes | No |
| Dynamic prefix peering (link-local prefix peering, rfc5549), (BP has IPv6 app disabled) | Yes | No | No | No | No |

(Continued)

| Dynamic BGP Peering CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|---|-----|-------|-------|----------|------------------|
| Dynamic prefix peering (IPv6 peering, IPv4 AFI, rfc5549), (BP has IPv6 app enabled) | No | No | No | No | No |

Routing Policy CT Type

| Routing Policy CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|--|-----|-------|-------|----------|------------------|
| Routing Policy on a BGP session with import/export IPv4 prefixes | Yes | Yes | Yes | Yes | Yes |
| Routing Policy on a BGP session with import/export IPv6 prefixes | Yes | Yes | Yes | Yes | No |
| Routing Policy on a BGP session with IPv4 aggregate prefixes | Yes | Yes | Yes | Yes | No |
| Routing Policy on a BGP session with IPv6 aggregate prefixes | Yes | Yes | Yes | Yes | No |

BGP Attributes (common to all BGP CTs)

| BGP Attributes (common to all BGP CTs) | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|---|-----|-------|-------|----------|---------------------|
| BGP: enable Password/MD5 based authentication | Yes | Yes | Yes | Yes | No |
| BGP: Custom BGP timers (Keep Alive timer, Hold timer) | Yes | Yes | Yes | Yes | Yes |
| BGP: Custom TTL | Yes | Yes | Yes | Yes | Yes |
| BGP: Enable Single-hop BFD | Yes | Yes | Yes | Yes | Yes |

DCI Features

| DCI Features | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|---------------------------|-----|-------|-------|----------|---------------------|
| Type 5 route filtering | No | Yes | No | Yes | No |

Apstra 4.1.0 Feature Matrix

IN THIS SECTION

- [Fabric Roles | 960](#)
- [Fabric Connectivity | 961](#)
- [Device Management | 961](#)

- [Connectivity \(from Leaf Layer\) | 962](#)
- [Connectivity \(from Access Layer\) | 963](#)
- [Routing Policies | 964](#)
- [Miscellaneous | 964](#)
- [Virtual Network CT Type | 965](#)
- [IP Link CT Type | 965](#)
- [Static Route CT Type | 967](#)
- [Custom Static Route CT Type | 968](#)
- [BGP to Generic CT Type | 968](#)
- [BGP to IP Endpoint CT Type | 973](#)
- [Dynamic BGP Peering CT Type | 976](#)
- [Routing Policy CT Type | 978](#)
- [BGP Attributes \(common to all BGP CTs\) | 979](#)
- [DCI Features | 979](#)

Fabric Roles

| Fabric Roles | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|---|-----|-------|-------|----------|------------------|
| Access Switch | No | No | No | Yes | No |
| Non-EVPN-VXLAN Leaf (IP forwarder only) | Yes | Yes | Yes | Yes | Yes |
| EVPN-VXLAN Leaf | Yes | Yes | Yes | Yes | No |
| Spine or Superspine | Yes | Yes | Yes | Yes | Yes |

Fabric Connectivity

| Fabric Connectivity | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|---------------------------------------|-----|-------|-------|----------|------------------|
| 3-stage Clos | Yes | Yes | Yes | Yes | Yes |
| 5-stage Clos | Yes | Yes | Yes | Yes | Yes |
| IP fabric (non-EVPN-VXLAN) | Yes | Yes | Yes | Yes | Yes |
| EVPN-VXLAN fabric | Yes | Yes | Yes | Yes | Limited |
| IPv6 fabric RFC-5549 (non-EVPN) | Yes | Yes | Yes | No | No |
| 3-stage Clos with access switch layer | No | No | No | Yes | Limited |
| Collapsed fabric | No | No | No | Yes | No |

Device Management

| Device Management | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|-------------------------|-----|-------|-------|--------------|------------------|
| On-box agent | Yes | Yes | Yes | Not possible | No |
| Off-box agent | Yes | Yes | No | Yes | Yes |
| Telemetry extensibility | Yes | Yes | Yes | Yes | No |
| Apstra ZTP | Yes | Yes | Yes | Yes | Yes |

(Continued)

| Device Management | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|---|---------|---------|-------------------------|----------|------------------|
| Device OS upgrade | Yes | Yes | Contact Juniper Support | Yes | Yes |
| Traffic draining (spine devices/ superspine devices - maintenance mode) | Yes | Yes | Yes | Yes | Yes |
| Traffic draining (leaf devices) | Limited | Limited | Limited | Limited | Limited |

Connectivity (from Leaf Layer)

| Connectivity (from Leaf Layer) | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|--------------------------------|-----|-------|--------------|--------------|------------------|
| LAG | Yes | Yes | Yes | Yes | Yes |
| MLAG/vPC | Yes | Yes | Yes | Not possible | Not possible |
| EVPN ESI (with LACP) | No | No | Not possible | Yes | No |
| 802.1x | Yes | No | Not possible | No | No |
| VLANs | Yes | Yes | Yes | Yes | Yes |
| Overlay protocol: static VXLAN | Yes | Yes | Not possible | No | No |

(Continued)

| Connectivity (from Leaf Layer) | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|--|-----|-------|-------|----------|---------------------|
| Overlay protocol: EVPN (3-stage and 5- stage) | Yes | Yes | Yes | Yes | No |
| IPv4 DHCP relay | Yes | Yes | Yes | Yes | Yes |
| IPv6 DHCP relay | Yes | Yes | Yes | Yes | Yes |
| EVPN DCI | Yes | Yes | Yes | Yes | No |
| IPv6 for applications (with EVPN and IPv4 fabric) | Yes | Yes | Yes | Yes | No |
| Policy Assurance (L3 ACLs) | Yes | Yes | No | Yes | Yes |

Connectivity (from Access Layer)

| Connectivity (from Access Layer) | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|--|-----|-------|-------|----------|---------------------|
| LAG | N/A | N/A | N/A | Yes | N/A |
| ESI LAG | N/A | N/A | N/A | Limited | N/A |

Routing Policies

| Routing Policies | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|---|-----|-------|-------|----------|------------------|
| Import all routes or default route or extra routes only | Yes | Yes | Yes | Yes | Yes |
| Export loopback, link and VN IP. Export extra routes | Yes | Yes | Yes | Yes | Yes |
| Export aggregate prefixes | Yes | Yes | Yes | Yes | No |
| Export L3 server link subnets | Yes | Yes | Yes | Yes | Yes |
| Route target import/export policies | Yes | Yes | Yes | Yes | No |

Miscellaneous

| Miscellaneous | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|---------------------------------------|-----|-------|-------|----------|------------------|
| Configlets | Yes | Yes | Yes | Yes | Yes |
| FFE: add racks/add links/change speed | Yes | Yes | Yes | Yes | Yes |
| Mixed leaf/spine link speed | Yes | Yes | Yes | Yes | Yes |

Virtual Network CT Type

| Virtual Network CT Type | EOS | NX-OS | SONiC | Juno OS | Junos OS Evolved |
|--|-----|-------|-------|---------|------------------|
| Single Virtual Network (VXLAN-based) | Yes | Yes | Yes | Yes | No |
| Multiple Virtual Network (VXLAN-based) | Yes | Yes | Yes | Yes | No |
| VLAN (default VRF, non-VXLAN) | Yes | Yes | Yes | Yes | Yes |

IP Link CT Type

| IP Link CT Type | EOS | NX-OS | 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 - for default VRF only |
| L3 Sub-interface on non-LAG physical interface (untagged/vlan tagged, default/non-default RZ, IPv6) | Yes | Yes | Yes | Yes | No |

(Continued)

| IP Link CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|---|-----|-------|-------|----------|------------------|
| L3 Sub-interface on LAG interface (untagged/vlan tagged, default/non-default RZ, IPv4) | Yes | Yes | Yes | Yes | Yes |
| L3 Sub-interface on LAG interface (untagged/vlan tagged, default/non-default RZ, IPv6) | Yes | Yes | Yes | Yes | Yes |
| L3 Sub-interface on LAG interface (untagged/vlan tagged, default RZ, IPv4) - spine/sspine | Yes | Yes | Yes | Yes | Yes |
| L3 Sub-interface on LAG interface (untagged/vlan tagged, default RZ, IPv6) - spine/sspine | Yes | Yes | Yes | Yes | Yes |

Static Route CT Type

| Static Route CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|--|-----|-------|-------|----------|------------------|
| Static Route (IPv4) applied on L3 Sub-interface | Yes | Yes | Yes | Yes | Yes |
| Static Route (IPv6) applied on L3 Sub-interface | Yes | Yes | Yes | Yes | Yes |
| Static Route (IPv4) applied on SVI | Yes | Yes | Yes | Yes | No |
| Static Route (IPv6) applied on SVI | Yes | Yes | Yes | Yes | No |
| Static Route with Share IP Endpoint Enabled (IPv4) | Yes | Yes | Yes | Yes | No |
| Static Route with Share IP Endpoint Enabled (IPv6) | Yes | Yes | Yes | Yes | No |

Custom Static Route CT Type

| Custom Static Route CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|--|-----|-------|-------|----------|------------------------|
| Custom Static Route (IPv4, default/non-default RZ) | Yes | Yes | Yes | Yes | Yes - default VRF only |
| Custom Static Route (IPv6, default/non-default RZ) | Yes | Yes | Yes | Yes | Yes |

BGP to Generic CT Type

| BGP to Generic CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|--|-----|-------|-------|----------|------------------|
| BGP session on L3 Sub-interface towards generic (IPv4, default/non-default RZ) | Yes | Yes | Yes | Yes | Yes |
| BGP session on L3 Sub-interface towards generic (IPv6, default/non-default RZ) | Yes | Yes | Yes | Yes | No |
| BGP session on SVI towards generic (IPv4, default RZ) | Yes | Yes | Yes | Yes | No |

(Continued)

| BGP to Generic CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|--|-----|-------|--------------|--------------|------------------|
| BGP session on SVI towards generic (IPv4, non-default RZ) | Yes | Yes | Yes | Yes | No |
| BGP session on SVI towards generic (IPv6, non-default RZ) | Yes | Yes | Yes | Yes | No |
| BGP session on SVI towards generic (IPv6, default RZ) | Yes | Yes | Yes | Yes | No |
| BGP session on SVI (mlag) towards dual-homed generic using secondary IPs (IPv4, default VRF) | 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 | Not possible | Yes | No |
| BGP session on SVI (mlag) towards dual-homed generic using secondary IPs (IPv6, default VRF) | Yes | Yes | Not possible | Not possible | No |

(Continued)

| BGP to Generic CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|--|-----|-------|--------------|----------|------------------|
| BGP session on SVI (mlag) towards dual-homed generic using secondary IPs (IPv6, non-default VRF) | Yes | Yes | Not possible | Yes | No |
| BGP session to generic with Share IP Endpoint Enabled (IPv4) | Yes | Yes | Yes | Yes | No |
| BGP session to generic with Share IP Endpoint Enabled (IPv6) | Yes | Yes | Yes | Yes | No |
| BGP session to generic with dynamic ASN (IPv4) | No | No | No | No | No |
| BGP session to generic with Static ASN (IPv4) | Yes | Yes | Yes | Yes | No |
| BGP session to generic with dynamic ASN (IPv6) | No | No | No | No | No |
| BGP session to generic with static ASN (IPv6) | Yes | Yes | Yes | Yes | No |

(Continued)

| BGP to Generic CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|--|-----|-------|-------|----------|------------------|
| BGP Unnumbered session (link-local peering) on L3 Sub-interface (BP has IPv6 app enabled, default VRF) | 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 | No | No |
| BGP Unnumbered session (link-local peering) on SVI (BP has IPv6 app enabled, default VRF) | Yes | Yes | Yes | No | No |
| BGP Unnumbered session (link-local peering) on SVI (BP has IPv6 app enabled, non-default VRF) | No | Yes | Yes | No | No |

(Continued)

| BGP to Generic CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|---|-----|-------|-------|----------|------------------|
| BGP Unnumbered session (link-local peering) on L3 Sub-interface (default VRF, BP has IPv6 app disabled) | 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 | No | No |
| BGP Unnumbered session (link-local peering) on SVI (BP has IPv6 app disabled, default VRF only) | No | No | No | No | No |
| BGP Peering combinations (Int to Int, Lo to Int, Int to Lo, Lo to Lo) | Yes | Yes | Yes | Yes | Yes |

(Continued)

| BGP to Generic CT Type | EOS | NX-OS | 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 |
| BGP session (IPv6 addressed) with IPv4 SAFI (rfc5549) with dynamic ASN (BP has IPv6 app enabled) | No | No | No | No | No |

BGP to IP Endpoint CT Type

| BGP to IP Endpoint CT Type | EOS | NX-OS | 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 |
| BGP session from L3 sub-interface to any IP endpoint in the network (IPv6, default/non-default RZ) | Yes | Yes | Yes | Yes | No |

(Continued)

| BGP to IP Endpoint CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|--|-----|-------|-------|----------|------------------|
| BGP session from SVI to any IP endpoint in the network (IPv4, default/non-default RZ) | Yes | Yes | Yes | Yes | No |
| BGP session from SVI to any IP endpoint in the network (IPv6, non-default RZ) | Yes | Yes | Yes | Yes | No |
| BGP session from SVI to any IP endpoint in the network (IPv6, default RZ) | 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 |
| BGP session from Loopback to any IP endpoint in the network (IPv6, default/non-default RZ) | Yes | Yes | Yes | Yes | No |

(Continued)

| BGP to IP Endpoint CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|---|-----|-------|-------|----------|------------------|
| BGP session with specific peer IP and and Static ASN (IPv4) | Yes | Yes | Yes | Yes | Yes |
| BGP session with specific peer IP and and Static ASN (IPv6) | Yes | Yes | Yes | Yes | No |
| BGP session with specific peer IP and and dynamic ASN (IPv4) | No | No | No | No | No |
| BGP session with specific peer IP and and dynamic ASN (IPv6) | No | No | No | No | No |
| BGP session (IPv6 addressed) with IPv4 SAFI (rfc5549) with static ASN (BP has IPv6 app enabled) | No | No | No | No | No |

(Continued)

| BGP to IP Endpoint CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|--|-----|-------|-------|----------|------------------|
| BGP session (IPv6 addressed) with IPv4 SAFI (rfc5549) with dynamic ASN (BP has IPv6 app enabled) | No | No | No | No | No |

Dynamic BGP Peering CT Type

| Dynamic BGP Peering CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|---|-----|-------|-------|----------|------------------|
| Dynamic BGP prefix peering on SVI (IPv4), default VRF | Yes | Yes | Yes | Yes | No |
| Dynamic BGP prefix peering on SVI (IPv4), non-default VRF | Yes | Yes | Yes | Yes | No |
| Dynamic BGP prefix peering on SVI (IPv6), default VRF | Yes | Yes | Yes | Yes | No |
| Dynamic BGP prefix peering on SVI (IPv6), non-default VRF | Yes | Yes | Yes | Yes | No |

(Continued)

| Dynamic BGP Peering CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|---|-----|-------|-------|----------|------------------|
| Dynamic BGP prefix peering on L3 sub-interface (IPv4), default VRF | Yes | Yes | Yes | Yes | No |
| Dynamic BGP prefix peering on L3 sub-interface (IPv4), non-default VRF | Yes | Yes | Yes | Yes | No |
| Dynamic BGP prefix peering on L3 sub-interface (IPv6), default VRF | Yes | Yes | Yes | Yes | No |
| Dynamic BGP prefix peering on L3 sub-interface (IPv6), non-default VRF | Yes | Yes | Yes | Yes | No |
| Dynamic prefix peering (link-local prefix peering, rfc5549), (BP has IPv6 app disabled) | Yes | No | No | No | No |

(Continued)

| Dynamic BGP Peering CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|---|-----|-------|-------|----------|------------------|
| Dynamic prefix peering (IPv6 peering, IPv4 AFI, rfc5549), (BP has IPv6 app enabled) | No | No | No | No | No |

Routing Policy CT Type

| Routing Policy CT Type | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|--|-----|-------|-------|----------|------------------|
| Routing Policy on a BGP session with import/export IPv4 prefixes | Yes | Yes | Yes | Yes | Yes |
| Routing Policy on a BGP session with import/export IPv6 prefixes | Yes | Yes | Yes | Yes | No |
| Routing Policy on a BGP session with IPv4 aggregate prefixes | Yes | Yes | Yes | Yes | No |
| Routing Policy on a BGP session with IPv6 aggregate prefixes | Yes | Yes | Yes | Yes | No |

BGP Attributes (common to all BGP CTs)

| BGP Attributes (common to all BGP CTs) | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|---|-----|-------|-------|----------|---------------------|
| BGP: enable Password/MD5 based authentication | Yes | Yes | Yes | Yes | No |
| BGP: Custom BGP timers (Keep Alive timer, Hold timer) | Yes | Yes | Yes | Yes | Yes |
| BGP: Custom TTL | Yes | Yes | Yes | Yes | Yes |
| BGP: Enable Single-hop BFD | Yes | Yes | Yes | Yes | Yes |

DCI Features

| DCI Features | EOS | NX-OS | SONiC | Junos OS | Junos OS Evolved |
|---------------------------|-----|-------|-------|----------|---------------------|
| Type 5 route filtering | No | Yes | No | Yes | No |

Qualified Devices and NOS Versions

IN THIS SECTION

● [Apstra Release 4.1.2 | 981](#)

- [Apstra Release 4.1.1 | 984](#)
- [Apstra Release 4.1.0 | 986](#)

Recommended qualified NOS versions and device (series) are listed below. Other versions in the same code train that contain only bug fixes are also expected to work. This is usually indicated with version numbers that differ only by the last digit; however, this is not strictly guaranteed by the NOS vendors.

If you plan to use a device or NOS version close to the qualified ones but not listed, it is highly recommended that you review the NOS release notes to ensure no backward incompatible or breaking changes are listed. We strongly advise testing the new version thoroughly in a staging environment before deploying it to production.

To request consideration for qualification for a release train not listed, contact your Juniper Apstra Sales representative.

Only Bug Fix NOS versions Examples:

- Junos and Junos Evolved
 - 20.2R2-S1 > 20.2R2-S3.5 (reason: only service release number change)
 - 20.2R2 > 20.2R3 (reason: R2 > R3 expected to contain only bugfixes)
- Arista EOS
 - 4.25.4M > 4.25.5M (reason: same code train, last digit change and M indicates Maintenance release)
- Cisco NXOS
 - 10.2(9)M > 10.2(10)M (reason: same code train, last digit change and M indicates Maintenance release)

Non-Bug Fix Examples:

- Junos and Junos Evolved
 - 20.2R1 > 20.2R2 (reason: R1 > R2 can have new features + bugfixes)
 - 20.2R2 > 20.4R2 (reason: different release trains)
- Arista EOS
 - 4.25.4M > 4.26.5M (reason: different release trains)
- Cisco NXOS

- 10.2(1)F > 10.2(3)F (reason: multiple last digit change, F indicates Feature release)

Apstra Release 4.1.2

| Device Operating System | Qualified NOS Versions | Supported Device (Series) |
|-------------------------|--|---|
| Juniper Junos OS | <ul style="list-style-type: none"> • 22.2R3 • 21.4R3 • 21.2R3 • 20.4R3 | <ul style="list-style-type: none"> • QFX5100 - Don't use as leaf with Layer 3 VNI • QFX5110 - Can't be used as a border leaf. It can't route between VXLAN IRB and L3 interface. • QFX5120 • QFX5200 • QFX5210 • QFX10002 • QFX10008 • QFX10016 • EX4650-48Y • EX4400-24MP • EX4400-24T • EX4400-48F • EX4400-48MP • EX4400-48T |

(Continued)

| Device Operating System | Qualified NOS Versions | Supported Device (Series) |
|--------------------------|---|--|
| Juniper Junos OS Evolved | <ul style="list-style-type: none"> IP Forwarder <ul style="list-style-type: none"> 22.2R3-EVO 21.4R3-EVO 21.2R3-EVO 20.4R3-EVO EVPN-VXLAN Leaf <ul style="list-style-type: none"> 22.2R3-EVO | <ul style="list-style-type: none"> QFX5700 chassis (5RU, 8-slots, 3 line card types) QFX5220 QFX5130 PTX10001-36MR PTX10004 chassis (7RU, 4-slots) PTX10008 chassis (13RU, 8-slots) PTX10016 chassis (21RU, 16-slots) ACX7100-32C ACX7100-48L |

(Continued)

| Device Operating System | Qualified NOS Versions | Supported Device (Series) |
|-------------------------|--|--|
| Enterprise SONiC | <ul style="list-style-type: none"> • SONiC-OS-4.0.5-GA-Enterprise-Advanced • SONiC-OS-4.0.5-GA-Enterprise-Base • SONiC-OS-3.5.4-GA-Enterprise-Advanced • SONiC-OS-3.5.4-GA-Enterprise-Base | <ul style="list-style-type: none"> • Dell Z9432F-ON (spine role) • Dell Z9332F-ON • Dell Z9264F-ON • Dell Z9100-ON • Dell S5296F-ON • Dell S5248F-ON • Dell S5232F-ON • Dell S5212F-ON • Dell N3248T • Edgecore/Accton AS7816-64X • Edgecore/Accton AS7726-32X • Edgecore/Accton S7712-32X • Edgecore/Accton AS7326-56X • Edgecore/Accton AS5712-54X |
| Cisco NX-OS | <ul style="list-style-type: none"> • 10.1(2) • 9.3(10) • 9.3(8) | Nexus 3000 or 9000 Platform |
| Arista EOS | <ul style="list-style-type: none"> • 4.27.6M • 4.27.4M • 4.25.3.1M • 4.24.5M | DCS-7000 Series |

Apstra Release 4.1.1

| Device Operating System | Qualified NOS Versions | Supported Device (Series) |
|--|--|---|
| Juniper Junos OS | <ul style="list-style-type: none"> • 21.4R2 • 21.2R3 • 20.4R3 | <ul style="list-style-type: none"> • QFX5100 - Don't use as leaf with Layer 3 VNI • QFX5110 - Can't be used as a border leaf. It can't route between VXLAN IRB and L3 interface. • QFX5120 • QFX5200 • QFX5210 • QFX10002 • QFX10008 • EX4650-48Y • EX4400-48F • EX4400-48T • EX4400-24T |
| Juniper Junos OS Evolved (spine, superspine and non-EVPN-VXLAN leaf) | <ul style="list-style-type: none"> • 21.4R2-EVO • 21.2R3-EVO • 20.4R3-EVO | <ul style="list-style-type: none"> • QFX5220 • QFX5130 • PTX10001-36MR |

(Continued)

| Device Operating System | Qualified NOS Versions | Supported Device (Series) |
|-------------------------|--|--|
| Enterprise SONiC | <ul style="list-style-type: none"> • SONiC-OS-3.5.3-GA-Enterprise-Advanced • SONiC-OS-3.5.3-GA-Enterprise-Base • SONiC-OS-3.4.1-GA-Enterprise-Advanced • SONiC-OS-3.4.1-GA-Enterprise-Base • SONiC-OS-3.3.0-GA-Enterprise-Advanced • SONiC-OS-3.3.0-GA-Enterprise-Base | <ul style="list-style-type: none"> • Dell Z9432F-ON (spine role) • Dell Z9332F-ON • Dell Z9264F-ON • Dell Z9100-ON • Dell S5296F-ON • Dell S5248F-ON • Dell S5232F-ON • Dell S5212F-ON • Dell N3248T • Edgecore/Accton AS7816-64X • Edgecore/Accton AS7726-32X • Edgecore/Accton S7712-32X • Edgecore/Accton AS7326-56X • Edgecore/Accton AS5712-54X |
| Cisco NX-OS | <ul style="list-style-type: none"> • 10.1(2) • 9.3(8) | Nexus 3000 or 9000 Platform |
| Arista EOS | <ul style="list-style-type: none"> • 4.27.4M • 4.25.3.1M • 4.24.5M • 4.23.6M | DCS-7000 Series |

Apstra Release 4.1.0

| Device Operating System | Qualified NOS Versions | Supported Device (Series) |
|--|--|--|
| Juniper Junos OS | <ul style="list-style-type: none"> • 21.2R3 • 20.4R3 | <ul style="list-style-type: none"> • QFX5100 - Don't use as leaf with Layer 3 VNI • QFX5110 - Can't be used as a border leaf. It can't route between a VXLAN IRB and an L3 interface. • QFX5120 • QFX5200 • QFX5210 • QFX10002 • QFX10008 • EX4650-48Y • EX4400-48F • EX4400-48T • EX4400-24T |
| Juniper Junos OS Evolved (spine, superspine and non-EVPN-VXLAN leaf) | <ul style="list-style-type: none"> • 21.2R3-EVO • 20.4R3-EVO | <ul style="list-style-type: none"> • QFX5220 • QFX5130 • PTX10001-36MR |

(Continued)

| Device Operating System | Qualified NOS Versions | Supported Device (Series) |
|-------------------------|--|--|
| Enterprise SONiC | <ul style="list-style-type: none"> SONiC-OS-3.4.1-GA-Enterprise-Advanced SONiC-OS-3.4.1-GA-Enterprise-Base SONiC-OS-3.3.0-GA-Enterprise-Advanced SONiC-OS-3.3.0-GA-Enterprise-Base | <ul style="list-style-type: none"> Dell Z9432F-ON (spine role) Dell Z9332F-ON Dell Z9264F-ON Dell Z9100-ON Dell S5296F-ON Dell S5248F-ON Dell S5232F-ON Dell S5212F-ON Dell N3248T Edgecore/Accton AS7816-64X Edgecore/Accton AS7726-32X Edgecore/Accton S7712-32X Edgecore/Accton AS7326-56X Edgecore/Accton AS5712-54X |
| Cisco NX-OS | <ul style="list-style-type: none"> 10.1(2) 9.3(8) | Nexus 3000 or 9000 Platform |
| Arista EOS | <ul style="list-style-type: none"> 4.25.3.1M 4.24.5M 4.23.6M 4.22.9M | DCS-7000 Series |

NOS Upgrade Paths (Devices)

IN THIS SECTION

- [Apstra Release 4.1.2 | 988](#)
- [Apstra Release 4.1.1 | 990](#)
- [Apstra Release 4.1.0 | 993](#)

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 824](#).

Apstra Release 4.1.2

Juniper Junos OS & Apstra 4.1.2

| From Version | To Version |
|--------------|--|
| 20.4R3-S3.4 | <ul style="list-style-type: none"> • 20.4R3-S2.6 • 22.2R3 |
| 21.2R1-S2.2 | 21.2R3.8 |
| 21.2R3-S2.9 | <ul style="list-style-type: none"> • 21.2R1-S2.2 • 21.4R3.16 |
| 21.4R3.16 | <ul style="list-style-type: none"> • 21.2R3-S2.9 • 22.2R3 |

(Continued)

| From Version | To Version |
|--------------|--|
| 22.2R3 | <ul style="list-style-type: none"> • 21.2R3-S2.9 • 21.4R3.16 |

Juniper Junos OS Evolved & Apstra 4.1.2

| From Version | To Version |
|--------------|--|
| 20.4R3-S3.5 | 21.2R3.10 |
| 21.2R3.10 | 21.4R3.13 |
| 21.4R3.13 | <ul style="list-style-type: none"> • 20.4R3-S3.5 • 22.2R2.12 |
| 22.2R2.12 | <ul style="list-style-type: none"> • 20.4R3-S3.5 • 21.4R3.13 |

Cisco NX-OS & Apstra 4.1.2

| From Version | To Version |
|--------------|---|
| 9.3.3 | 9.3(10) |
| 9.3.8 | 9.3(10) |
| 9.3.10 | <ul style="list-style-type: none"> • 9.3(8) • 10.1(2) |

Arista EOS & Apstra 4.1.2

| From Version | To Version |
|--------------|--|
| 4.23.6M | <ul style="list-style-type: none"> • 4.24.5M • 4.27.6M |
| 4.24.5M | <ul style="list-style-type: none"> • 4.23.6M • 4.27.4M |
| 4.25.3.1M | 4.27.6M |
| 4.27.6M | 4.25.3.1M |

SONiC & Apstra 4.1.2

| From Version | To Version |
|--------------|------------|
| 3.3.0 | 3.5.4 |
| 3.5.4 | 4.0.2 |
| 4.0.2 | 3.5.4 |

Apstra Release 4.1.1

Juniper Junos OS & Apstra 4.1.1

| From Version | To Version |
|--------------|-------------|
| 21.4R2.10 | 20.4R3-S3.4 |

(Continued)

| From Version | To Version |
|--------------|---|
| 21.2R3.8 | <ul style="list-style-type: none"> • 21.2R1-S2.2 • 21.2R1-S1.3 • 20.4R3-S3.4 |
| 21.2R1-S2.2 | 21.2R3.8 |
| 21.2R1-S1.3 | 21.2R3.8 |
| 20.4R3-S3.4 | 20.4R3-S2.6 |
| 20.4R3-S2.6 | <ul style="list-style-type: none"> • 21.4R2.10 • 21.2R3.8 • 20.4R3-S3.4 |
| 20.2R3-S2.5 | 20.4R3-S3.4 |

Juniper Junos OS Evolved & Apstra 4.1.1

| From Release | To Release |
|------------------|---|
| 21.4R2.14-EVO | 20.4R3-S3.5-EVO |
| 20.4R3-S3.5-EVO | <ul style="list-style-type: none"> • 21.4R2.14-EVO • 21.2R3-10-EVO • 20.4R3-S2.2-EVO |
| 20.4R3-S2.2-EVO | 20.4R3-S3.5-EVO |
| 20.4R2-S2.10-EVO | 20.4R3-S3.5-EVO |

Cisco NX-OS & Apstra 4.1.1

| From Version | To Version |
|--------------|---|
| 9.3.8 | <ul style="list-style-type: none"> • 10.1(2) • 9.2(2) |
| 9.3.3 | 9.3(8) |
| 9.2.3 | 9.3(8) |
| 7.0.3.I7.9 | 9.3(8) |

Arista EOS & Apstra 4.1.1

| From Version | To Version |
|--------------|--|
| 4.27.4M | 4.25.3M |
| 4.25.3.1M | 4.27.4M |
| 4.24.5M | <ul style="list-style-type: none"> • 4.27.4M • 4.23.6M |
| 4.23.6M | <ul style="list-style-type: none"> • 4.24.5M • 4.22.9M |
| 4.22.9M | 4.27.4M |

SONiC & Apstra 4.1.1

| From Version | To Version |
|--------------|------------|
| 3.4.1 | 3.3.0 |
| 3.3.0 | 3.4.1 |

Apstra Release 4.1.0

Table 23: Juniper Junos OS & Apstra 4.1.0

| From Version | To Version |
|--------------|---|
| 21.2R3.8 | <ul style="list-style-type: none"> • 21.2R1-S2.2 • 21.2R1-S1.3 • 20.4R3-S2.6 |
| 21.2R1-S2.2 | 21.2R3.8 |
| 21.2R1-S1.3 | 21.2R3.8 |
| 20.4R3-S2.6 | <ul style="list-style-type: none"> • 21.2R3.8 • 20.2R3-S2.5 • 20.4R3-S1.3 |
| 20.4R3-S1.3 | 20.4R3-S2.6 |
| 20.2R3-S2.5 | 20.4R3-S2.6 |

Table 24: Juniper Junos OS Evolved & Apstra 4.1.0

| From Release | To Release |
|--------------|-------------|
| 21.2R3.10 | 20.4R3-S2.2 |

Table 24: Juniper Junos OS Evolved & Apstra 4.1.0 *(Continued)*

| From Release | To Release |
|--------------|--|
| 20.4R3-S2.2 | <ul style="list-style-type: none"> • 21.2R3.10 • 20.4R3-S1.3 • 20.4R2-S2.10 |
| 20.4R3-S1.3 | 20.4R3-S2.2 |
| 20.4R2-S.10 | 20.4R3.S2.2 |

Table 25: Cisco NX-OS & Apstra 4.1.0

| From Version | To Version |
|--------------|---|
| 9.3.8 | <ul style="list-style-type: none"> • 10.1.2 • 9.2.2 |
| 9.3.3 | 9.3.8 |
| 9.2.3 | 9.3.8 |
| 7.0.3.17.9 | 9.3.8 |

Table 26: Arista EOS & Apstra 4.1.0

| From Version | To Version |
|--------------|------------|
| 4.24.5M | 4.22.9M |
| 4.23.6M | 4.22.9M |

Table 26: Arista EOS & Apstra 4.1.0 *(Continued)*

| From Version | To Version |
|--------------|---|
| 4.22.9M | <ul style="list-style-type: none"> • 4.25.3.1M • 4.24.5M • 4.23.6M |
| 4.21.14M | 4.23.6M |
| 4.20.11M | <ul style="list-style-type: none"> • 4.25.3.1M • 4.24.5M • 4.23.6M |

Table 27: SONiC & Apstra 4.1.0

| From Version | To Version |
|--------------|------------|
| 3.4.1 | 3.3.0 |
| 3.3.0 | 3.4.1 |

Predefined Dashboards (Analytics)

IN THIS SECTION

- [Device Environmental Health Summary Dashboard \(New in 4.1.2\) | 996](#)
- [Device Health Summary Dashboard | 996](#)
- [Device Telemetry Health Summary Dashboard \(New in 4.1.2\) | 997](#)
- [Drain Validation Dashboard | 997](#)
- [Throughput Health MLAG Dashboard | 997](#)

- [Traffic Trends Dashboard | 998](#)
- [Virtual Infra Fabric Health Check Dashboard | 998](#)
- [Virtual Infra Redundancy Check Dashboard | 998](#)

Device Environmental Health Summary Dashboard (New in 4.1.2)

| | |
|------------------|---|
| Goal | Show device environmental data |
| Trigger | Presence of at least one assigned system |
| Widgets / Probes | <ul style="list-style-type: none"> • Systems missing power supplies / Device Environmental Checks • Systems missing fans / Device Environmental Checks • Switch temperature alarm / Device Environmental Checks • Systems with inoperative power supplies / Device Environmental Checks • Systems with inoperative fans / Device Environmental Checks • Power supply temperature alarm / Device Environmental Checks • Systems with faulty power supply fans / Device Environmental Checks • Airflow direction mismatch / Device Environmental Checks |

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"> • Systems with high cpu utilization / Device System Health • Systems with high memory utilization / Device System Health • Systems with high disk utilization / Device System Health |

Device Telemetry Health Summary Dashboard (New in 4.1.2)

| | |
|------------------|--|
| Goal | Present sustained service execution anomalies under the device telemetry health probe |
| Trigger | Presence of at least one deployed system |
| Widgets / Probes | <ul style="list-style-type: none"> • Systems with degraded waiting time per service / Device Telemetry Health • Systems that sustained telemetry timeouts per service / Device Telemetry Health • Systems that sustained telemetry failures per service / Device Telemetry Health • Systems that sustained telemetry underruns per service / Device Telemetry Health |

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"> • LAG Imbalance / LAG Imbalance • MLAG Imbalance / MLAG Imbalance • Fabric ECMP Imbalance / ECMP Imbalance (Fabric Interfaces) |

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"> • Hypervisor VLANs missing in Fabric / Hypervisor & Fabric VLAN Config Mismatch • Hypervisor PNIC LAG Status / Hypervisor & Fabric LAG Config Mismatch • Hypervisor Low MTU anomalies / Hypervisor MTU Threshold Check • Critical Services affected by VLAN misconfig / VMs Without Fabric Configured VLANs • Hypervisor has inconsistent MTU / Hypervisor MTU Mismatch |

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
- Hypervisors without ToR switch redundancy / Hypervisor Redundancy Checks
 - Virtual Infra Networks without link redundancy / Hypervisor Redundancy Checks

Predefined Probes (Analytics)

IN THIS SECTION

- [BGP Session Flapping Probe | 1000](#)
- [Bandwidth Utilization Probe | 1002](#)
- [Critical Services: Utilization, Trending, Alerting Probe | 1005](#)
- [Device Environmental Checks Probe \(New in 4.1.2\) | 1006](#)
- [Device System Health Probe | 1007](#)
- [Device Telemetry Health Probe | 1009](#)
- [Device Traffic Probe | 1010](#)
- [Drain Traffic Anomaly Probe | 1014](#)
- [ECMP Imbalance \(External Interfaces\) Probe | 1015](#)
- [ECMP Imbalance \(Fabric Interfaces\) Probe | 1017](#)
- [ECMP Imbalance \(Spine to Superspine Interfaces\) Probe | 1020](#)
- [ESI Imbalance Probe | 1022](#)
- [EVPN Host Flapping Probe | 1024](#)
- [EVPN VXLAN Type-3 Route Validation Probe | 1025](#)
- [EVPN VXLAN Type-5 Route Validation Probe | 1027](#)
- [External Routes Probe | 1029](#)
- [Hot/Cold Interface Counters \(Fabric Interfaces\) Probe | 1030](#)
- [Hot/Cold Interface Counters \(Specific Interfaces\) Probe | 1034](#)
- [Hot/Cold Interface Counters \(Spine to Superspine Interfaces\) Probe | 1036](#)
- [Hypervisor and Fabric LAG Config Mismatch Probe \(Virtual Infra\) | 1038](#)
- [Hypervisor and Fabric VLAN Config Mismatch Probe \(Virtual Infra\) | 1039](#)
- [Hypervisor MTU Mismatch Probe \(Virtual Infra NSX-T Only\) | 1046](#)

- [Hypervisor MTU Threshold Check Probe \(Virtual Infra\) | 1046](#)
- [Hypervisor Missing LLDP Config Probe \(Virtual Infra\) | 1047](#)
- [Hypervisor Redundancy Checks Probe \(Virtual Infra\) | 1048](#)
- [Interface Flapping \(Fabric Interfaces\) Probe | 1049](#)
- [Interface Flapping \(Specific Interfaces\) Probe | 1051](#)
- [Interface Flapping \(Specific Interfaces\) Probe | 1052](#)
- [Interface Policy 802.1x Probe | 1054](#)
- [LAG Imbalance Probe | 1055](#)
- [Leafs Hosting Critical Services: Utilization, Trending, Alerting Probe | 1057](#)
- [Link Fault Tolerance in Leaf and Access LAGs Probe | 1058](#)
- [MLAG Imbalance Probe | 1060](#)
- [Multiagent Detector Probe | 1064](#)
- [Optical Transceivers Probe | 1065](#)
- [Packet Discard Percentage Probe | 1067](#)
- [Spine Fault Tolerance Probe | 1069](#)
- [Total East/West Traffic Probe | 1070](#)
- [VMs without Fabric Configured VLANs Probe \(Virtual Infra\) | 1072](#)
- [VXLAN Flood List Validation Probe | 1075](#)

Apstra software ships with many predefined probes that you can instantiate (Analytics > Probes > Create Probe > Instantiate Predefined Probe).

BGP Session Flapping Probe

IN THIS SECTION

- [BGP Session | 1002](#)
- [BGP Session Flapping | 1002](#)
- [Sustained BGP Session Flapping | 1002](#)

The BGP Session Flapping probe shows BGP session statuses for all switches and raises anomalies for flapping BGP sessions.

Instantiate Predefined Probe

Predefined Probe *

BGP Session Flapping

Probe Label *

BGP Session Flapping

Anomaly Time Window

5 Minutes

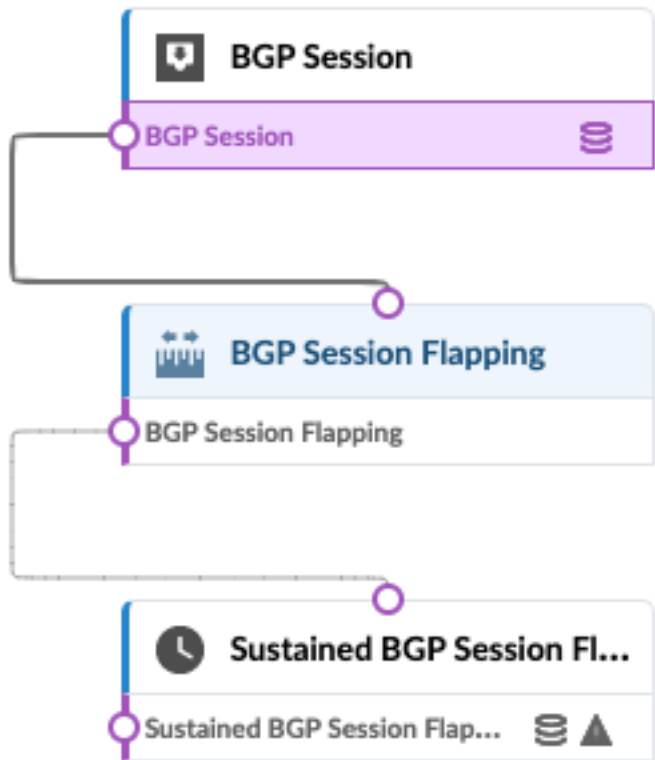
Anomaly Threshold (in %)

40

If the BGP flapping threshold is exceeded for more than or equal to percentage of Anomaly Time Window, an anomaly will be raised.

This probe shows BGP session statuses for all switches and raises anomalies for flapping BGP sessions.

It contains three stages: BGP Session, BGP Session Flapping, and Sustained BGP Session Flapping as described below:



BGP Session

The BGP Session stage shows a table with all available BGP sessions for switch systems. Each session is identified by the following parameters: Source IP, Source ASN, Destination IP, Destination ASN, Address Family and VRF name. The Flap Count column shows total number of BGP session flaps from the device. The Flap Count Increment column shows the number of new BGP session flaps for the service interval period. By default, it's two minutes. The FSM State and Status columns show the status of BGP sessions.

BGP Session Flapping

The BGP Session Flapping stage checks the Flap Count Increment value. If it's more than zero, the output value is 'true'. It only checks whether that BGP session has new flaps.

Sustained BGP Session Flapping

The Sustained BGP Session Flapping stage raises an anomaly for BGP sessions that have new BGP flaps for the specified time window. To illustrate, assume there are BGP flaps between leaf1 and spine1 nodes. The fabric BGP session between these nodes generates new BGP flaps when the interface status is changed on spine1 that's connected to leaf1. When shutdown and up interface is performed seven times on spine 1, it creates seven flaps for fabric BGP sessions between leaf1 and spine1. The seven new flaps are added and two 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.

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

Predefined Probe *

Bandwidth Utilization ▼

Probe Label *

Bandwidth Utilization

First summary average period

2 Minutes ▼

First summary history duration

1 Hour ▼

Second summary average period

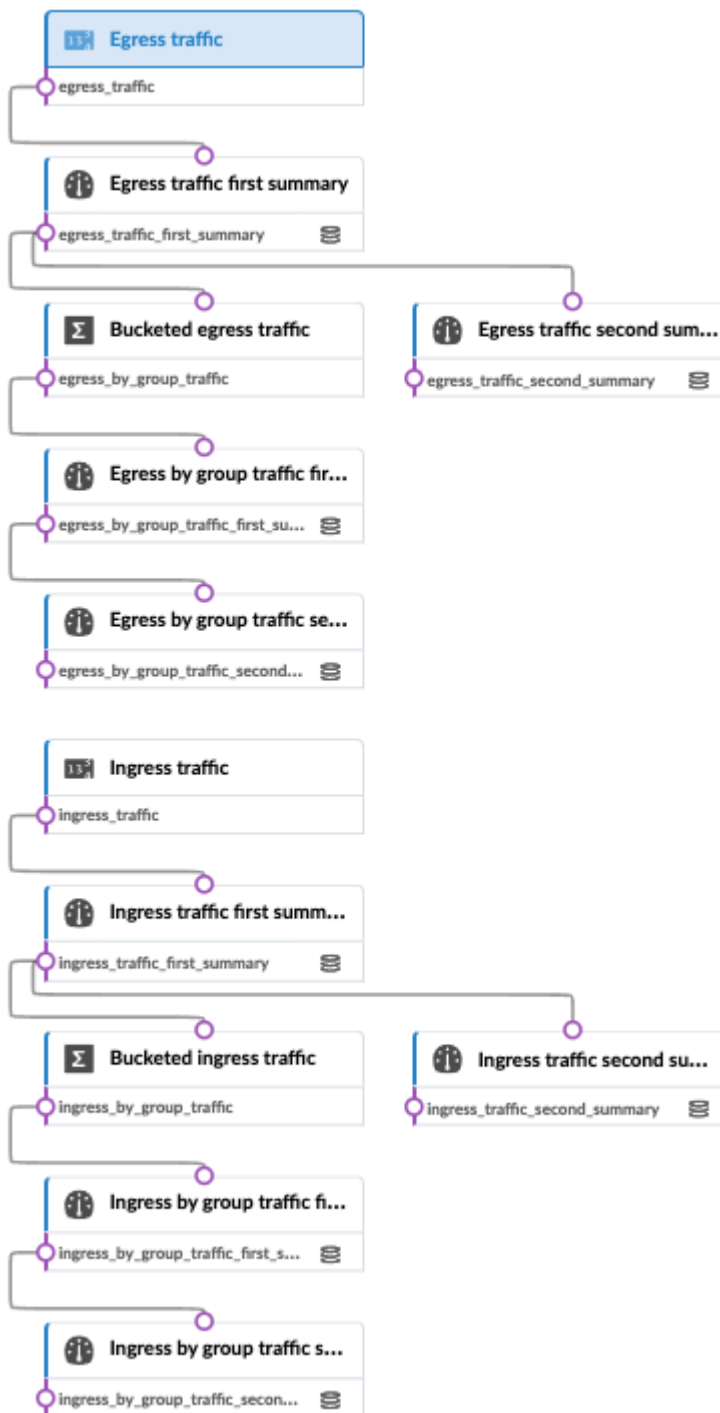
1 Hour ▼

Second summary history duration

30 Days ▼

Generate a probe to calculate bandwidth utilization

This probe captures history of bandwidth utilization trends at differing levels of aggregation.



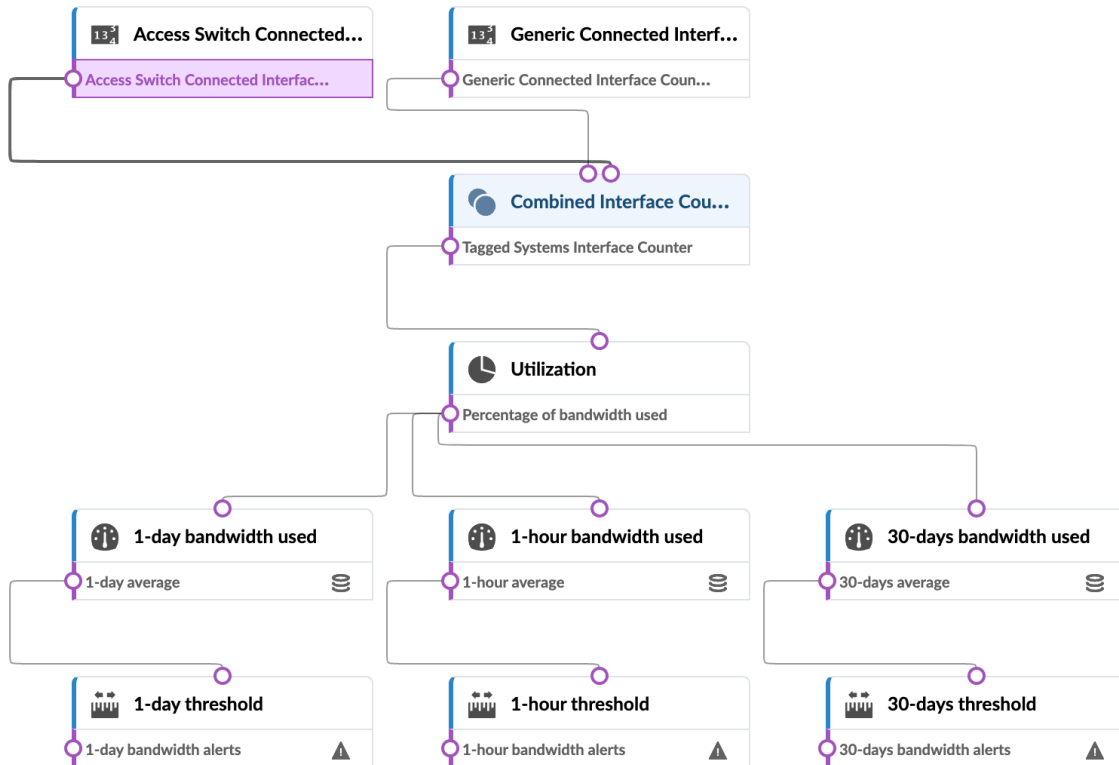
For more information about this probe, from the blueprint, navigate to **Analytics > Probes**, click **Create Probe**, then select **Instantiate Predefined Probe** from the drop-down list. Select the probe from the **Predefined Probe** drop-down list to see details specific to the probe.

Critical Services: Utilization, Trending, Alerting Probe

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

| | |
|---|--|
| <div><div>Predefined Probe *</div><div>Critical Services: Utilization, Trending, Alerting ▼</div></div> <div><div>Probe Label *</div><div>Critical Services: Utilization, Trending, Alerting</div></div> <div><div>Generic System Tags</div><div>No tags</div><div>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.</div></div> <div><div>Utilization threshold</div><div>80</div><div>If percentage bandwidth utilization reaches the threshold, an anomaly is raised.</div></div> | <div>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.</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.

Device Environmental Checks Probe (New in 4.1.2)

The device environmental checks probe monitors critical environmental metrics for managed switches including power supply, fan and temperature for real-time values of historical data retention over time.

When you instantiate this predefined probe, the instantiation menu displays a list of switch models in the blueprint. PSU count, fan count and air-flow direction information provide intent for deploying the switches.

If you have multiple blueprints that use the same switch model, you can set one expectation for the switch in one blueprint and a different expectation for the switch in a different blueprint.

Within one blueprint, all switches of the same model must have the same expectations. For example, you can't differentiate between specific QFX5120-48Y switches.

Instantiate Predefined Probe

Predefined Probe *

Device Environmental Checks

Probe Label *

Device Environmental Checks

History Retention Period

30 Days

Duration to maintain historical data.

Environment Expectations

Device Profile Label

default

Power Supply Count

2

Fan Tray Count

2

+ Add Environment Expectations

Table specifying expectations for power supply count and fan tray count on per device profile basis. device_profile_label: Device profile label. power_supply_count: Expected minimum number of power supplies for the device profile. fan_tray_count: Expected minimum number of fan trays for the device profile.

Built-in telemetry for device environment data is analysed in this probe.

☐ Create Another?

Create



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

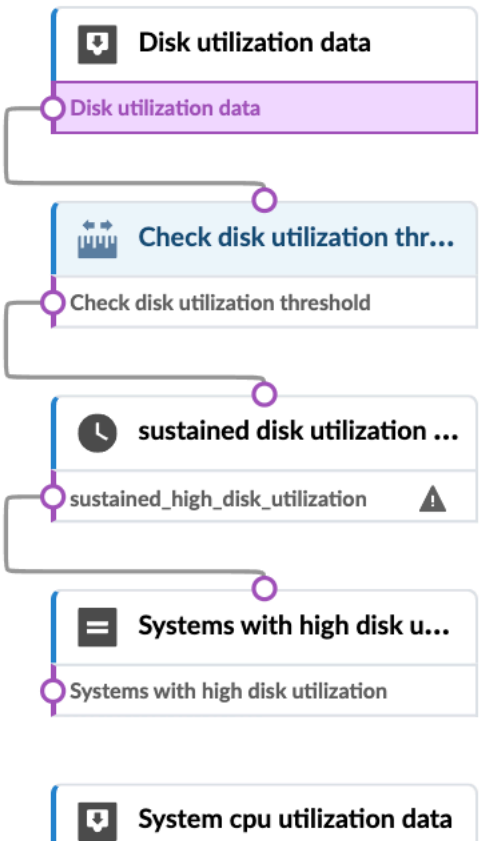
Time period in recent-history over which utilization data will be considered

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 Telemetry Health Probe

The device telemetry health probe verifies telemetry collector health. It runs analytics on the collection statistics from available service execution and if the telemetry collection health degrades, anomalies are raised.

Instantiate Predefined Probe

Predefined Probe *

Device Telemetry Health

Probe Label *

Device Telemetry Health

Max Waiting Time

120

Maximum time in seconds spent waiting for service to execute

Anomaly Time Window

10 Minutes

Threshold Duration

6 minutes

If any service running on a device, sustains telemetry collection failures/timeouts in this duration for over the Anomaly Time Window, an anomaly will be raised.

History retention period

7 Days

Time period to preserve historical data.

☒ **Enable telemetry stats history**

Maintain historical telemetry stats data

Generate a probe to verify the telemetry collector health. The probe utilizes the collection statistics from the available from service execution in order to run analytics and raise anomalies in the telemetry collection health degrades.

Create Another?

Create

```
graph TD; TS[Telemetry Stats] --> DWT[Degraded Wait Time]; TS --> SEF[Sustained Execution Failures]; TS --> SET[Sustained Execution Timeouts]; TS --> SEU[Sustained Execution Underruns];
```

The diagram illustrates the data flow for the Device Telemetry Health Probe. A central 'Telemetry Stats' data source (represented by a purple box with a database icon) feeds into four distinct monitoring components, each represented by a blue box with a status indicator (triangle) and a description below it:

- Degraded Wait Time**: Indicated by a status triangle and the description 'Degraded Wait Time'.
- Sustained Execution Failures**: Indicated by a status triangle and the description 'Sustained Execution Failures'.
- Sustained Execution Timeouts**: Indicated by a status triangle and the description 'Sustained Execution Timeouts'.
- Sustained Execution Underruns**: Indicated by a status triangle and the description 'Sustained Execution Underruns'.

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

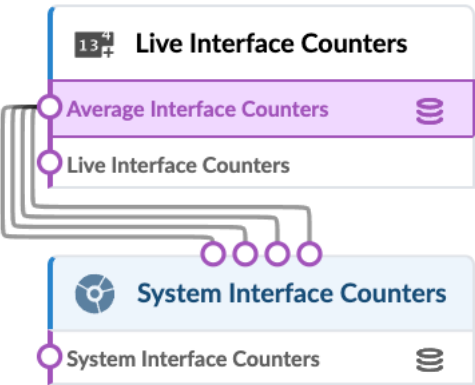
Maintain historical system interface counters data

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.

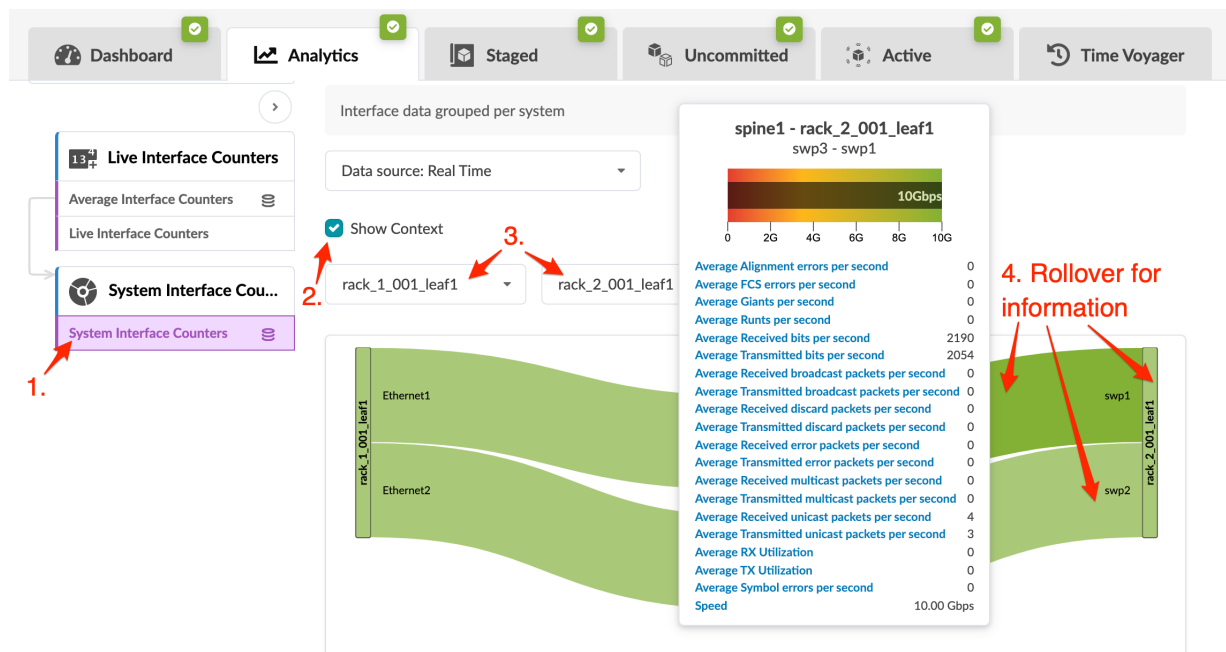


NOTE: You can change probe inputs, but if you change the probe processors then the probe is not a **predefined** probe anymore and the traffic layer view is not available in the active topology. For more information about the traffic layer view, see ["Physical Blueprint" on page 287](#).

| | | | | |
|-------------------------|--|--|---|--|
| Source Processor | Live Interface Counters ("Traffic Monitor" on page 1138) | 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 ("System Utilization" on page 1132) | 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 *

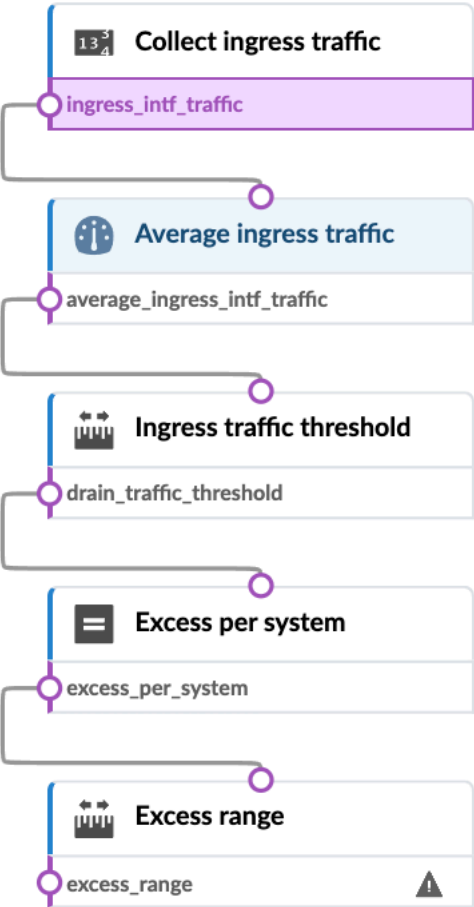
Drain Traffic Anomaly

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).

| | | |
|-------------------------|--|--|
| Source Processor | external interface traffic (Interface Counters) | 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

| | | |
|--------------------------------|--|---|
| Additional Processor(s) | external interface traffic avg (Periodic Average) | <p>Purpose: Calculate average traffic during period specified by average_period facade parameter. Unit is bytes per second.</p> <p>Input Stage: external_int_traffic</p> <p>Output Stage: external_int_traffic_avg</p> <p>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).</p> |
|--------------------------------|--|---|

| | |
|--|---|
| external interface std-dev (Standard Deviation) | <p>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'.</p> |
|--|---|

Input Stage: external_int_traffic_avg

| | |
|---|---|
| | <p>Output Stage: ext_int_std_dev</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> |
| std-dev percentage (Ratio) | <p>Input Stage: ext_int_std_dev</p> <p>Output Stage: std_dev_percentage</p> |
| live ecmp imbalance (Range) | <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>Output Stage: live_ecmp_imbalance</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> |
| links imbalanced percentage (Match Percentage) | <p>Input Stage: live_ecmp_imbalance</p> <p>Output Stage: links_imbalanced_percentage</p> |
| systems imbalanced (Range) | <p>Input Stage: links_imbalanced_percentage</p> <p>Output Stage: systems_imbalanced</p> |
| sustained ecmp imbalance (Time in State) | <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> |

| | | |
|---|---|---|
| | Output Stage: sustained_ecmp_imbalance | 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. |
| systems imbalanced count (Match Count) | Purpose: Count how many systems have external ecmp imbalance anomaly true at any instant in time. Input Stage: sustained_ecmp_imbalance | |
| | Output Stage: system_tx_imbalance_count | Number of systems with external ecmp imbalance. |
| live system imbalanced (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_tx_imbalance_count | |
| | Output Stage: live_system_imbalance_count | 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 (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.

| | | | |
|--------------------------------|---|--|--|
| Source Processor | leaf fabric interface traffic (Interface Counters-) | Purpose: wires in interface traffic samples (measured in bytes per second) from each spine-facing interface on each leaf. | |
| | Output Stage: leaf_fabric_int_traffic | 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). | |
| Additional Processor(s) | leaf fabric interface traffic avg (Periodic Average) | Purpose: Calculate average traffic during period specified by average_period facade parameter. Unit is bytes per second. | |
| | | Input Stage: leaf_fabric_int_traffic | |
| | Output Stage: leaf_fabric_int_tx_avg | 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). | |
| | leaf fabric interface std-dev (Standard Deviation) | 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'. | |
| | | Input Stage: leaf_fabric_int_tx_avg | |
| | Output Stage: leaf_fab_int_std_dev | 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

| | |
|---|---|
| Output Stage: imbalanced_system_count_out_of_range | 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.

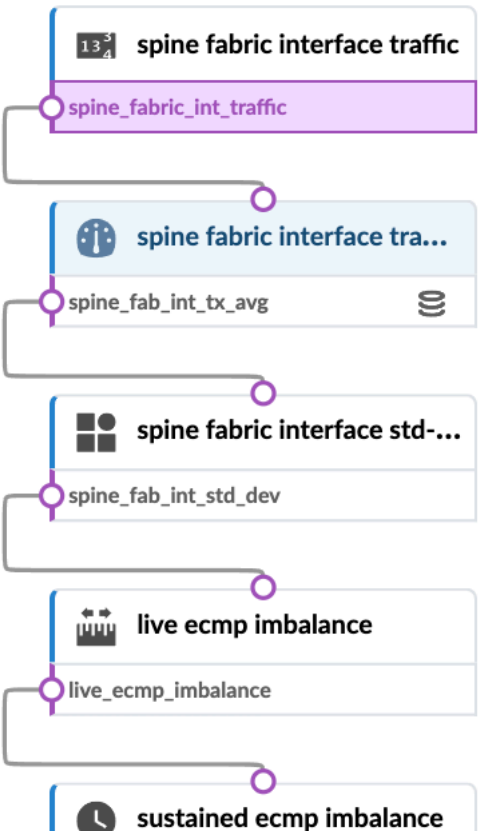
A given set of ECMP links (only calculated on spine to superspine links), identified by common system_id, is determined to be imbalanced if the standard-deviation of the tx_bytes counter (averaged periodically over the specified period) for the involved spine interfaces is above "Max Standard Deviation".

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

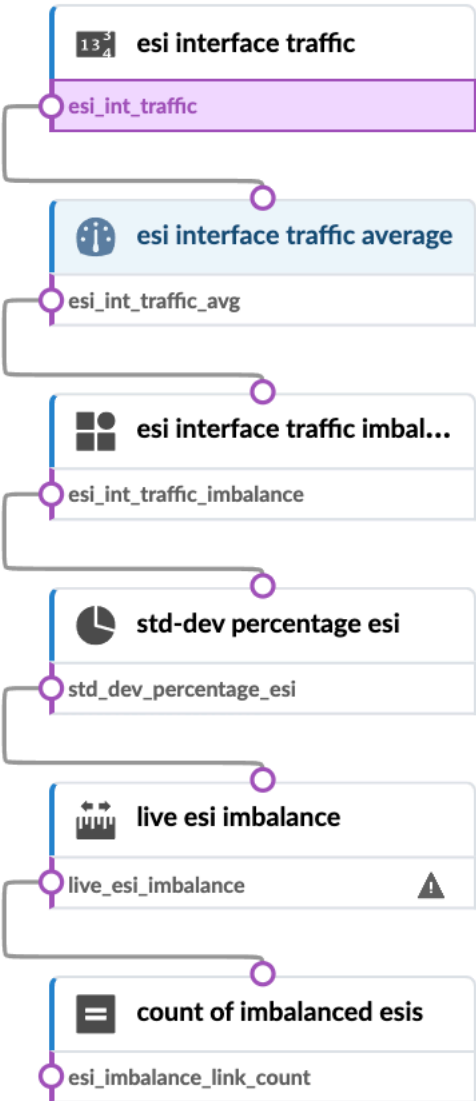
History duration

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 Host Flapping Probe

EVPN host flaps occur when an L2 loop is mistakenly created under the leaf devices by connecting a hub to two different leaf devices.

Instantiate Predefined Probe

Predefined Probe *

EVPN Host Flapping

Probe Label *

EVPN Host Flapping

Anomaly Time Window

2 Minutes

Anomaly Threshold (in %)

100

Collection period

2 Minutes

☒ Enable flapping hosts history

If enabled, probe will keep history of which leaf suppresses flapping MAC addresses and which specific addresses were suppressed.

History retention period

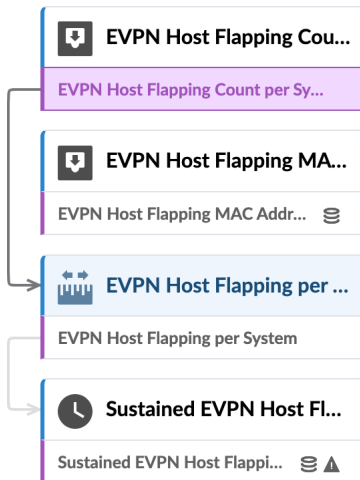
7 Days

On every leaf probe monitors MAC addresses that are being learned alternately from local and VTEP interfaces more often than it is allowed by constraints configured in the system.

If MAC address is suppressed for more than or equal to percentage of Anomaly Time Window, an anomaly will be raised.

Controls how often flapping MAC addresses will be collected on devices.

Duration to maintain flapping MAC addresses historical data.



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.

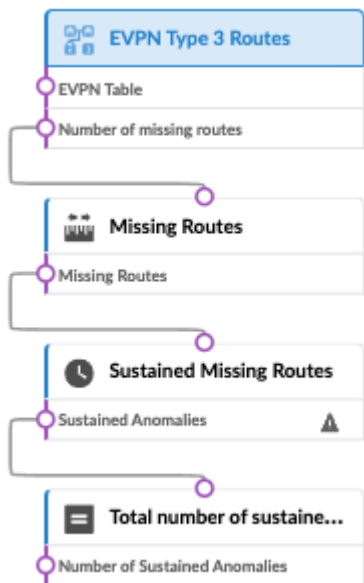
This probe validates EVPN Type-3 routes on every leaf in the network. It collects appropriate telemetry data, compares it to the set of Type-3 routes expected to be present and alerts if expected routes are missing on any device.

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

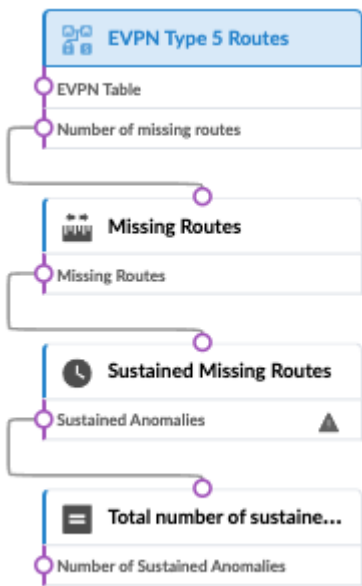
If routes are missing for more than or equal to percentage of Anomaly Time Window, an anomaly will be raised.

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.

| | | |
|-------------------------|--|--|
| Source Processor | leaf interface traffic (Interface Counters) | Purpose: wires in interface traffic samples (measured in bytes per second) from each spine facing interface on each leaf. |
| | Output Stage: leaf_int_traffic | 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'). |

| | | |
|--------------------------------|---|--|
| Additional Processor(s) | leaf interface tx avg (Periodic Average) | Purpose: Calculate average traffic during period specified by average_period facade parameter. Unit is bytes per second. |
| | Output Stage: leaf_int_tx_avg | 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').

**interface sum
per device
(Sum)**

Purpose: Sum average traffic for all interface under consideration per device.

Input Stage: leaf_int_tx_avg

**Output Stage:
if_counter_sum_per_device**

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).

**interface sum
per device
per link role
(Sum)**

Purpose: Sum average traffic for all interface under consideration per device, per interface role.

Input Stage: leaf_int_tx_avg

**Output Stage:
if_counter_sum_per_device_role**

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').

**live leaf
interface
cold (Range)**

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

**Output Stage:
live_leaf_int_cold**

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.

**live leaf
interface hot
(Range)**

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

**Output Stage:
live_leaf_int_hot**

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.

**sustained
cold leaf
interface
(Time in
State)**

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:
cold_leaf_int**

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: hot_leaf_int | 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) | <p>Purpose: Calculate percentage of interfaces that are cold on any given device under consideration.</p> <p>Input Stage: cold_leaf_int</p> |
| Output Stage: system_perc_cold | 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). |
| system percent hot (Match Percentage) | <p>Purpose: Calculate percentage of interfaces that are hot on any given device under consideration.</p> <p>Input Stage: hot_leaf_int</p> |
| Output Stage: system_perc_hot | 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). |
| device cold (Range) | <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> |
| Output Stage: device_cold_anomalous | 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).

| | |
|--|--|
| device hot (Range) | 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. |
| Input Stage: system_perc_hot | |
| Output Stage: device_hot_anomalous | 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). |

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"

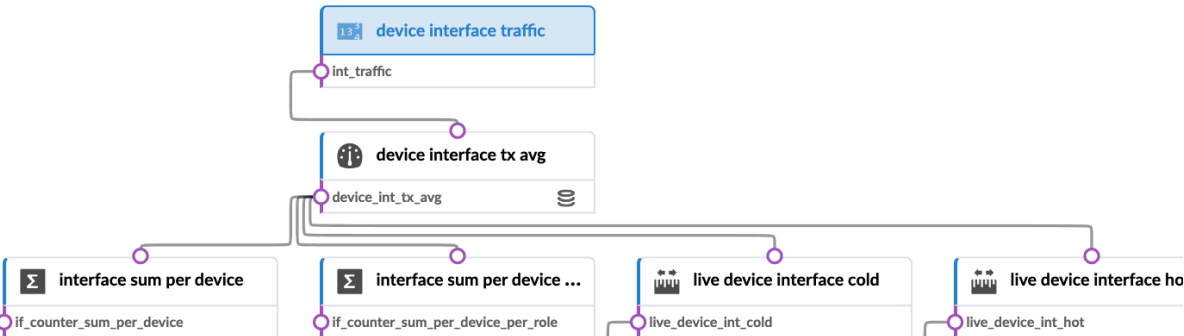
A given interface (out of the specified list) is considered to be in a cold state if its average counter value is less than "Min"

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"

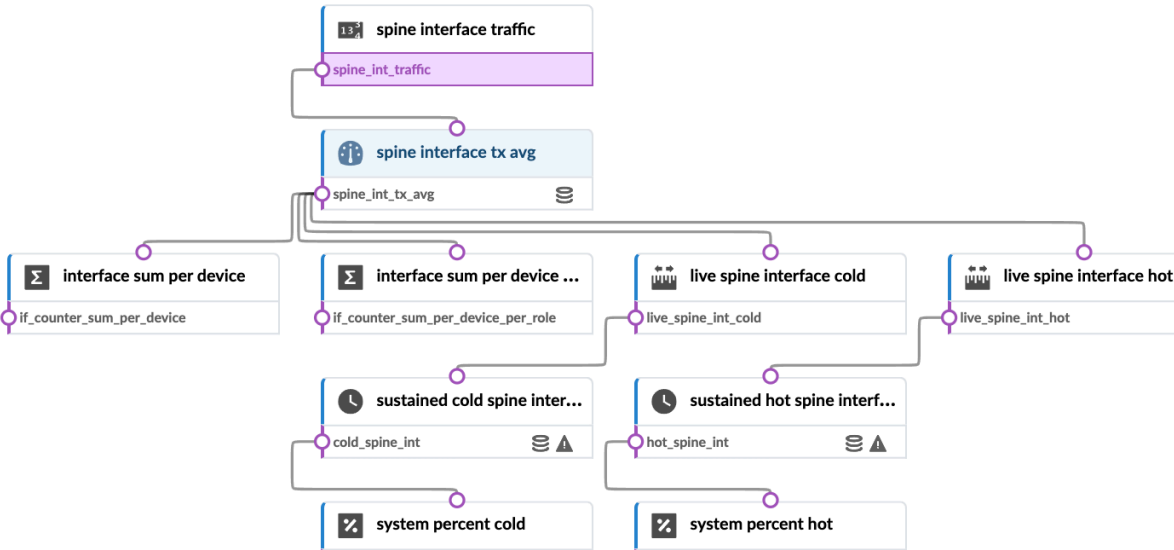
A given interface (considering only spine to superspine interfaces) is considered to be in a cold state if its average counter value is less than "Min"

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 and 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 leaf devices 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-leaf devices and ESXi hosts. LACP mode information is collected from the fabric LAG dual-leaf devices 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-leaf devices 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 leaf devices 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 leaf devices having LAG member ports. As a result in a blueprint after enabling this probe LAG mismatch anomalies are raised.

Probes > Hypervisor and Fabric LAG config mismatch

Stage: Hypervisor NIC LAG Mismatch Anomaly

| Fabric Interface | Fabric Lag | Hypervisor | Leaf | Pnic | Pnic Lag | Anomaly | Value | Updated |
|------------------|------------|---|---------------------|------|----------|---|-------|-----------|
| swp3 | bond1 | zz-karun-nsxt.cvx.2485377892354-3839439666-TN-2 | leaf-2-52540005BE0B | eth1 | | Anomalous value: mismatch Actual value: mismatch | true | a day ago |
| swp4 | bond1 | zz-karun-nsxt.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 leaf devices.

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 and Fabric VLAN Config Mismatch Probe (Virtual Infra)

IN THIS SECTION

- [Hypervisor & Fabric VLAN Config Mismatch Probe Overview | 1040](#)
- [Usage with NSX-T Integration | 1041](#)
- [Usage with VCenter Integration | 1045](#)

Hypervisor & Fabric VLAN Config Mismatch Probe Overview

| | | | |
|--------------------------------|---|---|--|
| Purpose | Calculate VLAN mismatch between configured virtual networks on leaf devices and VLANs needed by VMs running on hypervisors attached to leaf devices. (Formerly known as Virtual Infra VLAN Match). Detects misconfiguration of hypervisor trunk logical switches when VLAN tag is configured inside a VM (not on the bridge itself) (as of Apstra 4.1.0). | | |
| Source Processors | Fabric configured VLAN configs (generic graph collector) | output stage: Fabric VLAN configs (number set) (generated from graph) | |
| | Hypervisor expected VLAN configs (generic graph) | output stage: Hypervisor VLAN configs (number set) | |
| Additional Processor(s) | Hypervisor unique VLAN configs (set count) | input stage: Hypervisor VLAN configs output stage: Hypervisor unique VLAN configs (number set) | |
| | Differences between Hypervisor and Fabric (set comparison) | input stages: | Hypervisor unique VLAN configs |
| | | | Fabric VLAN configs |
| | | output stages: | Common in Fabric and Hypervisor (number set) |
| | | | Fabric Only (number set) |
| | | | Hypervisor Only (number set) |
| | Fabric missing VLAN configs accumulator (accumulate) | input stage: Hypervisor Only output stage: Hypervisor Only TimeSeries (number set time series) | |
| | Hypervisor missing VLAN configs accumulator (accumulate) | input stage: Fabric Only output stage: Fabric Only TimeSeries (number set time series) | |
| | Check for Fabric missing VLAN configs (range) | input stage: Hypervisor Only TimeSeries output stage: Fabric missing VLAN configs anomaly (discrete state set) | |

Check for Hypervisor missing VLAN configs (range)

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:

DashboardAnalyticsStagedUncommittedActiveTime Voyager

DashboardsAnomaliesWidgetsProbes

Probes > Hypervisor & Fabric VLAN Config Mismatch

OperationalNo anomaliesadmin an hour agoEnabled

Search stages...>

Fabric configured VLAN configs

Fabric VLAN configs

Hypervisor expected VLAN configs

Hypervisor VLAN configs

137229Hypervisor unique VLAN configs

Hypervisor unique VLAN configs

Differences between Hypervisor a...

Common in Fabric and Hypervisor

Processor: Differences between Hypervisor and Fabric

Set Comparison

| Input Name | Stage Name |
|------------|--------------------------------|
| A | Hypervisor unique VLAN configs |
| B | Fabric VLAN configs |

| Properties | |
|------------------|--|
| Significant Keys | server, vlan, interface, traffic, hypervisor, connected_to, fabric_interface |
| Enable Streaming | False |

- Click the **Fabric VLAN Configs** stage to show the VLANs tagged towards NSX-T transport nodes on fabric ToR leaf devices 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 | Vlans |
|---------------------|------------------|---|--------------------------------------|---------------------|---------|-------|
| 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 options:

- 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 options:

- 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' with '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 'Remediate Anomalies' button. Below this is a table of anomalies.

| 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 leaf devices 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

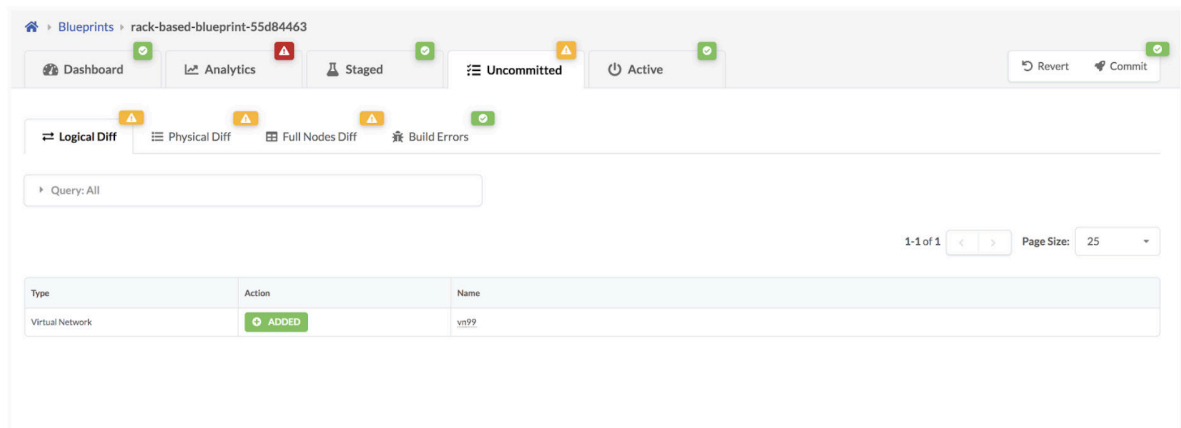
| connected To | Fabric Interface | Hypervisor | Interface | Server | Traffic | Vlan | Anomaly | Value | Updated |
|--------------------------------|------------------|------------|--------------------------------------|----------------------------|---------|------|--|-------|----------------|
| muc_leaf_5100_001_leaf_pair1 | 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-19876afc7071 | 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-19876afc7071 | 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.

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 NSX-T Only)

| | | |
|--------------------------------|--|---|
| Purpose | NSX-T Only - Detect maximum transmission unit (MTU) value deviations across hypervisor physical network adapters (pnics). | |
| Source Processor | Interface MTU (generic graph collector) | output stage: Interface MTU (number set) (generated from graph) |
| Additional Processor(s) | Check MTU mismatch between hypervisors (standard deviation) | input stage: Interface MTU output stage: Hypervisor MTU Deviation (number set) |
| | MTU Mismatch (range) | input stage: Hypervisor MTU Deviation (number set) output stage: MTU Mismatch (discrete state set) |
| | | |
| Example Usage | NSX Integration - 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. | |

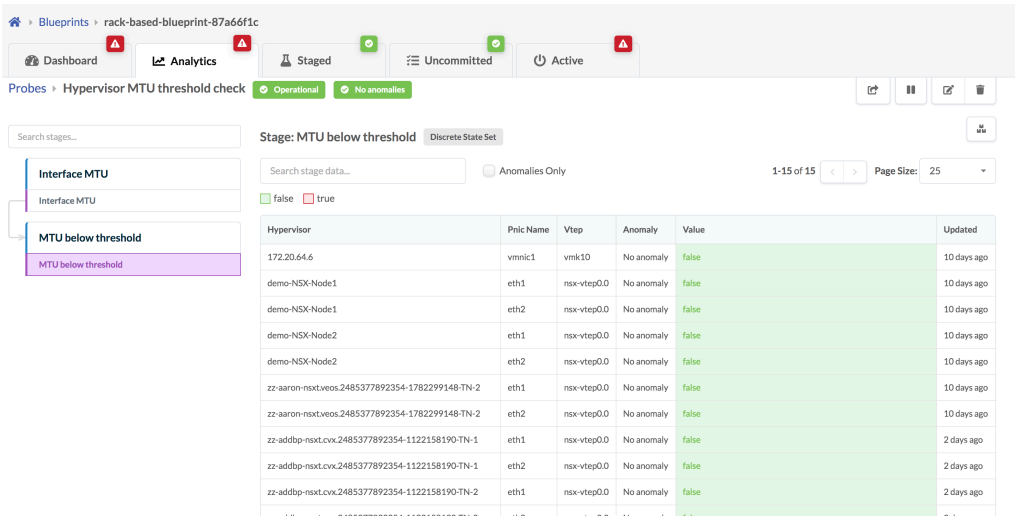
For more information about this probe, from the blueprint, navigate to **Analytics > Probes**, click **Create Probe**, then select **Instantiate Predefined Probe** from the drop-down list. Select the probe from the **Predefined Probe** drop-down list to see details specific to the probe.

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 leaf devices 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.



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)

| | | |
|--------------------------------|---|---|
| Purpose | Detect virtual infra hosts that are not configured for LLDP. (Formerly known as Virtual Infra missing LLDP config). | |
| Source Processor | Hypervisor NIC LLDP Config (generic graph) | output stage: Hypervisor NIC LLDP config (discrete state set) (generated from graph) |
| Additional Processor(s) | LLDP config by switch (match count) | input stage: Hypervisor NIC LLDP config output stage: LLDP config by switch (number set) |
| | Switches missing LLDP config (range) | input stage: LLDP config by switch |

output stage: Switches missing LLDP config anomaly
(discrete state set)

Example Usage **VMware Integration** - 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)

Purpose Detect hypervisor redundancy.

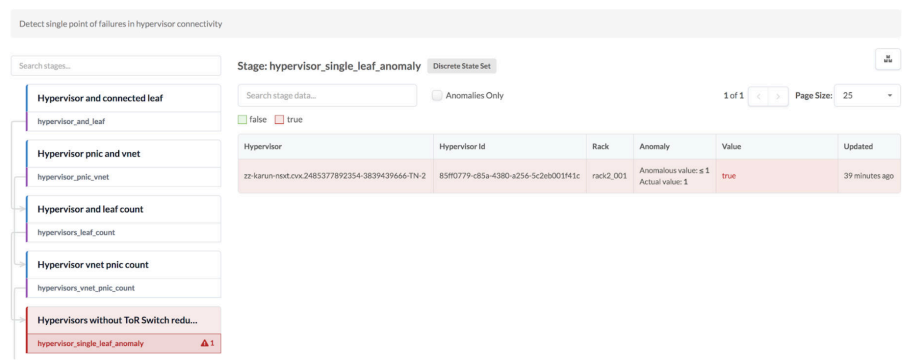
| | | |
|--------------------------|---|---|
| Source Processors | Hypervisor and connected leaf (generic graph) | output stage: hypervisor_and_leaf (text set) (generated from graph) |
| | Hypervisor pnic and vnet (generic graph collector) | output stage: hypervisor_pnic_vnet (text set) (generated from graph) |

| | | |
|--------------------------------|---|--|
| Additional Processor(s) | Hypervisor and leaf count (set count) | input stage: hypervisor_and_leaf output stage: hypervisors_leaf_count (number set) |
| | Hypervisor vnet pnic count (set count) | 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 leaf devices 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-leaf devices and ESXi hosts.



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.

Source Processor leaf fab int status (Service Data Collector) Purpose: wires in interface status telemetry for all fabric interfaces on the leaf devices.

Output Stage: leaf_if_status 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).

Additional Processor(s) leaf fabric interface status history (Accumulate) 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 | |
| | Output Stage: leaf_fab_int_status_accumulate | 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). |
| leaf fabric interface flapping (Range) | 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 | |
| | Output Stage: if_status_flapping | 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 | |
| | Output Stage: system_flapping | Set of statuses for each leaf, indicating if the leaf has higher than 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). |

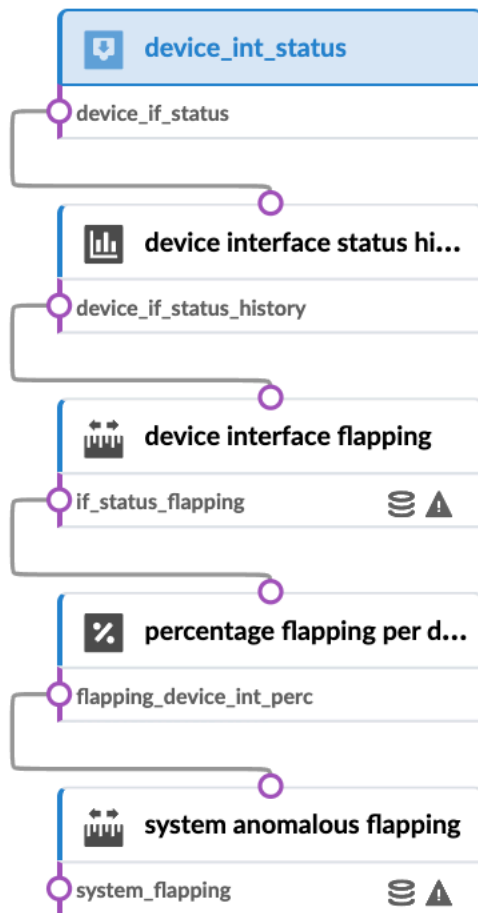
For more information about this probe, from the blueprint, navigate to **Analytics > Probes**, click **Create Probe**, then select **Instantiate Predefined Probe** from the drop-down list. Select the probe from the **Predefined Probe** drop-down list to see details specific to the probe.

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

| | |
|--|--|
| <div><div>Predefined Probe *</div><div>Interface Flapping (Specific Interfaces) ▼</div></div> <div><div>Probe Label *</div><div>Interface Flapping (Specific Interfaces)</div></div> <div><div>Interfaces *</div><div>No interfaces specified.</div><div>+ Add Interface</div></div> <div><div>Max Flapping Interfaces Percentage</div><div>10</div><div>Maximum percentage of flapping interfaces on a device</div></div> <div><div>Threshold</div><div>5</div><div>Sum total of number of flaps in recent-history for which an anomaly will be raised</div></div> <div><div>Duration</div><div>1 Minute ▼</div><div>Time period in recent-history in which interface flapping will be considered</div></div> | <div>Generate a probe to determine if specific interfaces are flapping</div> <div>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.</div> <div>If more-than "Max Flapping Interfaces Percentage" percent of interfaces on a given device are flapping, an anomaly will be raised for that device.</div> <div>Finally, the last "Anomaly History Count" anomaly state-changes are stored for observation.</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.

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 ▼

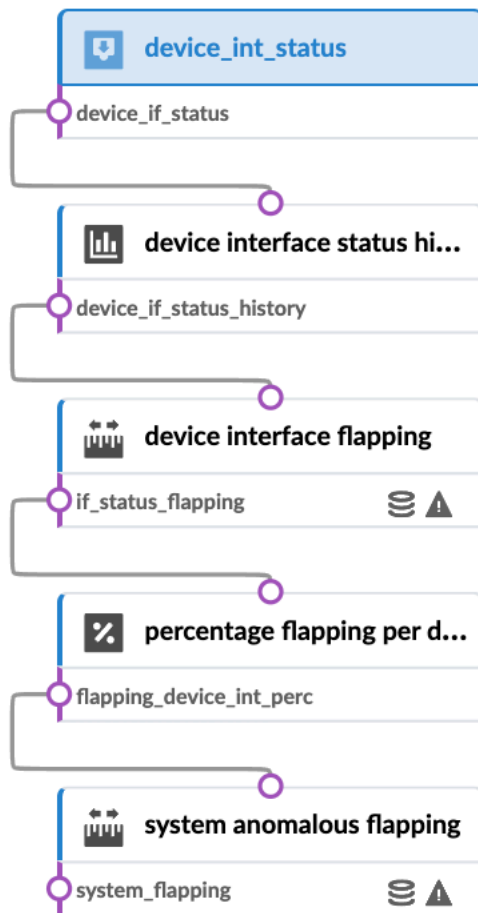
Time period in recent-history in which interface flapping will be considered

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).

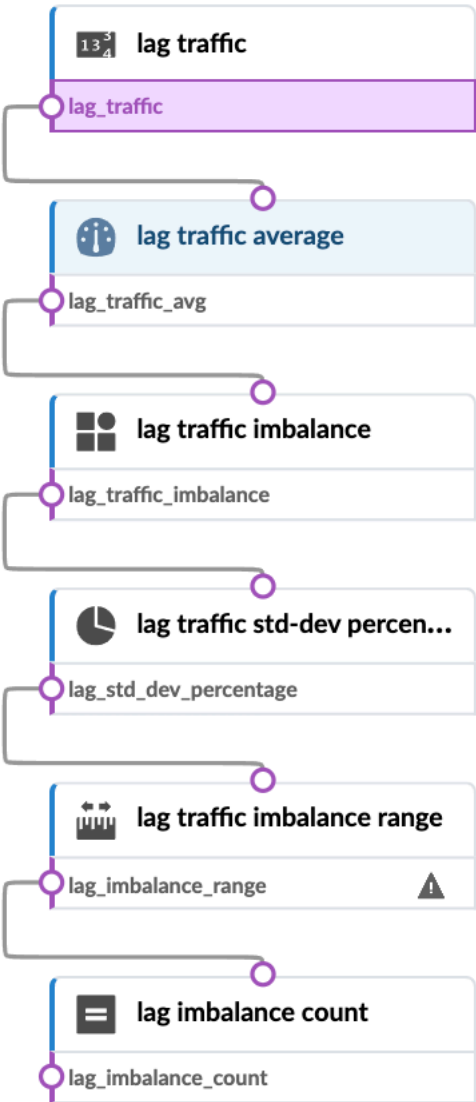
Duration

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

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.

Instantiate Predefined Probe

Predefined Probe *

Leafs Hosting Critical Services: Utilization, Trending, Alerting ▼

Probe Label *

Leafs Hosting Critical Services: Utilization, Trending, Alerting

Leaf Tags

No tags

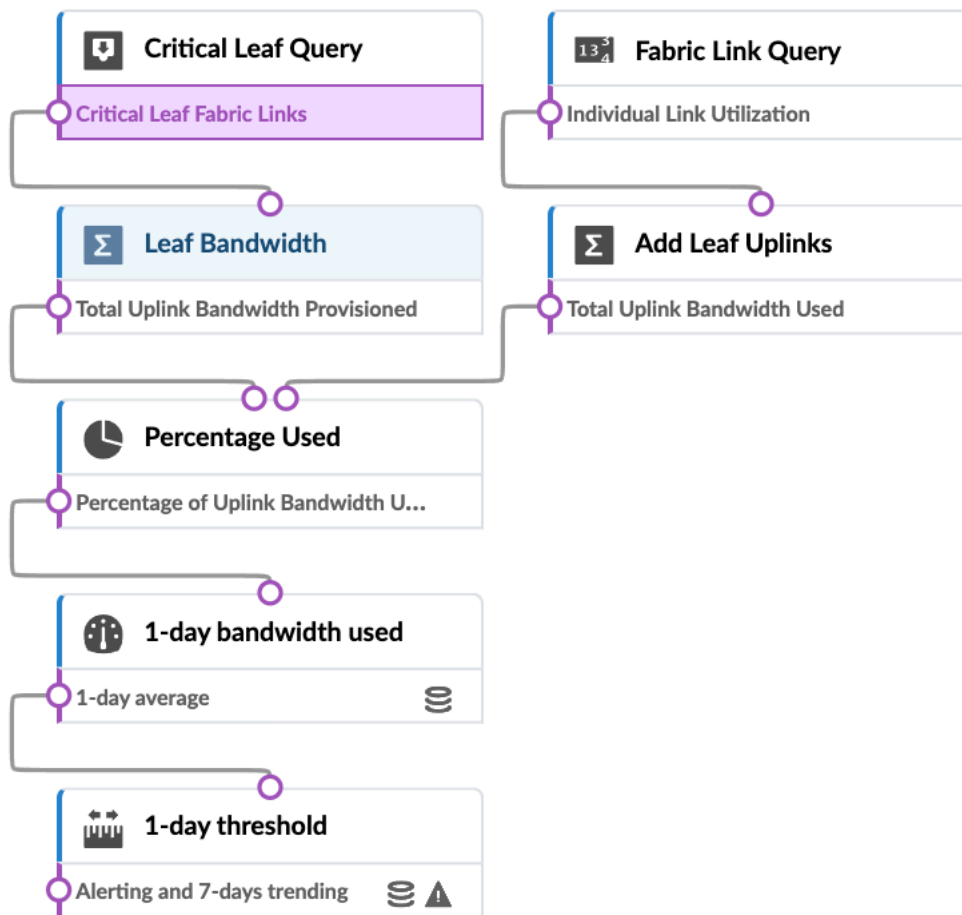
Bandwidth utilization is monitored for fabric interfaces hosted by leaf that have at least one of specified tags assigned.

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

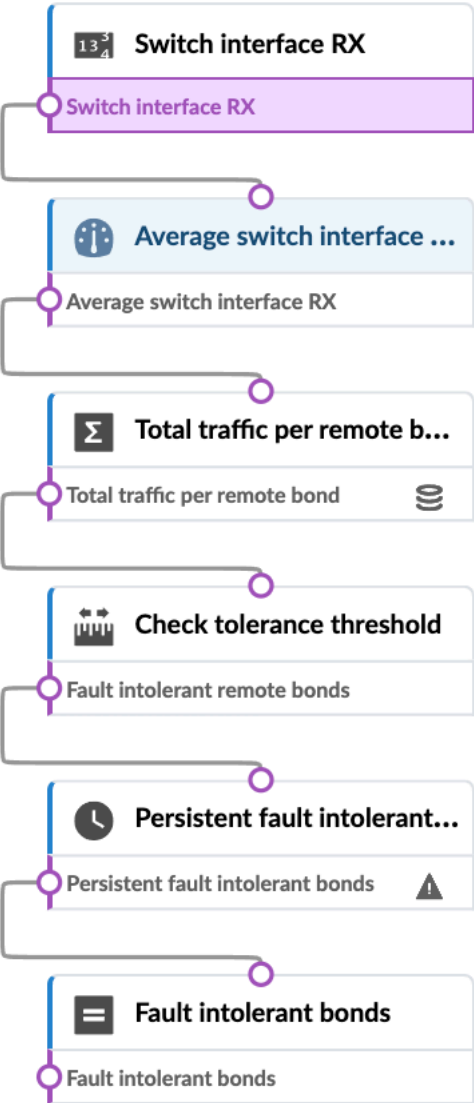
Time period in recent-history over which anomaly intolerant bonds will be considered

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.

| | | |
|--------------------------------|--|---|
| Source Processor | mlag interface traffic (Interface Counters) | 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. |
| | Output stage: mlag_int_traffic | 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). |
| Additional Processor(s) | mlag interface traffic average (Periodic Average) | Purpose: Calculate average traffic during period specified by average_period facade parameter. Unit is bytes per second. Input Stage: mlag_int_traffic Output Stage: mlag_int_traffic_avg 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. |
| | mlag interface traffic imbalance | 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) | Output Stage: mlag_int_traffic_imbalance | 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. |
| port-channel interface std-dev (Standard Deviation) | 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 Output Stage: port_channel_int_std_dev | 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. |
| port-channel total traffic (Sum) | Purpose: Calculate total traffic per port channel. Unit is byte per second. Input Stage: mlag_int_traffic_avg Output Stage: mlag_port_channel_total | 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. |
| mlag port-channel traffic std-dev (Standard Deviation) | 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 Output Stage: mlag_port_channel_imbalance | 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. |

| | |
|--|---|
| std-dev percentage mlag (Ratio) | Input Stage: mlag_int_traffic_imbalance Output Stage: std_dev_percentage_mlag |
| std-dev percentage port-channel (Ratio) | Input Stage: port_channel_int_std_dev Output Stage: std_dev_percentage_pc |
| live mlag imbalance (Range) | <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>Output Stage: live_mlag_imbalance 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> |
| live port- channel imbalance (Range) | <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>Output Stage: live_port_channel_imbalance 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> |
| std-dev percentage mlag port- channel (Ratio) | Input Stage: mlag_port_channel_imbalance Output Stage: std_dev_percentage_mlag_pc |
| live mlag port-channel imbalance (Range) | <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

Output Stage:
mlag_port_channel_imbalance_out_of_range 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.

mlag imbalance per link count (Match Count) Input Stage: live_mlag_imbalance
 Output Stage: mlag_imbalance_link_count

port-channel imbalance per rack (Match Percentage) Purpose: Calculate percentage of port channels on a given rack that have imbalance anomaly. Input Stage: live_port_channel_imbalance
Output Stage:
port_channel_imbalance_per_rack 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.

mlag port-channel imbalance per rack (Match Percentage) Purpose: Calculate percentage of MLAGs on a given rack that have port channel imbalance anomaly.
 Input Stage: mlag_port_channel_imbalance_out_of_range
Output Stage:
mlag_port_channel_imbalance_anomaly_per_rack 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.

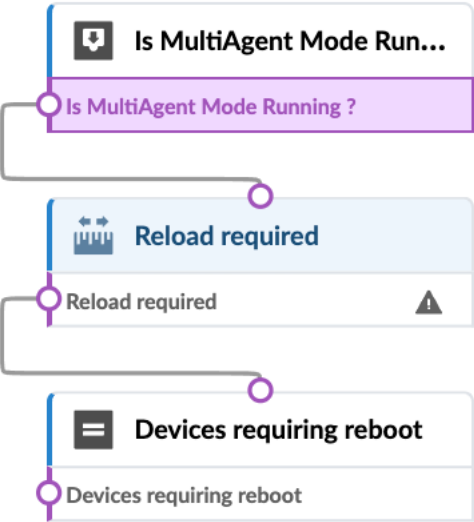
For more information about this probe, from the blueprint, navigate to **Analytics > Probes**, click **Create Probe**, then select **Instantiate Predefined Probe** from the drop-down list. Select the probe from the **Predefined Probe** drop-down list to see details specific to the probe.

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.

Optical Transceivers Probe

The Optical Transceivers probe is supported on Juniper and Arista. The Optical Transceivers probe (introduced in Apstra version 4.1.0) monitors optical statistics based on the following telemetry data:

- Temperature (C) of the physical port (interface stats)
- Voltage (V) of the physical port (interface stats)
- Transmit Power Level (dBm) of each optical lane (lane stats)
- Receive Power Level (dBm) of each optical lane (lane stats)
- Transmit Bias (mA) of each optical lane (lane stats)

If telemetry data falls outside the specified range for the specified amount of time, a warning or alarm is raised, as applicable.

In Apstra versions 4.1.0 and 4.1.1, the reason for a warning or alarm is not specified.

In Apstra version 4.1.2, warnings and alarms specify whether the value causing the anomaly was too high or too low.

Instantiate Predefined Probe

Predefined Probe *

Optical Transceivers

Probe Label *

Optical Transceivers

Anomaly Time Window

2 Minutes

Anomaly Threshold (in %)

100

If an optical metric's threshold is exceeded for more than or equal to percentage of Anomaly Time Window, an anomaly will be raised. For example, if Anomaly Time Window is 120 seconds and Threshold is 10%, an anomaly will be raised if a threshold is exceeded for more than 12 seconds.

History Retention Period

30 Days

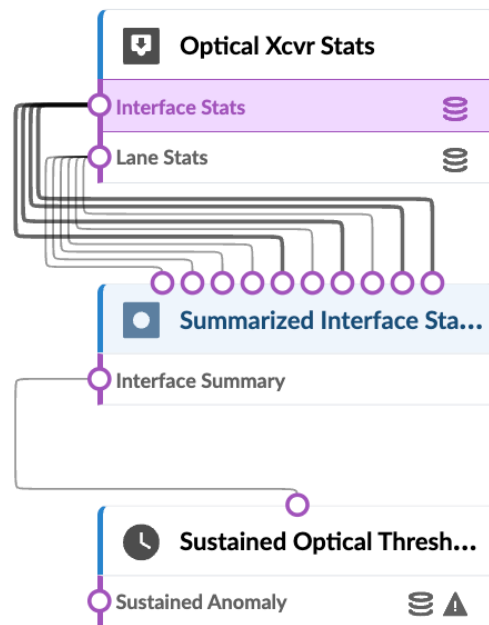
Duration to maintain historical data.

Built-in telemetry for optical interfaces is analysed in this probe. The real-time values are checked against the default thresholds specified in transceivers by manufacturers.

The probe summarizes the stats for every interface. If at least one threshold is exceeded during a specified time period, the interface is marked as anomalous and the anomaly is raised.

☐ Create Another?

Create



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

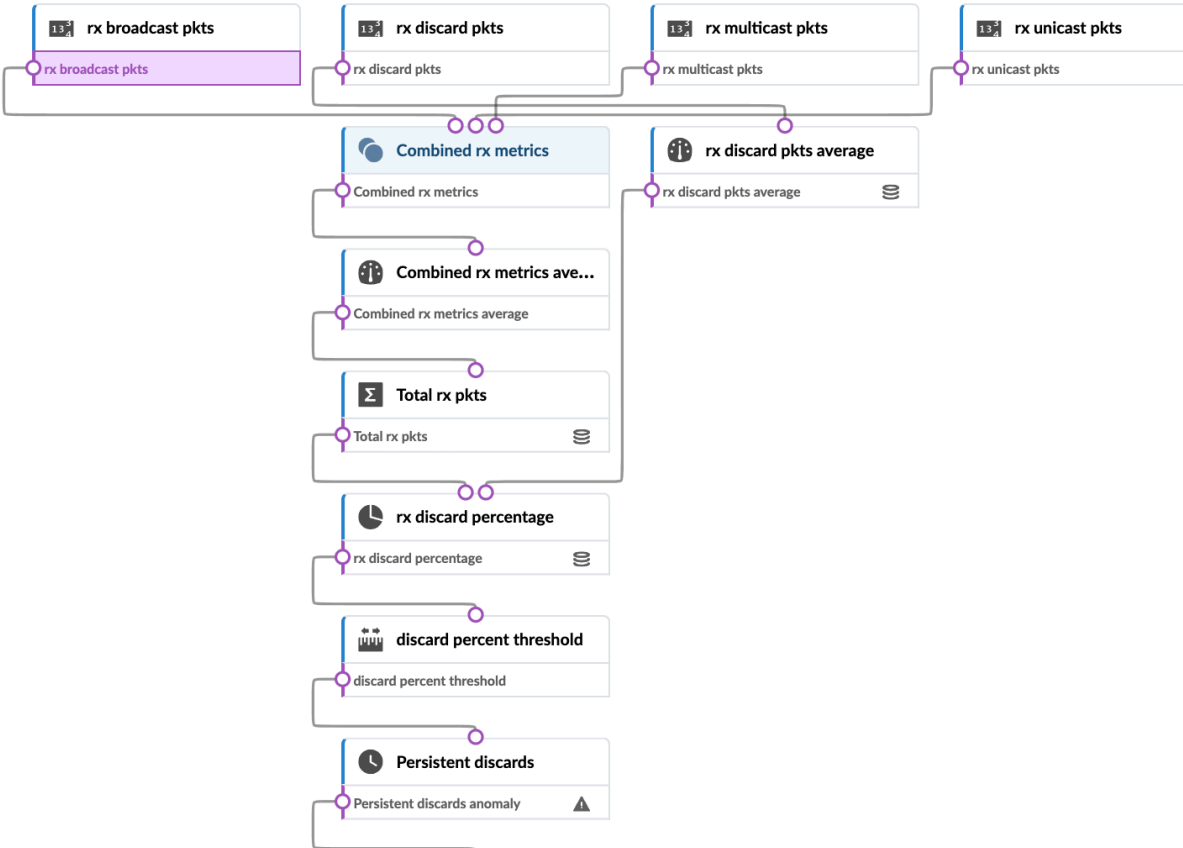
Time period in recent-history over which discard percentage data will be considered

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

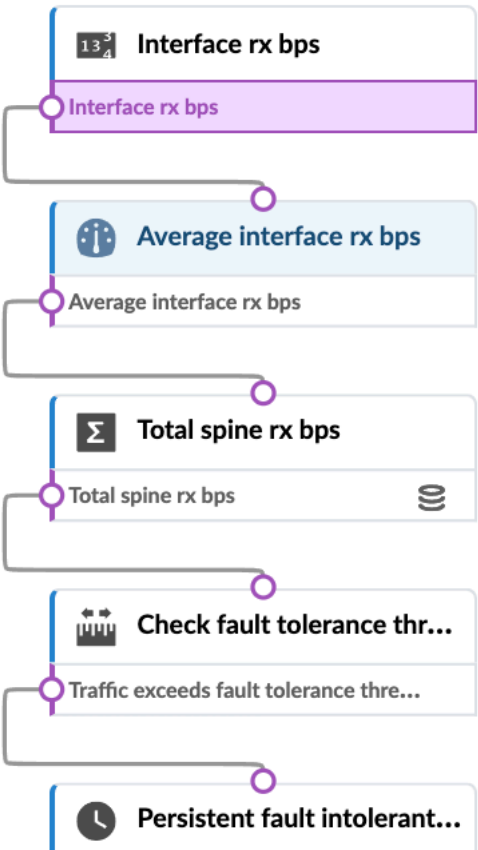
Time period in recent-history in which anomaly intolerant traffic will be considered

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 leaf devices 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).

| | | | |
|--------------------------------|---|--|--|
| Source Processors | external router south-north link traffic (Interface Counters) | Purpose: wires in interface traffic samples (measured in bytes per second) for traffic sent to external routers | |
| | | Output Stage: ext_router_interface_traffic | |
| | leaf server traffic counters (Interface Counters) | Purpose: wires in interface traffic samples (measured in bytes per second) for traffic received on leaf devices from the servers | |
| | | Output Stage: server_traffic_counters | 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). |
| Additional Processor(s) | external router south-north links traffic average (Periodic Average) | 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 | |
| | | Output Stage: ext_router_interface_traffic_avg | 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).

**server traffic average
(Periodic Average)**

Purpose: Calculate average server traffic during period specified by average_period facade parameter. Unit is bytes per second.

Input Stage: server_traffic_counters

**Output Stage:
server_traffic_avg**

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).

south-north traffic (Sum)

Purpose: Calculate total traffic by summing average traffic on each interface-facing external router. Unit is bytes per second.

Input Stage: ext_router_interface_traffic_avg

**Output Stage:
total_outgoing_traffic**

Total south-north traffic average in bytes per second.

total server traffic (Sum)

Purpose: Calculate total server traffic by summing average traffic on each interface attached to servers in receive direction. Unit is bytes per second.

Input Stage: server_traffic_avg

**Output Stage:
total_server_traffic**

Total server traffic average in bytes per second.

**outgoing_traffic_average
(Periodic Average)**

Purpose: Calculate total south-north traffic over average_period seconds, which is a facade parameter. Unit is bytes per second.

Input Stage: total_outgoing_traffic

| | | |
|--|---|---|
| | Output Stage: total_outgoing_traffic_average | Total south-north traffic average in bytes per second. |
| server generated traffic average (Periodic Average) | 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 | |
| | Output Stage: total_server_traffic_history | Time series showing total average server traffic over recent history. Unit is bytes per second. |
| east-west traffic (Subtract) | 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 | |
| | Output Stage: eastwest_traffic_history | 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)

| | | |
|--------------------------|--|--|
| Purpose | Calculate VMs missing a VLAN and calculate VMs not backed by VLANs on managed leaf devices connected to hypervisors. | |
| Source Processors | VMs backed by Fabric VLANs (generic graph collector) | output stage: VMs backed by Fabric VLANs (number set) (generated from graph) |
| | VMs on hypervisors connected to Fabric (generic) | 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

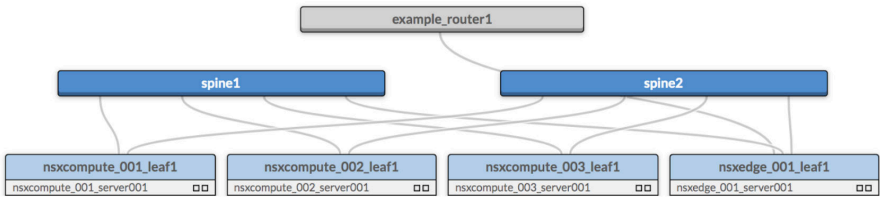
NSX-T Integration - VMs participating in a particular network are attached to an NSX logical switch. In NSX transport zone controls to which hypervisors or ESXi host an NSX logical switch can span. To have VXLAN connectivity for these VMs they need to be part of the same transport zone. This predefined 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 leaf devices 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 |
| View stage | | |

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-5260c3b671ac | 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-5260c3b671ac | 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

Collection period

5 Minutes

If routes are missing for more than or equal to percentage of Anomaly Time Window, an anomaly will be raised.

Telemetry collection interval.

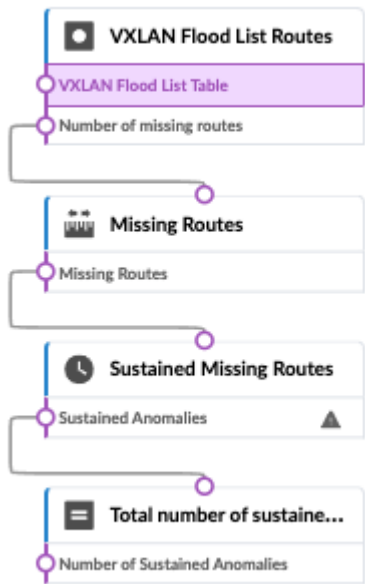
This probe validates the VXLAN flood list entries on every leaf in the network. It collects appropriate telemetry data, compares it to the set of flood list forwarding entries expected to be present and alerts if expected entries are missing on any device.

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 | 1078](#)
- [Processor: Average | 1082](#)
- [Processor: Comparison | 1083](#)
- [Processor: EVPN Type 3 | 1085](#)
- [Processor: EVPN Type 5 | 1085](#)
- [Processor: Extensible Service Data Collector | 1086](#)
- [Processor: Generic Graph Collector | 1090](#)
- [Processor: Generic Service Data Collector | 1093](#)
- [Processor: Interface Counters | 1096](#)
- [Processor: Logical Operator | 1099](#)
- [Processor: Match Count | 1100](#)
- [Processor: Match Percentage | 1102](#)
- [Processor: Match String | 1104](#)
- [Processor: Max | 1107](#)
- [Processor: Min | 1109](#)
- [Processor: Periodic Average | 1111](#)
- [Processor: Range | 1114](#)
- [Processor: Ratio | 1117](#)
- [Processor: Service Data Collector | 1119](#)
- [Processor: Set Comparison | 1122](#)
- [Processor: Set Count | 1124](#)
- [Processor: Standard Deviation | 1125](#)
- [Processor: State | 1127](#)
- [Processor: Subtract | 1130](#)

- [Processor: Sum | 1131](#)
- [Processor: System Utilization | 1132](#)
- [Processor: Time in State | 1133](#)
- [Processor: Traffic Monitor | 1138](#)
- [Processor: Union | 1141](#)
- [Processor: VXLAN Floodlist | 1143](#)

Processor: Accumulate

IN THIS SECTION

- [Example: Accumulate | 1080](#)

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.

| Parameter | 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) |

(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 leaf devices, or all interfaces between leaf and spine devices)</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</p> |

(Continued)

| Parameter | Description |
|--|--|
| | <p>accessed using the special <code>query_result</code> 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. <code>ps</code> is what the actual node is referred to in the query; the rest depends on the structure of the node. The <code>int()</code> casting is required because values of <code>property_set</code> nodes are strings. Here it's assumed that a property set node has the label <code>probe_propset</code> and that the value <code>accumulate_duration</code> 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, 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.</p> |
| Enable Streaming (<code>enable_streaming</code>) | 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) |

(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 grouping processors and they all take the Group by 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" on page 1125 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: Average

See ["standard deviation" on page 1125](#) example. It's the same except we calculate average instead of standard deviation.

Processor: Comparison

IN THIS SECTION

Example: Comparison | 1084

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) |
| 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 |

(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 leaf devices, or all interfaces between leaf and spine devices)</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"> • 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. • Any missing metric identity key under "ingestion_filter" is assumed to match. • 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. • The <code>data_type</code> must be one of the table data types. • 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. |
| 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 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"> • Value of query_group_by field - Semantics • 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. • Empty list ([]) - Produces one group containing all the query results. • 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. |
| 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 | 1093](#)

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 leaf devices, or all interfaces between leaf and spine devices)</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"> • Value of query_group_by field - Semantics • 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. • Empty list ([]) - Produces one group containing all the query results. • 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. |
| 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 leaf devices, or all interfaces between leaf and spine devices)</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"> • Value of query_group_by field - Semantics • 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. • Empty list ([]) - Produces one group containing all the query results. • 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. |
| 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 | 1099](#)

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 leaf devices, or all interfaces between leaf and spine devices)</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"> • Value of query_group_by field - Semantics • 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. • Empty list ([]) - Produces one group containing all the query results. • 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. |
| 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 | 1101](#)

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 grouping processors and they all take the Group by 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<processor_standard_deviation> 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 | 1103](#)

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 grouping processors and they all take the Group by 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 <processor_standard_deviation> 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 | 1106](#)

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 leaf devices, or all interfaces between leaf and spine devices)</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 | 1108

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 grouping processors and they all take the Group by 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" on page 1125 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 | 1110

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 grouping processors and they all take the Group by 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" on page 1125 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,iface_name=swp40] : 10
[system_id=leaf0,iface_name=swp41] : 11
[system_id=leaf0,iface_name=swp42] : 15
[system_id=spine0,iface_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](#) | 1113

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 leaf devices, or all interfaces between leaf and spine devices)</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 | 1116](#)

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 leaf devices, or all interfaces between leaf and spine devices)</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 | 1118](#)

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 | 1122](#)

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 leaf devices, or all interfaces between leaf and spine devices)</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"> • Value of query_group_by field - Semantics • 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. • Empty list ([]) - Produces one group containing all the query results. • 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. |
| 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 | 1123](#)

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 | 1125

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 grouping processors and they all take the Group by 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" on page 1125 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 1125](#) example. It's the same except we calculate the number of stage items.

Processor: Standard Deviation

IN THIS SECTION

Example: Standard Deviation | 1126

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 grouping processors and they all take the Group by 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" on page 1125 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 | 1129

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 leaf devices, or all interfaces between leaf and spine devices)</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 "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: 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 | 1132](#)

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 grouping processors and they all take the Group by 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" on page 1125 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 1125](#) 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 | 1135](#)

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 leaf devices, or all interfaces between leaf and spine devices)</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 |

(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 leaf devices, or all interfaces between leaf and spine devices)</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 | <p>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.</p> |

(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"> • Value of query_group_by field - Semantics • 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. • Empty list ([]) - Produces one group containing all the query results. • 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. |
| 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. |
| Period | Duration of the averaging period |

(Continued)

| Parameter | Description |
|--|---|
| 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 | 1142](#)

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](#) | 1144
- [Juniper Junos Configlet Example on 4.0.2: MTU \(section Interface-Level: Delete\)](#) | 1145
- [Juniper Junos Configlet Example on 4.0.2 Example: SNMP \(multiple sections\)](#) | 1145
- [Juniper Junos Configlet Example on 4.0.1 and 4.0.0: NTP \(section SYSTEM\)](#) | 1146
- [Cisco NX-OS Configlet Example: Syslog \(section SYSTEM\)](#) | 1146
- [Arista EOS Configlet Example: NTP \(section SYSTEM\)](#) | 1147
- [Arista EOS Configlet Example: Interface Speed \(section INTERFACE\)](#) | 1147
- [Enterprise SONiC Configlet Example: NTP \(section SYSTEM\)](#) | 1147
- [Enterprise SONiC Configlet Example: SNMP \(section SYSTEM\)](#) | 1148

- [Enterprise SONiC Configlet Example: Syslog \(section SYSTEM\) | 1148](#)
- [Enterprise SONiC Configlet Example: Static Route \(section FRR\) | 1148](#)
- [Enterprise SONiC Configlet Example: sonic-cli Commands \(section SYSTEM\) | 1149](#)

Juniper Junos Configlet Interface-Level Example on 4.0.2: gigheter-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 "gigheter-options", you can use a interface-level hierarchical or set configlet.

```
gigheter-options no-auto-negotiation
gigheter-options fec none
```

```
gigheter-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 gigheter-options no-auto-negotiation
set interfaces xe-0/0/0 gigheter-options fec none
```

For a Junos Interface-Level hierarchical configlet Apstra software will load Junos structured configuration:

```
interfaces {
    xe-0/0/0 {
        gigheter-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. You must use `sudo -u admin` at the beginning, and surround terms that contain spaces with single quotes 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
```

Apstra-CLI Commands

IN THIS SECTION

- [change-device-password](#) | 1149
- [config-syntax-check \(Juniper only\)](#) | 1150

The "[Apstra-CLI](#)" on page 859 utility comes with a built-in feature that auto-completes commands. Use the TAB key to learn about this tool, its functionality, and available commands. See below for just a few of the available commands. For assistance with using Apstra-CLI, contact "[Juniper Support](#)" on page 824.

change-device-password

```
scenario change-device-password --blueprint <bp_id> --system <sys_id> --old-password <old_password> --new-password <new_password>
```

As of Apstra version 4.1.2, you can change device credentials for device system agents using `apstra-cli`. The new scenario command is a category of `apstra-cli` commands that include multiple tasks. `scenario change-device-password` is a collection of eleven tasks as follows:

- Check old password by ssh connection

- State creation of configlet for password
- Commit blueprint
- Check new password by ssh connection
- Change system agent password
- Check system agent status
- Update device pristine config
- State deletion of configlet used for password change
- Commit blueprint
- Check new password by ssh connection
- Check system agent status

config-syntax-check (Juniper only)

Command Syntax for Datacenter blueprints:

```
blueprint --blueprint <bp_id> config-syntax-check --system <sys_id> --username <device_username> --password <device_password>
```

Command Syntax for Freeform blueprints:

```
blueprint --blueprint <bp_id> freeform-system config-syntax-check --system <sys_id> --username <device_username> --password <device_password>
```

With the `config-syntax-check` command, you can verify configuration syntax on your Juniper devices before committing your blueprint (as of Apstra version 4.1.2). This check is useful when working with configlets in Datacenter blueprints and when working with config templates in Freeform blueprints.

This command works only with hierarchical configuration to verify whether configuration syntax is correct. It doesn't work for set commands.

Apstra EVPN Support Addendum

IN THIS SECTION

- [Qualified Vendor and NOS | 1151](#)
- [Limitations | 1152](#)
- [TCAM Carving in NX-OS | 1153](#)
- [Arista EOS VxLAN Routing | 1154](#)
- [Graph Node VTEP Types | 1156](#)

When deploying EVPN on Apstra-supported devices and NOSs, 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 979](#).

Hardware ASIC Support

Apstra supports EVPN on the following hardware ASICs:

- Arista DCS 7280SE with Arad chipset
- Cisco Cloudscale
- Mellanox Spectrum A1
- Trident Trident2 (see below)
- Trident Trident2+ (see below)
- Trident Trident3 (see below)
- Trident Tomahawk (see below)
- Juniper Q5

Table 28: Apstra EVPN ASIC Support

| ASIC | Example Switches | Notes |
|-------------------------|------------------------------------|---|
| Arista Trident2 | Arista DCS-7050 | Can use as Spine, Leaf, or Border Leaf. Must set up EOS Recirculation interface(s) to use as a Layer3 Leaf (see Arista VXLAN documentation for more information). |
| Arista Trident3 | DCS-7050CX3 | Can use as Spine, Leaf, or Border Leaf. |
| Arista XP80 | Arista DCS-7160 | Can use as Spine, Leaf, or Border Leaf. |
| Arista Jericho | DCS-7280R | Can use as Spine, Leaf, or Border Leaf. |
| Cisco Cloudscale | Cisco 93180YC-EX | Can use as Spine, Leaf, or Border Leaf |
| Cisco Trident2 with ALE | Cisco 9396PX, 9372PX, 9332PQ, 9504 | Can use as Spine, Leaf, or Border Leaf (see TCAM Carving in NXOS section). |
| Cisco Trident2+ | Cisco 3132Q-V | Can't use as Border Leaf |
| Juniper Q5 | Juniper QFX10002 | Can use as Spine, Leaf, or Border Leaf |
| Juniper Trident2 | Juniper QFX5100 | Can use as Spine or Layer2 Leaf |
| Juniper Trident2+ | Juniper QFX5110 | Can use as Spine, Leaf, or Border Leaf |
| Juniper Trident3 | Juniper QFX5120 | Can use as Spine, Leaf, or Border Leaf |

For recommended NOS versions, refer to Qualified Devices and NOS <device_support>.

Limitations

IN THIS SECTION

- [EVPN Layer2 Limitations | 1153](#)
- [EVPN Layer3 Limitations | 1153](#)

EVPN Layer2 Limitations

- VLAN (Rack-local) Virtual networks must be in the default routing zone.
- VxLAN (Inter-rack) Virtual networks can't 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 leaf devices, spine devices or superspine devices.
- 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.

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 | 1155](#)
- [VxLAN Routing System Profile for Arista Jericho Devices | 1155](#)
- [VxLAN Routing Profile for Arista Arad Devices | 1156](#)

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
```

VxLAN Routing Profile for Arista Arad Devices

We recommend when using VxLAN Routing for Arista Arad devices (for example, on 7280SE platform) that you assign EOS VxLAN Routing 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
  profile vxlan-routing
```

Graph Node VTEP Types

IN THIS SECTION

- [Unicast VTEPs | 1156](#)
- [Logical VTEPs | 1157](#)
- [Anycast VTEP | 1158](#)

Unicast VTEPs

Unicast VTEPs do not apply to 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 leaf devices share same logical VTEP IP

- Singleton leaf device 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
```

Anycast VTEP

Anycast VTEPs do not apply to Cisco.

Arista Anycast VTEPs

Apstra IP Allocation

One anycast VTEP for entire blueprint, shared between all Arista leaf devices

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

```

Apstra Server Configuration File

IN THIS SECTION

- [Controller | 1160](#)
- [Security | 1160](#)
- [Log Rotate | 1161](#)
- [Auth Sysdb Log Rotator | 1161](#)
- [Main Sysdb Log Rotator | 1162](#)
- [Anomaly Sysdb Log Rotator | 1164](#)
- [Device Image Management | 1164](#)
- [Authentication | 1165](#)
- [Device Config Management | 1165](#)
- [Telemetry Init | 1165](#)
- [Telemetry Global Config | 1166](#)

- [Task API | 1166](#)
- [Statistics | 1167](#)
- [Enterprise | 1167](#)
- [Syslog | 1167](#)
- [Builtin Telemetry Disable | 1168](#)
- [Agent Management | 1169](#)
- [Show Tech | 1169](#)
- [System Operation Filesystem Thresholds | 1170](#)
- [System Operation Memory Thresholds | 1170](#)

/etc/aos/aos.conf

Controller

```
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

```
[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
# This encrypts sensitive data when sending configuration to device. This also
# enables aos agents to use appropriate credentials to access and/or configure
```



```
# device. Default behavior to configure or run commands using device root
# Note: Manual agent installation will not work if this is enabled.
enable_encryption_to_device = 0
```

Log Rotate

```
[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
# Maximum number of recent invalid persistence group kept
max_kept_invalid_persistence_groups = 3
```

Auth Sysdb Log Rotator

```
[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

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**.

Anomaly Sysdb Log Rotator

```
[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

```
[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

```

[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
# Enable ratelimiting. This mechanism protects against password bruteforce. By
# default ratelimiting is enabled. You can disable it by setting
# enable_ratelimit to 0
enable_ratelimit = 1

```

Device Config Management

```

[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

```

[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.
mlag = 4

```

Telemetry Global Config

```

[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

```

[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

```

[statistics]

# Enable or disable full validation for pod statistics. Disable if Racks and/or
# Pods tabs load times are excessive
pod_full_validation = enabled

```

Enterprise

```

[enterprise]

# Enable or disable Enterprise related features
enable = 0

```

Syslog

```

[syslog]

# Interval, specified as <hh:mm:ss>, at which collector will recollect hostname
hostname_check_interval = 00:00:10

```

Builtin Telemetry Disable

New in Apstra version 4.1.1

```
[builtin_telemetry_disable]

# Disable telemetry service lldp for the specified set of system IDs. System
# IDs can be provided as a comma separated list(eg: a, b, c, d). In order to
# disable the service for all devices, specify the value "all".
lldp_disable_devices =

# Disable telemetry service arp for the specified set of system IDs. System IDs
# can be provided as a comma separated list(eg: a, b, c, d). In order to
# disable the service for all devices, specify the value "all".
arp_disable_devices =

# Disable telemetry service hostname for the specified set of system IDs.
# System IDs can be provided as a comma separated list(eg: a, b, c, d). In
# order to disable the service for all devices, specify the value "all".
hostname_disable_devices =

# Disable telemetry service mac for the specified set of system IDs. System IDs
# can be provided as a comma separated list(eg: a, b, c, d). In order to
# disable the service for all devices, specify the value "all".
mac_disable_devices =

# Disable telemetry service xcvr for the specified set of system IDs. System
# IDs can be provided as a comma separated list(eg: a, b, c, d). In order to
# disable the service for all devices, specify the value "all".
xcvr_disable_devices =

# Disable telemetry service interface for the specified set of system IDs.
# System IDs can be provided as a comma separated list(eg: a, b, c, d). In
# order to disable the service for all devices, specify the value "all".
interface_disable_devices =

# Disable telemetry service interface_counters for the specified set of system
# IDs. System IDs can be provided as a comma separated list(eg: a, b, c, d). In
# order to disable the service for all devices, specify the value "all".
interface_counters_disable_devices =

# Disable telemetry service bgp for the specified set of system IDs. System IDs
# can be provided as a comma separated list(eg: a, b, c, d). In order to
# disable the service for all devices, specify the value "all".
bgp_disable_devices =

# Disable telemetry service mlag for the specified set of system IDs. System
# IDs can be provided as a comma separated list(eg: a, b, c, d). In order to
# disable the service for all devices, specify the value "all".
mlag_disable_devices =
```



```
# Disable telemetry service route for the specified set of system IDs. System
# IDs can be provided as a comma separated list(eg: a, b, c, d). In order to
# disable the service for all devices, specify the value "all".
route_disable_devices =
# Disable telemetry service lag for the specified set of system IDs. System IDs
# can be provided as a comma separated list(eg: a, b, c, d). In order to
# disable the service for all devices, specify the value "all".
lag_disable_devices =
```

Agent Management

New in Apstra version 4.1.1

```
[agent_management]

# Override the default heartbeat timeout for agents spawned dynamically by
# AgentManager. The value must be a non-negative number. The unit is seconds.
# The value 0 is used to turn off heartbeat-based agent timeouts and restarts.
# The minimum non-0 value allowed is 60. If not provided, then the default
# timeout value (600 seconds) is used.
heartbeat_period =
```

Show Tech

New in Apstra version 4.1.1

```
[show_tech]

# Minimum free space in the file system for /var/lib/aos/show_tech needed to
# initiate controller show tech collection via the Apstra API (in MBytes,
# default: 4096, min: 4096)
min_free_disk_space = 4096
# The directory /var/lib/aos/show_tech must be smaller than this size to
# initiate controller show tech collection via the Apstra API (in MBytes,
# default: 10240, min: 4096)
max_directory_size = 10240
# Maximum controller show tech collection duration before job times out (in
# seconds, default: 1200, min: 1200)
controller_timeout = 1200.0
```

System Operation Filesystem Thresholds

New in Apstra version 4.1.2

```
[system_operation_filesystem_thresholds]

# Default operation thresholds for filesystem utilization, used unless an
# option for a specific filesystem is specified in the section. Two thresholds
# are specified - warning and critical. When resource utilization passes each
# threshold, an operation anomaly is raised at the corresponding level. When a
# critical threshold is crossed the APIs are automatically transitioned into
# read-only mode. Numbers here are utilization levels, between 0.0 and 1.0.
# Note: Both 0.0 and 1.0 utilization levels are not allowed.
default = warning:0.8 critical:0.9
```

System Operation Memory Thresholds

New in Apstra version 4.1.2

```
[system_operation_memory_thresholds]

# Operation thresholds for memory utilization of the controller VM. Two
# thresholds are specified - warning and critical. When resource utilization
# passes each threshold, an operation anomaly is raised at the corresponding
# level. When a critical threshold is crossed the APIs are automatically
# transitioned into read-only mode. Numbers here are utilization levels,
# between 0.0 and 1.0. Note: Both 0.0 and 1.0 utilization levels are not
# allowed.
default = warning:0.8 critical:0.9
```

Agent Configuration File (Devices)

IN THIS SECTION

 [Controller Section | 1171](#)

- [Service Section | 1172](#)
- [Logrotate Section | 1173](#)
- [Device Info Section | 1174](#)
- [Device Profile Section | 1174](#)

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
```

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` To push telemetry (alerts) only, leave the default value of 0. Configuration files won't be modified unless a network administrator specifies it.

`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

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
```

Graph

IN THIS SECTION

- [Graph Overview | 1175](#)
- [Query Specification | 1176](#)
- [Change Notification | 1178](#)
- [Notification Processing | 1178](#)
- [Putting It All Together | 1180](#)
- [Convenience Functions | 1181](#)
- [Apstra Graph Datastore | 1190](#)

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 run searches on the graph specified by graph queries to find particular patterns (subgraphs) in a graph. 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)
```

Name objects to refer to them and use 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 GUI.

```
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

The query 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

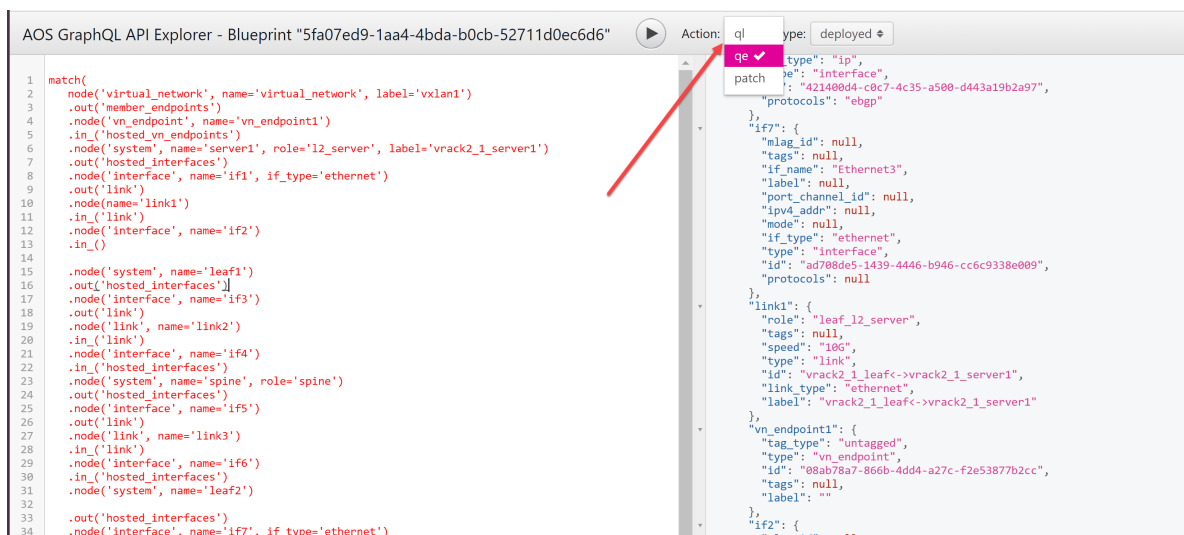
    """
    spine = path['spine_device']
    leaf = path['leaf_device']
```

```
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 GUI.

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 leaf devices and spine devices 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 Preview

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 824](#) 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 © 2025 Juniper Networks, Inc. All rights reserved.