
Apstra Documentation

Release 3.3.0

Jun 07, 2021

CONTENTS

| | | |
|----------|---|----------|
| 1 | Getting Started | 3 |
| 1.1 | Up and Running with Juniper Apstra Software | 3 |
| 1.2 | Design / Build Physical Network | 3 |
| 1.2.1 | Devices | 3 |
| 1.2.2 | Design | 4 |
| 1.2.3 | Resources | 4 |
| 1.2.4 | External Systems | 4 |
| 1.2.5 | Blueprints | 4 |
| 1.2.6 | Next Steps | 5 |
| 2 | Apstra Server | 7 |
| 2.1 | Apstra Server Requirements/References | 7 |
| 2.1.1 | Hypervisors | 7 |
| 2.1.2 | Apstra Server VM Resources | 7 |
| 2.1.3 | Network Security Protocols | 8 |
| 2.1.4 | Other Network Protocols | 9 |
| 2.1.5 | Network Client Services | 9 |
| 2.2 | Installing Apstra Server | 10 |
| 2.2.1 | Install Apstra Server on VMware ESXi | 10 |
| 2.2.2 | Install Apstra Server on KVM | 15 |
| 2.2.2.1 | Method One: via Virtual Machine Manager | 15 |
| 2.2.2.2 | Method Two: Via Command Line | 20 |
| 2.2.3 | Install Apstra Server on Hyper-V | 21 |
| 2.2.4 | Install Apstra Server on VirtualBox | 30 |
| 2.2.5 | Configuring Apstra Server | 31 |
| 2.2.5.1 | Initial Setup | 31 |
| 2.2.5.2 | Configuring Static Management IP Address | 34 |
| 2.2.5.3 | Updating SSH Host Keys | 35 |
| 2.2.5.4 | Changing Default Docker Networks | 36 |
| 2.2.5.5 | Changing Apstra Server Hostname | 36 |
| 2.2.5.6 | Apstra Server Configuration File | 37 |
| 2.3 | Managing Apstra Server | 43 |
| 2.3.1 | Monitoring Apstra Server via CLI | 43 |
| 2.3.2 | Restarting Apstra Server | 43 |
| 2.3.3 | Reinstalling Apstra Server | 44 |
| 2.3.4 | Apstra Server Database Management | 46 |
| 2.3.4.1 | Backing up Database | 46 |
| 2.3.4.2 | Validating Backup | 47 |
| 2.3.4.3 | Restoring Database | 47 |
| 2.3.4.4 | Validating Restore | 49 |

| | | |
|----------|---|-----------|
| 2.3.4.5 | Resetting Database | 50 |
| 2.3.4.6 | Migrating Database | 50 |
| 2.3.5 | Resetting Apstra Server VM Password | 54 |
| 2.4 | Upgrading Apstra Server | 59 |
| 2.4.1 | Supported Upgrade Paths | 59 |
| 2.4.2 | Known Limitations and Issues | 60 |
| 2.4.2.1 | API Changes | 61 |
| 2.4.2.2 | Rack Types and Rack Based Templates | 61 |
| 2.4.2.3 | Virtual Network Creation | 61 |
| 2.4.3 | Upgrading Apstra server VM In-Place | 62 |
| 2.4.3.1 | Pre-Upgrade Validation (Same VM) | 62 |
| 2.4.3.2 | Apstra Server In-Place Upgrade To Version 3.2 | 63 |
| 2.4.3.3 | Apstra Server In-Place Upgrade To Version 3.1 | 67 |
| 2.4.3.4 | Apstra Server Agents In-Place Upgrade | 69 |
| 2.4.3.5 | AOS Cluster In-Place VM Upgrade | 71 |
| 2.4.4 | Upgrading Apstra Server onto Different VM (VM-VM) | 72 |
| 2.4.4.1 | Pre-Upgrade Validation (Different VM) | 72 |
| 2.4.4.2 | Deploy New Apstra Server (Different VM) | 73 |
| 2.4.4.3 | Import State (Different VM) | 73 |
| 2.4.4.4 | Cluster Upgrade to Different VM (using Import State) | 75 |
| 2.4.4.5 | Apstra Server Rollback (Different VM) | 76 |
| 2.4.4.6 | System Agents Upgrade (Different VM) | 76 |
| 2.4.4.7 | Proxy and DNS updates, Shutdown of old Apstra Server (Different VM) | 76 |
| 2.4.5 | Updating SSH Host Keys | 77 |
| 2.4.6 | Q & A | 77 |
| 3 | Web Interface (UI) | 79 |
| 3.1 | Accessing Web Interface | 79 |
| 3.2 | Resetting Admin Password | 80 |
| 3.3 | Replacing SSL Certificate | 80 |
| 3.3.1 | Replacing Existing Certificate with Signed Certificate | 80 |
| 3.3.2 | Replacing Existing Certificate with Self-Signed Certificate | 82 |
| 3.4 | Checking Web Interface Version | 83 |
| 3.5 | Updating Web Interface Version | 83 |
| 3.6 | Restoring Web Interface Version | 84 |
| 4 | Blueprints | 85 |
| 4.1 | Creating Blueprint | 85 |
| 4.2 | Dashboard | 85 |
| 4.2.1 | Blueprint Dashboard Overview | 85 |
| 4.2.2 | Deleting Blueprint | 86 |
| 4.3 | Analytics | 87 |
| 4.3.1 | Intent-Based Analytics | 87 |
| 4.3.1.1 | Disk Usage | 87 |
| 4.3.2 | Dashboards | 88 |
| 4.3.2.1 | Analytics Dashboard Overview | 88 |
| 4.3.2.2 | Configuring Auto-Enabled Dashboards | 89 |
| 4.3.2.3 | Instantiating Predefined Dashboard | 89 |
| 4.3.2.4 | Creating Dashboard | 91 |
| 4.3.2.5 | Editing Dashboard | 91 |
| 4.3.2.6 | Deleting Dashboard | 91 |
| 4.3.3 | Anomalies (IBA) | 91 |
| 4.3.4 | Widgets | 92 |
| 4.3.4.1 | Widget Overview | 92 |

| | | |
|---------|--|-----|
| 4.3.4.2 | Creating Anomaly Heat Map Widget | 92 |
| 4.3.4.3 | Creating Stage Widget from Widgets View | 93 |
| 4.3.4.4 | Creating Stage Widget from Probes View | 93 |
| 4.3.4.5 | Editing Widget | 93 |
| 4.3.4.6 | Deleting Widget | 93 |
| 4.3.5 | Probes | 94 |
| 4.3.5.1 | IBA Probes Overview | 94 |
| 4.3.5.2 | Importing Probe | 96 |
| 4.3.5.3 | Exporting Probe | 96 |
| 4.3.5.4 | Instantiating Predefined Probe | 97 |
| 4.3.5.5 | Creating Probe | 146 |
| 4.3.5.6 | Editing Probe | 181 |
| 4.3.5.7 | Deleting Probe | 182 |
| 4.4 | Staged | 182 |
| 4.4.1 | Physical | 182 |
| 4.4.1.1 | Build (Physical) | 182 |
| 4.4.1.2 | Topology | 192 |
| 4.4.1.3 | Nodes | 205 |
| 4.4.1.4 | Links | 209 |
| 4.4.1.5 | Racks | 213 |
| 4.4.1.6 | Pods | 217 |
| 4.4.2 | Virtual | 218 |
| 4.4.2.1 | Build - Virtual | 218 |
| 4.4.2.2 | Virtual Networks | 219 |
| 4.4.2.3 | Security Zones | 242 |
| 4.4.2.4 | Data Center Interconnect / EVPN Gateways | 251 |
| 4.4.2.5 | Virtual Infra | 264 |
| 4.4.2.6 | Endpoints | 264 |
| 4.4.3 | Policies | 268 |
| 4.4.3.1 | Security Policies | 268 |
| 4.4.3.2 | Interface Policies | 276 |
| 4.4.4 | Catalog | 283 |
| 4.4.4.1 | Logical Devices | 283 |
| 4.4.4.2 | Interface Maps | 284 |
| 4.4.4.3 | External Routers | 285 |
| 4.4.4.4 | Property Sets | 286 |
| 4.4.4.5 | Configlets | 287 |
| 4.4.4.6 | AAA Servers | 288 |
| 4.4.5 | Settings | 290 |
| 4.4.5.1 | Fabric Addressing Policy | 290 |
| 4.4.5.2 | L3 Edge IP Connectivity | 291 |
| 4.4.5.3 | Virtual Network Policy | 291 |
| 4.4.6 | Tasks | 293 |
| 4.5 | Uncommitted | 293 |
| 4.5.1 | Uncommitted Overview | 293 |
| 4.5.2 | Committing Staged Changes | 294 |
| 4.5.3 | Reverting Staged Changes | 294 |
| 4.6 | Active | 295 |
| 4.6.1 | Physical | 295 |
| 4.6.1.1 | Status | 295 |
| 4.6.1.2 | Selection | 296 |
| 4.6.1.3 | Topology | 296 |
| 4.6.1.4 | Nodes | 306 |
| 4.6.1.5 | Links | 307 |

| | | |
|----------|---|------------|
| 4.6.1.6 | Racks | 308 |
| 4.6.1.7 | Pods | 309 |
| 4.6.2 | Query | 309 |
| 4.6.2.1 | Searching the Active Blueprint | 310 |
| 4.6.3 | Anomalies (Service) | 310 |
| 4.6.3.1 | Discovery Anomalies | 310 |
| 4.6.3.2 | Configuration Deviation | 314 |
| 4.6.4 | Root Causes | 317 |
| 4.6.4.1 | Root Cause Overview | 317 |
| 4.6.4.2 | Enabling Root Cause Analysis | 317 |
| 4.6.4.3 | Viewing Root Cause Analysis | 317 |
| 4.7 | Time Voyager | 318 |
| 4.7.1 | Listing Blueprint Revisions | 319 |
| 4.7.2 | Keeping a Saved Blueprint Revision | 319 |
| 4.7.3 | Deleting a Kept Blueprint Revision | 320 |
| 4.7.4 | Jumping to a Previous Revision in a Blueprint | 320 |
| 5 | Devices | 321 |
| 5.1 | Managed Devices | 321 |
| 5.1.1 | Acknowledge Device(s) | 321 |
| 5.1.2 | Setting Admin State on Devices | 321 |
| 5.1.3 | Updating User Config | 322 |
| 5.1.4 | Editing System | 322 |
| 5.1.4.1 | Modular Devices and Device Profiles | 323 |
| 5.1.4.2 | Hyperloop Interface | 324 |
| 5.1.5 | Editing Pristine Config | 324 |
| 5.1.6 | Updating Pristine Config from Device | 325 |
| 5.1.7 | Deleting System(s) | 326 |
| 5.1.8 | Managed Device Telemetry | 326 |
| 5.2 | Telemetry | 326 |
| 5.2.1 | Telemetry Overview | 326 |
| 5.2.1.1 | Telemetry in the Web Interface | 326 |
| 5.2.1.2 | Telemetry Services | 328 |
| 5.2.1.3 | Telemetry Collection Statistics | 329 |
| 5.2.2 | L2 Server Telemetry | 330 |
| 5.2.3 | External Router Telemetry | 330 |
| 5.2.4 | Telemetry Streaming | 330 |
| 5.2.5 | Route Anomalies for a Host - Example | 331 |
| 5.2.6 | Telemetry Command Reference | 333 |
| 5.2.6.1 | Cisco | 333 |
| 5.2.6.2 | Arista | 333 |
| 5.2.6.3 | Cumulus | 334 |
| 5.2.6.4 | Linux Servers | 335 |
| 5.2.7 | Extensible Telemetry | 335 |
| 5.2.7.1 | AOS Device Drivers | 335 |
| 5.2.7.2 | Telemetry Collectors | 335 |
| 5.2.8 | Debugging Telemetry | 345 |
| 5.3 | Agents | 345 |
| 5.3.1 | Agents Overview | 345 |
| 5.3.1.1 | On-box Agents | 346 |
| 5.3.1.2 | Off-box Agents | 346 |
| 5.3.1.3 | Agent Details | 346 |
| 5.3.2 | Before Creating Agent | 348 |
| 5.3.2.1 | On-Box Agent Minimum Configuration | 348 |

| | | |
|----------|---|-----|
| 5.3.2.2 | Off-Box Agent Minimum Configuration | 348 |
| 5.3.2.3 | Uploading Packages | 350 |
| 5.3.2.4 | Retain Pre-existing Configuration | 350 |
| 5.3.3 | Creating Agent | 350 |
| 5.3.3.1 | On-box | 350 |
| 5.3.3.2 | Off-box | 352 |
| 5.3.3.3 | Agents on L2 Servers | 355 |
| 5.3.4 | Uninstalling Agent | 355 |
| 5.3.5 | Editing Agent | 355 |
| 5.3.6 | Deleting Agent | 355 |
| 5.3.7 | Upgrading Device Operating System | 355 |
| 5.3.7.1 | Device Operating System Upgrade | 355 |
| 5.3.8 | Vendor Specific Agent Information | 364 |
| 5.3.8.1 | Juniper Device Agent | 364 |
| 5.3.8.2 | Arista Device Agent | 367 |
| 5.3.8.3 | Cisco NX-OS Device Agent | 382 |
| 5.3.8.4 | Cumulus Device Agent | 391 |
| 5.3.8.5 | SONiC Device Agent | 406 |
| 5.3.8.6 | Server Agent | 412 |
| 5.4 | Agent Profiles | 416 |
| 5.4.1 | Creating Agent Profile | 417 |
| 5.4.2 | Editing Agent Profile | 418 |
| 5.4.3 | Deleting Agent Profile | 418 |
| 5.5 | Packages | 418 |
| 5.5.1 | Uploading Packages | 418 |
| 5.6 | OS Images | 418 |
| 5.6.1 | Registering Device OS Image | 419 |
| 5.6.1.1 | Method One: Upload Image | 420 |
| 5.6.1.2 | Method Two: Provide Image URL | 420 |
| 5.6.1.3 | Adding Checksum | 421 |
| 5.6.2 | Editing OS Image | 422 |
| 5.6.3 | Deleting OS Image | 422 |
| 5.7 | Apstra ZTP | 422 |
| 5.7.1 | Overview | 422 |
| 5.7.2 | Apstra ZTP 2.0.0 | 423 |
| 5.7.2.1 | Apstra ZTP 2.0.0 VM Server Resource Requirements | 423 |
| 5.7.2.2 | Apstra ZTP 2.0.0 Network requirements | 424 |
| 5.7.2.3 | Download and Deploy Apstra ZTP VM on Standalone Apstra ZTP VM | 424 |
| 5.7.2.4 | Configuring Static Management IP Address | 425 |
| 5.7.2.5 | Configure DHCP Server | 425 |
| 5.7.2.6 | Configure ZTP User on Apstra Server | 427 |
| 5.7.2.7 | Configure Apstra Server IP for ZTP | 427 |
| 5.7.2.8 | Edit ZTP Configuration File | 428 |
| 5.7.3 | Apstra ZTP 1.0.0 | 431 |
| 5.7.3.1 | Apstra ZTP 1.0.0 VM Server Resource Requirements | 431 |
| 5.7.3.2 | Apstra ZTP 1.0.0 Network requirements | 432 |
| 5.7.3.3 | Download and Deploy the Apstra ZTP VM | 432 |
| 5.7.3.4 | Deploy Apstra ZTP on the Apstra Server VM | 432 |
| 5.7.3.5 | Using the Apstra Server VM as an HTTP image server for ZTP | 435 |
| 5.7.3.6 | Configure the Apstra ZTP Server VM | 436 |
| 5.7.3.7 | Management IP address | 436 |
| 5.7.3.8 | Configure Apstra ZTP 1.0.0 User on Apstra Server | 437 |
| 5.7.3.9 | Configuring the DHCP Server | 438 |
| 5.7.3.10 | Editing the ZTP Configuration File | 440 |

| | | |
|----------|--|------------|
| 5.7.4 | Platform Specific Information | 442 |
| 5.7.4.1 | Cumulus Linux | 442 |
| 5.7.4.2 | Cisco NX-OS | 445 |
| 5.7.4.3 | Arista EOS | 447 |
| 5.7.4.4 | Juniper Junos | 450 |
| 5.7.4.5 | Enterprise SONiC | 452 |
| 5.7.5 | Monitoring DHCP, TFTP and HTTP Logs | 454 |
| 5.7.5.1 | ZTP Logs | 455 |
| 5.8 | Device Profiles | 458 |
| 5.8.1 | Cumulus Device Profile | 458 |
| 5.8.1.1 | Background | 458 |
| 5.8.1.2 | Problem Statement | 458 |
| 5.8.1.3 | Solution | 458 |
| 5.8.1.4 | User Interface | 459 |
| 5.8.1.5 | DP Data Model | 459 |
| 5.8.1.6 | Capabilities | 469 |
| 5.8.1.7 | Port specific semantics | 470 |
| 5.8.1.8 | Inter port constraints | 475 |
| 5.8.1.9 | Debugging and Recovery | 481 |
| 5.8.1.10 | Appendix | 482 |
| 5.8.1.11 | Example for DP JSON for Edgecore / Accton AS5712-54X | 482 |
| 5.8.1.12 | References | 528 |
| 5.8.2 | SONiC Device Profile | 528 |
| 5.8.2.1 | Background | 528 |
| 5.8.2.2 | Problem Statement | 528 |
| 5.8.2.3 | Solution | 529 |
| 5.8.2.4 | User Interface | 529 |
| 5.8.2.5 | Selector information | 529 |
| 5.8.2.6 | Capabilities | 529 |
| 5.8.2.7 | Port Specific Semantics | 530 |
| 5.8.2.8 | Debugging and recovery | 530 |
| 5.8.2.9 | Example of DP and port_config.ini | 531 |
| 5.8.3 | Juniper Device Profile | 562 |
| 5.8.3.1 | Overview | 562 |
| 5.8.3.2 | Juniper QFX10002 | 562 |
| 5.8.4 | Device Profiles Overview | 563 |
| 5.8.4.1 | Hardware Capabilities | 563 |
| 5.8.4.2 | Control Plane Policing | 565 |
| 5.8.4.3 | Autonomous Systems Sequencing Support | 566 |
| 5.8.4.4 | Interface Breakout | 566 |
| 5.8.5 | Listing Device Profiles | 567 |
| 5.8.6 | Viewing a Device Profile | 568 |
| 5.8.6.1 | Vendor Specific Schema for Port Settings | 572 |
| 5.8.7 | Creating a Device Profile | 572 |
| 5.8.8 | Cloning a Device Profile | 572 |
| 5.8.9 | Editing a Device Profile | 573 |
| 5.8.9.1 | Use Case: Enabling Auto-negotiation | 573 |
| 5.8.10 | Deleting a Device Profile | 576 |
| 5.8.11 | Device Profiles and REST API | 576 |
| 6 | Design | 577 |
| 6.1 | Logical Devices | 577 |
| 6.1.1 | Logical Device Overview | 577 |
| 6.1.2 | Creating Logical Device | 579 |

| | | |
|----------|---|------------|
| 6.1.2.1 | Creating Logical Device - Example | 579 |
| 6.1.3 | Editing Logical Device | 581 |
| 6.1.4 | Deleting Logical Device | 582 |
| 6.2 | Interface Maps | 582 |
| 6.2.1 | Interface Map Overview | 582 |
| 6.2.2 | Creating Interface Map | 584 |
| 6.2.2.1 | Creating Interface Map - Example: Breakout Ports | 584 |
| 6.2.2.2 | Use Case: Inter Port Constraints - Disabled Ports | 587 |
| 6.2.3 | Editing Interface Map | 590 |
| 6.2.4 | Deleting Interface Map | 591 |
| 6.3 | Rack Types | 591 |
| 6.3.1 | Rack Type Overview | 591 |
| 6.3.2 | Creating Rack Type | 593 |
| 6.3.2.1 | Creating Rack Type - Example | 593 |
| 6.3.3 | Editing Rack Type | 596 |
| 6.3.3.1 | Editing Rack types in Templates | 596 |
| 6.3.3.2 | Editing Rack types in Blueprints | 597 |
| 6.3.4 | Deleting Rack Type | 597 |
| 6.4 | Templates | 597 |
| 6.4.1 | Template Overview | 597 |
| 6.4.2 | Creating Rack Based Template | 599 |
| 6.4.3 | Creating Pod Based Template | 600 |
| 6.4.4 | Editing Template | 601 |
| 6.4.4.1 | Updating Rack Type in Rack Type Template | 601 |
| 6.4.5 | Deleting Template | 601 |
| 6.5 | Configlets | 601 |
| 6.5.1 | Configlet Overview | 601 |
| 6.5.1.1 | Rendering Order | 603 |
| 6.5.1.2 | Configlet Details | 603 |
| 6.5.2 | Creating Configlet | 606 |
| 6.5.2.1 | Configlets on Cumulus | 607 |
| 6.5.2.2 | Configlets and Property Sets | 607 |
| 6.5.2.3 | Cumulus Linux Configlet Examples | 608 |
| 6.5.2.4 | Cisco NX-OS Configlet Examples | 614 |
| 6.5.2.5 | Arista EOS Configlet Examples | 615 |
| 6.5.2.6 | Juniper Junos Configlet Examples | 618 |
| 6.5.2.7 | Enterprise SONiC Configlet Examples | 619 |
| 6.5.3 | Editing Configlet in Global Catalog | 623 |
| 6.5.4 | Deleting Configlet from Global Catalog | 623 |
| 6.5.5 | Troubleshooting | 623 |
| 6.5.5.1 | Configlets and Config Deviation | 623 |
| 6.6 | Property Sets | 624 |
| 6.6.1 | Property Set Overview | 624 |
| 6.6.2 | Creating Property Set | 624 |
| 6.6.3 | Editing Property Set | 625 |
| 6.6.4 | Deleting Property Set | 625 |
| 6.7 | TCP/UDP Port Aliases | 625 |
| 6.7.1 | TCP/UDP Port Alias Overview | 625 |
| 6.7.2 | Creating TCP/UDP Port Alias | 626 |
| 6.7.3 | Editing TCP/UDP Port Alias | 626 |
| 6.7.4 | Deleting TCP/UDP Port Alias | 626 |
| 7 | Resources | 627 |
| 7.1 | ASN Pools | 627 |

| | | |
|----------|---|------------|
| 7.1.1 | ASN Pool Overview | 627 |
| 7.1.2 | Creating ASN Pool | 628 |
| 7.1.3 | Editing ASN Pool | 628 |
| 7.1.4 | Deleting ASN Pool | 628 |
| 7.2 | VNI Pools | 629 |
| 7.2.1 | VNI Pool Overview | 629 |
| 7.2.2 | Creating VNI Pool | 629 |
| 7.2.3 | Editing VNI Pool | 630 |
| 7.2.4 | Deleting VNI Pool | 630 |
| 7.3 | IP Pools | 630 |
| 7.3.1 | IP Pool Overview | 630 |
| 7.3.2 | Creating IPv4 Pool | 631 |
| 7.3.2.1 | Warning for Overlapping IP Pools | 632 |
| 7.3.3 | Editing IPv4 Pool | 634 |
| 7.3.4 | Deleting IPv4 Pool | 635 |
| 7.4 | IPv6 Pools | 635 |
| 7.4.1 | IPv6 Pool Overview | 635 |
| 7.4.2 | Creating IPv6 Pool | 636 |
| 7.4.3 | Editing IPv6 Pool | 636 |
| 7.4.4 | Deleting IPv6 Pool | 636 |
| 8 | External Systems | 639 |
| 8.1 | Providers | 639 |
| 8.1.1 | Provider Overview | 639 |
| 8.1.2 | Creating LDAP Provider | 639 |
| 8.1.3 | Creating Active Directory Provider | 640 |
| 8.1.4 | Creating TACACS+ Provider | 642 |
| 8.1.4.1 | Configuring TACACS+ Provider | 642 |
| 8.1.5 | Creating RADIUS Provider | 643 |
| 8.1.5.1 | RADIUS Limitations | 644 |
| 8.1.6 | Editing Provider | 644 |
| 8.1.7 | Deleting Provider | 644 |
| 8.1.8 | Provider Role Mapping | 645 |
| 8.1.8.1 | Creating Role Map | 645 |
| 8.1.8.2 | Editing Role Map | 646 |
| 8.1.8.3 | Deleting Role Map | 646 |
| 8.2 | External Routers | 646 |
| 8.2.1 | External Router Overview | 646 |
| 8.2.1.1 | Characteristics and Requirements | 646 |
| 8.2.1.2 | BGP Specific Requirements | 646 |
| 8.2.1.3 | OSPF Specific Requirements | 647 |
| 8.2.1.4 | Policies | 647 |
| 8.2.1.5 | External Router Config - BGP Example | 647 |
| 8.2.1.6 | External Router Config - OSPF Example | 648 |
| 8.2.1.7 | External Router Details | 649 |
| 8.2.2 | Creating External Router | 650 |
| 8.2.3 | Editing External Router | 650 |
| 8.2.4 | Deleting External Router | 650 |
| 9 | Platform | 651 |
| 9.1 | User Management | 651 |
| 9.1.1 | Creating User Profile | 652 |
| 9.1.1.1 | Use Case 1: Only Create Virtual Networks (not Including Allocating Resources) | 652 |
| 9.1.1.2 | Use Case 2: Create Virtual Networks and Allocate Resources | 652 |

| | | |
|-----------|--|------------|
| 9.1.2 | Editing User Profile | 653 |
| 9.1.3 | Changing User Password | 653 |
| 9.1.4 | Logging Out User | 654 |
| 9.1.5 | Deleting User Profile | 654 |
| 9.2 | Role Management | 654 |
| 9.2.1 | Role Management Overview | 654 |
| 9.2.2 | Creating User Role | 657 |
| 9.2.2.1 | Use Case 1: Read, Write and Commit Specified Blueprints | 657 |
| 9.2.2.2 | Use Case 2: Manage VN Endpoints on Specified Blueprints | 658 |
| 9.2.2.3 | Use Case 3: Create Virtual Networks (not Including Allocating Resources) | 659 |
| 9.2.2.4 | Use Case 3A: Create Virtual Networks and Allocate Resources | 660 |
| 9.2.2.5 | Use Case 4: Read and Write Resources for All Blueprints | 660 |
| 9.2.3 | Editing User Role | 661 |
| 9.2.4 | Deleting User Role | 662 |
| 9.3 | Syslog Configuration | 662 |
| 9.3.1 | Configuring Syslog Servers | 662 |
| 9.4 | Streaming Architecture | 664 |
| 9.4.1 | AOSOM-Streaming | 664 |
| 9.4.1.1 | Using Aosom-streaming | 665 |
| 9.4.1.2 | Aosom-Streaming configuration | 669 |
| 9.4.1.3 | (Optional) Build Aosom-Streaming VM | 672 |
| 9.4.1.4 | Troubleshooting | 675 |
| 9.5 | Receiver Configuration | 676 |
| 9.5.1 | Configuring Using Telegraf Plugin | 676 |
| 9.5.1.1 | AOS Plugin Configuration | 677 |
| 9.5.2 | Configuring Using AOS Web Interface | 677 |
| 9.5.2.1 | Creating Receiver | 678 |
| 9.5.2.2 | Deleting Receiver | 678 |
| 9.6 | Event Log | 678 |
| 9.6.1 | Exporting Event Log to CSV File | 679 |
| 9.6.2 | Sending Event Log with Syslog | 680 |
| 9.7 | AOS Cluster | 680 |
| 9.7.1 | Creating Apstra VM | 682 |
| 9.7.2 | Editing Apstra VM | 682 |
| 9.7.3 | Deleting Apstra VM | 682 |
| 9.8 | Developers | 683 |
| 9.8.1 | API Documentation | 683 |
| 9.8.1.1 | API Examples | 683 |
| 9.8.2 | Tools | 716 |
| 9.8.2.1 | REST API Explorer | 716 |
| 9.9 | Technical Support | 717 |
| 9.9.1 | Providing Technical Support Data | 718 |
| 9.9.1.1 | Show Tech for On-box Agents and Apstra Server (Web Interface) | 718 |
| 9.9.1.2 | Show Tech for Junos Off-box Agents (CLI) | 719 |
| 9.9.1.3 | Show Tech for Off-box Agents (CLI) | 720 |
| 9.9.1.4 | Show Tech for Apstra Server (CLI) | 721 |
| 9.9.1.5 | Show Tech for Arista On-box Agents (CLI) | 722 |
| 9.9.1.6 | Show Tech for Cisco On-box Agents (CLI) | 722 |
| 9.9.1.7 | Show Tech for Cumulus On-box Agents (CLI) | 723 |
| 9.9.1.8 | Show Tech for SONiC On-box Agents (CLI) | 723 |
| 10 | Favorites | 725 |
| 11 | User | 727 |

| | | |
|-----------|--|------------|
| 11.1 | Profile | 727 |
| 11.1.1 | Changing Your Profile Password | 727 |
| 11.1.2 | Changing Your Profile Name/Email | 727 |
| 11.1.3 | Managing Favorites | 728 |
| 11.2 | Log Out | 728 |
| 12 | Guides | 729 |
| 12.1 | Device Guides | 729 |
| 12.1.1 | AOS Device Configuration Lifecycle | 729 |
| 12.1.1.1 | Terminology | 729 |
| 12.1.1.2 | Configuration stages: Overview | 730 |
| 12.1.1.3 | Configuration stages: Detail | 731 |
| 12.1.1.4 | Configuration Deviations | 734 |
| 12.1.1.5 | Device Offline (Unavailable) | 735 |
| 12.1.1.6 | Manually Applying Full Config | 735 |
| 12.1.1.7 | Deploy Modes | 735 |
| 12.1.2 | Adding Device | 736 |
| 12.1.3 | Deploying Device | 737 |
| 12.1.3.1 | Vendor specifics | 737 |
| 12.1.4 | Draining Device Traffic | 738 |
| 12.1.4.1 | Monitoring Traffic Draining | 738 |
| 12.1.5 | Removing Device | 740 |
| 12.1.5.1 | Removing Device from Blueprint | 740 |
| 12.1.5.2 | Removing Device from AOS Management | 741 |
| 12.1.5.3 | Replacing Device | 742 |
| 12.1.6 | Device AAA support | 742 |
| 12.1.6.1 | Supported Platforms | 742 |
| 12.2 | 5-stage Clos Architecture | 743 |
| 12.2.1 | 5-Stage Clos Overview | 743 |
| 12.2.1.1 | 5-Stage Clos Limitations | 744 |
| 12.2.1.2 | 5-Stage Clos and EVPN | 745 |
| 12.2.2 | Creating 5-Stage Clos Network | 745 |
| 12.2.3 | Modifying 5-stage Clos Network | 746 |
| 12.3 | Integrations | 746 |
| 12.3.1 | VMware vSphere Integration | 746 |
| 12.3.1.1 | VMware vSphere Integration Overview | 746 |
| 12.3.1.2 | Enabling vSphere Integration | 747 |
| 12.3.1.3 | VM Visibility | 747 |
| 12.3.1.4 | Validating Virtual Infra Integration | 748 |
| 12.3.1.5 | vCenter Policy-Based Auto-Remediation | 748 |
| 12.3.1.6 | Disabling Virtual Infra Integration | 750 |
| 12.3.2 | VMware NSX-T Integration | 750 |
| 12.3.2.1 | Overview | 750 |
| 12.3.2.2 | Enabling NSX-T Integration | 750 |
| 12.3.2.3 | Virtual Infrastructure Visibility | 752 |
| 12.3.2.4 | Validating Virtual Infra Integration | 754 |
| 12.3.2.5 | Disabling Virtual Infra Integration | 754 |
| 12.3.3 | VMware NSX-T Inventory mapping to AOS Virtual Infrastructure | 755 |
| 12.3.3.1 | Overview | 755 |
| 12.3.3.2 | NSX-T Networking Terminology and correlation | 755 |
| 12.3.3.3 | NSX Inventory Model | 758 |
| 12.3.3.4 | Exact Model details and Relationship | 759 |
| 12.3.4 | NSX-T Security Policy Integration | 772 |
| 12.3.4.1 | Introduction | 772 |

| | | |
|-----------|--|------------|
| 12.3.4.2 | Use Cases | 772 |
| 12.3.4.3 | Overview | 773 |
| 12.3.4.4 | Setup AOS NSX-T Proxy | 773 |
| 12.3.4.5 | Create Distributed firewall(DFW) policy on NSX-T | 777 |
| 12.3.4.6 | Commit Policy for NSX-T proxy | 781 |
| 12.3.4.7 | Policy Translation Details | 782 |
| 12.3.4.8 | EVPN Support | 785 |
| 12.3.4.9 | Error Handling | 785 |
| 12.3.4.10 | AOS Policy Conflicts | 786 |
| 12.3.4.11 | Limitations | 786 |
| 12.4 | Other Guides | 787 |
| 12.4.1 | Apstra IBA Getting Started Tutorial | 787 |
| 12.4.1.1 | Install AOS CLI | 787 |
| 12.4.1.2 | Install Custom Collector Package | 787 |
| 12.4.1.3 | Bulk Update System Agent Profile | 791 |
| 12.4.1.4 | Apstra IBA Probes | 793 |
| 12.4.1.5 | Sending IBA Info with Syslog | 797 |
| 12.4.2 | Enable VXLAN Routing on Cumulus Tomahawk | 798 |
| 12.4.2.1 | Creating a Device Profile for Hyperloop | 798 |
| 12.4.2.2 | Updating Logical Device for Hyperloop | 800 |
| 12.4.2.3 | Creating an Interface Map for Hyperloop | 800 |
| 12.4.2.4 | Updating Managed Devices for Hyperloop | 802 |
| 12.4.2.5 | Assigning Hyperloop Device Profile | 803 |
| 12.4.2.6 | Verifying Loopback Port | 803 |
| 12.4.3 | Adding Access Layer Switches | 804 |
| 12.4.3.1 | Design | 804 |
| 12.4.3.2 | Blueprints | 806 |
| 12.4.4 | Creating a LAG on a Server | 810 |
| 12.4.4.1 | Prerequisites | 810 |
| 12.4.4.2 | Server Configuration | 811 |
| 12.4.4.3 | Validation | 812 |
| 12.4.4.4 | Procfs diagnostics | 812 |
| 12.4.5 | L3 Server Configurations | 813 |
| 12.4.5.1 | Introduction | 813 |
| 12.4.5.2 | Managed Devices | 817 |
| 12.4.5.3 | Cloning device-profiles | 820 |
| 12.4.6 | Building a Virtual Lab | 822 |
| 12.4.6.1 | AOS Virtual Lab on EVE-NG | 822 |
| 12.4.6.2 | Build an ESXi vEOS Lab | 862 |
| 12.4.7 | Juniper EVPN Support | 891 |
| 12.4.7.1 | Overview | 891 |
| 12.4.7.2 | Understanding EVPN multi-homing Terminology and Concepts | 891 |
| 12.4.7.3 | EVPN Services | 893 |
| 12.4.7.4 | Configuration Rendering | 894 |
| 12.4.7.5 | Limitations | 897 |
| 12.4.8 | Mixed Uplink Speeds between Leafs and Spines | 897 |
| 12.5 | AOS-CLI | 900 |
| 12.5.1 | Installing AOS-CLI | 900 |
| 12.5.2 | Accessing AOS-CLI | 901 |
| 13 | Reference | 903 |
| 13.1 | AOS Feature Matrix | 903 |
| 13.2 | Device and NOS Support | 905 |
| 13.2.1 | Juniper Junos | 905 |

| | | |
|----------|---|-----|
| 13.2.1.1 | Supported Juniper Devices | 905 |
| 13.2.1.2 | Recommended Junos NOS Versions | 905 |
| 13.2.2 | Arista EOS | 905 |
| 13.2.2.1 | Supported Arista Devices | 905 |
| 13.2.2.2 | Recommended EOS NOS Versions | 906 |
| 13.2.3 | Cisco NX-OS | 906 |
| 13.2.3.1 | Supported Cisco Devices | 906 |
| 13.2.3.2 | Recommended NX-OS Versions | 906 |
| 13.2.4 | Cumulus Linux | 906 |
| 13.2.4.1 | Supported Cumulus Devices | 906 |
| 13.2.4.2 | Recommended Linux Versions | 907 |
| 13.2.5 | Enterprise SONiC | 907 |
| 13.2.5.1 | Supported Enterprise SONiC Devices | 907 |
| 13.2.5.2 | Recommended Enterprise SONiC Version | 907 |
| 13.2.6 | Requesting NOS Support | 908 |
| 13.3 | AOS Device Agent Configuration File | 908 |
| 13.3.1 | Controller Section | 908 |
| 13.3.1.1 | metadb | 908 |
| 13.3.1.2 | web | 909 |
| 13.3.1.3 | interface | 909 |
| 13.3.2 | Service Section | 909 |
| 13.3.2.1 | enable_configuration_service | 909 |
| 13.3.2.2 | backup_config_restoration_timeout | 909 |
| 13.3.3 | Logrotate Section | 910 |
| 13.3.4 | Device Info Section | 910 |
| 13.3.4.1 | model | 910 |
| 13.3.5 | Device Profile Section | 911 |
| 13.4 | AOS EVPN Support Addendum | 911 |
| 13.4.1 | Vendor Device OS Support | 911 |
| 13.4.2 | Hardware ASIC Support | 911 |
| 13.4.3 | Limitations | 912 |
| 13.4.3.1 | EVPN Layer2 Limitations | 912 |
| 13.4.3.2 | EVPN Layer3 Limitations | 913 |
| 13.4.4 | Cumulus RN-766 Support | 913 |
| 13.4.5 | TCAM Carving in NXOS | 913 |
| 13.4.5.1 | Cisco NXOSv TCAM Carving | 914 |
| 13.4.5.2 | Cisco Trident2 TCAM Carving | 914 |
| 13.4.6 | Arista EOS VxLAN Routing | 914 |
| 13.4.6.1 | Recirculation Interface for Arista Trident2 Devices | 914 |
| 13.4.6.2 | VxLAN Routing System Profile for Arista Jericho Devices | 915 |
| 13.4.7 | AOS Graph Node VTEP Types | 915 |
| 13.4.7.1 | Unicast VTEPs | 915 |
| 13.4.7.2 | Logical VTEPs | 916 |
| 13.4.7.3 | Anycast VTEP | 917 |
| 13.5 | Graph | 918 |
| 13.5.1 | Reference Design Schema | 919 |
| 13.5.2 | Query Specification | 919 |
| 13.5.3 | Change Notification | 920 |
| 13.5.4 | Notification Processing | 921 |
| 13.5.5 | Putting It All Together | 922 |
| 13.5.6 | Convenience Functions | 923 |
| 13.5.6.1 | Functions | 923 |
| 13.5.6.2 | PathQueryBuilder nodes | 924 |
| 13.5.6.3 | Property matchers | 926 |

| | | |
|--------|--|-----|
| 13.5.7 | AOS Graph Datastore | 929 |
| 13.6 | Juniper Apstra Technology Previews (Tech Previews) | 929 |

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.

Juniper Apstra 3.3.0 User Guide 3.3 documentation in PDF format is available here: [Apstra_33_Docs.pdf](#)

GETTING STARTED

1.1 Up and Running with Juniper Apstra Software

Welcome! To get started, you'll install and configure the Apstra server. Then for security purposes, we recommend that you replace the SSL certificate and change default passwords. Follow the links below for details.

1. Ensure that the server you're going to use for the Apstra server meets *requirements*.
2. *Install and configure the Apstra server*.
3. Replace the SSL certificate and change the default password for the *web interface (UI)*.

Now you're ready to design, build, deploy, operate and validate networks.

1.2 Design / Build Physical Network

Depending on the complexity of your design, other tasks may be required in addition to the ones included in this general workflow. You can work with devices, design, resources and external systems in any order before proceeding to the blueprint section.

1.2.1 Devices

1. *Device profiles* represent physical devices. Many device profiles are predefined for you. Check the list, and if one that you need is not included, you can create it.

```
Devices > Device Profiles
```

2. Create and install *agents* for the devices in your network. If you have many of the same devices using the same configuration you might consider creating *agent profiles*, which can streamline the task of creating agents.

```
Devices > System Agents > Agents
```

3. When agents are created, they appear in the managed devices list in the quarantined state. *Acknowledge* them to put them in the ready state which allows them to be managed. (If you have a *modular device* in your network, you may need to change the associated device profile. It's best to do this before acknowledging.)

```
Devices > Managed Devices
```

1.2.2 Design

1. *Logical devices* are abstractions of physical devices. Check existing logical devices for ones that meet your requirements and create them as needed for your design.

```
Design > Logical devices
```

2. *Interface maps* are used to create links between physical devices (device profiles) and logical devices. Check existing interface maps, and if they don't meet your needs, you can create them.

```
Design > Interface maps
```

3. *Rack types* are logical representations of racks. If a rack type that you need for your design is not included with the predefined ones, you can create one.

```
Design > Rack Types
```

4. *Templates* are used to build rack designs. Check the predefined templates and if one doesn't meet the needs of your design, you can create one.

```
Design > Templates
```

1.2.3 Resources

Create pools for resources (*ASNs*, *IPv4 addresses*, *IPv6 addresses*) that will be used in the network.

```
Resources
```

1.2.4 External Systems

Create *external routers* to represent switches that the network uses for traffic exiting the fabric.

```
External Systems > External Routers
```

1.2.5 Blueprints

1. Create a *blueprint* based on a predefined template or one that you created in the design section.

```
Blueprints
```

2. *Build* the network by assigning resources, device profiles, device system IDs, and external routers.

```
Blueprints > <your_blueprint_name> > Staged > Physical > Build
```

3. Review the calculated *cabling map* and 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. You can also use *AOS CLI* to discover existing cabling and override the blueprint with that cabling.

```
Blueprints > <your_blueprint_name> > Staged > Physical > Links
```

4. When all assignments have been made and the blueprint is error-free, *commit* the blueprint. Committing a blueprint initiates work on the intent and realizes it on the network by pushing configuration changes on assigned devices.

```
Blueprints > <your_blueprint_name> > Uncommitted
```

5. Review the blueprint dashboard for anomalies. If you have cabling anomalies, the likely reason is a mismatch in calculated cabling and actual cabling. Either re-cable the switches, recreate the blueprint with appropriate interface maps or use AOS CLI to override the cabling in the blueprint with discovered cabling.

```
Blueprints > Dashboard
```

1.2.6 Next Steps

After your deployment is running, you can proceed to *build the virtual environment* with virtual networks and security zones, as needed. You can also refer to the *guides* to learn about *intent-based analytics* and other capabilities.

APSTRA SERVER

2.1 Apstra Server Requirements/References

2.1.1 Hypervisors

The Apstra server can be deployed on the following hypervisors:

VMware ESXi Supported versions - 6.7, 6.5, 6.0, 5.5

QEMU / KVM for Ubuntu Supported versions - 18.04 LTS

Microsoft Hyper-V Supported version - Windows Server 2016 Datacenter Edition

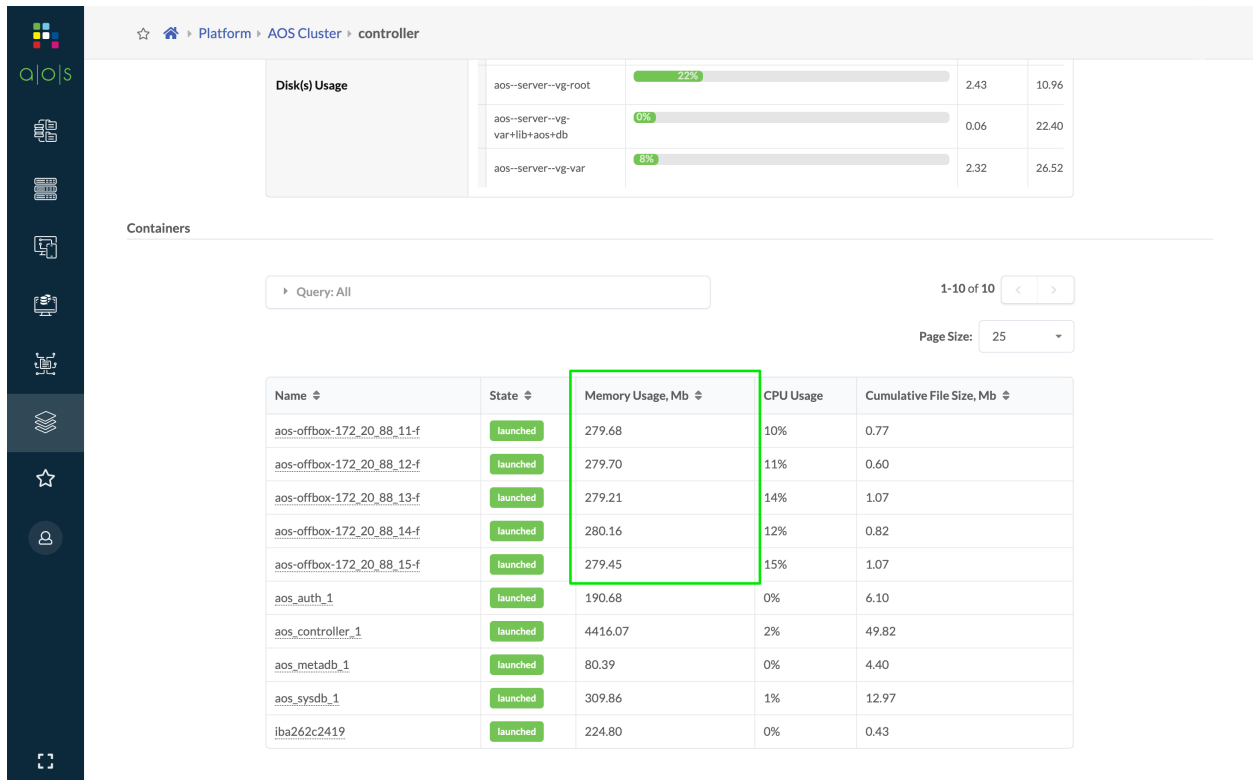
Oracle VirtualBox / VMware Workstation For lab / evaluation purposes only

2.1.2 Apstra Server VM Resources

The required VM resources for the Apstra server may be greater than the recommendations below. Requirements are based on the size of the network (blueprint), the scaling of off-box agents and the use of Intent Based Analytics (IBA). If one VM is insufficient for your needs, you can increase resources by *clustering several VMs*.

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

Note: * Off-box agent memory usage is dependent on the number of IBA collectors enabled. Apstra recommends using the web interface *AOS Cluster* feature to monitor off-box container memory usage (e.g. aos-offbox-172_20_88_11-f). Additional AOS cluster worker nodes can be added to scale off-box agent capacity.



Important: Although, an Apstra server VM might run with fewer resources than specified above, depending on the size of the network, CPU and RAM allocations may be insufficient. In this case, the system encounters errors or a critical “segmentation fault” (core dump). If this happens, delete the VM and redeploy it with additional resources.

2.1.3 Network Security Protocols

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

Table 1: Apstra Server Network Protocol Requirements

| Source | Destination | Protocol | Description |
|----------------------------------|-----------------|--|---|
| User workstation | Apstra Server | tcp/22 (ssh) | CLI access to the server |
| User workstation | Apstra Server | tcp/80 (http) | Redirects to tcp/443 (https) |
| User workstation | Apstra Server | tcp/443 (https) | Web UI and REST API |
| Network Device for device agents | Apstra Server | tcp/80 (http) | Redirects to tcp/443 (https) |
| Network Device or Off-box Agent | Apstra Server | tcp/443 (https) | Device agent installation and upgrade, Rest API |
| Network Device or Off-box Agent | Apstra Server | tcp/29730-29739 | Agent binary protocol (Sysdb) |
| ZTP Server | Apstra Server | tcp/443 (https) | Rest API for Device System Agent Install |
| Apstra Server | Network Devices | tcp/22 (ssh) | Device agent installation and upgrade |
| Off-box Agent | Network Devices | tcp/443 (https) tcp/9443 (nxapi) tcp/830 (for Junos) | Management from Off-box Agent |

2.1.4 Other Network Protocols

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

Table 2: Other Network Protocols

| Source | Destination | Protocol | Description |
|----------------|----------------|---|---|
| Administrator | Network Device | tcp/22 (ssh) | Device management from Administrator |
| Network Device | DNS Server | udp/53 (dns) | DNS Discovery for Apstra server IP (if applicable) |
| Network Device | DHCP Server | udp/67-68 (dhcp) | DHCP for automatic management IP (if applicable) |
| | | (icmp type 0, type 8 for echo and response) | As necessary for network troubleshooting. Not required for the Apstra server. |

2.1.5 Network Client Services

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

Table 3: Apstra Server Network Client Services

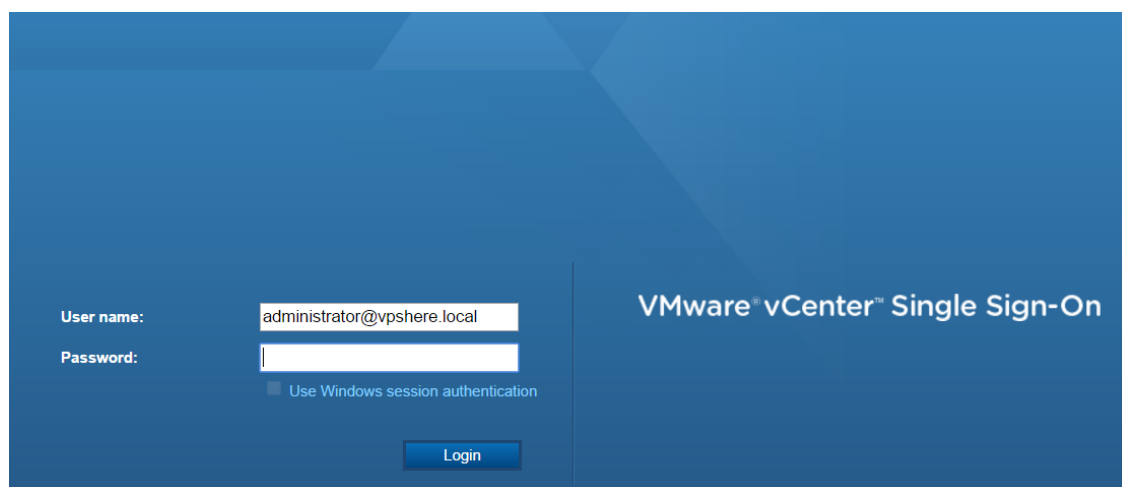
| Source | Destination | Protocol | Description |
|---------------|----------------|--------------------------------|--|
| Apstra Server | DNS Server | udp/53 (dns) | Server DNS Client |
| Apstra Server | LDAP Server | tcp/389 (ldap) tcp/636 (ldaps) | Apstra Server LDAP Client (if configured) |
| Apstra Server | TACACS+ Server | tcp/udp/49 (tacacs) | Apstra Server TACACS+ Client (if configured) |
| Apstra Server | RADIUS Server | tcp/udp/1812 (radius) | Apstra Server RADIUS Client (if configured) |
| Apstra Server | Syslog Server | udp/514 (syslog) | Apstra Server Syslog Client (if configured) |

2.2 Installing Apstra Server

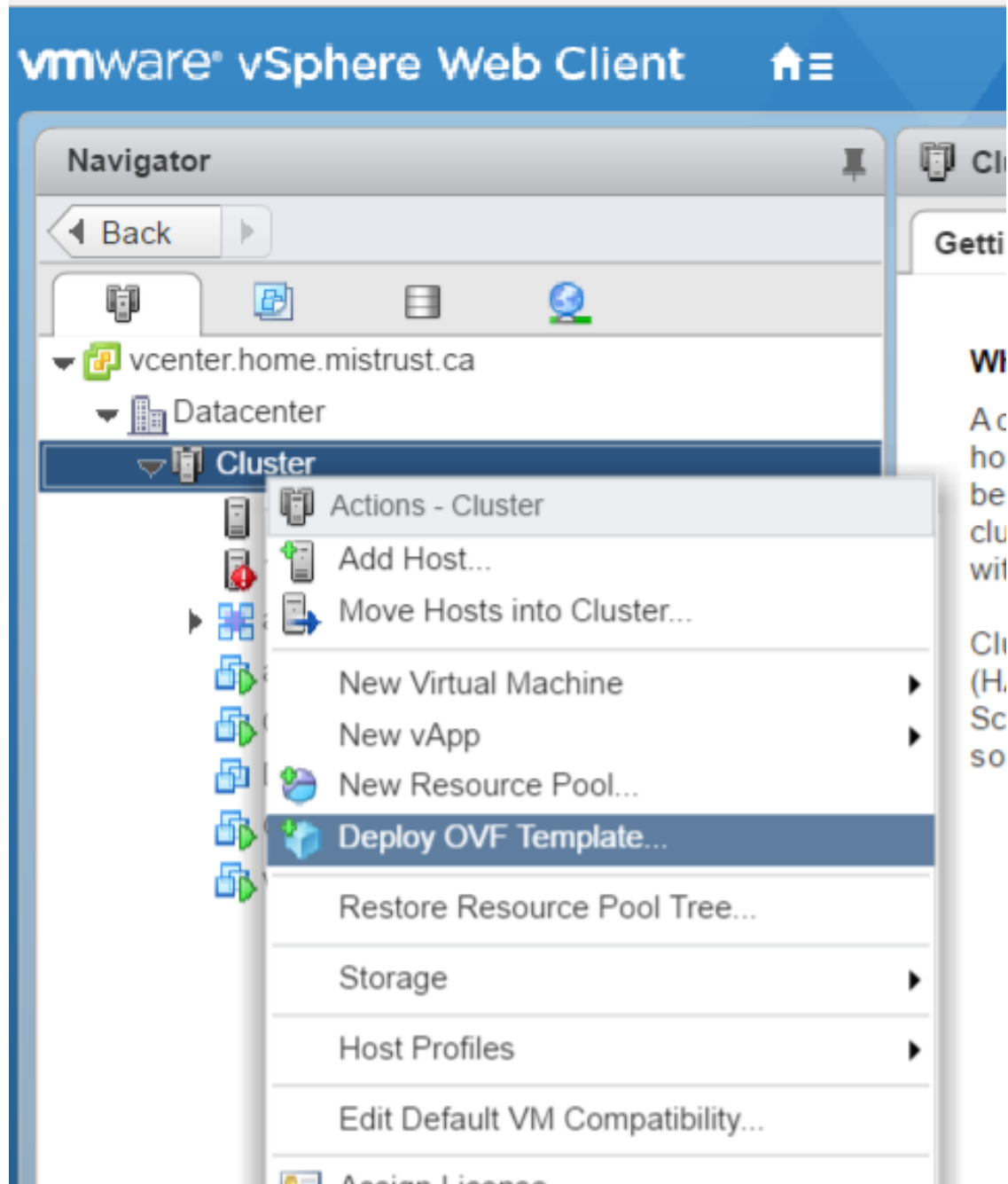
1. Refer to *Apstra Server Requirements and Other References* to confirm that your system meets resource requirements.
2. The Apstra server and agent software is delivered pre-installed on a single virtual machine (VM) in OVA and QCOW2 formats. As a registered support user, download the software from <https://support.juniper.net/support/downloads/?p=afc>.
3. Install the Apstra server VM on one of the supported hypervisors. See the links below for example installations. For specific hypervisor use instructions, please seek assistance from the hypervisor provider (e.g. VMware, Ubuntu, RedHat, Microsoft) or the open-source community.

2.2.1 Install Apstra Server on VMware ESXi

- (a) Log into vCenter.



- (b) Right-click your target deployment environment, and click **Deploy OVF Template**.



(c) Click **Choose Files** and locate the OVF file that you downloaded.

Deploy OVF Template

1 Select an OVF template

- 2 Select a name and folder
- 3 Select a compute resource
- 4 Review details
- 5 Select storage
- 6 Ready to complete

Select an OVF template

Select an OVF template from remote URL or local file system

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

☐ URL

<http> | <https://remoteserver-address/filetodeploy.ovf> | .ova

☒ Local file

aos_server_3.2.2-12.ova

(d) Specify **Unique Name and Target Location** for the VM.

Deploy OVF Template

✓ 1 Select an OVF template

2 Select a name and folder

- 3 Select a compute resource
- 4 Review details
- 5 Select storage
- 6 Ready to complete

Select a name and folder

Specify a unique name and target location

Virtual machine name:

Select a location for the virtual machine.

(e) We recommend Thick Provisioning for the Apstra server.

Deploy OVF Template

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

Select storage

Select the storage for the configuration and disk files


☐ Encrypt this virtual machine

Select virtual disk format:

Thick Provision Lazy Zeroed ▾

VM Storage Policy:

Datastore Default ▾

| Name | Capacity | Provisioned | Free | Type |
|--|----------|-------------|-----------|------|
|  datastore1 | 469.5 GB | 72.83 GB | 426.95 GB | VM |

Compatibility

✓ Compatibility checks succeeded.

CANCEL

BACK

NEXT

- (f) Map the AOS Management network to enable it to reach the virtual networks that the Apstra server will manage on ESXi.

Deploy OVF Template

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

Select networks

Select a destination network for each source network.

| Source Network | Destination Network |
|----------------|---------------------|
| VM Network | VM Network |
| 1 items | |

IP Allocation Settings

IP allocation: Static - Manual

IP protocol: IPv4

[CANCEL](#)[BACK](#)[NEXT](#)

- (g) Verify that all details are correct, then click **Finish**. When the VM is up and running you are ready to *configure the Apstra server*.

Deploy OVF Template

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

Ready to complete

Click Finish to start creation.

| | |
|------------------------|--|
| Provisioning type | Deploy from template |
| Name | aos_server_3.2.2-12 |
| Template name | aos_server_3.2.2-12 |
| Download size | 1.8 GB |
| Size on disk | 64.0 GB |
| Folder | dc1 - bs7 |
| Resource | dc1 - compute cluster |
| Storage mapping | 1 |
| All disks | Datastore: datastore1; Format: Thick provision lazy zeroed |
| Network mapping | 1 |
| VM Network | VM Network |
| IP allocation settings | |
| IP protocol | IPv4 |
| IP allocation | Static - Manual |

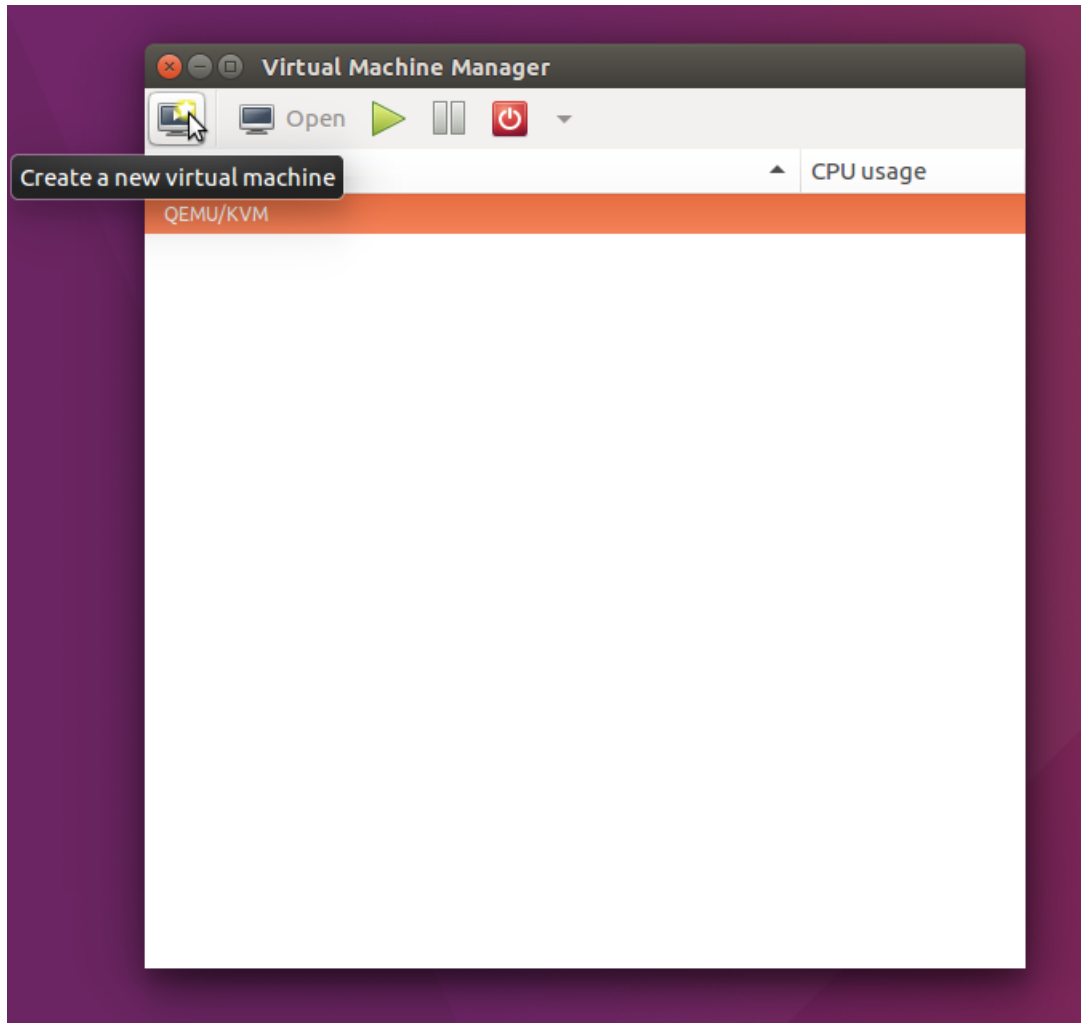
[CANCEL](#)
[BACK](#)
[FINISH](#)

2.2.2 Install Apstra Server on KVM

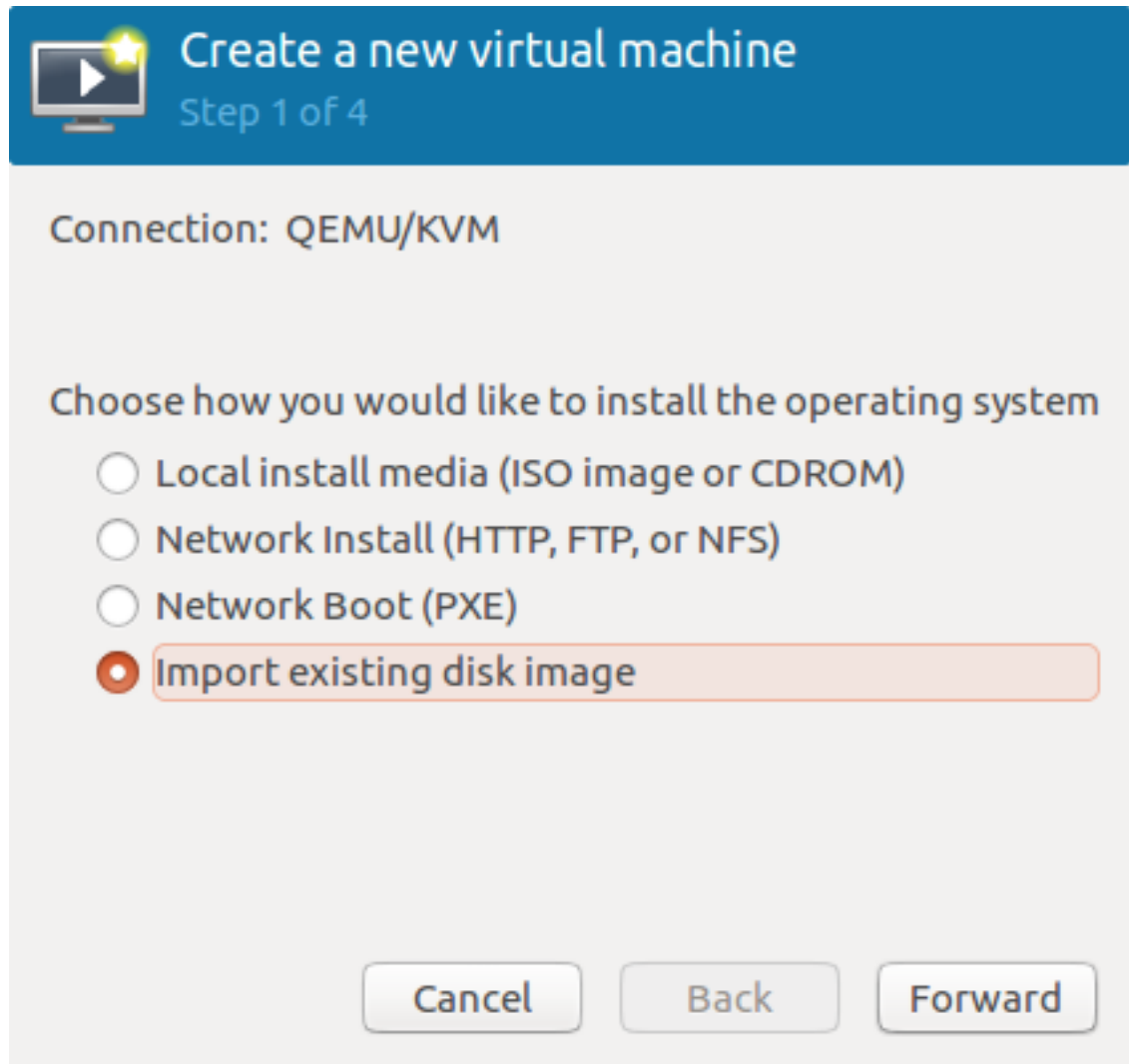
Make sure that KVM is installed on your Linux distribution before continuing.


2.2.2.1 Method One: via Virtual Machine Manager

- (a) Click the **Create a new virtual machine** button.



(b) Import the existing disk image.



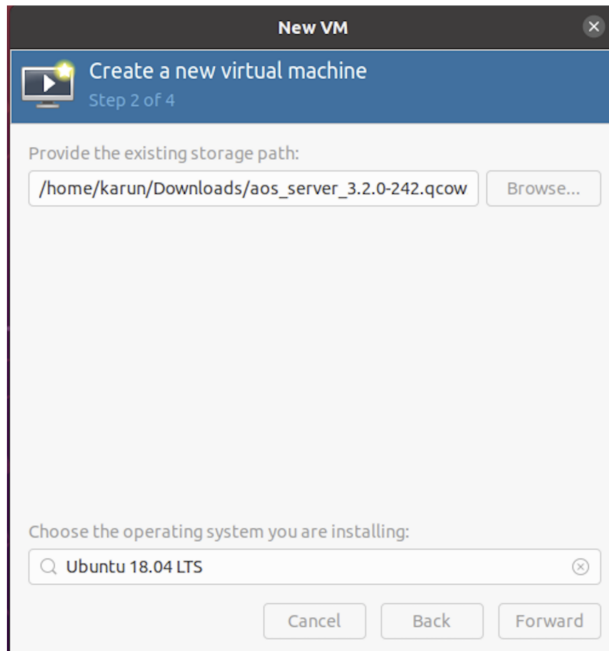
 **Create a new virtual machine**
Step 1 of 4

Connection: QEMU/KVM

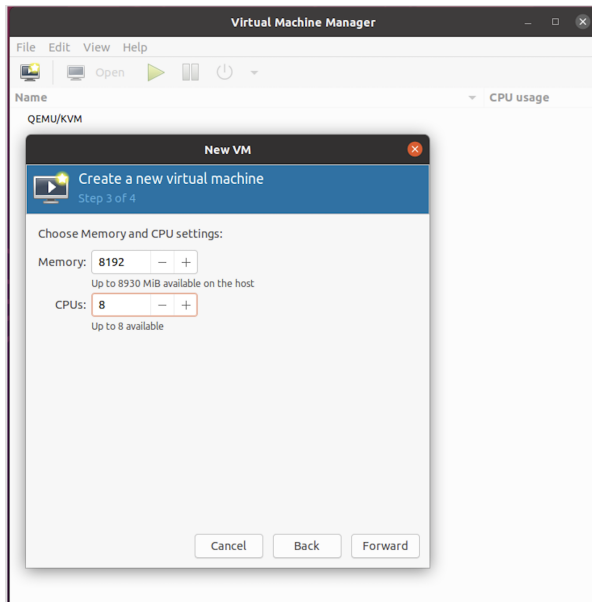
Choose how you would like to install the operating system

- ☐ Local install media (ISO image or CDROM)
- ☐ Network Install (HTTP, FTP, or NFS)
- ☐ Network Boot (PXE)
- ☒ Import existing disk image

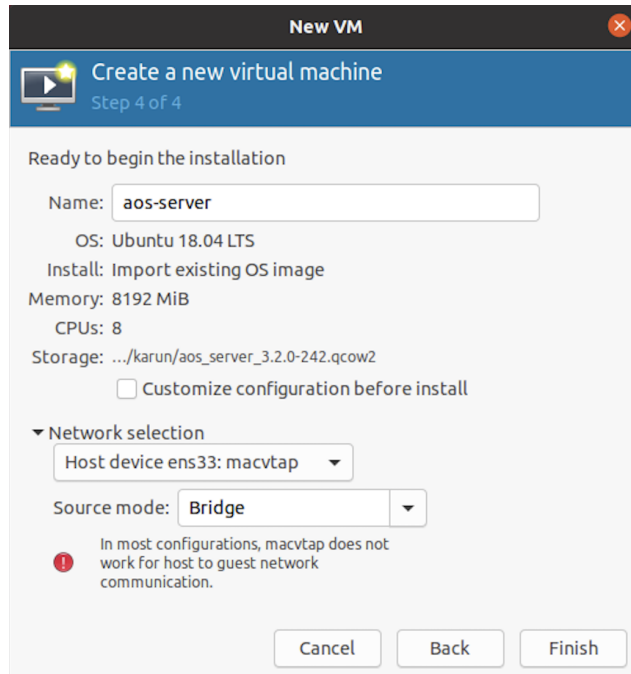
(c) Specify the path for the qcow2 image and provide the following OS parameters.



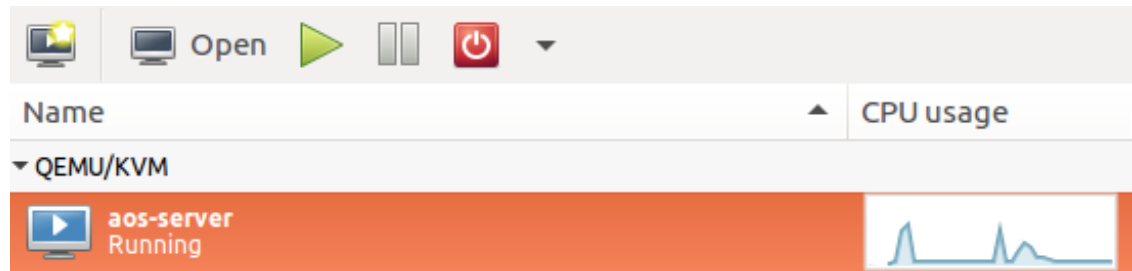
(d) Enter the required CPU and RAM values.



(e) Select a network and provide respective bridge settings.



- (f) When the VM is up and running you are ready to *configure the Apstra server*. Open a console to it with one of the KVM frontends (such as virt-manager).



2.2.2.2 Method Two: Via Command Line

Warning: Users must use the `e1000` or `virtio` Linux KVM network drivers. Using other drivers such as `rtl8139` may result in high CPU utilization for the `ksoftirqd` process.

You can verify the network driver with the `ethtool -i eth0` command in the Apstra server VM.

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

- (a) Linux KVM requires that the QEMU environment and bridge networking be installed and configured. Please refer to the following document as examples for QEMU install and configuration.

For RHEL

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/virtualization_deployment_and_administration_guide/index

For Ubuntu

<https://help.ubuntu.com/community/KVM/Installation>

- (b) Uncompress the qcow2.gz image with **gunzip**.

```
ubuntu@ubuntu:~$ ls -l
total 1873748
-rw-r--r-- 1 ubuntu ubuntu 1918712115 Feb  4 22:28 aos_server_3.3.0.2-46.
↪qcow2.gz
ubuntu@ubuntu:~$ gunzip aos_server_3.3.0.2-46.qcow2.gz
ubuntu@ubuntu:~$ ls -l
total 1905684
-rw-r--r-- 1 ubuntu ubuntu 1951413760 Feb  4 22:28 aos_server_3.3.0.2-46.qcow2
ubuntu@ubuntu:~$
```

- (c) Create the VM with the **virt-install** command line tool. For example, installing the **aos_server_3.3.0.2-46.qcow2** image using existing bridge network (named **br0**).

```
ubuntu@ubuntu:~$ sudo virt-install --name=aos-server --disk=aos_server_3.3.0.
↪2-46.qcow2 --os-type=linux --os-variant ubuntu18.04 --import --
↪noautoconsole --vcpu=8 --ram=65535 --network bridge=br0,model=virtio

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

ubuntu@ubuntu:~$
```

- (d) Connect to the VM console. For example.

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

Apstra Operating System (AOS)

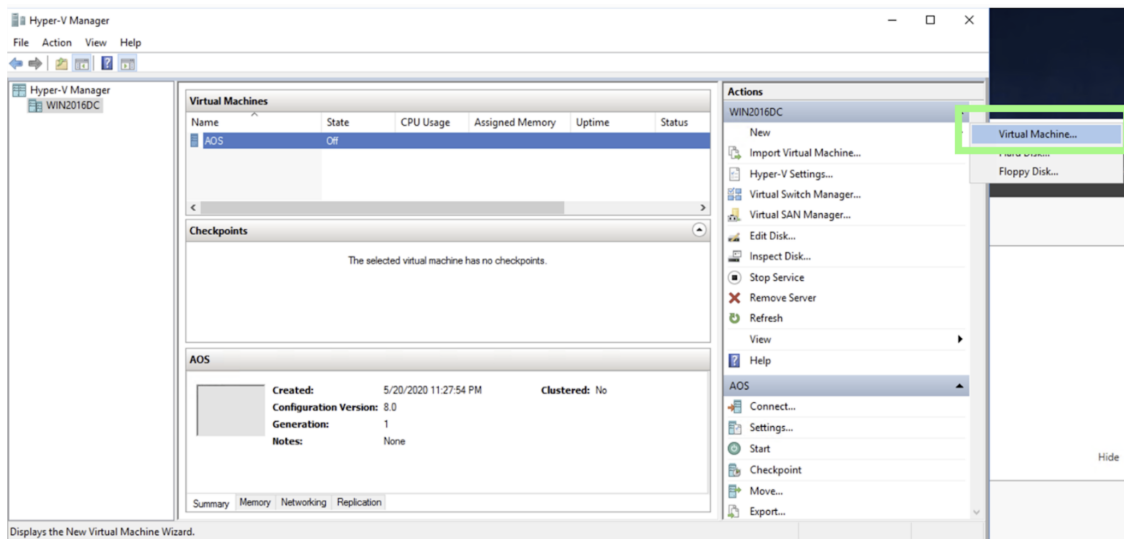
aos-server login:
```

- (e) When the VM is up and running you are ready to *configure the Apstra server*.

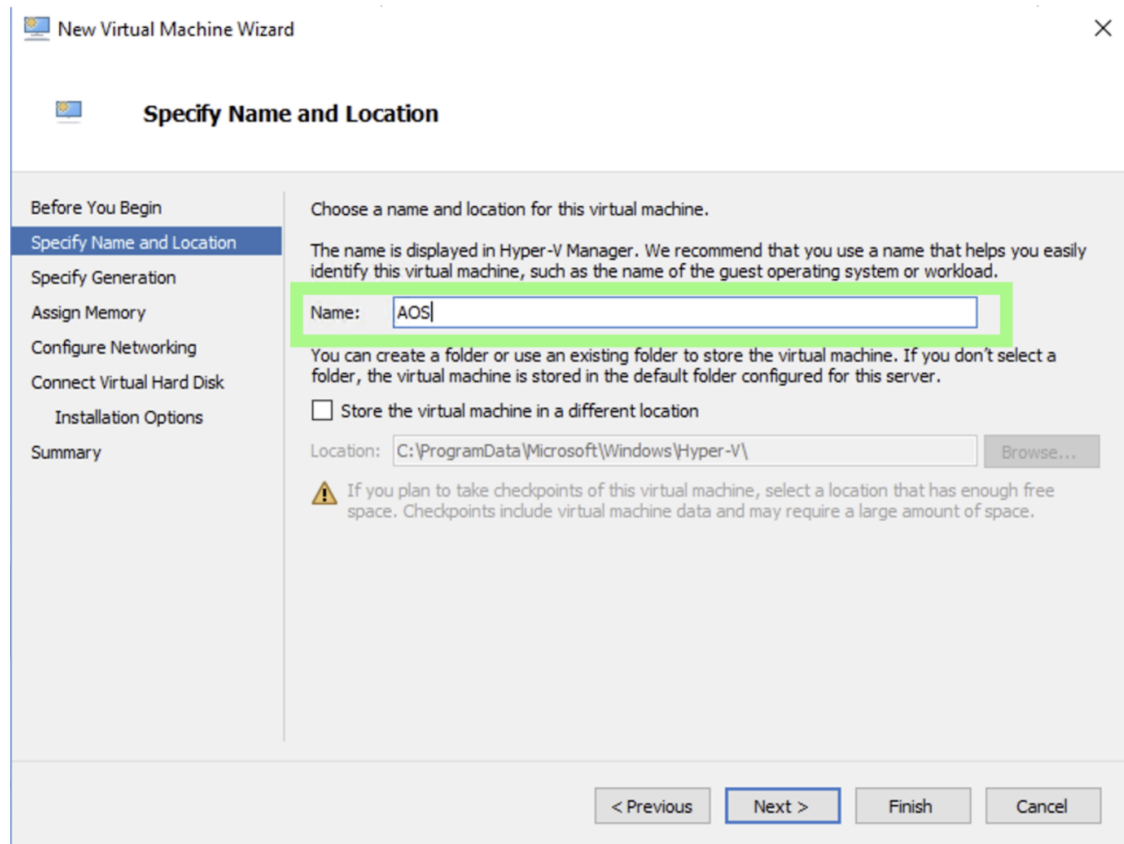
2.2.3 Install Apstra Server on Hyper-V

Kindly ensure Microsoft Hyper-V Manager is available in the Hyper-V systems based environment. These instructions are based on using Hyper-V Manager on a Windows Server 2016 Datacenter Edition:

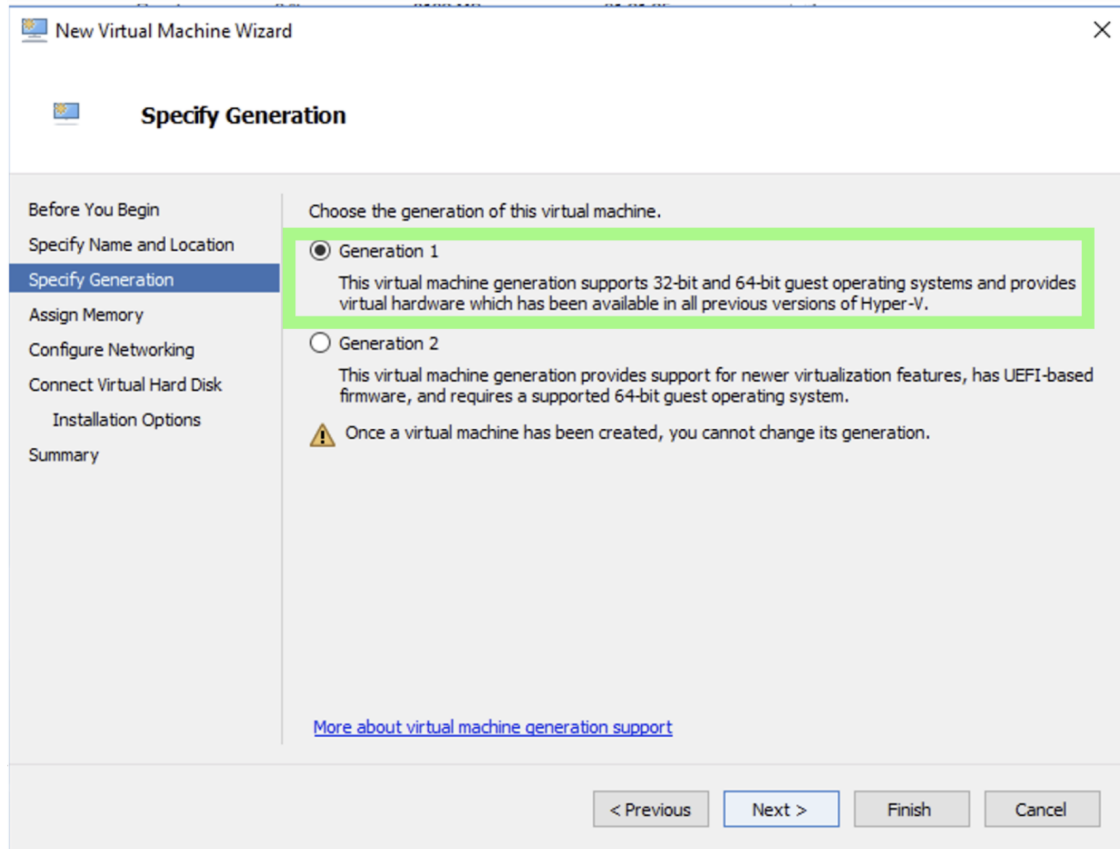
- (a) In **Hyper-V Manager**, in the **Actions** panel, click **New**, then click **Virtual Machine**.



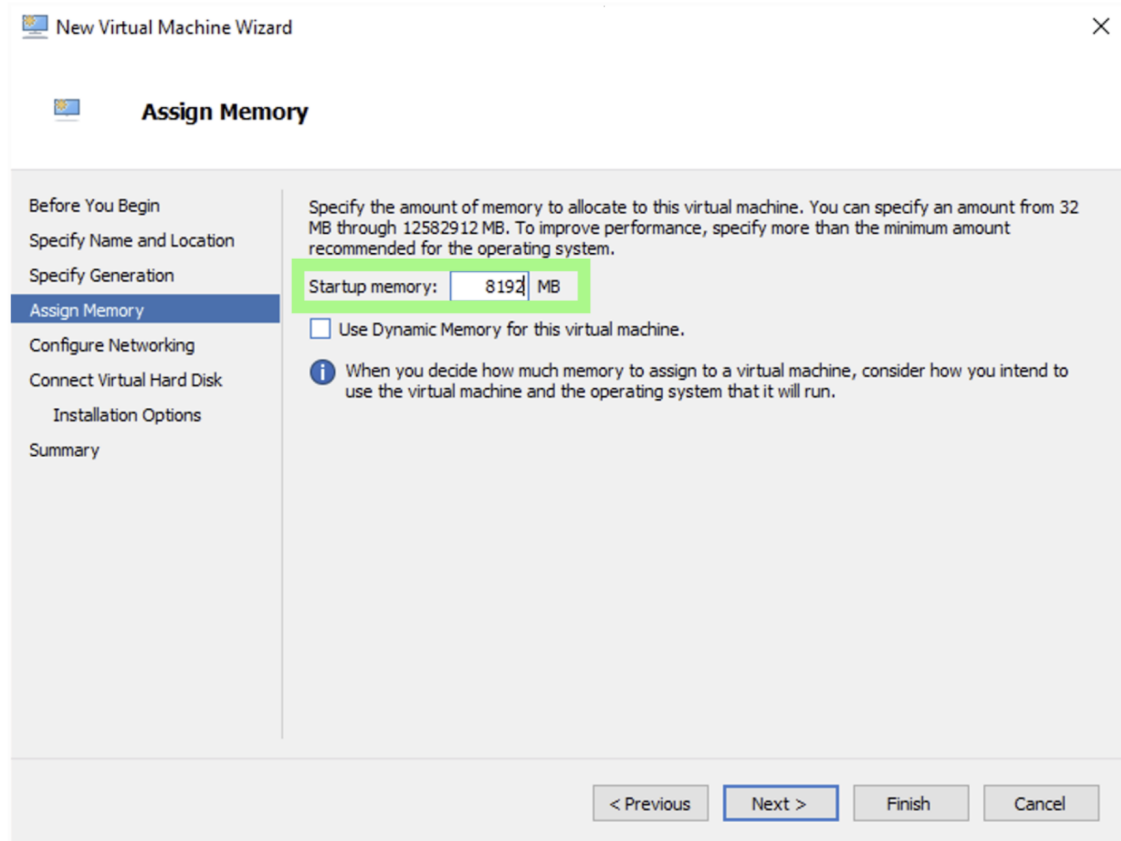
- (b) Specify the VM name.



- (c) Specify the generation type as **Generation1**.



- (d) Provide the required vCPU and RAM values. The example below allocates 2vCPU and 8 GB RAM. However, in production the VM resources should be allocated based on *Apstra Server Resource Requirements*.



The screenshot shows the 'New Virtual Machine Wizard' window, specifically the 'Assign Memory' step. The window has a title bar with a close button (X) in the top right corner. Below the title bar, there is a sub-header 'Assign Memory' with a small icon to its left. On the left side, there is a vertical list of steps: 'Before You Begin', 'Specify Name and Location', 'Specify Generation', 'Assign Memory' (which is highlighted with a blue background), 'Configure Networking', 'Connect Virtual Hard Disk', 'Installation Options', and 'Summary'. The main area of the wizard contains instructions: 'Specify the amount of memory to allocate to this virtual machine. You can specify an amount from 32 MB through 12582912 MB. To improve performance, specify more than the minimum amount recommended for the operating system.' Below this text, there is a text input field labeled 'Startup memory:' with the value '8192' entered, followed by a 'MB' unit label. To the right of the input field is a checkbox labeled 'Use Dynamic Memory for this virtual machine.' which is currently unchecked. Below the checkbox is an information icon (i) followed by a note: 'When you decide how much memory to assign to a virtual machine, consider how you intend to use the virtual machine and the operating system that it will run.' At the bottom of the window, there are four buttons: '< Previous', 'Next >' (which is highlighted with a blue border), 'Finish', and 'Cancel'.

New Virtual Machine Wizard

Assign Memory

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

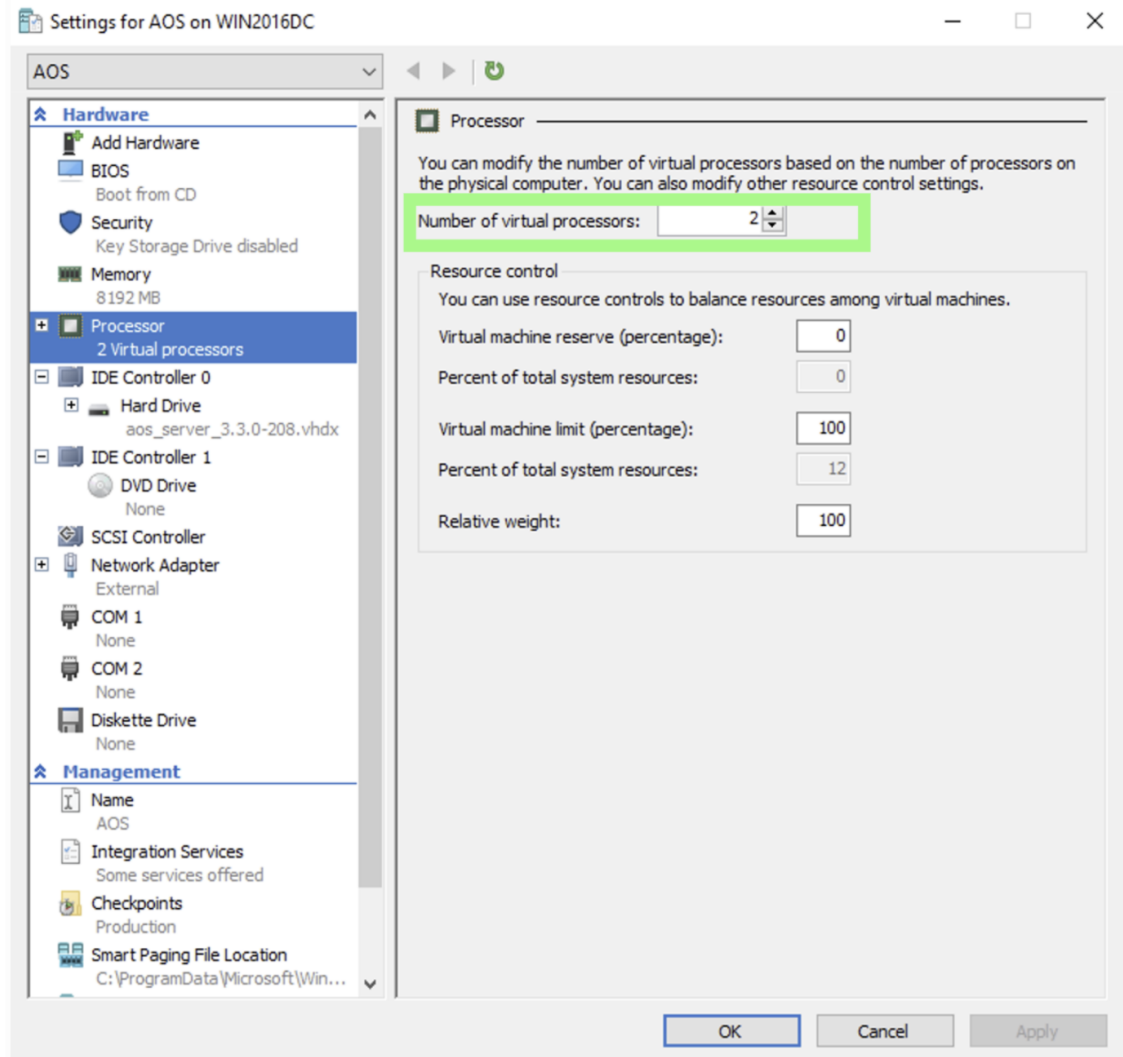
Specify the amount of memory to allocate to this virtual machine. You can specify an amount from 32 MB through 12582912 MB. To improve performance, specify more than the minimum amount recommended for the operating system.

Startup memory: 8192 MB

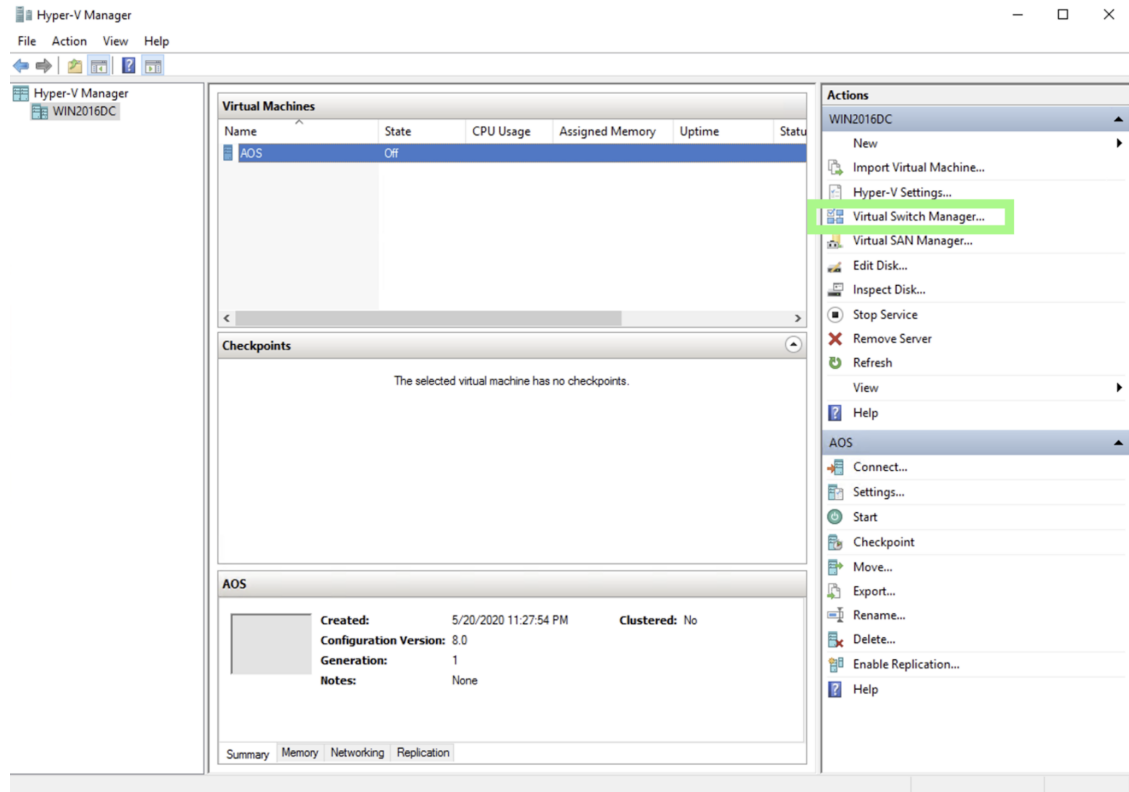
☐ Use Dynamic Memory for this virtual machine.

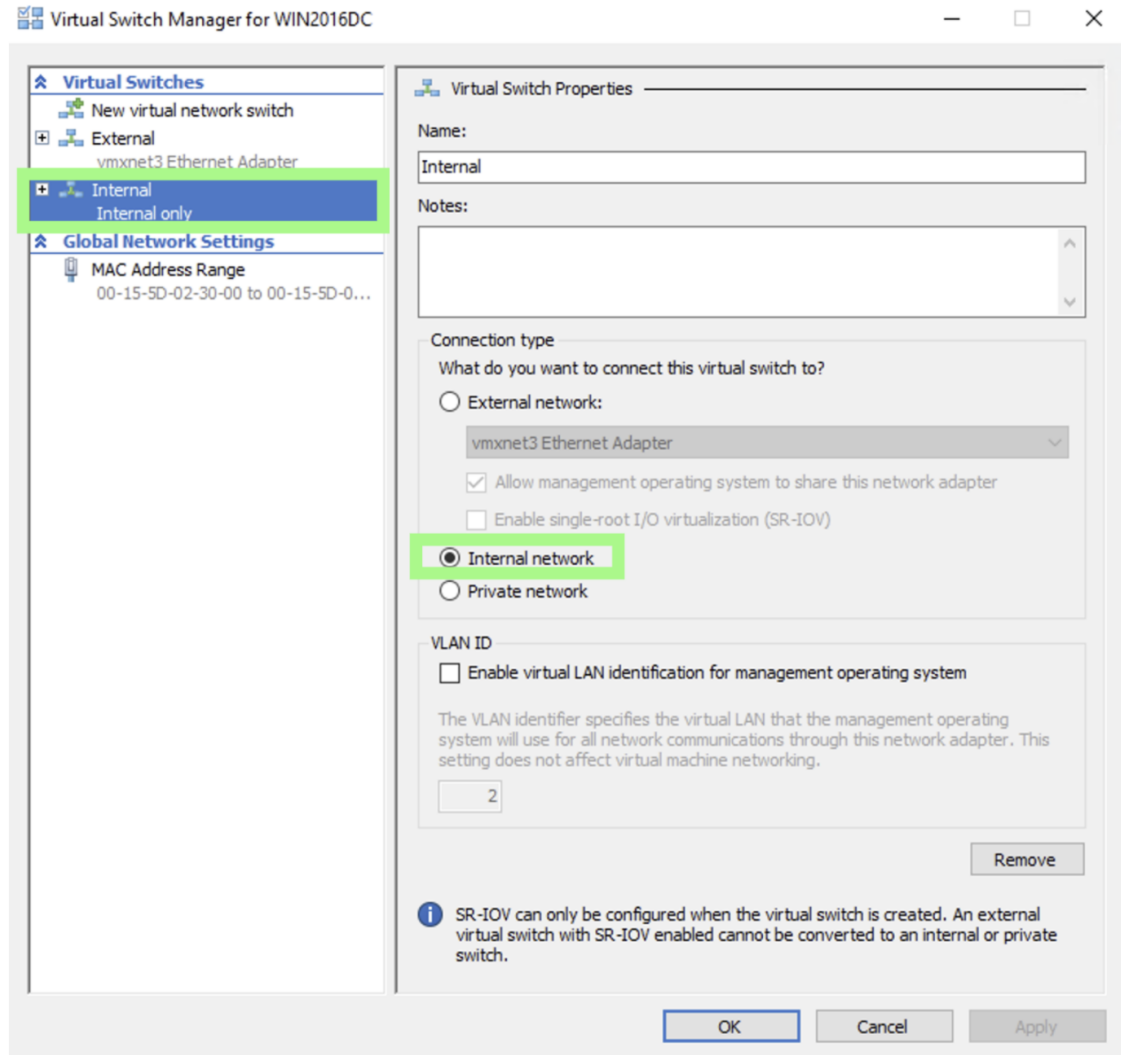
i When you decide how much memory to assign to a virtual machine, consider how you intend to use the virtual machine and the operating system that it will run.

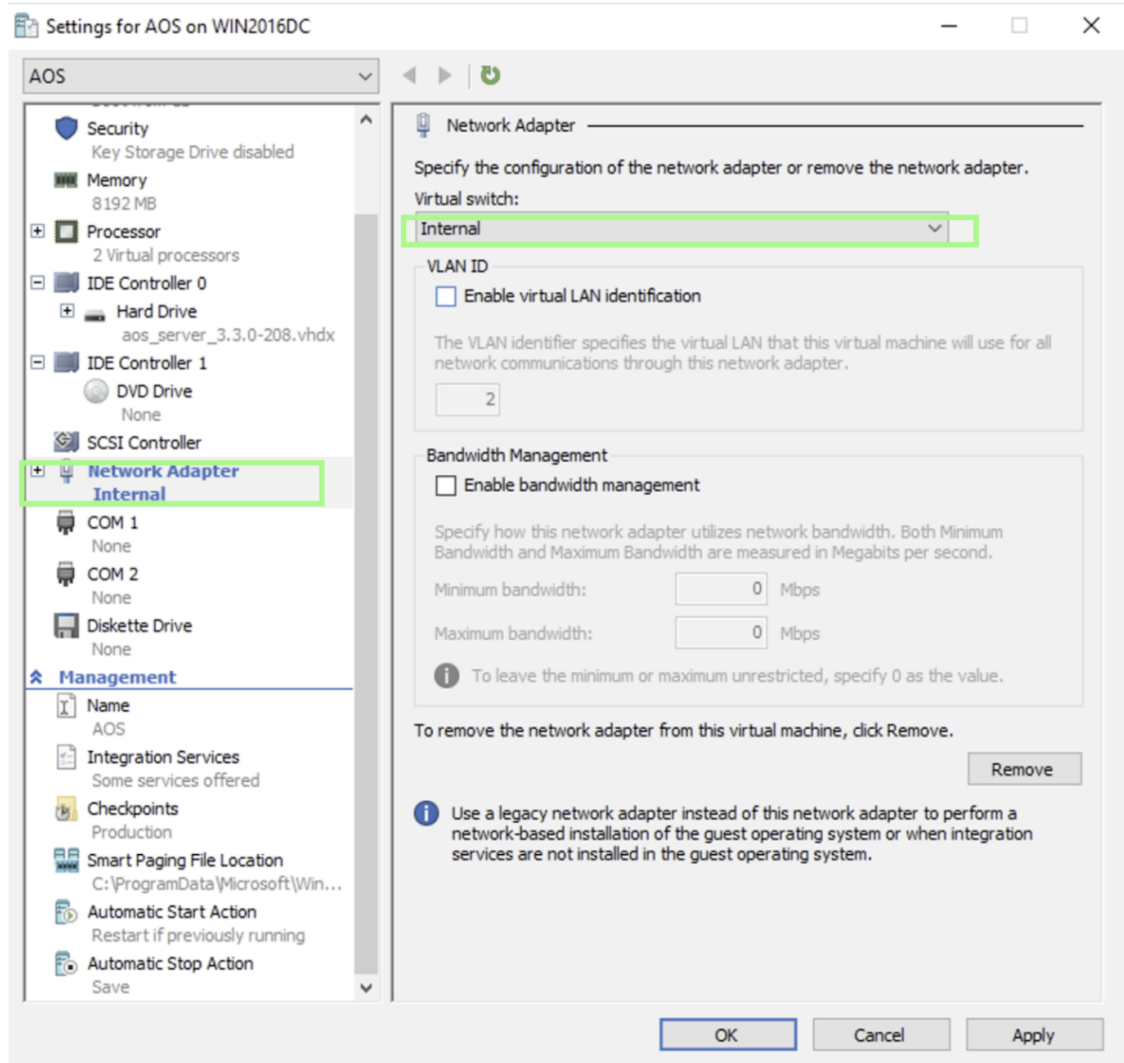
< Previous Next > Finish Cancel



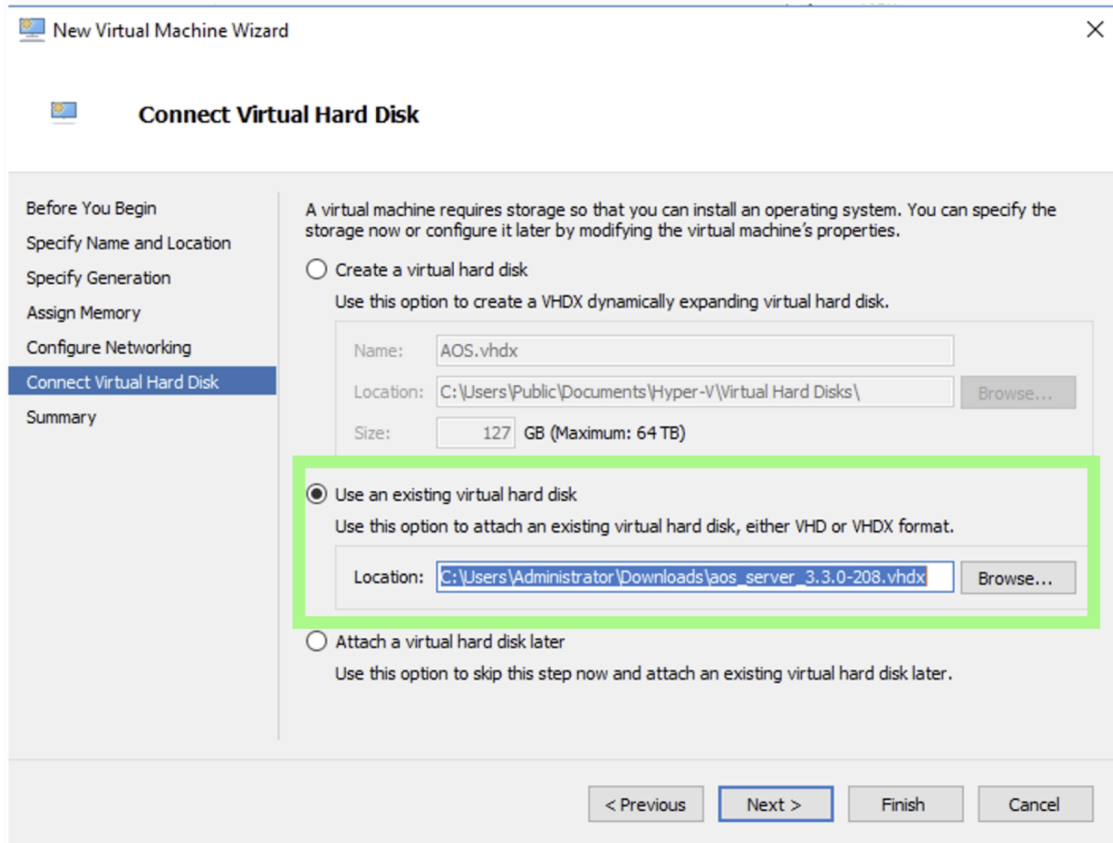
- (e) To set up a network environment, select **Virtual Switch Manager** and create the virtual switch. Under VM settings select the appropriate virtual switch under Hyper-V network adapter. This should help the Apstra server connect on the management network to the devices managed by it. In the example below, the management network is residing on Internal Network **Virtual Switch**. However, in a production setup, select the appropriate virtual switch as per your deployment environment.



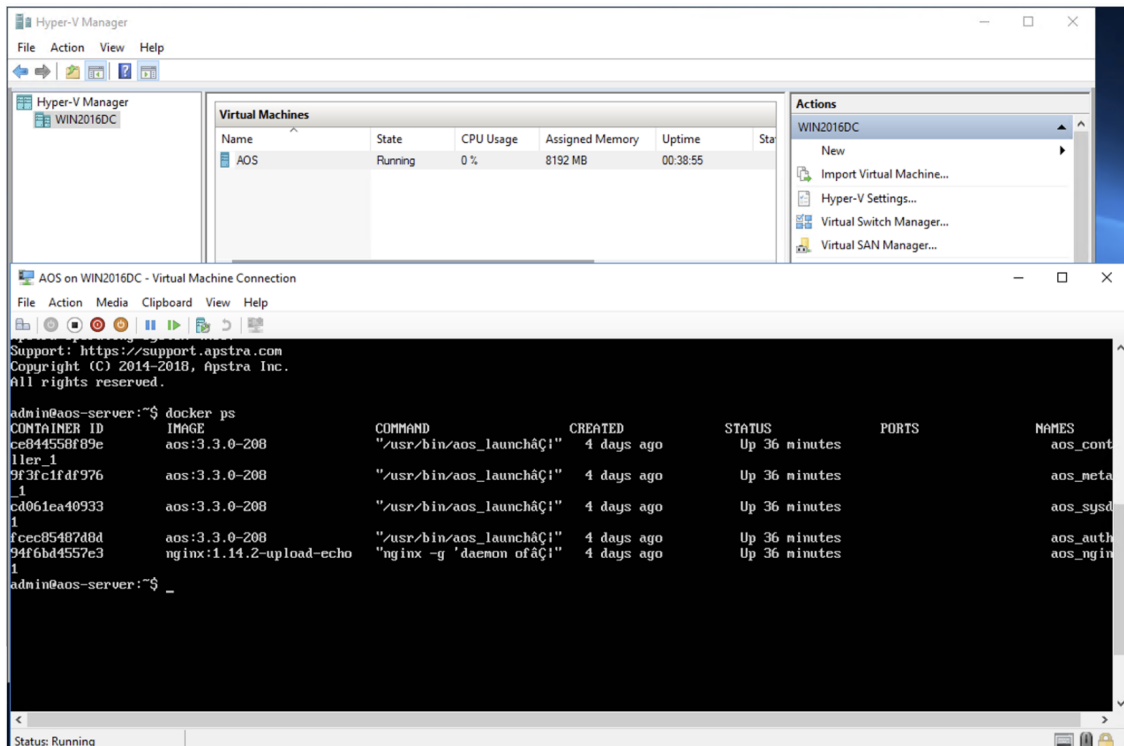




(f) Locate and select the disk image in the **VHD** or **VHDX** format on your computer:



- (g) When the VM is up and running you are ready to *configure the Apstra server*. Open a console to it using Hyper-V Manager. After the Apstra server is configured, Docker daemon will run properly.

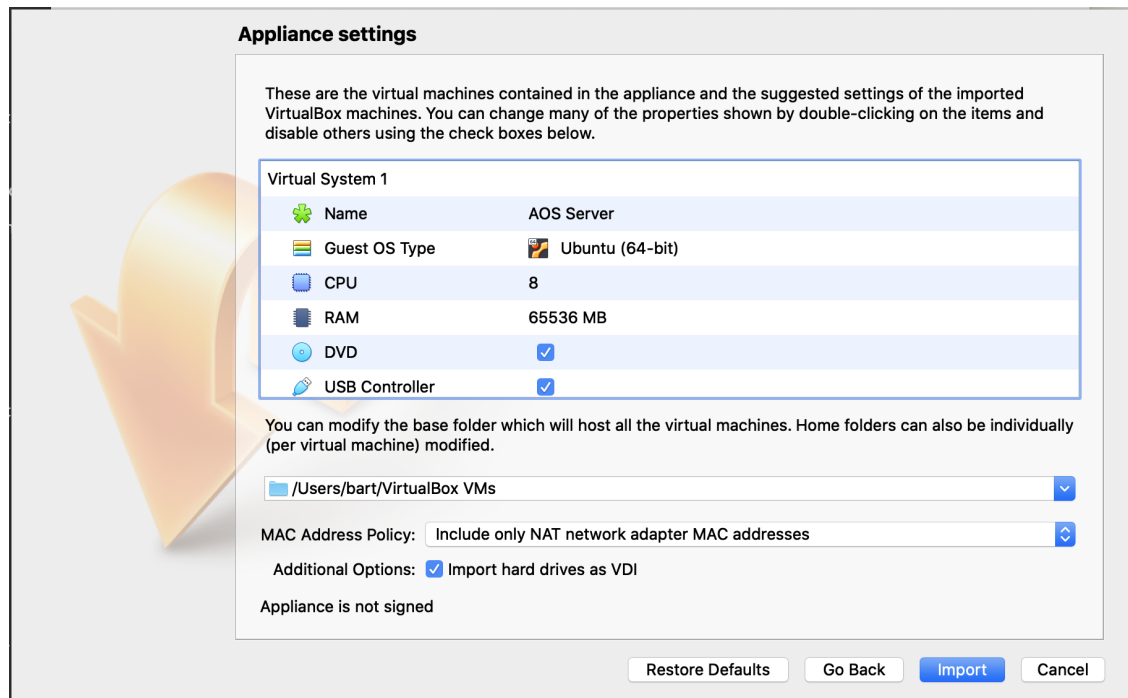


2.2.4 Install Apstra Server on VirtualBox

Note: VirtualBox is meant for demonstration and lab purposes only. For production usage, please use a proper enterprise-scale virtualization solution.

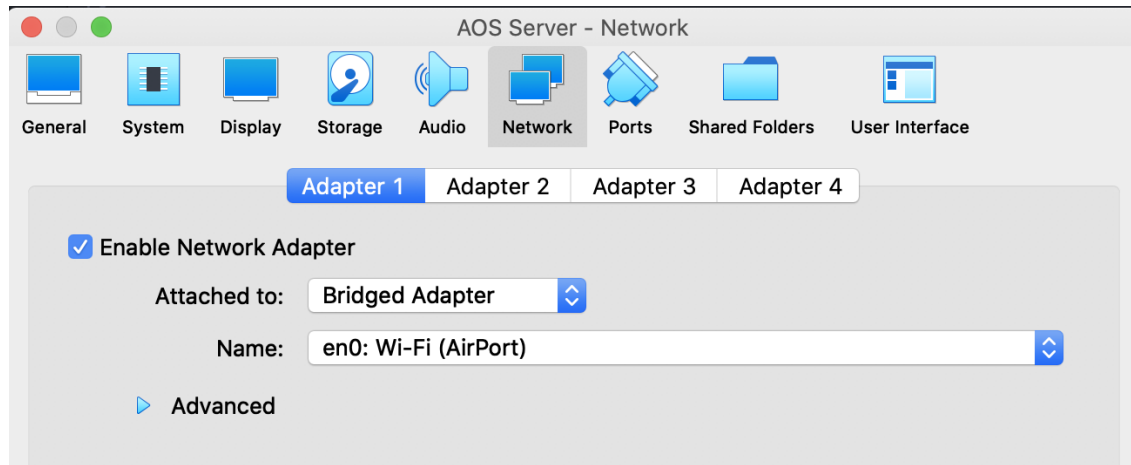
The steps below are for running Apstra software on VirtualBox. For full details of using VirtualBox, please refer to the official VirtualBox documentation at <https://www.virtualbox.org/>.

- (a) Download the OVA to your local workstation from the Support Portal at <https://support.juniper.net/support/downloads/?p=afc>.
- (b) Start VirtualBox, select **File > Import Appliance**, navigate to the OVA file and select it for import.
- (c) Verify the appliance settings. By default VirtualBox uses 2 vCPUs. Change it to 8 vCPUs. For lab / testing purposes, 8GB RAM is sufficient:



- (d) When the import has finished (this will take a while), power up the VM to verify that it is available.
- (e) Ensure your VM settings match requirements. In particular, check the network settings for the adapter that is attached to your management network. If this value is not set correctly the Apstra server will not get an IP address.

Note that by default VirtualBox has one network adapter attached to the Bridged Adapter using the Active network. This means you should have full connectivity from your workstation to the VM (http & ssh) out of the box:



- (f) *Configure the Apstra server.*
 - (g) Finally, verify connectivity using ssh from your workstation to the IP address of the VM's active network adapter, and by pointing a web browser to this same IP address (the IP address can be obtained by running `ifconfig -a` on the VM).
4. Configure the Apstra server.

2.2.5 Configuring Apstra Server

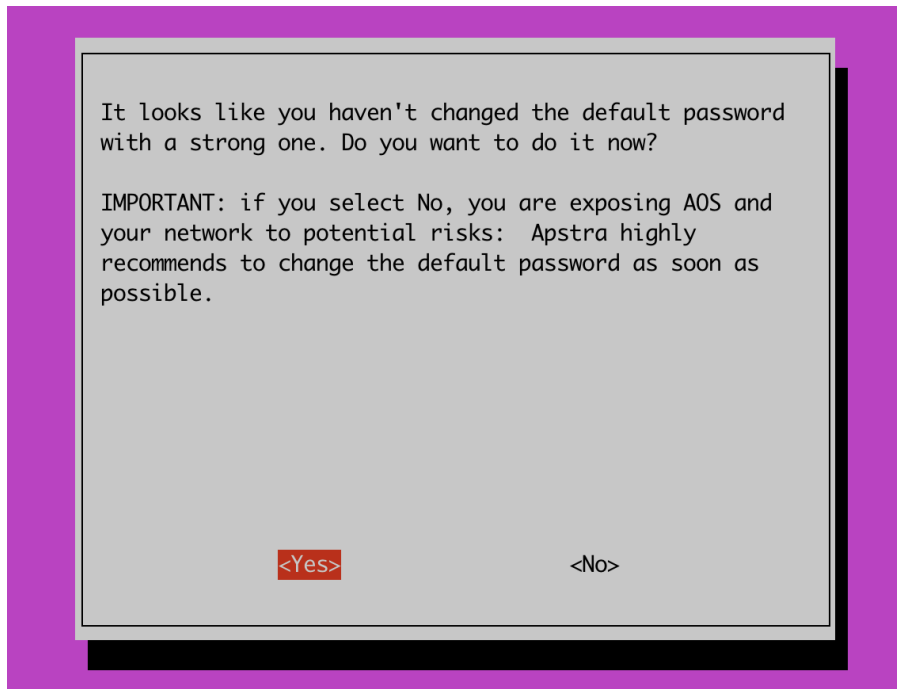
After you've installed the Apstra server VM, boot the VM to access the first boot configuration script.

2.2.5.1 Initial Setup

(Versions 2.3.1 and Later)

- (a) When you connect to the Apstra server via CLI for the first time, a script runs automatically to assist with basic settings. You can access the script at any time with the following command:

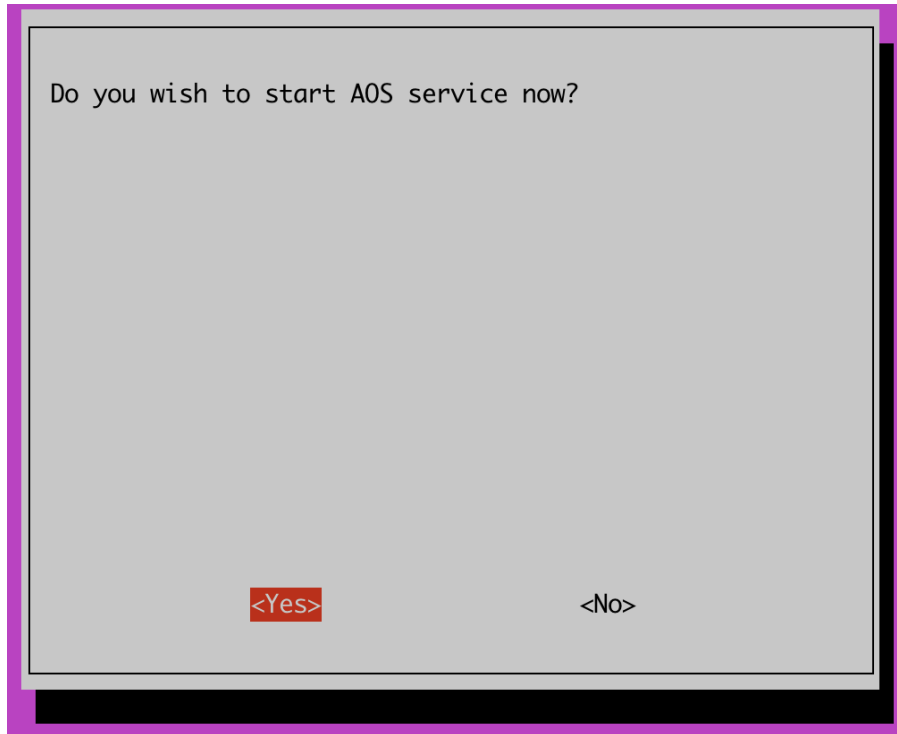
```
admin@aos-server:~$ aos_config
```



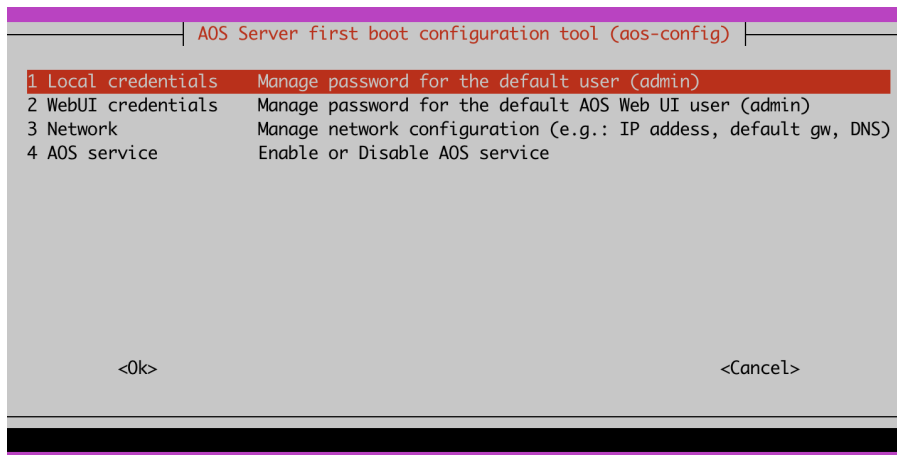
- (b) Choose **<Yes>** to change the default administrator password. For enhanced security, choose a strong password that has a minimum of fourteen characters, one uppercase character, one digit, and one that does not contain the current username in any form.

Warning: User *admin* has full root access. We *highly* recommend that you change the default password. Apstra cannot be held responsible for security-related incidents due to failing to change the default password.

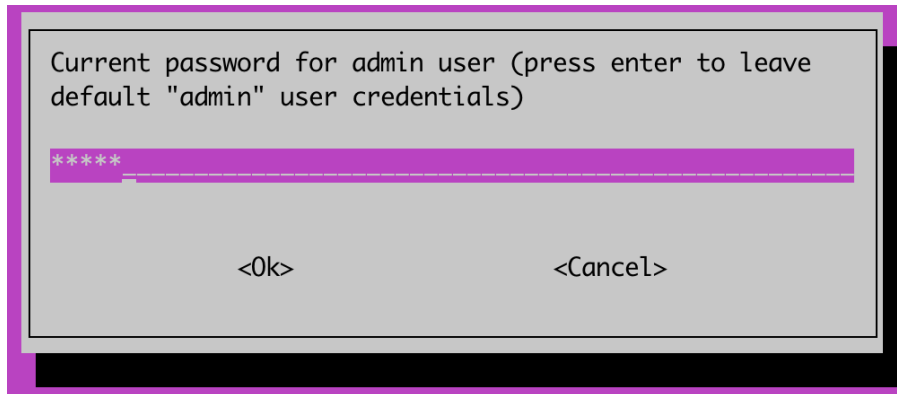
- (c) After changing the **admin** password, choose **<Yes>** when asked if you want to start service.



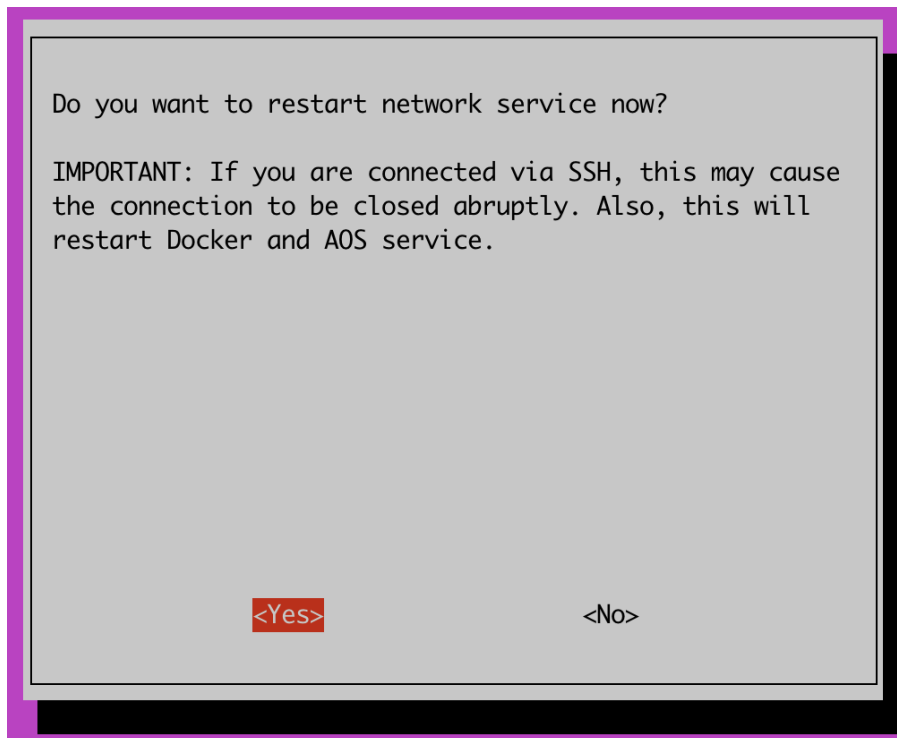
- (d) The main menu appears. Choose **Local credentials** and change the password for the default CLI user **admin**.



- (e) Choose **WebUI credentials**, and change the password for the default Web UI user **admin**. Services must be up and running in order to change this password.



- (f) Choose **Network** to change the machine's network settings. By default **DHCP** is used. If you change the default to **static** you'll have the option of providing a CIDR IP address, gateway, primary / secondary DNS and domain values.
- (g) After you've completed the configuration, choose **<Yes>** to restart network service, Docker and AOS service.



- (h) AOS Service is disabled by default. If you enable services from this configuration tool, it invokes `/etc/init.d/aos`. This is the equivalent of running `service aos start` at the command line.

2.2.5.2 Configuring Static Management IP Address

(Versions 2.3.0 and Later)

Log into the server as user **admin**. To statically configure the management IP address, edit the `/etc/netplan/01-netcfg.yaml` file per standard Ubuntu 14.04 practice.

Listing 1: Configure IP address - versions 2.3.0 and later

```

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

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

```

Apply the IP address change with the `sudo netplan apply` command or reboot the Apstra server with the `sudo reboot` command.

See <https://netplan.io/examples> for more information on using netplan.

The user can also use the `sudo aos_config` CLI command to change the server network configuration via netplan.

2.2.5.3 Updating SSH Host Keys

Warning: In versions 3.2.x and earlier, due to the way the Apstra server VM is packaged, all instances installed from the same OVA/QCOW image have the same SSH host keys.

As a result, this issue allows an attacker to more easily bypass remote host verification when a user connects by SSH to what is believed to be a previously used Apstra server host but is really the attacker's host performing a man-in-the-middle attack.

To update the the SSH Host Keys on the Apstra server, follow this procedure to generate new SSH Host Keys for a new or existing Apstra server VM:

- (a) Remove existing SSH Host Keys

```
admin@aos-server:~$ sudo rm /etc/ssh/ssh_host*
```

- (b) Configure new SSH Host Keys

```

admin@aos-server:~$ sudo dpkg-reconfigure openssh-server
Creating SSH2 RSA key; this may take some time ...
2048 SHA256:EWRFCs4V6BmOILR3T2PsnxgluE0qXQ/z9IKkXrnLpJs root@aos-server (RSA)
Creating SSH2 ECDSA key; this may take some time ...
256 SHA256:THaXEia8VW6Jfw6OBXFegulCav0zcGSVOy9RkNOPxf4 root@aos-server (ECDSA)
Creating SSH2 ED25519 key; this may take some time ...
256 SHA256:0HOn0nnF+7oRaF5HggI4vWeyxT+UNsHcbvNpBJdaKhQ root@aos-server_
↵ (ED25519)

```

- (c) Restart SSH server process

```
admin@aos-server:~$ sudo systemctl restart ssh
```

2.2.5.4 Changing Default Docker Networks

Docker containers used by the Apstra server require two networks for internal connectivity. These are configured automatically with a default IPv4 network address. These networks are:

```
docker0: inet 172.17.0.1/16
docker_gwbridge: inet 172.18.0.1/16
```

If these subnets are required elsewhere in the setup, modify the default values to avoid conflicts.

docker0 network

To modify the default value for the docker0 network, update the *bip* value with the subnet of your choice. If the json file does not already exist, you can create it in the following format:

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

{
    "bip": "172.26.0.1/16"
}

$ sudo service docker restart
```

docker_gwbridge network

To modify the default value for the docker_gwbridge network, replace the subnet and gateway IP with the values of your choice:

```
$ sudo service aos stop
$ docker swarm leave --force
$ sudo service docker stop
$ sudo ip link set docker_gwbridge down
$ sudo ip link del dev docker_gwbridge
$ sudo service docker start
$ docker network rm docker_gwbridge
$ docker network create \
  --subnet 10.11.0.0/16 \
  --gateway 10.11.0.1 \
  --opt com.docker.network.bridge.name=docker_gwbridge \
  --opt com.docker.network.bridge.enable_icc=false \
  --opt com.docker.network.bridge.enable_ip_masquerade=true \
  docker_gwbridge
$ sudo service aos restart
```

2.2.5.5 Changing Apstra Server Hostname

You can change the Apstra server hostname with the `hostnamectl set-hostname` command.

- (a) Run `hostnamectl set-hostname` command

```
admin@aos-server:~$ hostname
aos-server
admin@aos-server:~$ sudo hostnamectl set-hostname xyz-server
admin@aos-server:~$ hostname
xyz-server
```

(b) Edit /etc/hosts file and replace the existing hostname with new one

```
admin@aos-server:~$ cat /etc/hosts
127.0.0.1    localhost
127.0.1.1    aos-server  <= replace this with new hostname e.g. 'xyz-server'
..snip..
admin@aos-server:~$ sudo vi /etc/hosts
```

2.2.5.6 Apstra Server Configuration File

/etc/aos/aos.conf

Controller Section

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

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

Security Section

```
[security]

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

Log Rotate Section

```
[logrotate]

# AOS has builtin log rotate functionality. You can disable it by setting
# <enable_log_rotate> to 0 if you want to use linux logrotate utility to manage
# your log files. AOS agent reopens log file on SIGHUP
enable_log_rotate = 1
# Log file will be rotated when its size exceeds <max_file_size>
```

(continues on next page)

(continued from previous page)

```

max_file_size = 1M
# The most recent <max_kept_backups> rotated log files will be saved. Older
# ones will be removed. Specify 0 to not save rotated log files, i.e. the log
# file will be removed as soon as its size exceeds limit.
max_kept_backups = 5
# Interval, specified as <hh:mm:ss>, at which log files are checked for
# rotation.
check_interval = 1:00:00

```

Auth Sysdb Log Rotator Section

```

[auth_sysdb_log_rotator]

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

```

Main Sysdb Log Rotator Section

Four parameters for configuring the main graph datastore retention policy.

```

[main_sysdb_log_rotator]

# AOS has builtin main sysdb persistence file rotation functionality. Default
# value is 1 which means sysdb retention policy is enabled. You can disable it
# by setting it to 0 and you also can enable it again by setting it to 1. All
# retention policy parameters will be reloaded by restarting AOS service, or
# sending SIGHUP signal to SysdbResourceManager agent via "sudo kill -s 1

```

(continues on next page)

(continued from previous page)

```
# $(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, AOS keeps a minimum of three groups of valid Main Graph Datastore persistence files.

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

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

Credential Sysdb Log Rotator Section

[credential_sysdb_log_rotator]

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

Anomaly Sysdb Log Rotator Section

[anomaly_sysdb_log_rotator]

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

(continues on next page)

(continued from previous page)

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

Device Image Management Section

[device_image_management]

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

Authentication Section

[authentication]

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

Device Config Management Section

[device_config_management]

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

Telemetry Init Section

```
[telemetry_init]

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

Telemetry Global Config Section

```
[telemetry_global_config]

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

Task API Section

```
[task_api]

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

(continues on next page)

(continued from previous page)

```
# in sysdb.
max_uncompressed_data_size = 1000
```

Statistics Section

```
[statistics]

# Enable or disable full validation for pod statistics. Disable if Racks and/or
# Pods tabs load times are excessive
pod_full_validation = enabled
```

2.3 Managing Apstra Server

2.3.1 Monitoring Apstra Server via CLI

The Apstra server CLI command `sudo service aos status` can be used for simple status check.

```
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:~$
```

The Apstra server script `aos_Apstra_server_health_check` is available for further error collection. The command runs a search for known error signatures in the Apstra server logs, such as agent crashes, and returns the output. The information can be used for troubleshooting.

Usage:

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

The script outputs errors it finds. If no errors are found, it returns no output.

2.3.2 Restarting Apstra Server

Telemetry Alarm

Device agents may temporarily log a “liveness” telemetry alarm if the Apstra server is down, but this clears in a minute or two after services are restored.

To restart the Apstra server, you can simply reboot the VM, or use the service command `sudo service aos stop`.

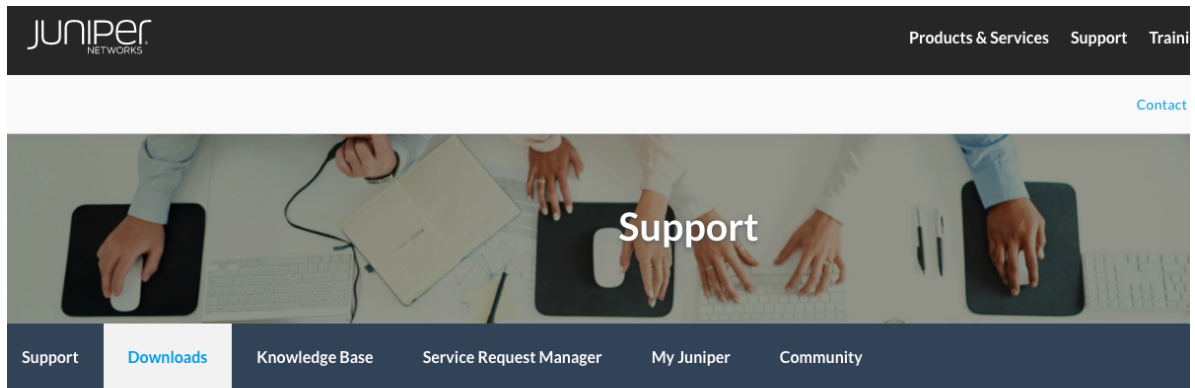
```
admin@aos-server:~$ sudo service aos stop
admin@aos-server:~$ sudo service aos start
admin@aos-server:~$
```

2.3.3 Reinstalling Apstra Server

This is a *nuclear* option that wipes and reinstalls the software on the existing server. This is mostly helpful when doing a *proof of concept* or demo install. If you have problems that require you to reinstall the software, please contact the Juniper JTAC Apstra Support group at <https://support.juniper.net/>.

Warning: The below procedure removes ALL Apstra data from the Apstra server VM. Use with care.

1. Download the “Installer” .run file from the support portal, <https://support.juniper.net/support/downloads/?p=afc>.



Find a Product

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

[All Products](#) ▾
 Apstra Fabric Conductor

[View all products >](#)

Download Results for: Apstra Fabric Conductor

Select: OS Apstra Fabric Conductor ▾
 VERSION 3.3 ▾
 SUPPORTING PLATFORMS Show All ▾

| ✕ Application Package | | | |
|---------------------------|---------|-------------|---|
| Description | Release | File Date | Downloads |
| AIS Installer for Upgrade | 3.3.0.2 | 05 Feb 2021 | run (783.22MB) Checksums |

2. Stop AOS service, if possible.

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

3. Delete the Apstra server database.

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

4. Remove the aos-compose package.

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

5. Reinstall the software from the .run file.

```

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

```

You can now restore an `aos_backup` or build a new blueprint.

2.3.4 Apstra Server Database Management

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

Requirements

- Source and Target database versions must be the same version. If versions are different, before proceeding please contact the Juniper JTAC Apstra Support group at <https://support.juniper.net/> to obtain assistance with the procedure.
- Source and Target must have the same IP address after starting the Apstra server to ensure that device agents can ‘call home’ properly after database restoration. The software can be restored to a different IP address, but then each device agent must be reconfigured (`/mnt/flash/aos-config`, `/etc/aos/aos.conf`) to point to the new Apstra server IP.

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

2.3.4.1 Backing up Database

Backups can be performed while the software is running. They are saved in the Apstra server as a dated snapshot: `/var/lib/aos/snapshot/`.

If the Apstra server needs to be restored or if its disk image becomes corrupt, the Apstra server is lost, including any backups. We recommend that you periodically move backups from the Apstra server to a secure location.

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.

Back up the Apstra server with the `aos_backup` command.

If all IBA probes have been disabled, the following message appears.

```

admin@aos-server:~$ sudo aos_backup
=====
Backup operation completed successfully.

```

(continues on next page)

(continued from previous page)

```
=====
New AOS snapshot: 2020-07-28_20-56-26
admin@aos-server:~$
```

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

```
admin@aos-server:~$ sudo aos_backup
=====
Warning:
Backup operation has been completed successfully. However AOS state
has been changed while this script was running, which means some
changes might not have been captured in the snapshot created in this
backup. You may choose to invoke aos_backup script again if you wish
to capture these changes right now instead of waiting for the next
backup operation.
=====
New AOS snapshot: 2019-12-06_16-15-57
admin@aos-server:~$
```

In this case, disable IBA probes and run the `aos_backup` command again.

Note: Backup/restore currently does not support Devices/OS Images information. Devices/OS Images information will be lost after the restore operation.

2.3.4.2 Validating Backup

The backup is stored in `/var/lib/aos/snapshot/<date>/aos.data.tar.gz`

Copy the contents of the snapshot directory to your backup infrastructure. This also includes the `aos_restore` script.

```
admin@aos-server:~$ sudo ls -lah /var/lib/aos/snapshot/
total 20K
drwx----- 5 root root 4.0K Jul 28 20:58 .
drwxr-xr-x 7 root root 4.0K Jul 28 02:43 ..
drwx----- 2 root root 4.0K Jul 28 02:43 2020-07-28_02-43-12
drwx----- 2 root root 4.0K Jul 28 20:56 2020-07-28_20-56-26
drwx----- 2 root root 4.0K Jul 28 20:58 2020-07-28_20-58-54
admin@aos-server:~$
```

2.3.4.3 Restoring Database

Note: If restoring a backup to a new Apstra server that uses a different network interface for access (e.g. `eth1` vs `eth0`), the `metadb` variable in the `[controller]` section of the `/etc/aos/aos.conf` configuration file must be updated and the Apstra server must be restarted.

1. Confirm the contents of the snapshot folder exists somewhere on the filesystem. For example, we have copied the restoration data to `/tmp/aos_test_restore`.

```
admin@aos-server:~$ sudo ls -lah /var/lib/aos/snapshot/2020-07-28_20-56-26/
total 21M
drwx----- 2 root root 4.0K Jul 28 20:56 .
drwx----- 5 root root 4.0K Jul 28 20:58 ..
-rw----- 1 root root 21M Jul 28 20:56 aos.data.tar.gz
-rwxr-xr-x 1 root root 1.3K Jul 28 20:56 aos_restore
-rw----- 1 root root 1 Jul 28 20:56 comment.txt
admin@aos-server:~$
```

2. Run the restore command from the snapshotted restoration file. The restore process first backs up the current database.

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

(continues on next page)

(continued from previous page)

```

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

```

2.3.4.4 Validating Restore

The database has been restored and migrated to a new server. The entire system state has been copied over from the backed up installation to the new target.

```

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

Jul 28 00:36:32 aos-server aos[1078]: [240B blob data]
Jul 28 00:36:32 aos-server systemd[1]: Started LSB: Start AOS management system.
Jul 28 02:45:45 aos-server systemd[1]: Stopping LSB: Start AOS management system...
Jul 28 02:45:46 aos-server aos[4968]: Stopping aos_controller_1 ...
Jul 28 02:45:46 aos-server aos[4968]: Stopping aos_metadb_1      ...
Jul 28 02:45:46 aos-server aos[4968]: Stopping aos_auth_1       ...
Jul 28 02:45:46 aos-server aos[4968]: Stopping aos_sysdb_1      ...

```

(continues on next page)

(continued from previous page)

```

Jul 28 02:45:46 aos-server aos[4968]: Stopping aos_nginx_1    ...
Jul 28 02:45:58 aos-server aos[4968]: [240B blob data]
Jul 28 02:45:58 aos-server systemd[1]: Stopped LSB: Start AOS management system.
admin@aos-server:~$ docker ps -a

```

| CONTAINER ID | IMAGE | COMMAND | CREATED |
|--------------|--------------------------|--------------------------|--------------------|
| 8bc9c1dd7a3a | aos:3.3.0-660 | "/usr/bin/aos_launch..." | About a minute ago |
| b0191320d2bd | aos:3.3.0-660 | "/usr/sbin/aos_launc..." | About a minute ago |
| 136736759f45 | aos:3.3.0-660 | "/usr/sbin/aos_launc..." | About a minute ago |
| 00a12eb03ae5 | aos:3.3.0-660 | "/usr/sbin/aos_launc..." | About a minute ago |
| c9b18cd4f55a | aos:3.3.0-660 | "/usr/sbin/aos_launc..." | About a minute ago |
| 90f35781d2a0 | aos:3.3.0-660 | "/usr/sbin/aos_launc..." | About a minute ago |
| f5c2d249176b | aos:3.3.0-660 | "/usr/sbin/aos_launc..." | About a minute ago |
| ab6c532a37ad | aos:3.3.0-660 | "/usr/bin/aos_launch..." | 20 hours ago |
| 8e6fd8ae8f08 | aos:3.3.0-660 | "/usr/bin/aos_launch..." | 20 hours ago |
| 5b6359e21386 | aos:3.3.0-660 | "/usr/bin/aos_launch..." | 20 hours ago |
| f665ce206f46 | aos:3.3.0-660 | "/usr/bin/aos_launch..." | 20 hours ago |
| 335dec5fba44 | nginx:1.14.2-upload-echo | "nginx -g 'daemon of..." | 20 hours ago |

```

admin@aos-server:~$

```

2.3.4.5 Resetting Database

This procedure deletes *all* data on the Apstra server to a fresh state.

```

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:~$

```

2.3.4.6 Migrating Database

Warning: 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 for 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 on the new server. This process is recommended for users using on-box device system agents.

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

To migrate an active instance from one server to another:

1. Backup the original Apstra server using the `sudo aos_backup` command and copy the snapshot to the new server using a temporary IP on the new 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. 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 -C /var/lib/aos/snapshot/ 2020-07-27_22-49-34
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:~$
```

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

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

4. 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. See [Configuring Static Management IP Address](#) for more information.

5. 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:~$
```

6. Restore the database on the new Apstra server with the snapshot `aos_restore` command.

```
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:
=====
```

(continues on next page)

(continued from previous page)

```

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

```

(continues on next page)

(continued from previous page)

```

/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$

```

7. The `aos_restore` command automatically starts the service when it is finished restoring the database. Verify 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$

```

8. Verify that your devices are online in their “Active” state.

☆ 🏠 > Devices > Managed Devices

1-6 of 6

Page Size: 25

| Device Key | Device Profile | Operation Mode | Management IP | AOS Version | Hostname | Location | OS | Acknowledged? | State | Blueprint | Comms |
|--------------|----------------|----------------|---------------|------------------|--------------------------|----------|----------------|---------------|-----------|-----------------|-------|
| 525400C610DE | Juniper vQFX | FULL CONTROL | 172.20.34.7 | AOS_3.3.0_OB.660 | spine1 | | Junos 18.4R1.8 | ✓ | IS-ACTIVE | L2 Virtual EVPN | 📶 |
| 525400DB8FAD | Juniper vQFX | FULL CONTROL | 172.20.34.8 | AOS_3.3.0_OB.660 | I2-virtual-ext-004-leaf1 | | Junos 18.4R1.8 | ✓ | IS-ACTIVE | L2 Virtual EVPN | 📶 |
| 525400B09AAD | Juniper vQFX | FULL CONTROL | 172.20.34.9 | AOS_3.3.0_OB.660 | spine2 | | Junos 18.4R1.8 | ✓ | IS-ACTIVE | L2 Virtual EVPN | 📶 |
| 52540081AD32 | Juniper vQFX | FULL CONTROL | 172.20.34.10 | AOS_3.3.0_OB.660 | I2-virtual-ext-001-leaf1 | | Junos 18.4R1.8 | ✓ | IS-ACTIVE | L2 Virtual EVPN | 📶 |
| 5254002CF610 | Juniper vQFX | FULL CONTROL | 172.20.34.11 | AOS_3.3.0_OB.660 | I2-virtual-ext-003-leaf1 | | Junos 18.4R1.8 | ✓ | IS-ACTIVE | L2 Virtual EVPN | 📶 |
| 525400F5415B | Juniper vQFX | FULL CONTROL | 172.20.34.12 | AOS_3.3.0_OB.660 | I2-virtual-ext-002-leaf1 | | Junos 18.4R1.8 | ✓ | IS-ACTIVE | L2 Virtual EVPN | 📶 |

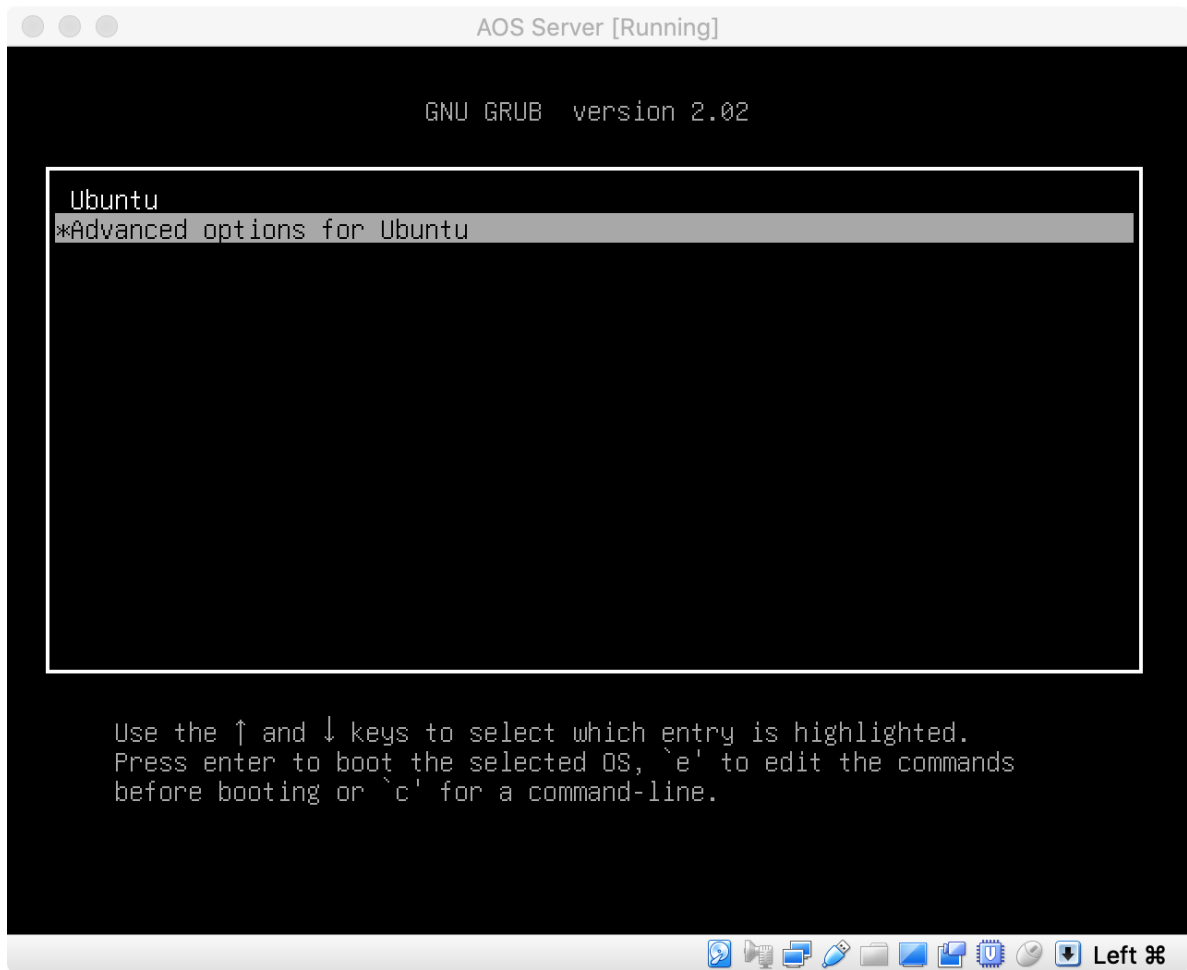
2.3.5 Resetting Apstra Server VM Password

Your Fabric is safe.

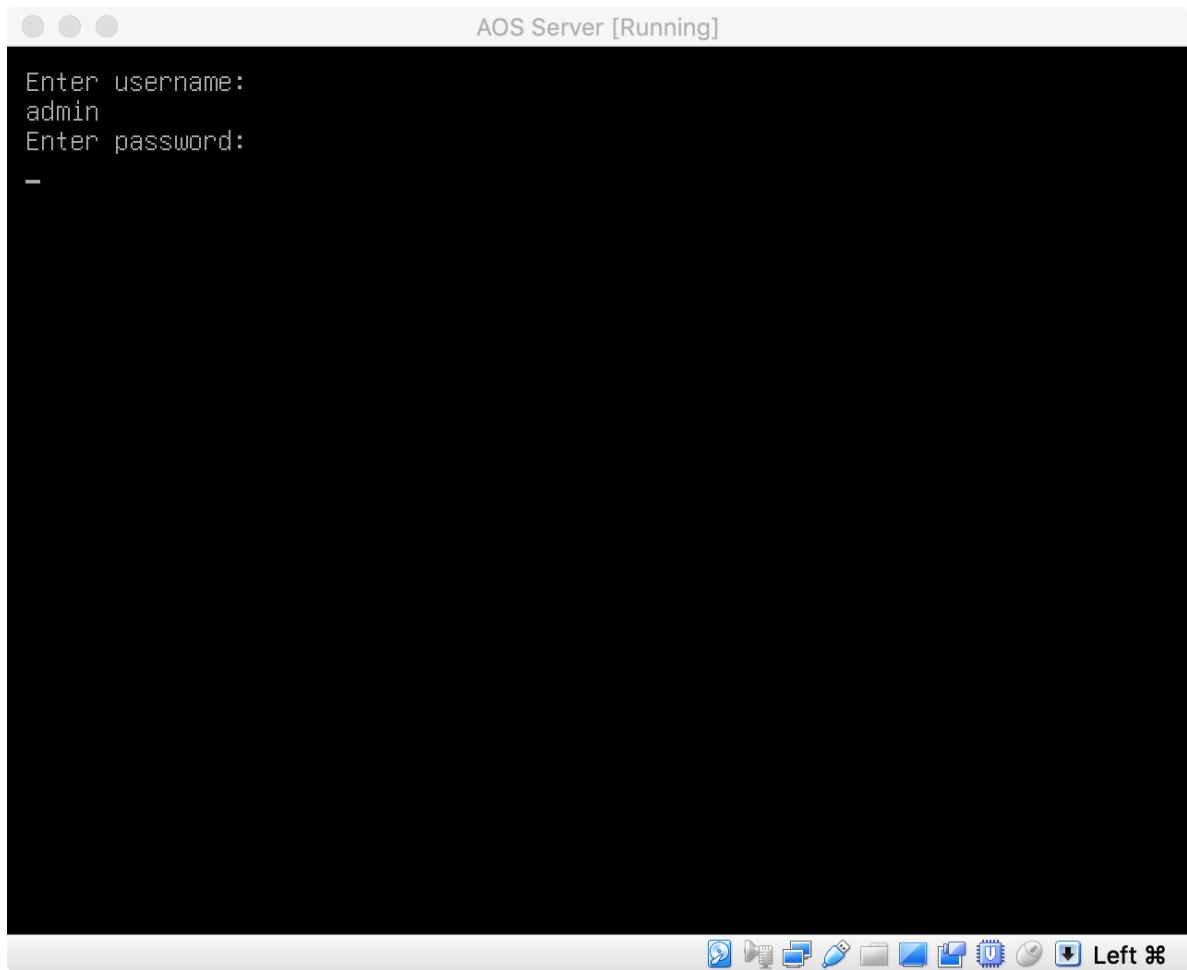
You'll need to reboot the Apstra server VM, but any deployed Fabric will not be impacted.

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 the password.

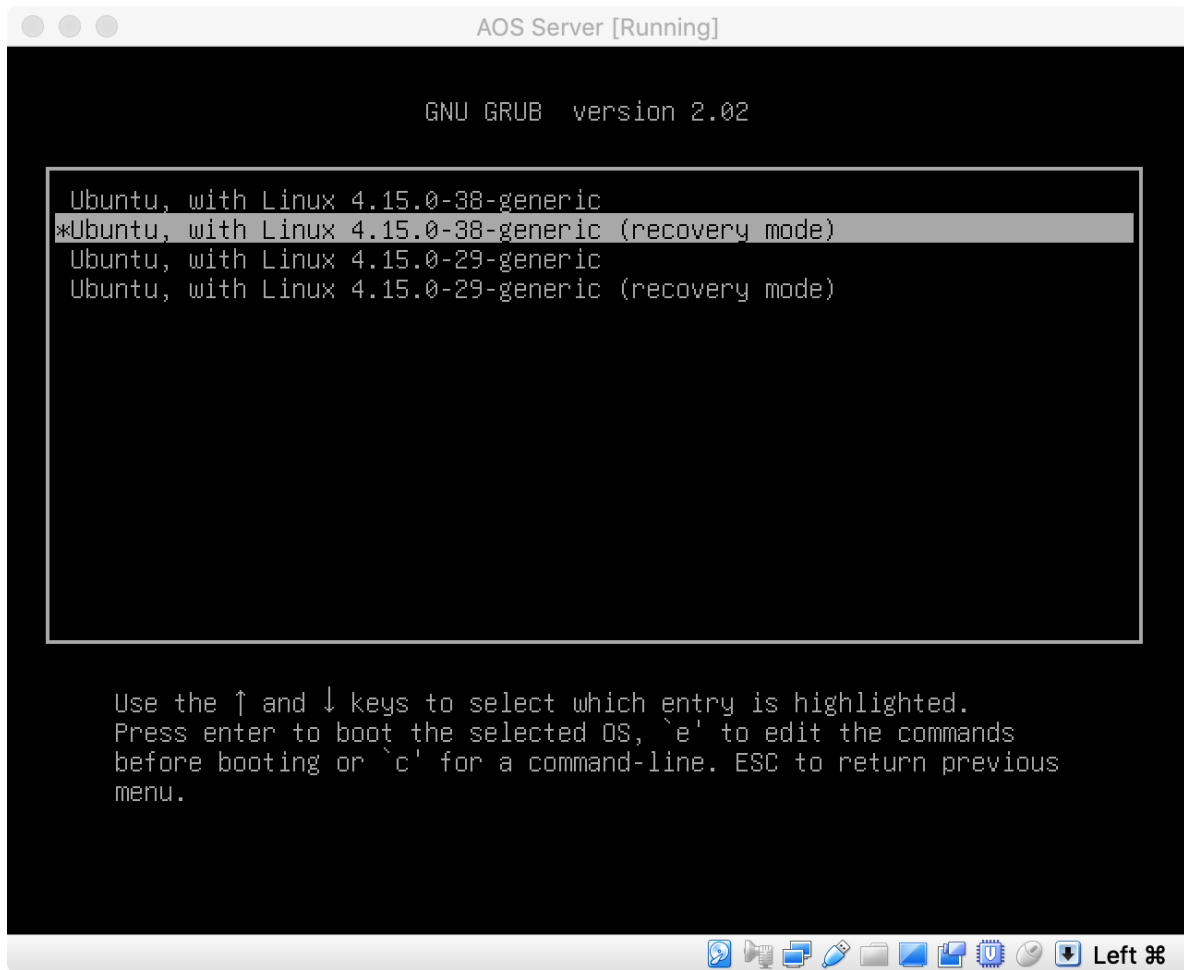
1. Attach to the Apstra server console and send a “reset” signal to the VM. To access the GRUB menu, immediately press the **esc** or **shift** key in the console on reboot.



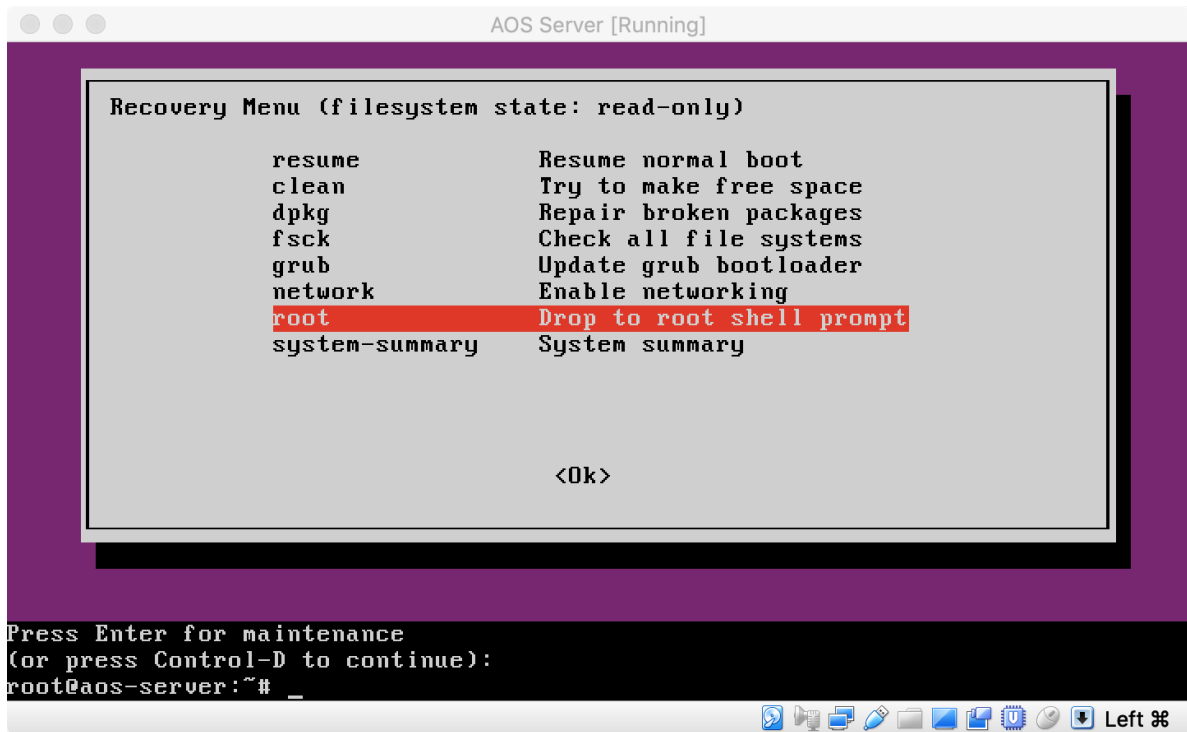
2. Select **Advanced options for Ubuntu**. For versions 2.3 and later, use username **admin** and password **apstra**.



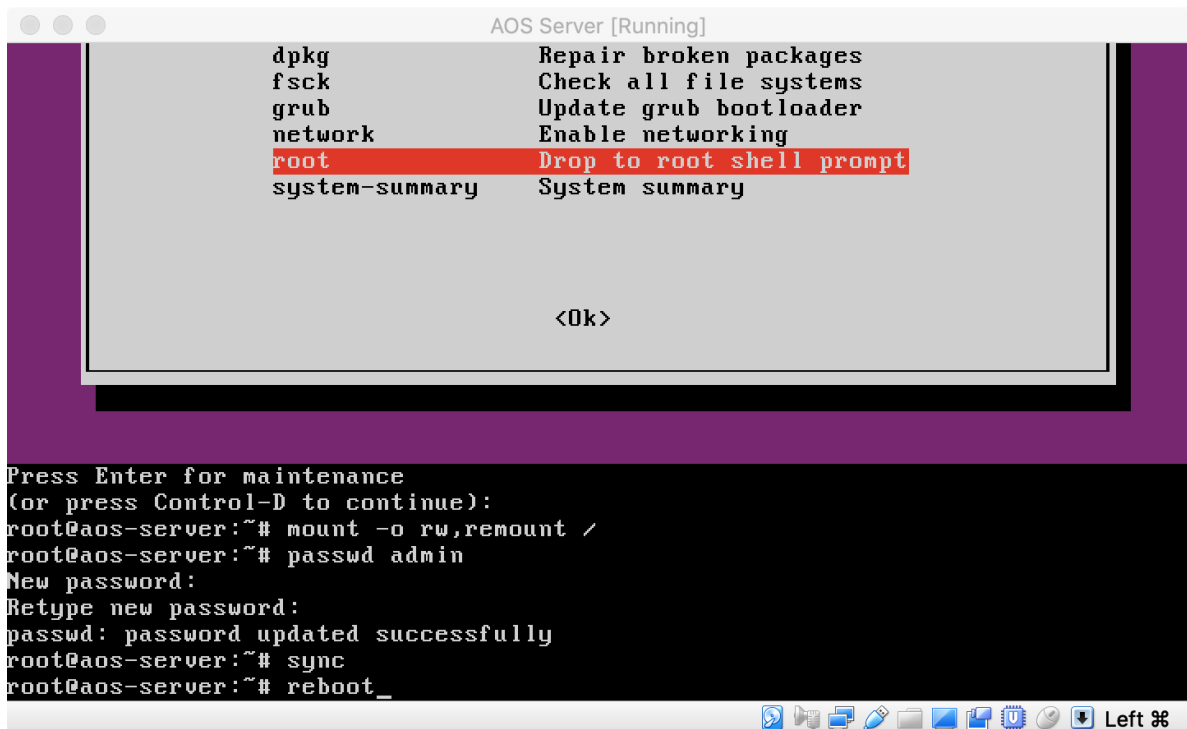
3. At the next GRUB menu, select the first (**recovery mode**) option.



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



5. At the root shell prompt enter `mount -o rw,remount /` then `passwd admin` to reset the password for the default **admin** user for the Apstra server VM Linux CLI.
6. Enter `sync` then `reboot` to reboot the VM.



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

2.4 Upgrading Apstra Server

For new installations with new blueprints, only version 3.3.0 is available.

Note: In 3.3.0 minor releases, upgrade is supported from **3.3.0.1** to **3.3.0.2**.

2.4.1 Supported Upgrade Paths

Warning: “.0” major feature release versions (e.g. 3.3.0, 3.2.0, 3.1.0) are for new installations ONLY. You CANNOT UPGRADE to major feature release versions (starting with version 3.1.0). Supported upgrades include maintenance release versions “.1” and later (e.g. 3.1.1, 3.1.2, etc.). See below for supported upgrade path details.

Check your Software Version

From the web interface, navigate to **Platform > About**.

From the Apstra server VM CLI run the command `service aos show_version`.

To 3.2.4-65, the following upgrade paths are supported:

- 3.2.3-46 to 3.2.4-65 (All)
- 3.2.2.4-9 to 3.2.4-65 (All)
- 3.2.2.3-3 to 3.2.4-65 (All)
- 3.2.2.2-3 to 3.2.4-65 (All)
- 3.2.2.1-2 to 3.2.4-65 (All)
- 3.2.2-12 to 3.2.4-65 (All)
- 3.2.1-298 to 3.2.4-65 (All)
- 3.2.0-242 to 3.2.4-65 (All)
- 3.1.1-179 to 3.2.4-65 (All)

To 3.2.3-46, the following upgrade paths are supported:

- 3.2.2-12 to 3.2.3-46 (All)
- 3.2.1-298 to 3.2.3-46 (All)
- 3.2.0.2-4 to 3.2.3-46 (All)
- 3.2.0.1-2 to 3.2.3-46 (All)
- 3.2.0-242 to 3.2.3-46 (All)
- 3.1.1-179 to 3.2.3-46 (All)

To 3.2.2-12, the following upgrade paths are supported:

- 3.2.1-298 to 3.2.2-12 (All)
- 3.2.0.2-4 to 3.2.2-12 (All)

- 3.2.0.1-2 to 3.2.2-12 (All)
- 3.2.0-242 to 3.2.2-12 (All)
- 3.1.1-179 to 3.2.2-12 (All)

To 3.2.1-298, the following upgrade paths are supported:

- 3.2.0-242 to 3.2.1-298 (All)
- 3.2.0.1-2 to 3.2.1-298 (All)
- 3.1.1-179 to 3.2.1-298 (All)

To 3.1.1-179, the following upgrade paths are supported:

- 2.3.1-129 (or higher) to 3.1.1-179 (All)
- 3.0.0-151 (or higher) to 3.1.1-179 (All)
- 3.0.1-96 to 3.1.1-179 (All)
- 3.0.2-133 to 3.1.1-179 (All)
- 3.1.0-206 to 3.1.1-179 (All)

Note: A number of Linux kernel and Nginx security updates were added to version 3.1.1. In order to receive Ubuntu Linux OS and Nginx updates, you **must** perform a VM to VM upgrade.

To 3.0.1-96, the following upgrade paths are supported:

- 2.2.1-166 to 3.0.1-96 (VM to VM)
- 2.3.0-181 to 3.0.1-96 (All)
- 2.3.1-129 to 3.0.1 (All)
- 2.3.2-130 to 3.0.1-96 (All)

For information about additional upgrade paths that may be supported, contact [Support](#).

2.4.2 Known Limitations and Issues

- In versions 3.2.x and earlier versions, the Apstra server VMs deployed from the same image share the same SSH keys. Due to the way the VM is packaged, all instances installed from the same OVA/QCOW image have the same SSH host keys. As a result, an attacker can more easily bypass remote host verification when a user connects by SSH to what is believed to be a previously used Apstra server host but is really the attacker's host performing a man-in-the-middle attack. To update the SSH Host Keys, see [Updating SSH Host Keys](#).
- To versions 3.2.x, as NCLU syntax is no longer used in Cumulus Interface type configlets since 3.2.0, when upgrading from versions 3.1.1 or earlier, Cumulus Interface type configlets must be modified in Linux command style to exactly match /etc/network/interfaces syntax.
- To version 3.1.1, due to AOS bug (AOS-15213), the device system agent upgrade process for Cumulus Linux devices will cause the device *switchd* process to restart temporarily disrupting traffic on the device. Please contact Support for a maintenance version of version 3.1.1 which will support upgrades of networks with Cumulus Linux devices without disruption.
- To version 3.1.1, because of a known AOS bug (AOS-14681), AOS health check on the Apstra server fails and indicates that RCI (Root Cause Identification)-related agents have terminated after the upgrade. To prevent this from happening, disable RCI before upgrading. Please allow 10 minutes after disabling RCI before starting the upgrade. After the upgrade is complete, the user can re-enable RCI.

- To version 3.1.1, a number a Linux kernel and security updates have been added to the Ubuntu 18.04 base OS in version 3.1.1. To receive Ubuntu Linux OS updates, you **must** do a VM to VM upgrade.
- To version 3.1.1, AOS bug (AOS-14708) is fixed where users using macOS Catalina and Google Chrome, users cannot accept default self-signed HTTPS/SSL certificate provided with Google Chrome. However, upgrading users must do a VM to VM upgrade or generate a new self-signed HTTPS/SSL certificate. See [Replacing SSL Certificate](#) for more information.
- To any version, saved show tech files are discarded (AOS-14416). To prevent file loss (in case they are subsequently needed) download show tech files before upgrading the Apstra server.
- To any version, configlets are not updated during upgrade. You must maintain configlets manually.
- To any version, built-in device profiles and interface maps are updated to the shipped built-in IMs and DPs of the target release without warning. Any changes in built-in DPs and IMs made by users are not reflected. This is known as AOS-13125.
- To any version, ensure there are no configuration deviation anomalies prior to starting the upgrade. If there are configuration deviation anomalies, the devices may restart processes which may cause traffic disruption on the device.
- To any version, ensure there are no devices in “Undeploy” deploy mode. Upgrade cannot proceed when some devices are set to Undeploy.
- To any version, user **must** delete Device AAA/TACACS configlets from the blueprint before upgrading the Apstra server, device agent, or NOS. You can re-apply the configlets after the upgrades.

Check release notes for known limitations and issues.

2.4.2.1 API Changes

- All API changes introduced in version 2.3.1 are backward compatible with 2.3.0.
- The following non-backward compatible API changes exist between 2.2.1 and 2.3.1:
 - See [Rack Types and Rack Based Templates](#)
 - See [Virtual Network Creation](#)

2.4.2.2 Rack Types and Rack Based Templates

Advanced rack design, introduced in version 2.3, allows users to define rack types with non-MLAG leaf pairs, multiple MLAG leaf pairs, etc. The API for both rack types and rack-based templates have been modified in a breaking fashion. The breaking change is in the JSON payloads in the CRUD APIs.

```
POST/PUT/GET /api/design/rack-types
POST/PUT/GET /api/design/templates
```

2.4.2.3 Virtual Network Creation

Per-port virtual networks, introduced in version 2.3, allow users to create virtual network endpoints on individual interfaces of L2 servers that have multiple leaf-facing interfaces. For example, given an L2 server that has 1 link per leaf to a non-MLAG leaf pair, users can create virtual network endpoints on one or both of the leaf-facing server interfaces.

The virtual network facade API has been updated for this. The breaking change is in the JSON payload where the virtual network endpoints are specified. Instead of specifying a system node ID, API client must now specify an interface node ID.

2.4.3 Upgrading Apstra server VM In-Place

Same VM: Pros and Cons

Pro - No need to deploy a new VM, but the VM memory may need to be increased.

Con - OS updates are not part of the Apstra server in-place upgrade procedure. If OS updates are required, you can migrate to a different VM (VM-VM).

Make sure you have the following Apstra server permissions:

- AOS Operating System admin user privileges.
- AOS admin user group permissions.

Apstra server In-Place VM upgrade consists of:

1. *Pre-Upgrade Validation*
2. *Apstra server In-Place Upgrade To AOS 3.2*
3. *Apstra server In-Place Upgrade To AOS 3.1*
4. *Apstra server Agents In-Place Upgrade*
5. *Apstra server Cluster In-Place VM Upgrade*

2.4.3.1 Pre-Upgrade Validation (Same VM)

1. Verify that the upgrade is supported for your version. See *Supported Upgrade Paths*.
2. Verify that you have the required VM resources for upgrading. Run `free -h` and verify < 50% utilization before starting the upgrade.

```
admin@aos-server:~$ free -m
              total        used        free      shared  buff/cache   available
Mem:           64422         5663        54129           6         4628        58078
Swap:           4331           0         4331
admin@aos-server:~$
```

3. If utilization is > 50%, gracefully shutdown the Apstra server, add resources, then restart the Apstra server.
4. Validate the Apstra server health by logging in as **admin** and running the following command:

```
admin@aos-server:~$ service aos status
* aos.service - LSB: Start AOS management system
   Loaded: loaded (/etc/init.d/aos; generated)
   Active: active (exited) since Sun 2019-10-20 19:45:08 UTC; 6min ago
     Docs: man:systemd-sysv-generator(8)
    Tasks: 0 (limit: 4915)
   CGroup: /aos.service

admin@aos-server:~$
```

5. For each blueprint, review service and probe anomalies, resolve open anomalies as much as possible, and take notes of the remaining ones.
6. Perform a backup of the old Apstra server by running the `sudo aos_backup` command:

```
admin@aos-server:~$ sudo aos_backup
[sudo] password for admin:
=====
Backup operation completed successfully.
=====
New AOS snapshot: 2019-10-19_00-05-06
admin@aos-server:~$
```

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

2.4.3.2 Apstra Server In-Place Upgrade To Version 3.2

1. After you've performed the pre-upgrade validation, download the software installer `.run` image (e.g. `aos_3.2.1-298.run`) and transfer it to the Apstra server.

```
admin@aos-server:~$ ls -l *run
-rw----- 1 root root 800916510 Apr  2 17:45 aos_3.2.1-298.run
```

2. Start the upgrade by running the installer `.run` (e.g. `aos_3.2.1-298.run`) image.

```
admin@aos-server:~$ sudo bash aos_3.2.1-298.run
[sudo] password for admin:
Verifying archive integrity... All good.
Uncompressing AOS installer 100%
=====
Backup operation completed successfully.
=====
AOS[2020-04-09_23:12:33]: Loading AOS 3.2.1-298 image
AOS[2020-04-09_23:13:32]: Initiating upgrade pre-checker
AOS[2020-04-09_23:13:33]: Initiating docker library import DONE
AOS[2020-04-09_23:14:31]: Preparing to retrieve data from running AOS Server. DONE
AOS[2020-04-09_23:14:52]: Retrieving data from running AOS Server. This step can
↳take up to 10 minutes DONE
AOS[2020-04-09_23:17:24]: Importing retrieved state to AOS pre-checker. This step
↳can take up to 20 minutes DONE
Waiting for blueprint <evpn-veos-virtual> processing to finish.. Done
Summary saved to /tmp/aos-upgrade-config-summary-2020.04.09-231738
```

3. (new in version 3.2) Review a summary of configuration pushed to devices during this upgrade. Page through output then hit `q`.

```

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

BLUEPRINT: evpn-veos-virtual
(test-evpn.veos.2485377892357-1139715540 - evpn-veos-virtual)
                                Section
↳
Systems
=====
UPGRADE_BGP_AF_ROUTE_MAPS
↳spine1 [5054003468A8, 172.20.98.12]
~~~~~
↳leaf1 [505400DBF2ED, 172.20.98.11]

```

(continues on next page)

(continued from previous page)

```

Move route-maps from 'router bgp' context into per-address-family and
↳spine2 [5054007AA372, 172.20.98.13]
per-neighbor route-maps to account for advanced external routing policy
↳leaf3 [5054000F612A, 172.20.98.14]
config. Without this change, route-map policies are ambiguous between
↳leaf2 [5054004ED91F, 172.20.98.15]
ipv4, ipv6, and evpn address-families. This operation is performed
atomically using EOS 'configure session' feature to prevent traffic
loss during the change when the configuration is temporarily removed.

configure session bgp_af_route_map_upgrade
router bgp 64514
  default neighbor 172.16.0.13 route-map MlagPeer out
  default neighbor 198.51.100.2 route-map RoutesFromExt in
  default neighbor 198.51.100.2 route-map RoutesToExt out
  default neighbor l3clos-1 route-map AdvLocal out
  address-family ipv4
    neighbor 172.16.0.13 route-map MlagPeer out
    neighbor 198.51.100.2 route-map RoutesFromExt in
    neighbor 198.51.100.2 route-map RoutesToExt out
    neighbor l3clos-1 route-map AdvLocal out
  exit
  default neighbor l3clos-1 route-map AdvLocal out
  address-family ipv6
    neighbor l3clos-1 route-map AdvLocal out
  exit
  vrf blue
    default neighbor 172.16.0.15 route-map MlagL3PeerIn in
    default neighbor 172.16.0.15 route-map MlagL3PeerOut out
    default neighbor 198.51.100.2 route-map RoutesFromExt-blue in
    default neighbor 198.51.100.2 route-map RoutesToExt-blue out
    address-family ipv4
      neighbor 172.16.0.15 route-map MlagL3PeerIn in
      neighbor 172.16.0.15 route-map MlagL3PeerOut out
      neighbor 198.51.100.2 route-map RoutesFromExt-blue in
      neighbor 198.51.100.2 route-map RoutesToExt-blue out
    exit
  exit
  vrf blue
    default neighbor fc01:a05:198:51:100::2 route-map RoutesFromExt-blue in
    default neighbor fc01:a05:198:51:100::2 route-map RoutesToExt-blue out
    default neighbor fc01:a05:fab::7 route-map MlagL3PeerIn in
    default neighbor fc01:a05:fab::7 route-map MlagL3PeerOut out
    address-family ipv6
      neighbor fc01:a05:198:51:100::2 route-map RoutesFromExt-blue in
      neighbor fc01:a05:198:51:100::2 route-map RoutesToExt-blue out
      neighbor fc01:a05:fab::7 route-map MlagL3PeerIn in
      neighbor fc01:a05:fab::7 route-map MlagL3PeerOut out
    exit
  exit
  vrf red
    default neighbor 172.16.0.17 route-map MlagL3PeerIn in
    default neighbor 172.16.0.17 route-map MlagL3PeerOut out
    default neighbor 198.51.100.2 route-map RoutesFromExt-red in
    default neighbor 198.51.100.2 route-map RoutesToExt-red out
    address-family ipv4
      neighbor 172.16.0.17 route-map MlagL3PeerIn in

```

(continues on next page)

(continued from previous page)

```

neighbor 172.16.0.17 route-map MlagL3PeerOut out
neighbor 198.51.100.2 route-map RoutesFromExt-red in
neighbor 198.51.100.2 route-map RoutesToExt-red out
exit
exit
vrf red
default neighbor fc01:a05:198:51:100::2 route-map RoutesFromExt-red in
default neighbor fc01:a05:198:51:100::2 route-map RoutesToExt-red out
default neighbor fc01:a05:fab::d route-map MlagL3PeerIn in
default neighbor fc01:a05:fab::d route-map MlagL3PeerOut out
address-family ipv6
neighbor fc01:a05:198:51:100::2 route-map RoutesFromExt-red in
neighbor fc01:a05:198:51:100::2 route-map RoutesToExt-red out
neighbor fc01:a05:fab::d route-map MlagL3PeerIn in
neighbor fc01:a05:fab::d route-map MlagL3PeerOut out
exit
exit
exit
commit
no configure session bgp_af_route_map_upgrade
configure terminal

MLAG_PEER_POLICY_COMMUNITIES
↪leaf1 [505400DBF2ED, 172.20.98.11]
~~~~~
↪leaf2 [5054004ED91F, 172.20.98.15]
AOS 3.2.0 adds support for OSPF connectivity points. As part of
loop-prevention, BGP communities are used to satisfy loop prevention
for mutual redistribution between OSPF and BGP. When OSPF redistributed
routes are advertised across an MLAG Peer link (SVI or L3 Peer link), these
community values must be preserved to be used by table-map filtering.
This upgrade plugin ensures mlag leafs send BGP communities between
each other. Future AOS releases will also increase usage of communities.

router bgp 64514
neighbor mlag-peer send-community
neighbor mlag-peer send-community extended
exit

(END)

```

4. (new in version 3.2) Use the upgrade interactive menu to display config change summary, list all devices with config changes, dump all config changes, and so on. You can continue with the upgrade, or quit the upgrade.

```

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

aos-upgrade (h for help) # s

```

(continues on next page)

(continued from previous page)

```

aos-upgrade (h for help) # 1
Blueprint: evpn-veos-virtual
(test-evpn.veos.2485377892357-1139715540 - evpn-veos-virtual)
  spine1 [5054003468A8, 172.20.98.12]
  leaf1 [505400DBF2ED, 172.20.98.11]
  spine2 [5054007AA372, 172.20.98.13]
  leaf3 [5054000F612A, 172.20.98.14]
  leaf2 [5054004ED91F, 172.20.98.15]

aos-upgrade (h for help) # d
Dumping all configs to /tmp/aos-upgrade-configs-2020.04.09-232155
BLUEPRINT: evpn-veos-virtual
(test--evpn.veos.2485377892357-1139715540 - evpn-veos-virtual)
[1/5] 23:21:55 system: 5054003468A8
[2/5] 23:22:05 system: 505400DBF2ED
[3/5] 23:22:07 system: 5054007AA372
[4/5] 23:22:08 system: 5054000F612A
[5/5] 23:22:10 system: 5054004ED91F

aos-upgrade (h for help) #

```

Note: (d)ump : It dumps all config changes to a file. As of version 3.2.1, it takes about 4~5 seconds per device, i.e. 4~5 minutes for 60 devices, only for the first time, due to AOS bug (AOS-16728).

Warning: Upgrading the Apstra server is a disruptive process. When upgrading to the same VM, if you select **c** to continue, you cannot *rollback* the upgrade. The only way to return to the previous version is to reinstall a new VM with the previous version and restore the database from backup.

5. Successful upgrade is confirmed. You can also check the current version in the web interface at **Platform > About**.

```

aos-upgrade (h for help) # h

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

aos-upgrade (h for help) # c

AOS[2020-04-09_23:23:21]: Loading AOS Device Installer image
AOS[2020-04-09_23:24:40]: Stopping upgrade pre-checker
cc4590d6a718: Loading layer [=====>]
↪ 65.58MB/65.58MB
8c98131d2d1d: Loading layer [=====>]
↪ 991.2kB/991.2kB
03c9b9f537a4: Loading layer [=====>]
↪ 15.87kB/15.87kB

```

(continues on next page)

(continued from previous page)

```

1852b2300972: Loading layer [=====>]
↪ 3.072kB/3.072kB
583f37d385c1: Loading layer [=====>]
↪ 85.08MB/85.08MB
fc0141fa6aa5: Loading layer [=====>]
↪ 3.584kB/3.584kB
Loaded image: nginx:1.14.2-upload-echo
AOS[2020-04-09_23:25:11]: Removing installed (3.1.1-179) AOS package
AOS[2020-04-09_23:25:12]: Installing AOS 3.2.1-298 package
=====
AOS upgrade successful. Please find logs at:
/var/tmp/aos_upgrade_logs_20200409_232331.tgz
=====
admin@aos-server:~$
admin@aos-server:~$ service aos show_version
3.2.1-298

```

2.4.3.3 Apstra Server In-Place Upgrade To Version 3.1

1. After you've performed the pre-upgrade validation, download the software installer **.run** image (e.g. aos_3.1.1-179.run) and transfer it to the Apstra server.

```

admin@aos-server:~$ ls -l
total 810076
-rw----- 1 admin admin 829510572 Oct 15 20:45 aos_3.1.1-179.run
admin@aos-server:~$

```

2. Start the upgrade by running the installer **.run** (e.g. aos_3.1.1-179.run) image.

```

admin@aos-server:~$ sudo bash aos_3.1.1-179.run
Verifying archive integrity... All good.
Uncompressing AOS installer 100%
=====
Backup operation completed successfully.
=====
Loading AOS 3.1.1-179 image
Initiating upgrade pre-checker
AOS[2019-10-21_04:16:09]: Initiating docker library import DONE
AOS[2019-10-21_04:16:29]: Preparing to retrieve data from running AOS Server. DONE
AOS[2019-10-21_04:20:19]: Retrieving data from running AOS Server. This step can
↪take up to 10 minutes DONE
AOS[2019-10-21_04:26:52]: Importing retrieved state to AOS pre-checker. This step
↪can take up to 20 minutes DONE

```

3. Review any device configuration changes planned to be pushed after the upgrade. Page through output then hit **q**. Then answer (y/n) to continue.

Warning: This is a disruptive process. When upgrading to the same VM, if you select y to continue, you cannot *rollback* the upgrade. The only way to return to the previous version is to reinstall a new VM with the previous version and restore the AOS database from backup.

```

Device configuration will be updated for the following device(s):
=====
Device: 525400144A92 (FQDN: l2-virtual-ext-001-leaf1, Management IP Address: 172.
↪20.51.9)
=====
Additional configuration that would be pushed on device agent upgrade:
{
  "hostname": [{
    "filename": "/etc/hostname",
    "data": [
      "l2-virtual-ext-001-leaf1"
    ]
  },
  {
    "command": "/bin/hostname -F /etc/hostname"
  },
  {
    "command": "systemctl reset-failed lldpd.service"
  },
  {
    "command": "/usr/sbin/service lldpd restart"
  }
], "interfaces": [{
  "filename": "/etc/network/interfaces",
  "data": [
    "# This file was generated by AOS. Do not edit by hand.",
    "#",
    "# The loopback interface",
    "auto lo",
    "iface lo inet loopback",
    "    address 10.0.0.0/32",
    "",
    "# Fabric interfaces",
    "auto swp1",
    "iface swp1",
    "    address 10.0.0.7/31",
    "    alias facing_spine1:swp1",
    "",
    "auto swp2",
    "iface swp2",
    "    address 10.0.0.15/31",
    "    alias facing_spine2:swp1",
    "",
    "# L3Edge interfaces",
    :
Do you want to continue?(y/n):

```

4. Successful upgrade is confirmed. You can also check the version in the web interface at **Platform > About**.

```

Loading AOS Device Installer image
alaa3da2a80a: Loading layer [=====>]
↪ 65.56MB/65.56MB
ef1a1ec5bba9: Loading layer [=====>]
↪ 991.2kB/991.2kB
6c3332381368: Loading layer [=====>]
↪ 15.87kB/15.87kB
e80c789bc6ac: Loading layer [=====>]
↪ 3.072kB/3.072kB

```

(continues on next page)

(continued from previous page)

```

21e2934acccd: Loading layer [=====>]
↪ 86.59MB/86.59MB
43fd92dffd9: Loading layer [=====>]
↪ 3.584kB/3.584kB
Loaded image: nginx:1.14.2-upload
Removing installed (3.0.1-96) AOS package
Installing AOS 3.1.1-179 package
=====
AOS upgrade successful. Please find logs at:
/var/tmp/aos_upgrade_logs_20191021_043637.tgz
=====
admin@aos-server:~$ service aos show_version
3.1.1-179
admin@aos-server:~$

```

2.4.3.4 Apstra Server Agents In-Place Upgrade

When you upgrade the Apstra server to a new version, you must also upgrade the device agents version to match the Apstra server version.

Version 3.2

1. Log in to the web interface as **admin**.
2. Navigate to **Devices > System Agents > Agents**, select the devices that require upgrading, and click the 'Install' button. This re-initiates the installation process, which includes a version check. If there is a version mismatch, the agent will automatically be upgraded.

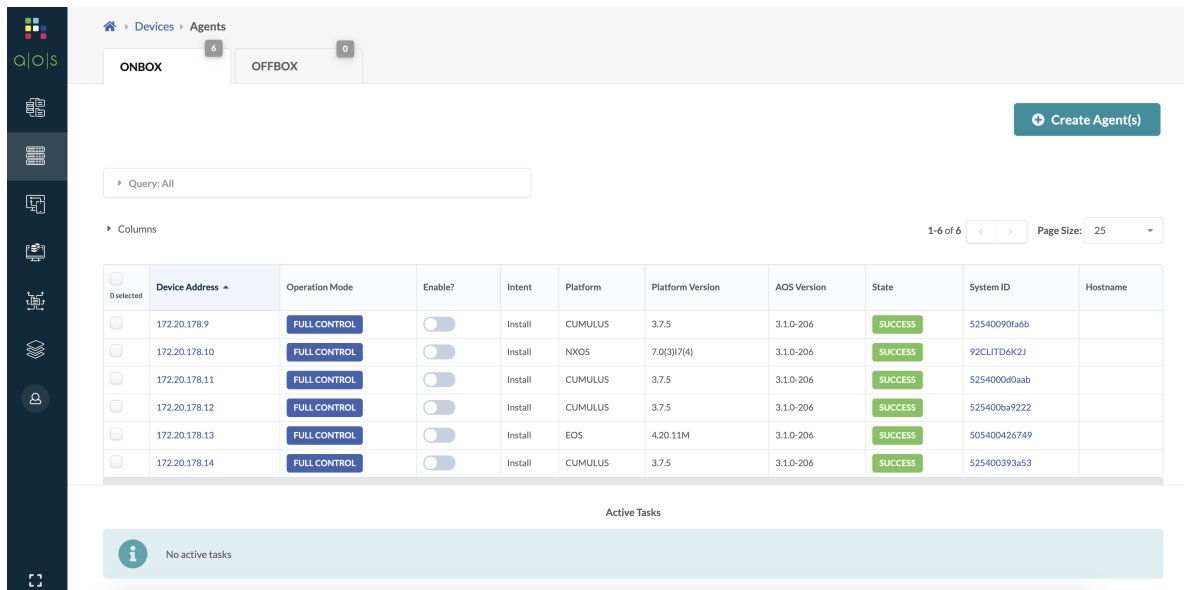
1-7 of 7 Page Size: 25

| Device Address | Operation Mode | Platform | Platform Version | AOS Version | Job State | System ID | Hostname | Device State | Actions |
|----------------|----------------|----------|------------------|-------------|-----------|--------------|----------|--------------|---------|
| 172.20.87.11 | FULL CONTROL | CUMULUS | 3.7.11 | 3.2.1-298 | SUCCESS | 52540040EAF | leaf-1-2 | IS-ACTIVE | ✓ |
| 172.20.87.12 | FULL CONTROL | CUMULUS | 3.7.11 | 3.2.1-298 | SUCCESS | 525400708CD3 | spine-2 | IS-ACTIVE | ✓ |
| 172.20.87.10 | FULL CONTROL | CUMULUS | 3.7.11 | 3.2.1-298 | SUCCESS | 5254000F6C9F | leaf-2 | IS-ACTIVE | ✓ |

3. On the Dashboard, under Liveness, verify there are no anomalies shown for the upgraded devices.

Versions 3.1 and Earlier

1. Log in to the web interface as **admin**.
2. Navigate to **Devices > System Agents > Agents**, and enable the system agents again, in a rolling fashion, to trigger the upgrade of the system agents.



Devices > Agents

ONBOX 6 OFFBOX 0

Create Agent(s)

Query: All

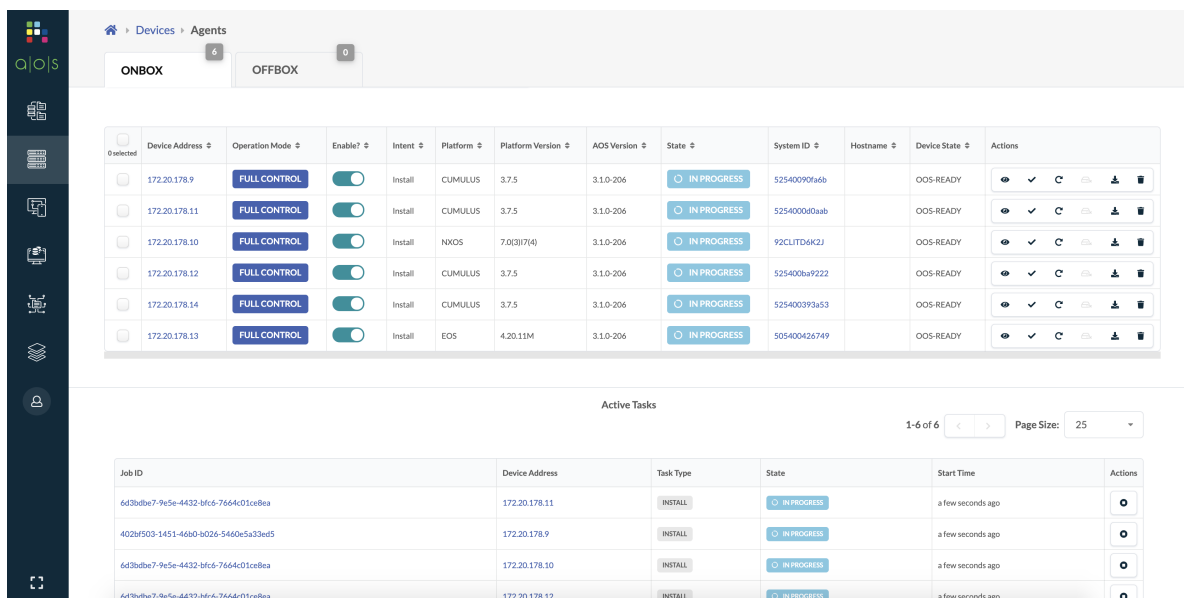
Columns 1-6 of 6 Page Size: 25

| Device Address | Operation Mode | Enable? | Intent | Platform | Platform Version | AOS Version | State | System ID | Hostname |
|----------------|----------------|--------------------------|---------|----------|------------------|-------------|---------|--------------|----------|
| 172.20.178.9 | FULL CONTROL | <input type="checkbox"/> | Install | CUMULUS | 3.7.5 | 3.1.0-206 | SUCCESS | 52540090fa6b | |
| 172.20.178.10 | FULL CONTROL | <input type="checkbox"/> | Install | NXOS | 7.0(3)(7/4) | 3.1.0-206 | SUCCESS | 92CLITD6K2J | |
| 172.20.178.11 | FULL CONTROL | <input type="checkbox"/> | Install | CUMULUS | 3.7.5 | 3.1.0-206 | SUCCESS | 5254000d0aab | |
| 172.20.178.12 | FULL CONTROL | <input type="checkbox"/> | Install | CUMULUS | 3.7.5 | 3.1.0-206 | SUCCESS | 525400ba9222 | |
| 172.20.178.13 | FULL CONTROL | <input type="checkbox"/> | Install | EOS | 4.20.11M | 3.1.0-206 | SUCCESS | 505400426749 | |
| 172.20.178.14 | FULL CONTROL | <input type="checkbox"/> | Install | CUMULUS | 3.7.5 | 3.1.0-206 | SUCCESS | 525400393a53 | |

Active Tasks

No active tasks

3. Wait for the upgrade to complete.



Devices > Agents

ONBOX 6 OFFBOX 0

| Device Address | Operation Mode | Enable? | Intent | Platform | Platform Version | AOS Version | State | System ID | Hostname | Device State | Actions |
|----------------|----------------|-------------------------------------|---------|----------|------------------|-------------|-------------|--------------|----------|--------------|---------|
| 172.20.178.9 | FULL CONTROL | <input checked="" type="checkbox"/> | Install | CUMULUS | 3.7.5 | 3.1.0-206 | IN PROGRESS | 52540090fa6b | | OOS-READY | |
| 172.20.178.11 | FULL CONTROL | <input checked="" type="checkbox"/> | Install | CUMULUS | 3.7.5 | 3.1.0-206 | IN PROGRESS | 5254000d0aab | | OOS-READY | |
| 172.20.178.10 | FULL CONTROL | <input checked="" type="checkbox"/> | Install | NXOS | 7.0(3)(7/4) | 3.1.0-206 | IN PROGRESS | 92CLITD6K2J | | OOS-READY | |
| 172.20.178.12 | FULL CONTROL | <input checked="" type="checkbox"/> | Install | CUMULUS | 3.7.5 | 3.1.0-206 | IN PROGRESS | 525400ba9222 | | OOS-READY | |
| 172.20.178.14 | FULL CONTROL | <input checked="" type="checkbox"/> | Install | CUMULUS | 3.7.5 | 3.1.0-206 | IN PROGRESS | 525400393a53 | | OOS-READY | |
| 172.20.178.13 | FULL CONTROL | <input checked="" type="checkbox"/> | Install | EOS | 4.20.11M | 3.1.0-206 | IN PROGRESS | 505400426749 | | OOS-READY | |

Active Tasks

1-6 of 6 Page Size: 25

| Job ID | Device Address | Task Type | State | Start Time | Actions |
|--------------------------------------|----------------|-----------|-------------|-------------------|---------|
| 6d3b0be7-9e5e-4432-bfcd-7664c01ce8ea | 172.20.178.11 | INSTALL | IN PROGRESS | a few seconds ago | |
| 402bf503-1451-46d0-b026-5460e533ed5 | 172.20.178.9 | INSTALL | IN PROGRESS | a few seconds ago | |
| 6d3b0be7-9e5e-4432-bfcd-7664c01ce8ea | 172.20.178.10 | INSTALL | IN PROGRESS | a few seconds ago | |
| 6d3b0be7-9e5e-4432-bfcd-7664c01ce8ea | 172.20.178.12 | INSTALL | IN PROGRESS | a few seconds ago | |

4. Verify that the new version is running on the system agents.

Devices > Agents

ONBOX 4 OFFBOX 0

Create Agent(s)

Query: All 1-6 of 6 Page Size: 25

| Device Address | Operation Mode | Enable? | Intent | Platform | Platform Version | AOS Version | State | System ID | Hostname | Device State | Actions |
|----------------|----------------|---------|---------|----------|------------------|-------------|---------|--------------|-----------|--------------|---------|
| 172.20.178.9 | FULL CONTROL | ON | Install | CUMULUS | 3.7.5 | 3.1.1-173 | SUCCESS | 52540090fa6b | cumulus | OOS-READY | [Icons] |
| 172.20.178.11 | FULL CONTROL | ON | Install | CUMULUS | 3.7.5 | 3.1.1-173 | SUCCESS | 525400080a8b | cumulus | OOS-READY | [Icons] |
| 172.20.178.10 | FULL CONTROL | ON | Install | NXOS | 7.0(3)17(4) | 3.1.1-173 | SUCCESS | 92CLTD6K2J | switch | OOS-READY | [Icons] |
| 172.20.178.12 | FULL CONTROL | ON | Install | CUMULUS | 3.7.5 | 3.1.1-173 | SUCCESS | 525400ba9222 | cumulus | OOS-READY | [Icons] |
| 172.20.178.14 | FULL CONTROL | ON | Install | CUMULUS | 3.7.5 | 3.1.1-173 | SUCCESS | 525400393a53 | cumulus | OOS-READY | [Icons] |
| 172.20.178.13 | FULL CONTROL | ON | Install | EOS | 4.20.11M | 3.1.1-173 | SUCCESS | 505400426749 | localhost | OOS-READY | [Icons] |

Active Tasks

No active tasks

5. When no liveness anomalies are present, you have successfully updated the Apstra server.

Devices > Managed Devices

Stock Device

Query: Hostname = ^ 1-6 of 6 Page Size: 25

| Device Key | Device Profile | Operation Mode | Management IP | AOS Version | Hostname | Location | OS | Acknowledged? | State | Blueprint | Comms |
|--------------|----------------|----------------|---------------|------------------|-----------|----------|------------------|---------------|-----------|--------------|-------|
| 525400393A53 | Cumulus VX | FULL CONTROL | 172.20.178.14 | AOS_3.1.1_OB.173 | cumulus | | Cumulus 3.7.5 | ✓ | OOS-READY | Not assigned | 📶 |
| 5054000ACFAE | Arista vEOS | FULL CONTROL | 172.20.178.13 | AOS_3.1.1_OB.173 | localhost | | EOS 4.20.11M | ✓ | OOS-READY | Not assigned | 📶 |
| 5254008A9222 | Cumulus VX | FULL CONTROL | 172.20.178.12 | AOS_3.1.1_OB.173 | cumulus | | Cumulus 3.7.5 | ✓ | OOS-READY | Not assigned | 📶 |
| 5254000D0AAB | Cumulus VX | FULL CONTROL | 172.20.178.11 | AOS_3.1.1_OB.173 | cumulus | | Cumulus 3.7.5 | ✓ | OOS-READY | Not assigned | 📶 |
| 5254006D5023 | Cisco NXOSv | FULL CONTROL | 172.20.178.10 | AOS_3.1.1_OB.173 | switch | | NXOS 7.0(3)17(4) | ✓ | OOS-READY | Not assigned | 📶 |
| 52540090FA68 | Cumulus VX | FULL CONTROL | 172.20.178.9 | AOS_3.1.1_OB.173 | cumulus | | Cumulus 3.7.5 | ✓ | OOS-READY | Not assigned | 📶 |

Note: If you need to roll back to the previous version, you must build a new VM with the previous version and restore the configuration to that VM. For assistance, please contact [Support](#).

2.4.3.5 AOS Cluster In-Place VM Upgrade

Upgrade process steps described above can also be used for In-Place VM upgrade for AOS Cluster. The workflow to upgrade AOS Cluster will be as:

1. *Pre-Upgrade Validation*
2. Download the aos.run file on the Apstra controller and worker nodes.
3. Install the aos.run in the Apstra controller node first. Once the installation is complete, the worker nodes will disconnect from the Apstra controller, and the state of worker node changes to **FAILED**.
4. For on-box system agents, refer to *AOS System Agents In-Place Upgrade*
5. Then install the aos.run file in the worker AOS nodes. Once the installation is complete, the worker nodes will reconnect to the Apstra controller and the state will show **ACTIVE** in AOS Cluster.

Note: **FAILED** state means that off-box agents and IBA probe containers located on that worker node will be unavailable, but devices managed by the off-box agents will remain in service.

Note: For in-place upgrade, post upgrade any off-box system agents will immediately apply any device configuration changes resulting from the upgrade.

2.4.4 Upgrading Apstra Server onto Different VM (VM-VM)

Different VM: Pros and Cons

Pro - Include any Ubuntu 18.04 OS updates (ex: security updates).

Con - Requires more infra level tasks (new VM deployment).

The upgrade operation requires the following AOS System permissions:

- AOS Operating System admin user privileges.
- AOS admin user group permissions.

Upgrading the Apstra server onto a different VM consists of:

1. *Pre-Upgrade Validation*
2. *Deploy New Apstra server*
3. *Import State*
4. *Apstra server cluster Upgrade to a Different VM*
5. *Apstra server Rollback (in case of failure)*
6. *System Agents Upgrade*
7. *Proxy and DNS updates, Shutdown of old AOS Server*

2.4.4.1 Pre-Upgrade Validation (Different VM)

1. Verify whether the upgrade is supported for your version. See *Supported Upgrade Paths*.
2. Validate system health by logging in as **admin** and running the following command:

```
admin@aos-server:~$ service aos status
* aos.service - LSB: Start AOS management system
   Loaded: loaded (/etc/init.d/aos; generated)
   Active: active (exited) since Mon 2019-01-21 22:11:36 UTC; 22h ago
     Docs: man:systemd-sysv-generator(8)
  Process: 6465 ExecStop=/etc/init.d/aos stop (code=exited, status=0/SUCCESS)
  Process: 6828 ExecStart=/etc/init.d/aos start (code=exited, status=0/SUCCESS)
```

3. For each blueprint, review service and probe anomalies, resolve open anomalies as much as possible, and take notes of the remaining ones.
4. Log in as **root**, and perform a backup of the old Apstra server by running the `sudo aos_backup` command:

```

root@aos-server:/# aos_backup
=====
Backup operation completed successfully.
=====
New AOS snapshot: 2019-01-08_15-24-15

```

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

2.4.4.2 Deploy New Apstra Server (Different VM)

Note: The Apstra server upgrade procedure does not include the migration of any customization done in `/etc/aos/aos.conf` file. If you had performed any customization of this file in the current Apstra server that needs to be migrated (ex: updated the `metadb` field to use a different network interface) you must re-apply the change yourself in the new Apstra server VM.

1. Download the Apstra server image from the portal.
2. Deploy a new Apstra server VM and configure the new Apstra server VM with a new IP address (same or new FQDN may be used), See [Apstra Server Installation](#) for instructions.

Note: Make sure your new VM has sufficient server resources. See [Apstra Server VM Resources](#) for more information.

3. Verify that the new Apstra server has SSH access to the old Apstra server.
4. Verify that the new Apstra server can reach the Systems Agents. See [Network Security Protocols](#).
5. Verify that the new Apstra server can reach any used external system (ex: NTP, DNS, vSphere server, LDAP or TACACs+ server, etc).

2.4.4.3 Import State (Different VM)

Note: Avoid API/GUI write operations on the old Apstra server during and after the upgrade step (see next steps) as these updates won't be automatically copied over to the new Apstra server.

1. Log in to the new Apstra server VM as **admin**.
2. Start the "Import State" operations on the new Apstra server by running the `sudo aos_import_state` script where `--ip-address` is the old Apstra server IP address. You will be prompted for the SSH admin password and for the root password of the old Apstra server.

```

root@aos-server:/home/admin# aos_import_state --ip-address 10.10.10.10 --username_
↪admin
SSH password for remote AOS VM:
Root password for remote AOS VM:
AOS[20190108_232845]: Preparing to retrieve data from remote AOS Server.
AOS[20190108_232858]: Retrieving data from remote AOS Server. This step can take_
↪upto 10 minutes
AOS[20190108_232958]: Successfully retrieved data from remote AOS Server.
AOS[20190108_232959]: Importing retrieved state to AOS. This step can take upto_
↪20 minutes

```

Note: The size of the blueprint and the Apstra server VM resources determine how long it takes to complete. As of version 3.2.2-12, if the database import exceeds twenty minutes, the operations may fail as “Timed Out” error. In such a case, the timeout value (seconds) can be extended as follows.

```
root@aos-server:/home/admin# AOS_UPGRADE_DOCKER_EXEC_TIMEOUT=3000 aos_import_
↪state --ip-address 10.10.10.10 --username admin
```

Contact [Apstra Global Support](#) for more information.

3. You will be prompted to approve the device configuration changes. Review any device configuration changes planned to be pushed after the upgrade.

Note: To come out of the config review mode and continue, press **q**.

```
=====
Device: 525400001DD0 (FQDN: spine-1, Management IP Address: 172.20.182.14)
=====
Additional configuration that would be pushed on device agent upgrade:
interface Ethernet1
  description facing_rack-002-leaf1:Ethernet3
!
interface Ethernet2
  description facing_rack-001-leaf2:swp4
!
interface Ethernet3
  description facing_rack-001-leaf1:swp2
!
=====
Device: 52540009BE9D (FQDN: spine-2, Management IP Address: 172.20.182.13)
=====
Additional configuration that would be pushed on device agent upgrade:
interface Ethernet1
  description facing_rack-002-leaf1:Ethernet4
!
interface Ethernet2
  description facing_rack-001-leaf2:swp3
!
interface Ethernet3
  description facing_rack-001-leaf1:swp4
!
=====
All existing onbox system agents will be disabled
AOS controller will not automatically upgrade AOS device agents. Use system_
↪agents to upgrade AOS device agents.
The incremental configurations that will be pushed to the device is saved at /tmp/
↪tmpJL92fp
```

4. (new in version 3.2) Use the Upgrade Interactive Menu to display config change summary, list all devices with config changes, dump all config changes, and so on. You can continue with the upgrade, or quit the upgrade from the menu.

```
AOS Upgrade: Interactive Menu
=====
```

(continues on next page)

(continued from previous page)

```

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

aos-upgrade (h for help) #

```

5. (Versions 3.1 and earlier) Approve the configuration changes and proceed by entering **y** to the next question (press “n” to interrupt the upgrade).

```

root@aos-server:/home/admin# aos_import_state --ip-address 172.20.145.3 --
↪username admin
SSH password for remote AOS VM:
Root password for remote AOS VM:
AOS[20190109_214903]: Preparing to retrieve data from remote AOS Server.
AOS[20190109_214915]: Retrieving data from remote AOS Server. This step can take ↪
↪upto 10 minutes
AOS[20190109_214926]: Successfully retrieved data from remote AOS Server.
AOS[20190109_214927]: Importing retrieved state to AOS. This step can take upto ↪
↪20 minutes
Do you want to continue?(y/n):

```

6. Verify that the prompt shows “Importing state successful” message.

```

AOS[20190109_221646]: Importing state successful.
AOS[20190109_221646]: Please find logs at /var/tmp/aos_upgrade_logs_20190109_
↪221646.tgz

```

7. Log in as admin into the new Apstra server web interface and verify that all system agents are in “Disabled” mode.

Note: At this point, it is expected to have liveness anomalies raised against the system agents until the complete upgrade is complete.

2.4.4.4 Cluster Upgrade to Different VM (using Import State)

The `aos_import_state` script can be used for two scenarios when upgrading a cluster.

Scenario-1: New VM for the controller node with reuse of worker node VMs.

Scenario-2: New VMs for controller node with new worker node VMs.

For **Scenario-1**, When a new VM is used for the controller node VM and the worker VMs are being reused, the workflow to follow is:

1. *Pre-Upgrade Validation*
2. Deploy the new Apstra server VM only for the controller node with the required version and resources. See *Deploy New Apstra Server*
3. Import the state from the running instance to the new instance using the `aos_import_state` script. Once the import is complete, the upgraded instance shows the state of worker nodes as **INACTIVE**. Run `aos_import_state` script on the new instance controller node. See *Import State*

4. Upgrade system agents. See *System Agents Upgrade*.
5. Download the .run file for the new version in the worker VMs.
6. Execute the .run file for the new version in the worker nodes. Once upgrade is complete for worker nodes, the cluster state changes to **ACTIVE**.
7. Remove the old controller VM.

For **Scenario-2**, When new VMs are used for the controller node VM and for the worker VMs, the workflow to follow is :

1. *Pre-Upgrade Validation*
2. Create new Apstra server VMs for the controller and worker nodes with the required version. For example, if the existing deployment has 2 worker VMs and a controller, create 3 new VMs. See *Deploy New Apstra Server*.
3. Designate one of the new VMs to be the new controller.
4. Execute `aos_import_state` from the new controller VM. Specify the **--cluster-node-address-mapping** argument to map between old cluster worker VMs and the new cluster worker VMs. See *Import State* If there were two worker nodes with IP 1.1.1.1 and 1.1.1.2 in old cluster and the new cluster worker node VMs have IP 2.2.2.1 and 2.2.2.2, specify the argument as

```
aos_import_state --ip-address <old_aos_controller> --username admin --password
admin --cluster-node-address-mapping 1.1.1.1 2.2.2.1 --cluster-node-address-
↪mapping
1.1.1.2 2.2.2.2.
```

5. Upgrade system agents See *System Agents Upgrade*
6. After completion of `aos_import_state` script and **system agents upgrade**, the new controller node shows the cluster with new worker node IPs.
7. Remove the old controller and worker VM nodes.

Note: It is possible to specify a subset of worker VMs as well in Step 2. That is, in the above scenario, if the argument **--cluster-node-address-mapping 1.1.1.2 2.2.2.2** is not specified, then the new Apstra controller VM will come up with worker nodes as 2.2.2.1 and 1.1.1.2. To upgrade the worker node in 1.1.1.2, the .run file can be used.

2.4.4.5 Apstra Server Rollback (Different VM)

In the case of a failed Apstra server upgrade, or other malfunction, you may rollback to the old Apstra server VM, as long as the system agents upgrade has not yet started. To do so:

1. Gracefully shutdown the new Apstra server.
2. Start the old Apstra server VM.

2.4.4.6 System Agents Upgrade (Different VM)

The agent upgrade process for a different VM is the same as for a same-VM upgrade. See *System Agents Upgrade*.

2.4.4.7 Proxy and DNS updates, Shutdown of old Apstra Server (Different VM)

1. Update any DNS entries to include the new Apstra server IP/FQDN based on your configuration.

2. If a proxy is used for the Apstra server, make sure the proxy now points to the new Apstra server.
3. Gracefully shutdown the old Apstra server VM. You have successfully upgraded the Apstra server.

2.4.5 Updating SSH Host Keys

Warning: In versions 3.2.x and earlier, due to the way the Apstra server VM is packaged, all Apstra server instances installed from the same OVA/QCOW image have the same SSH host keys.

As a result, this issue allows an attacker to more easily bypass remote host verification when a user connects by SSH to what is believed to be a previously used host but is really the attacker's host performing a man-in-the-middle attack.

To update the the SSH Host Keys on a Apstra server, follow this procedure to generate new SSH Host Keys for a new or existing Apstra server VM:

1. Remove the existing SSH host keys.

```
admin@aos-server:~$ sudo rm /etc/ssh/ssh_host*
```

2. Configure new SSH host keys.

```
admin@aos-server:~$ sudo dpkg-reconfigure openssh-server
Creating SSH2 RSA key; this may take some time ...
2048 SHA256:EWRFcs4V6BmOILR3T2PsxngluE0qXQ/z9IKkXrnLpJs root@aos-server (RSA)
Creating SSH2 ECDSA key; this may take some time ...
256 SHA256:THaXEia8VW6Jfw6OBXFegulCav0zcGSVOy9RkNOPxf4 root@aos-server_
↵ (ECDSA)
Creating SSH2 ED25519 key; this may take some time ...
256 SHA256:0HOn0nnF+7oRaF5HggI4vWeyxT+UNsHcbvNpBJdaKhQ root@aos-server_
↵ (ED25519)
```

3. Restart the SSH server process.

```
admin@aos-server:~$ sudo systemctl restart ssh
```

2.4.6 Q & A

- **Why are System Agents temporarily in Disabled mode in the new Apstra server after the upgrade?** It tells the Apstra server to not perform any commit onto the System Agents as the System Agents are running an older version. By enabling the System Agents again in the new Apstra server, the System Agents gets upgraded to match Apstra server version.
- **Will the device configurations change after the Apstra server upgrade?** Device configuration changes may be suggested when running the upgrade script: manually verify and commit these changes if accepted.
- **How does getting a new IP impact existing VMware vSphere integration?** Make sure any firewall between the new Apstra server and the vSphere server is allowing the connection. No other impact as Apstra server initiates the connection with vSphere Server, not vice-versa.

WEB INTERFACE (UI)

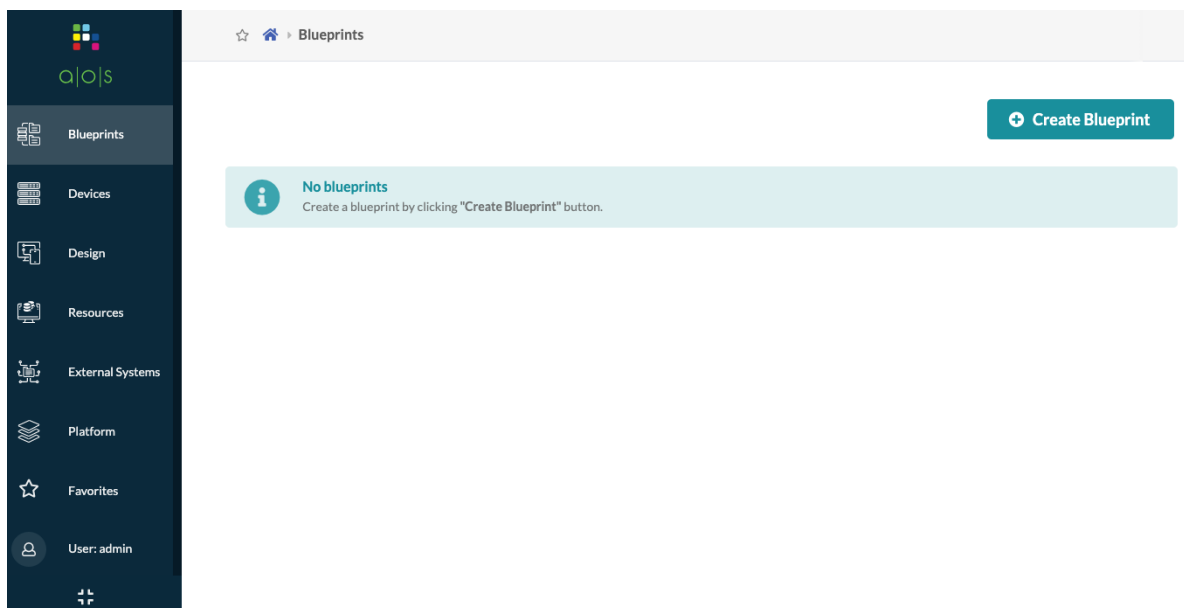
After the *Apstra server* is deployed, you can design, build, deploy, operate and validate the network from the web interface.

3.1 Accessing Web Interface

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.

Important: For security, please replace the default self-signed *SSL certificate* with one from your own certificate authority.

3. From the login page, enter username **admin** and password **admin** to go to the main screen.



Important: For security, please *change the web interface password* for **admin** after you first log in.

We recommend that you also change the operating system (OS) password.

1. Log into the configuration tool: `admin@aos-server:~$ aos_config`.
2. Choose **Local credentials** and change the OS password. (You can also change the webUI credentials password from here.)

For guidance on designing and building the network, see *Getting Started*.

3.2 Resetting Admin Password

To recover a forgotten or lost admin password for the web interface, log into the Apstra server as the default admin user via ssh, and type the command `aos_reset_admin_password`.

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

Important: For security, please *change the admin password* after resetting it to the default.

3.3 Replacing SSL Certificate

Encryption Only

The certificate is used only for encrypting the Apstra server and REST API, not for any internal device-server connectivity. We support and recommend replacing the default SSL certificate.

A unique self-signed certificate is automatically generated on each Apstra server at first boot. The default certificate files are stored on the Apstra server at `/etc/aos/nginx.conf.d`.

The HTTPS certificate is not retained in system backups. Backups of the `/etc/aos` folder must be performed manually when performing system backups.

- `nginx.crt` - public key for webserver
- `nginx.key` - private key for webserver

3.3.1 Replacing Existing Certificate with Signed 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. 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)
```

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

3. Create a certificate signing request.

If your certificate requires Subject Alternative Name (SAN), you will need your own OpenSSL template, which is beyond the scope of this document. If you need more advanced certificate support please contact support.

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

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

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

4. Submit your Certificate Signing Request (`nginx.csr`) to your Certificate Authority.

The required steps are outside the scope of this document - CA instructions differ per implementation. Any valid SSL certificate will work.

The example below is of self-signing the certificate.

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

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

```

root@aos-server:/etc/aos/nginx.conf.d# openssl rsa -noout -modulus -in nginx.key
↳| openssl md5
(stdin)= 60ac4532a708c98d70fee0dbcaable75

root@aos-server:/etc/aos/nginx.conf.d# openssl req -noout -modulus -in nginx.csr
↳| openssl md5
(stdin)= 60ac4532a708c98d70fee0dbcaable75

root@aos-server:/etc/aos/nginx.conf.d# openssl x509 -noout -modulus -in nginx.crt
↳| openssl md5
(stdin)= 60ac4532a708c98d70fee0dbcaable75

```

- Restart the nginx container to load the new certificate.

```

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

```

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

3.3.2 Replacing Existing Certificate with Self-Signed Certificate

Users on versions 3.1.0 and earlier that use macOS Catalina and Google Chrome cannot accept the default self-signed HTTPS/SSL certificate that is provided by Google Chrome. The self-signed certificate must be replaced. (AOS bug AOS-14708).

- 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

```

- Verify a Random Number Generator seed file .rnd exists in /home/admin. If not, create one.

```

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

```

- 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

```

(continues on next page)

(continued from previous page)

```

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. Restart the nginx container to load the new certificate.

```

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

```

3.4 Checking Web Interface Version

From the web interface, navigate to **Platform > About**.

3.5 Updating Web Interface Version

You can install an optional Apstra server UI update to add additional 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. Upload the Apstra server UI update file to the Apstra server. For this example, the file is named **aos-web-ui_2.2.0-67.run**.
2. Change to the root user and run the following file.

```

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:~#

```

3. During update, the current UI is copied to the `/opt/aos/frontend/snapshot/` snapshot directory.
4. From the web interface, navigate to **Platform > About** to confirm that the **UI version** has been updated.

3.6 Restoring Web Interface Version

1. You can restore the previous web interface version at any time without affecting the state of the Apstra server. From the snapshot directory, run the `webui_restore` file.

```
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#
```

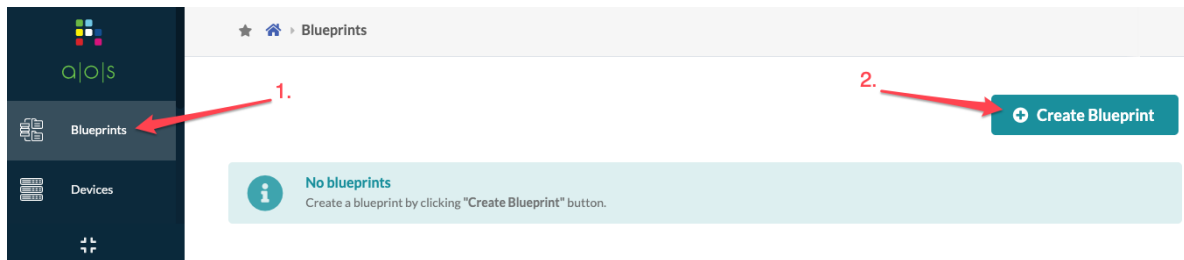
2. From the web interface, navigate to **Platform > About** to confirm that the **UI version** has been rolled back.

BLEUPRINTS

4.1 Creating Blueprint

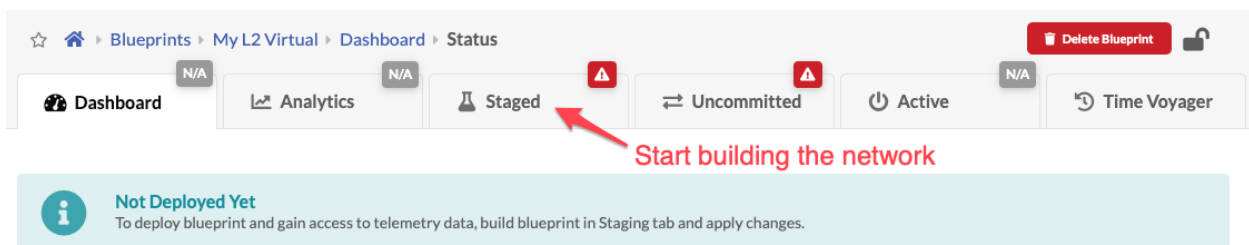
Before creating a blueprint, confirm that a template exists that meets the structure and policy needs of your network. If needed, you can *create a custom template*.

1. From the *web interface*, click **Blueprints**, then click **Create Blueprint**.



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

The next step after creating a blueprint is to *build* the network in the **Staged** view.



4.2 Dashboard

4.2.1 Blueprint Dashboard Overview

The blueprint dashboard shows the overall health and status of a committed blueprint. Statuses are indicated by color: green for succeeded, yellow for pending, and red for failed. The deployment status section includes deployment statuses for service config, discovery config, and (as of AOS version 3.3.0) drain config. The anomalies section includes statuses for all probes, IP fabric, external routing, L2 connectivity, liveness, deployment status, and route

verification. The nodes status section includes statuses for deployment, BGP, cabling, config, interface, liveness, route, and hostname.

From the [web interface](#), click **Blueprints** (in left menu), then click the name of a blueprint to go to its dashboard.

★ 🏠 > Blueprints

+ Create Blueprint

My L2 Virtual

L3 CLOS

Structure:

2 spines, 4 leaves, 8 L2 servers

Analytics

Deployment Status

N/A

Service Anomalies

N/A

Probe Anomalies

N/A

Root Causes:

N/A

Version 1

Last modified 18 minutes ago

☆ 🏠 > Blueprints > pod1 > Dashboard > Status
 Delete Blueprint

Dashboard Analytics Staged Uncommitted Active Time Voyager

Deployment Status

Service Config

5 SUCCEEDED

0 PENDING

0 FAILED

Discovery Config

0 SUCCEEDED

0 PENDING

0 FAILED

Drain Config

0 SUCCEEDED

0 PENDING

0 FAILED

Anomalies

All Probes

0 anomalies

IP Fabric

BGP

Cabling

Interface

Hostname

External Routing

BGP

Interface

L2 Connectivity

Interface

MLAG

LAG

Liveness

Spine

Leaf

L2 Server

Deployment Status

Deployment

Config Dev

Route Verification

Route Table

Nodes Status

Deployment

BGP

Cabling

Config

Interface

Liveness

Route

Hostname

evpn_mlag_001_leaf1

evpn_mlag_001_leaf2

evpn_single_001_leaf1

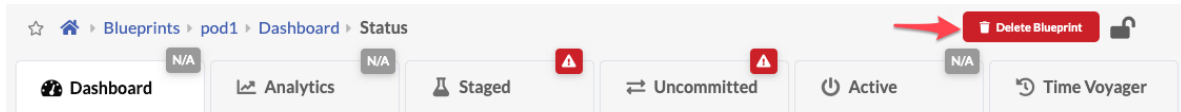
spine1

spine2

4.2.2 Deleting Blueprint

AOS administrators can delete blueprints.

1. From the blueprint, navigate to **Dashboard**, then click **Delete Blueprint** (top-right).



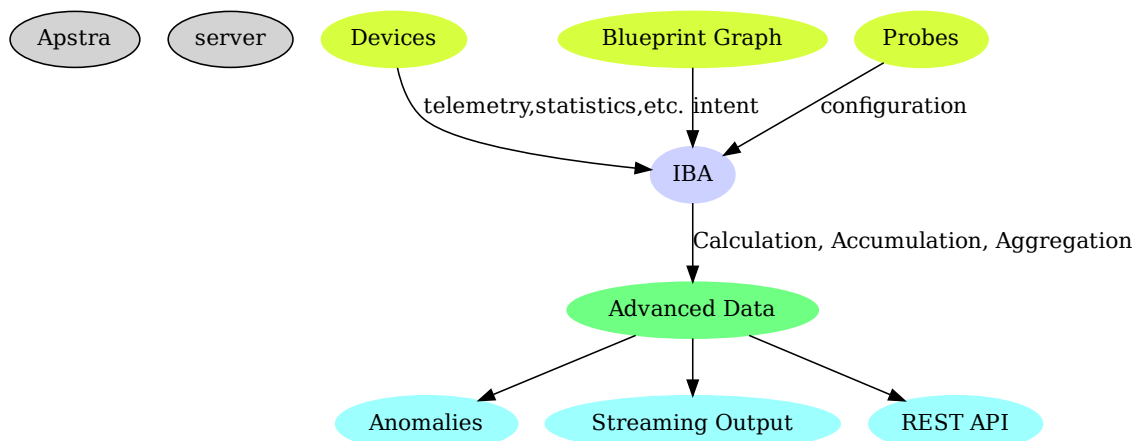
2. Enter the blueprint name, then click **Delete** to delete the blueprint and go to the blueprint summary view.

4.3 Analytics

4.3.1 Intent-Based Analytics

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* with current and historic data from devices to reason about the network at-large.

Data generated by devices are ingested via *agents* and sent to the Apstra server. With the use of IBA *probes*, data can be aggregated across devices in response to operator configuration. Combining probes with intent from the blueprint graph generates a reduced set of data that can be more easily reasoned about. You can directly inspect advanced data from the web interface or from *REST API* to gain real-time insight about the network. It can also be streamed out with our existing streaming infrastructure. Also, based on the state of this advanced data, *anomalies* can be raised.



4.3.1.1 Disk Usage

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 *Apstra Server Configuration File* `/etc/aos/aos.conf` indicate how often to rotate logs and remove old checkpoint files. Using IBA can increase disk usage to tens of gigabytes. If this is an issue, you can adjust the log rotation settings to reduce disk usage.

Additional space may be used by system snapshots and old images from any in-place Apstra server upgrades. These can be deleted or moved off the system to increase free disk space.

4.3.2 Dashboards

4.3.2.1 Analytics Dashboard Overview

Analytics dashboards are automatically created based on the state of the *active (operational) blueprint*. These dashboards contain *widgets* that monitor different aspects of the network and raise alerts to any anomalies. You cannot configure the trigger logic that determines when dashboards are auto-created, but you can create/instantiate your own dashboards. System-generated dashboards are labeled with **System** and user-generated dashboards are labeled with the user's name. Dashboards can be shown in various levels of detail by choosing a **Display mode**: summary, preview, expanded.

Additional characteristics of analytics dashboards include:

- You can display analytics dashboards on the *blueprint dashboard* to have additional network information on one screen. To add them, turn **ON** the analytics dashboards' default toggles.
- Auto-generated dashboards are not deleted automatically.
- Probes that already exist (already created by users) and that are not modified after initial creation, are reused instead of creating duplicates of those probes.
- Required probes and widgets are ensured only when a dashboard is initially enabled. Post auto-creation, any user action to update or delete referenced probes and widgets may cause the dashboard to enter an invalid state. Invalid dashboards are not automatically repaired.
- 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**.

4.3.2.2 Configuring Auto-Enabled Dashboards

Certain auto-created dashboards generate anomalies that are expected, so you may not want to see them. To suppress these anomalies, either proactively set the auto-creation toggle for the dashboard to **OFF**, or delete the dashboard after it has been enabled. Once a dashboard is disabled it will not be re-enabled unless the auto-enable toggle is set to **ON** and the respective trigger is satisfied.

1. From the blueprint, navigate to **Analytics > Dashboards**, then click **Configure Auto-Enabled Dashboards**. Dashboard are listed with descriptions giving more details including the trigger for when dashboards are auto-created. Widgets that are used in each dashboard are also listed.
2. Toggle the dashboards on to auto-enable them or off to disable them.

4.3.2.3 Instantiating Predefined Dashboard

Several predefined dashboards are available that can be instantiated and modified to show analytics in multiple ways. You can instantiate more than one instance of any predefined dashboard.

1. From the blueprint, navigate to **Analytics > Dashboards**, then click **Create Dashboard** and select **Instantiate Predefined Dashboard** from the drop-down list.
2. Select a predefined dashboard from the drop-down list. See below for descriptions of some of the predefined dashboards.
3. Click **Create** to instantiate the dashboard and return to the list view.

Device Health Summary Predefined Dashboard

| Goal | Trigger | Probes and Widgets |
|--|---------|--|
| Find issues in the system health of managed devices present in the BluePrint | Always | <ol style="list-style-type: none"> 1. Check that memory usage is below 80% 2. Check that average CPU usage is below 80% 3. Check that filesystem usage on every writable mount point is below a certain threshold |

Note: Ensure same metric is not collected twice from the same device.

Drain Validation Predefined Dashboard

| Goal | Trigger | Probes and Widgets |
|--|---|--|
| Ensure drained switches are indeed drained of traffic by ensuring total bandwidth is minimal | Presence of at-least one drained switch | <ol style="list-style-type: none"> 1. Drain Traffic |

Throughput Health Predefined Dashboard

| Goal | Trigger | Probes and Widgets |
|---|---------|---|
| 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 | Manual | All imbalance thresholds are expressed as % of the link speed and not absolute values. 1. Fabric ECMP imbalance 2. External ECMP imbalance 3. LAG imbalance - should include both MLAG and LAG |

Traffic Trends Predefined Dashboard

| Goal | Trigger | Probes and Widgets |
|---|---------|--|
| Visualize traffic trends for general insights into fabric usage | Manual | 1. Total East/West Traffic 2. Bandwidth utilization history |

Virtual Infra Fabric Health Check Predefined Dashboard

| Goal | Trigger | Probes and Widgets |
|---|---|---|
| Find problems in physical or virtual infrastructure that affect workload connectivity | Presence of at-least one virtual Infra Manager in the BluePrint | 1. Hypervisor VLANs missing Fabric 2. Hypervisor Low MTU anomalies 3. Critical Services affected by VLAN misconfiguration 4. Hypervisor with inconsistent MTU 5. Hypervisor PNIC LAG Status |

Virtual Infra Redundancy Checks Predefined Dashboard

| Goal | Trigger | Probes and Widgets |
|---|---|--|
| Find single points of failure in physical or virtual infrastructure that affect high availability and available bandwidth for workloads | Presence of at-least one virtual Infra Manager in the BluePrint | Hypervisor VLANs missing Fabric, Hypervisor Low MTU anomalies, Critical Services affected by VLAN misconfiguration, Hypervisor with inconsistent MTU, Hypervisor PNIC LAG Status |

Cloning Analytics Dashboard

Instead of entering all details for a new dashboard, you can clone an existing one, give it a new name and customize it.

4.3.2.4 Creating Dashboard

Some probes and dashboards are automatically created giving you instant value without a learning curve. 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 (as of version 3.0).

1. From the blueprint, navigate to **Analytics > Dashboards**, then click **Create Dashboard** and 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 **Dashboards**, toggle on **Default**.
4. Add and/or *create widgets* to include in the dashboard.
5. Click **Create Dashboard** to create the dashboard and return to the list view. A large dashboard may take some time to create. You can check the status at the bottom of the screen under **Active Tasks**.

4.3.2.5 Editing Dashboard

Auto-created dashboards can be modified, but defaults should work in most cases.

1. From the list view (Analytics > Dashboards) or the details view, click the **Edit** button for the dashboard to edit.
2. Make your changes.
3. Click **Update** to stage the changes and return to the list view.

4.3.2.6 Deleting 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 list view (Analytics > Dashboards) or the details view, click the **Delete** button for the dashboard to delete.
2. If you would like all referenced widgets and probes that are exclusively referenced from this dashboard to be deleted for you so you don't have to delete them manually, check the checkbox. Deleting unnecessary widgets and probes frees up resources.
3. Click **Delete Dashboard** to stage the deletion and return to the list view.

4.3.3 Anomalies (IBA)

Blueprint Dashboard

The *blueprint dashboard* shows a summary of all anomalies including those generated by IBA probes. Clicking the **All Probes** gauge on the dashboard takes you to a list of anomalies (Analytics > Anomalies).

From the blueprint, navigate to **Analytics > Anomalies**. Any anomalies that are discovered by IBA probes are listed. 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.

4.3.4 Widgets

4.3.4.1 Widget Overview

Widgets generate data based on IBA *probes*. Depending on the type of widget, it returns either a total count of a particular type of anomaly, or it displays outputs generated from stages and processors in an IBA probe. Some widgets are created automatically (but they are not deleted automatically). Widgets can be viewed by themselves or they can be added to *analytics dashboards*. You can create custom widgets before or during dashboard creation.

From the blueprint, navigate to **Analytics > Widgets**.

1. Click **Analytics**

2. Click **Widgets**

Create Widget

☐ Show widget contents
☐ Anomalies Only

1-9 of 9 Page Size: 25

| <input type="checkbox"/> 0 selected | Name ▲ | Type | Properties | Updated by | Actions |
|-------------------------------------|---------------------------------|-------|--|-------------------|---------|
| <input type="checkbox"/> | Drained Switches Excess Traffic | Stage | Stage Drain traffic anomaly / excess_range | System 5 days ago | |
| <input type="checkbox"/> | External ECMP Imbalance | Stage | Stage ECMP Imbalance (External Interfaces) / system_tx_imbalance_count | System 5 days ago | |
| <input type="checkbox"/> | Fabric ECMP Imbalance | Stage | Stage ECMP Imbalance (Fabric Interfaces) / system_imbalance_count | System 5 days ago | |

Cloning Widgets

Instead of entering all details for a new widget, you can clone an existing one, give it a new name and customize it.

4.3.4.2 Creating Anomaly Heat Map Widget

Anomaly heatmap widgets count the anomalies from tagged IBA probes and stages.

1. From the blueprint, navigate to **Analytics > Widgets** and click **Create Widget**.
2. Select **Anomaly Heat Map** from the **Type** drop-down list and enter a name.
3. Enter row tags, column tags, and (optional) description.

4. Click **Create** to create the widget and return to the list view. Creating a large widget may take some time. You can check the status under the **Active Tasks** section at the bottom of the screen.

Creating Widget from Within a Probe

You can also create a widget from within a probe.

4.3.4.3 Creating Stage Widget from Widgets View

Stage widgets contain outputs from a stage of an IBA probe.

1. From the blueprint, navigate to **Analytics > Widgets** and click **Create Widget**.
2. Select **Stage** from the **Type** drop-down list and enter a name.
3. Select a probe and a stage, then customize the output as needed.
4. Click **Create** to create the widget and return to the list view. Creating a large widget may take some time. You can check the status under the **Active Tasks** section at the bottom of the screen.

4.3.4.4 Creating Stage Widget from Probes View

In addition to creating a stage widget from the widgets view, you can also create one while you are in the detail view of a probe.

1. From the blueprint, navigate to **Analytics > Probes** and select a probe.
2. Select a stage within the probe and click the **Create dashboard widget** button (right-side). The stage is preselected for you in the dialog that appears.
3. Configure the parameters as needed.
4. Click **Create** to create the widget and return to the detail view of the probe. The widget appears in the widgets list view (Analytics > Widgets) and when you create or update an analytics dashboard, the new widget appears as an option.

4.3.4.5 Editing Widget

Auto-created widgets can be modified, although defaults should work in most cases. When you edit widgets that are used in a dashboard you'll get the message 'This widget is currently in use from dashboard(s). Editing this widget will affect those dashboard(s)'.

1. From the list view (Analytics > Widgets) or the details view, click the **Edit** button for the widget to edit.
2. Make your changes.
3. Click **Update** to stage the changes and return to the list view.

4.3.4.6 Deleting Widget

A widget that is used in a dashboard cannot be deleted.

1. From the list view (Analytics > Widgets) or the details view, click the **Delete** button for the widget to delete.
2. Click **Delete Widget** to stage the deletion and return to the list view.

4.3.5 Probes

4.3.5.1 IBA Probes Overview

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, e.g. `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, e.g. 900, or as an expression, e.g. `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. See the example of *Service Data Collector* for more information.

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 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 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 the 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.

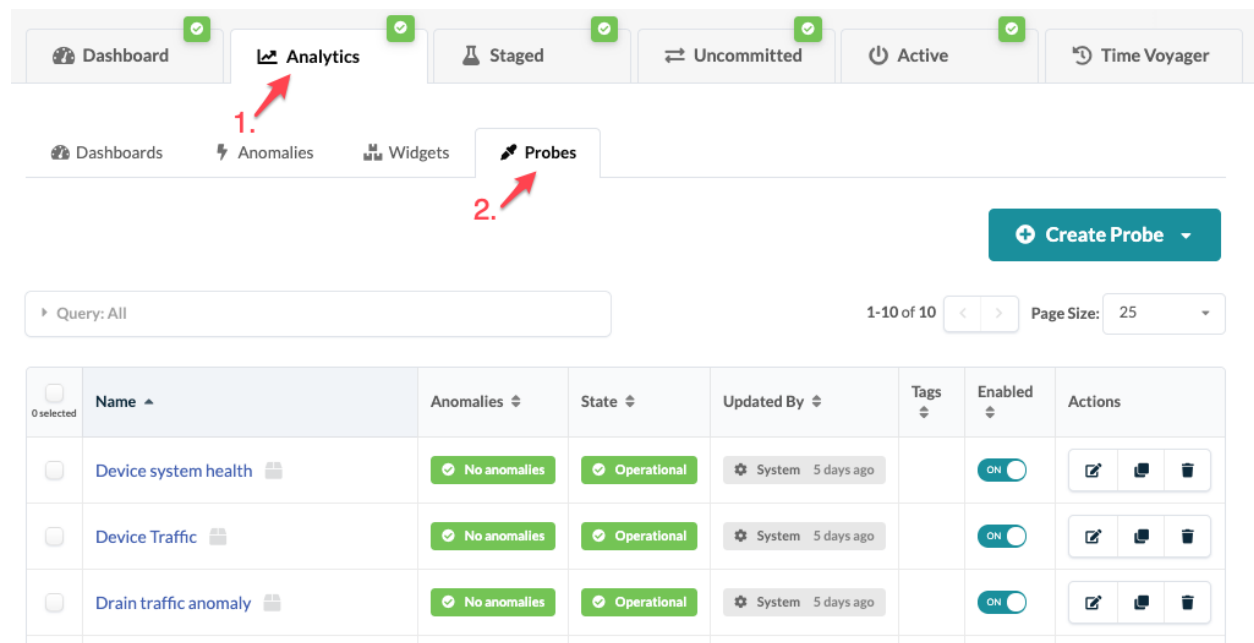
Note: The schema presented here is only for communicating the requirements and engineering is free to choose any way that accomplishes stated use cases.

Collector Processors `additional_properties` specified in collector processors' configuration can be accessed using the special `context.namespace`. E.g. 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, e.g. any grouping processor, it's no longer available.

From the blueprint, navigate to **Analytics / Probes**.



| Name | Anomalies | State | Updated By | Tags | Enabled | Actions |
|-----------------------|--------------|-------------|-------------------|------|---------|------------------------|
| Device system health | No anomalies | Operational | System 5 days ago | | ON | [Edit] [Copy] [Delete] |
| Device Traffic | No anomalies | Operational | System 5 days ago | | ON | [Edit] [Copy] [Delete] |
| Drain traffic anomaly | No anomalies | Operational | System 5 days ago | | ON | [Edit] [Copy] [Delete] |

4.3.5.2 Importing Probe

1. From the blueprint, navigate to **Analytics > Probes**, then click **Create Probe** and select **Import Probes** from the drop-down list.
2. Either click **Choose Files** and navigate to the file(s) on your computer, or drag and drop the file(s) from your computer into the dialog window.
3. Click **Import** to import the probe and return to the list view.

4.3.5.3 Exporting Probe

1. From the blueprint, navigate to **Analytics > Probes**, then click the name of the probe to export.

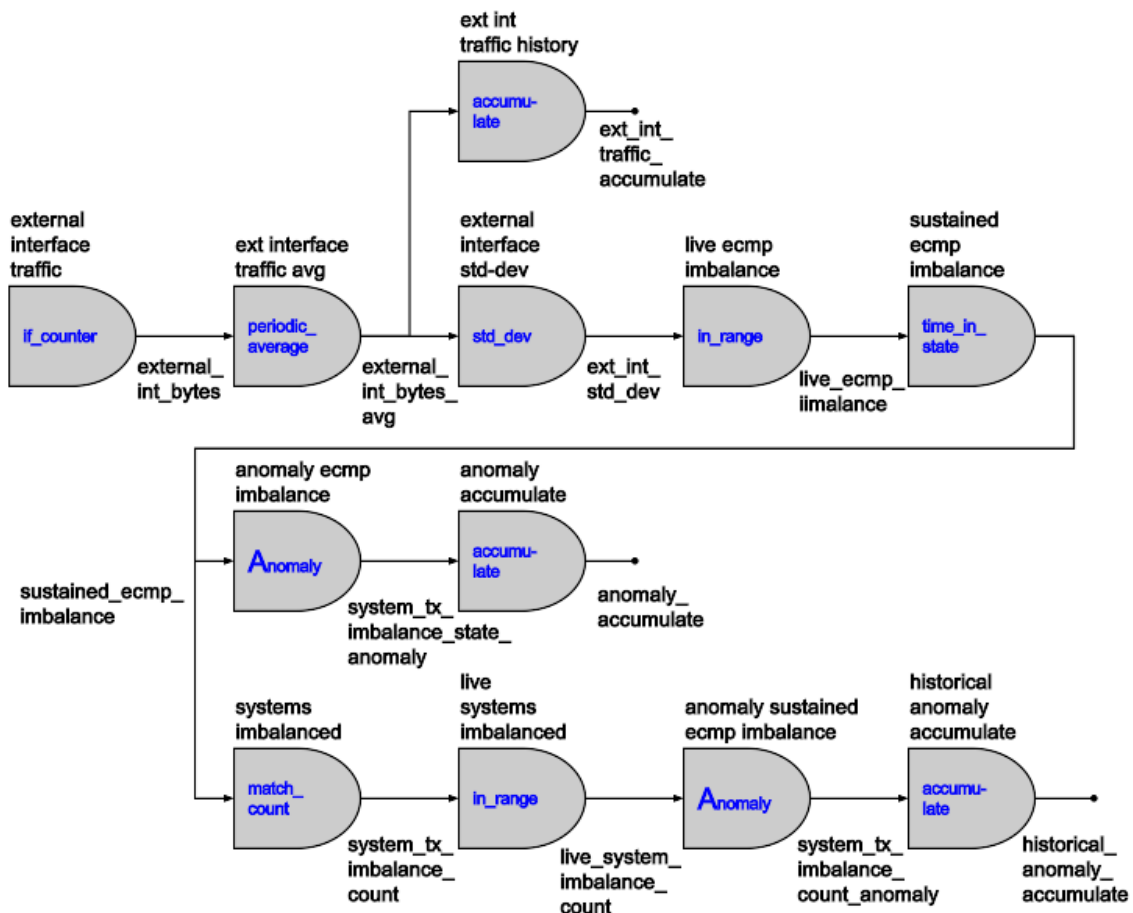
2. Click the **Export** button (top-right) to see a preview of the file that will be exported.
3. To copy the contents, click **Copy**, then paste it.
4. To download the json file to your local computer, click **Save as File**.
5. When you've copied and/or downloaded the file, click the **X** to close the dialog.

4.3.5.4 Instantiating Predefined Probe

The Two stage L3 Clos reference design comes with a set of predefined probes that can be instantiated via the web interface or via the facade API at /predefined_probes (Platform > Developers > Two stage L3 Clos). For the exact input and output parameters necessary for these probes, please refer to the API documentation.

1. From the blueprint, navigate to **Analytics > Probes**, then click **Create Probe** and select **Instantiate Predefined Probe** from the drop-down list.
2. Select a predefined probe from the drop-down list. For more information about some of the predefined probes, see the links below.
3. Configure the probe to suit your anomaly detection requirements.
4. Click **Create** to instantiate the probe and return to the list view.

ECMP Imbalance (External Interfaces) Probe



Found at `/predefined_probes/external_ecmp_imbalance`

It first identifies all the interfaces on all deployed and operational devices that are attached to external routers. It then collects samples for each, generates time series and calculates average traffic across configurable time interval. It then calculates imbalance between traffic averages for these interfaces on each device by calculating standard deviation. It then checks if this imbalance is within acceptable range (which is also configurable) and if not, how long has it been outside acceptable range and if this time exceeds configurable interval it raises an anomaly. It also creates time series for this anomaly so that one can observe recent history. Probe also calculates how many systems are imbalanced in total and if this number is out of configurable range, it raises the system level anomaly and keeps track about the history for this system level anomaly.

“external interface traffic” processor

Purpose: wires in interface traffic samples (measured in transmitted bytes per second) from each interface connected to external router.

Outputs of “external interface traffic” processor

`‘external_int_bytes’`: set of traffic samples (for each external router 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).

“external interface traffic avg” processor

Purpose: Calculate average traffic during period specified by `average_period` facade parameter. Unit is bytes per second.

Outputs of “external interface traffic avg” processor

`‘external_int_bytes_avg’`: set of traffic average values (for each external router 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).

“ext int traffic history” processor

Purpose: create recent history time series out of traffic samples from the `external_int_bytes` output. 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). Samples unit is bytes per second.

Outputs of “ext int traffic history” processor

`ext_int_traffic_accumulate`: set of traffic samples time series (for each external router facing interface). Each set member has the following keys to identify it: `label` (human readable name of the leaf), `system_id` (id of the system, usually serial number), `interface` (name of the interface). Samples unit is bytes per second.

“external interface std-dev” processor

Purpose: calculate standard deviation for a set consisting of traffic averages for each external router facing interface on a given system. Grouping per system is achieved using ‘group_by’ property set to ‘system_id’ and ‘label’.

Outputs of “external interface std-dev” processor

ext_int_std_dev: set of values, each indicating standard deviation (as a measure of ECMP imbalance) for traffic averages for each external router facing interface on a given system. Each set member has ‘system_id’ and ‘label’ key to identify system whose ECMP imbalance the value represents.

“live ecmp imbalance” processor (external router)

Purpose: Evaluate if standard deviation between external router 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).

Outputs of “live ecmp imbalance” processor (external router)

‘live_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 leaf is within acceptable range. Each set member has system_id key to identify system whose ECMP imbalance the value represents.

“sustained ecmp imbalance” processor (external router)

Purpose: Evaluate if standard deviation between external router 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.

Outputs of “sustained ecmp imbalance” processor (external router)

‘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.

“anomaly ecmp imbalance” processor

Purpose: Export sustained ecmp imbalance when true as an anomaly for each system.

Outputs of “anomaly ecmp imbalance” processor

‘system_tx_imbalance_state_anomaly’: 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.

“anomaly accumulate” processor

Purpose: Create time series showing ecmp anomaly being raised and cleared for each system under consideration. This time series may contain up to ‘anomaly_history_count’ anomaly state changes.

Outputs of “anomaly accumulate” processor

‘anomaly_accumulate’: Time series showing ecmp anomaly being raised and cleared for each system under consideration. This time series may contain up to ‘anomaly_history_count’ anomaly state changes.

“systems imbalanced count” processor

Purpose: Count how many systems have external ecmp imbalance anomaly true at any instant in time.

Outputs of “systems imbalanced count” processor

‘systems_tx_imbalance_count’: Number of systems with external ecmp imbalance.

“live systems imbalanced” processor

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

Outputs of “live systems imbalanced” processor

‘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

“anomaly sustained ecmp imbalance” processor

Purpose: Export as anomaly when the number of imbalanced systems is not within acceptable range, where acceptable range is defined as less than ‘max_systems_imbalanced’ value.

Outputs of “anomaly sustained ecmp imbalance” processor

‘system_tx_imbalance_count_anomaly’: Boolean indicating if the number of imbalanced systems is within accepted range, i.e. less than ‘max_systems_imbalanced’ which is a facade parameter

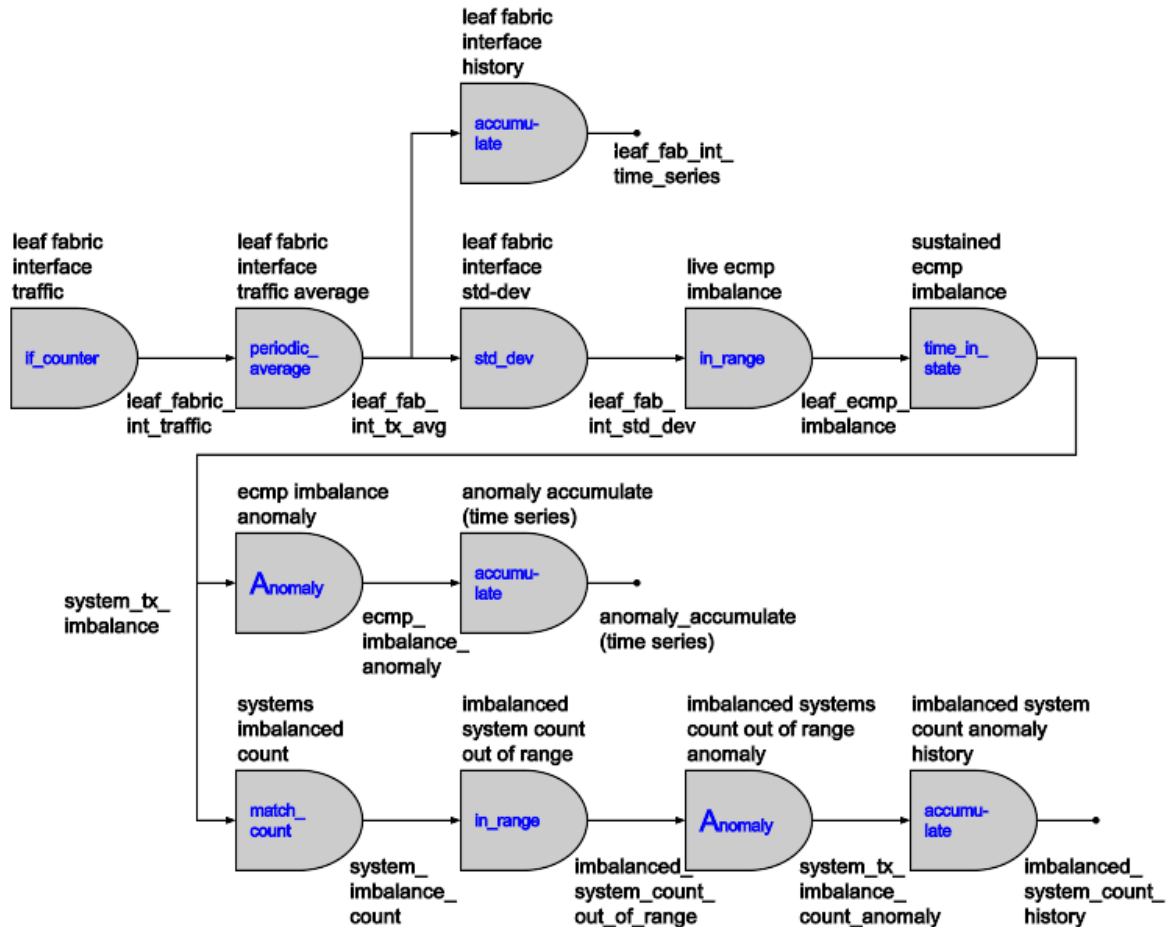
“historical anomaly accumulate” processor

Purpose: Create time series showing imbalanced system count out of range anomaly being raised and cleared. This time series may contain up to ‘system_imbalance_history_count’ anomaly state changes.

Outputs of “historical anomaly accumulate” processor

‘historical_anomaly_accumulate’: time series showing imbalanced system count out of range anomaly being raised and cleared. This time series may contain up to ‘system_imbalance_history_count’ anomaly state changes.

ECMP Imbalance (Fabric Interfaces) Probe



Found at /predefined_probes/fabric_ecmp_imbalance

It first identifies all the fabric interfaces on all deployed and operational leafs. Fabric interfaces are defined as those facing spines, which also satisfy the constraint that they are deployed and operational. It then collects samples for each, generates time series and calculates average traffic across configurable time interval. It then calculates imbalance between traffic averages for fabric interfaces on each leaf by calculating standard deviation. It then checks if this imbalance is within acceptable range (which is also configurable) and if not, how long has it been outside acceptable range and if this time exceeds configurable interval it raises anomaly. It also creates time series for this anomaly so that one can observe recent history. Probe also calculates how many systems are imbalanced in total and if this number is out of configurable range, it raises the system level anomaly and keeps track about the history for this system level anomaly.

“leaf fabric interface traffic” processor

Purpose: wires in interface traffic samples (measured in bytes per second) from each spine facing interface on each leaf.

Outputs of “leaf fabric interface traffic” processor

‘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).

“leaf fabric interface traffic average” processor

Purpose: Calculate average traffic during period specified by average_period facade parameter. Unit is bytes per second.

Outputs of “leaf fabric interface traffic average” processor

‘leaf_fab_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 history” processor

Purpose: create recent history time series out of traffic samples from the leaf_fabric_int_traffic output. 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). Samples unit is bytes per second.

Outputs of “leaf fabric interface history” processor

leaf_fab_int_time_series: set of traffic samples time series (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). Samples unit is bytes per second.

“leaf fabric interface std-dev” processor

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’.

Outputs of “leaf fabric interface std-dev” processor

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.

“live ecmp imbalance” processor (spine)

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).

Outputs of “live ecmp imbalance” processor (spine)

‘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” processor (spine)

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.

Outputs of “sustained ecmp imbalance” processor (spine)

‘system_tx_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 has been outside acceptable range for more than specified period of time. Each set member has system_id key to identify leaf whose ECMP imbalance the value represents.

“ecmp imbalance anomaly” processor

Purpose: Export sustained ecmp imbalance when true as an anomaly for each system.

Outputs of “ecmp imbalance anomaly” processor

‘ecmp_imbalance_anomaly’: 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 has been outside acceptable range for more than specified period of time. Each set member has system_id key to identify leaf whose ECMP imbalance the value represents.

“anomaly acumulate” processor

Purpose: Create time series showing ecmp anomaly being raised and cleared for each system under consideration. This time series may contain up to ‘anomaly_history_count’ anomaly state changes.

Outputs of “anomaly acumulate” processor

‘anomaly_accumulate’: Time series showing ecmp anomaly being raised and cleared for each system under consideration. This time series may contain up to ‘anomaly_history_count’ anomaly state changes.

“systems imbalanced count” processor

Purpose: Count how many systems have ecmp imbalance anomaly true at any instant in time.

Outputs of “systems imbalanced count” processor

‘systems_imbalance_count’: Number of systems with ecmp imbalance.

“imbalanced system count out of range” processor

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

Outputs of “imbalanced system count out of range” processor

‘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

“imbalanced system count out of range anomaly” processor

Purpose: Export as anomaly when the number of imbalanced systems is not within acceptable range, where acceptable range is defined as less than ‘max_systems_imbalanced’ value.

Outputs of “imbalanced system count out of range anomaly” processor

‘imbalanced_system_count_out_of_range_anomaly’: Boolean indicating if the number of imbalanced systems is within accepted range, i.e. less than ‘max_systems_imbalanced’ which is a facade parameter

“imbalanced system count anomaly history” processor

Purpose: Create time series showing imbalanced system count out of range anomaly being raised and cleared. This time series may contain up to ‘system_imbalance_history_count’ anomaly state changes.

Outputs of “imbalanced system count anomaly history” processor

‘imbalanced_system_count_anomaly_history’: time series showing imbalanced system count out of range anomaly being raised and cleared. This time series may contain up to ‘system_imbalance_history_count’ anomaly state changes.

EVPN Probes

To troubleshoot EVPN routes, three pre-defined EVPN probes are available:

- *EVPN VXLAN Type-3 Route Validation*
- *EVPN VXLAN Type-5 Route Validation*

- *VXLAN Flood List Validation*

You can use these probes for troubleshooting EVPN routing.

From the Probes view of your Analytics Blueprint, click Instantiate Predefined Probe button, then find any of these probes in the list that pops up, then click Create.

EVPN dashboard is NOT auto-enabled. Customers must manually enable it as needed. The dashboard, if enabled, auto enables two probes:

- **VXLAN Flood Probe:** To monitor all VNs on all devices
- **EVPN VXLAN Type-3 Probe:** This probe is created with empty “monitored_vn” list, i.e. the probe monitors no VN at all. Customers need to manually update this configuration option as needed.

Note: EVPN Type 5 Routes Probe is not part of EVPN dashboard.

The following parameters in these probes can be configured:

- **Probe Label:** Name to identify the probe.
- **Anomaly Time Window :** Average period duration for interface counters. Six minute default for VXLAN Flood List and 11 minute default for EVPN VXLAN Type-3 and EVPN VXLAN Type-5.
- **Anomaly Threshold:** If routes are missing for more than or equal to percentage of Anomaly Time Window, an anomaly will be 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 will be raised.
- **Collection period:** All these probes are polling based so has polling period. Default polling period is **10 minutes** for VXLAN Type-3 and VXLAN Type-5 probes, but **5 minutes** for VXLAN Flood List probe. These can be modified at creation time or can be updated later.
- **Monitored VNs:** Only for EVPN VXLAN Type-3 Probe. Specify the virtual networks to be monitored. Either list of desired VNs e.g. “1-3,6,8,10-13” or “*” to monitor all VNs.

Each of these Probe has 3 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.
- **Unintended:** No expectations are rendered for this route. No anomaly is raised

Processors

Once the probe has been created, you can click it to see a breakdown of the probe. Click the desired option on the left side, you can select the processors:

- **Routes:** Purpose: Collects specific routes of route table from all leafs. Each route has a “state” which is either Missing, Expected, or Unintended. Shows route table VXLAN Flood List, or EVPN Type 3 or EVPN Type 5.
- **Missing Routes :** This processor consumes in “Number of missing routes”. Calculates number of missing routes. For all probes, the number of missing routes are calculated per leaf per leaf. For VXLAN Floodlist and EVPN Type 3 probes, the probe checks for routes per Virtual Network. For EVPN Type 5 probe, the probe checks for routes per route target.
- **Sustained Missing Routes:** Raises an anomaly per System and VNI when missing routes are present longer than Anomaly Time Window.
- **Total number of sustained anomalies:** Show total number of sustained anomalies.

Search stages...

VXLAN Flood List Routes

VXLAN Flood List Table

Number of missing routes

Missing Routes

Missing Routes

Sustained Missing Routes

Sustained Anomalies ▲ 14

Total number of sustained anomalies ...

Number of Sustained Anomalies

Stage: VXLAN Flood List Table Table

VXLAN Flood List Routing Table.

Query: All

>_ 1-25 of 44 < >

Page Size: 25

| System ID | Endpoint | Next Hop | Vni | State | Telemetry Service Status | Updated |
|-------------------------------|-------------------------------|-----------|-------|----------|--------------------------|-------------------|
| 5254000C4449 leaf2 Leaf | 5254006DEFA5 leaf3 Leaf | 10.0.0.8 | 10000 | Expected | No warnings | a few seconds ago |
| 5254000C4449 leaf2 Leaf | 5254003857CA leaf1 Leaf | 10.0.0.6 | 10000 | Missing | No warnings | a few seconds ago |
| 5254000C4449 leaf2 Leaf | 525400BAE07E leaf5 Leaf | 10.0.0.10 | 10000 | Expected | No warnings | a few seconds ago |
| 5254000C4449 leaf2 Leaf | 52540034B75B leaf4 Leaf | 10.0.0.9 | 10000 | Expected | No warnings | a few seconds ago |
| 5254000C4449 | 5254006DEFA5 | | | | | |

VXLAN Flood List Validation

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 & alerts if expected entries are missing on any device. This probe will monitor all Virtual Networks from all devices.

Default collection period for this probe is **5 minutes**. It is not recommended to change the default collection interval.

EVPN VXLAN Type-3 Route Validation

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 & alerts if expected routes are missing on any device.

This probe allows customers at creation time to specify a list of Virtual Networks for which routes are collected.

Note: The maximum number of Virtual Networks in the list is limited to 10. However customers are allowed to specify "*" in which case all VNs are monitored.

Warning: Customers must exercise caution if they choose "*" (i.e. all VNs) as the monitored VN list due to high cpu/memory/network I/O overhead associated with BGP routing table iteration on the device side.

EVPN VXLAN Type-5 Route Validation

This probe validates EVPN Type-5 routes on every leaf in the network. It collects appropriate telemetry data; compares it to the set of Type-5 routes expected to be present & alerts if expected routes are missing on any device.

Note: If enabled it will monitor all VNs from all devices. It does not provide the "monitored VN list" configuration option like VXLAN Type-3 probe.

Instantiate Predefined Probe

Predefined Probe *

VXLAN Flood List Validation

Probe Label *

VXLAN Flood List Validation

Anomaly Time Window (in secs)

6 minutes

Anomaly Threshold (in %)

100

If routes are missing for more than or equal to percentage of Anomaly Time Window, an anomaly will be raised.

Collection period

300

Telemetry collection interval in seconds.

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.

Unintended: There are no expectations rendered (by AOS) for this route.

☐ Create Another?
 Create

Warning: Customers must exercise caution if they choose * (i.e. all VNs) as the monitored VN list due to high cpu/memory/network I/O overhead associated with BGP routing table iteration on the device side.

External Routes Probe

To troubleshoot external network connectivity problems, this predefined IBA probe automatically activates the collection of received or advertised routes across all BGP sessions established with external routers. This predefined IBA probe uses IBA Ingestion Filters introduced in version 3.1.

To start troubleshooting external routes follow these steps:

1. From the blueprint, navigate to **Analytics > Probes**, then click **Create Probe**, and select **Instantiate Predefined Probe** from the drop-down list.
2. Select “External routes”

Instantiate Predefined Probe

Predefined Probe *

EVPN VXLAN Type-3 Route Validation

Probe Label *

EVPN VXLAN Type-3 Route Validation

Anomaly Time Window (in secs)

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

600

Telemetry collection interval in seconds.

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".

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.

Unintended: There are no expectations rendered (by AOS) for this route.

☐ Create Another?

Instantiate Predefined Probe

Predefined Probe *

EVPN VXLAN Type-5 Route Validation

Probe Label *

EVPN VXLAN Type-5 Route Validation

Anomaly Time Window (in secs)

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

600

Telemetry collection interval in seconds.

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.

☐ Create Another?
 Create

Instantiate Predefined Probe

Probe *

External routes

Probe Label *

External routes

AFI

Address Family Identifiers

Type

Security Zone (VRF)

Prefix

IPv4 or IPv6 network address, specified in the form of network/prefixlen. Host routes OK in format 192.168.1.1, or a05:fab::1

More-specific prefixes mask

Match more-specific prefixes from a parent prefix, up until le_mask prefix len. Range is 0-32 for IPv4, 0-128 for IPv6. If not specified, implies the prefix-list entry should be an exact match. This option can be optionally be used in combination with ge_mask. le_mask must be longer than the subnetprefix length. If le_mask and ge_mask are both specified, then le_mask must be greater than ge_mask.

Less-specific prefixes mask

Match less-specific prefixes from a parent prefix, up from ge_mask to the prefix length of the route. Range is 0-32 for IPv4, 0-128 for IPv6. If not specified, implies the prefix-list entry should be an exact match. This option can be optionally be used in combination with le_mask. ge_mask must be longer than

Generate a probe to display external routes

This probe shows received and advertised routes across all BGP sessions established with external routers.

3. The predefined probe has the following parameters that you can configure at creation time or update subsequently:

- **AFI:** IPv4 or IPv6
- **Type:** Either “advertised-routes” or “received-routes”
- **Security 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.

When you click **Create**, external routes and present routes are collected in a single stage output table (mixing received, used and advertised routes).

Processor: external_routes Extensible Service Data Collector

| Data Type | table_dss |
|------------------|---|
| Graph Query | match((node('system', name='system', deploy_mode=is_in(['deploy', 'drain']), system_id=not_none()).out('hosted_interfaces').node('interface', if_name=not_none()).out('link').node('link', link_type=is_in(['ethernet', 'aggregate_link']).in('link').node('interface').in('hosted_interfaces').node('system', role='external_router')), distinct(['system']) |
| Ingestion filter | network → prefix 69.16.128.0/18 route_status → equals "received-routes" afi → equals "ipv4" vrf → equals "all" |
| Keys | |
| Value Map | 1 → up |
| Service name | bgp_route |
| System ID | system.system_id |
| Execution count | 1 |
| Service input | "" |
| Service interval | 120 |

Device Traffic Probe (aka Headroom probe)

Note: As of version 3.3.0, **Headroom Probe** is renamed **Device Traffic probe**. This probe is enabled automatically after blueprint creation.

One of the most useful default probes is the Device Traffic Probe (previously known as Headroom Probe), which provides helpful insights about link capacity between two points in the network. This probe is now extended to provide multiple interface counters (rx, tx, discard, errors etc.) for all managed devices. By default, All counters are collected realtime every 5 seconds. The probe provides utilization on a per port and aggregated utilization per system basis, and raises an anomaly in case of rule violation.

To manually enable this probe, navigate to **Analytics > Probes > Create Probe > Instantiate Predefined Probe**, and select **Device Traffic** as per below image.

The screenshot shows the 'Instantiate Predefined Probe' dialog box. The 'Predefined Probe' dropdown is set to 'Device Traffic'. The 'Probe Label' field contains 'Device Traffic'. The 'Interface counters average period' dropdown is set to '2 Minutes'. Below this, there is a checkbox for 'Enable interface counters history' which is checked, with the text 'Maintain historical interface counters data'. The 'Interface counters history retention period' dropdown is set to '30 Days'. Below this, there is a checkbox for 'Enable system counters history' which is checked, with the text 'Maintain historical system interface counters data'. The 'System interface counters history retention period' dropdown is set to '30 Days'. At the bottom right, there is a 'Create' button and a 'Create Another?' checkbox.

The predefined probe has the following parameters that you can configure at creation time or update subsequently:

- **Probe Label:** Custom name for the probe.
- **Interface counters average period:** The average period duration for interface counters in seconds. **Default value** = 2 Minutes.
- **Enable Interface counters history:** Maintain historical interface counters data.
- **Interface counters history retention period:** Duration to maintain historical interface counters data. **Default value** = 30 Days.
- **Enable system counters history:** Maintain historical system interface counters data.
- **System interface counters history retention period:** Duration to maintain historical system interface counters data. **Default value** = 30 Days.

Warning: User can change the probe inputs but if user changes the probe processor then the probe is not a **predefined** probe anymore and the traffic layer view will disappear. See [Topology Selection Views](#) for more information about traffic layer view.

Once the probe has been created, you can click it to see a breakdown of the probe. Click the desired option on the left side, you can select the processors Live Interface Counters, System Interface Counters.

Search stages...

Stage: Average Interface Counters Table Persisted 30 days / 624.32 MB

Average interface counter data

Real Time ☐ Time Series

Query: All

| System ID | Interface | Port Speed | Average TX Utilization | Average RX Utilization | Average Transmitted unicast packets per second | Average Transmitted broadcast packets per second | Average Transmitted multicast packets per second | Average Transmitted bits per second | Average Transmitted error packets per second | Average Transmitted discard packets per second | Average Received unicast packets per second |
|--|------------|------------|------------------------|------------------------|--|--|--|-------------------------------------|--|--|---|
| OE15A7450000 single-leaf-rack-121-leaf1 Leaf | Ethernet1 | 10Gbps | 29% | 29% | 731kpps | 731kpps | 731kpps | 2.99Gbps | 731kpps | 731kpps | 731kpps |
| OE15A7450000 single-leaf-rack-121-leaf1 Leaf | Ethernet10 | 10Gbps | 29% | 29% | 731kpps | 731kpps | 731kpps | 2.99Gbps | 731kpps | 731kpps | 731kpps |
| OE15A7450000 single-leaf-rack-121-leaf1 Leaf | Ethernet11 | 10Gbps | 29% | 29% | 731kpps | 731kpps | 731kpps | 2.99Gbps | 731kpps | 731kpps | 731kpps |

Live Interface Counters processor

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.

Outputs of “Live Interface Counters” processor

‘Average Interface Counters’: set of interface counters samples (for each port of each managed device) based on specified average time with historical data.

Search stages...

Stage: Average Interface Counters Table Persisted 30 days / 789.92 MB

Average interface counter data

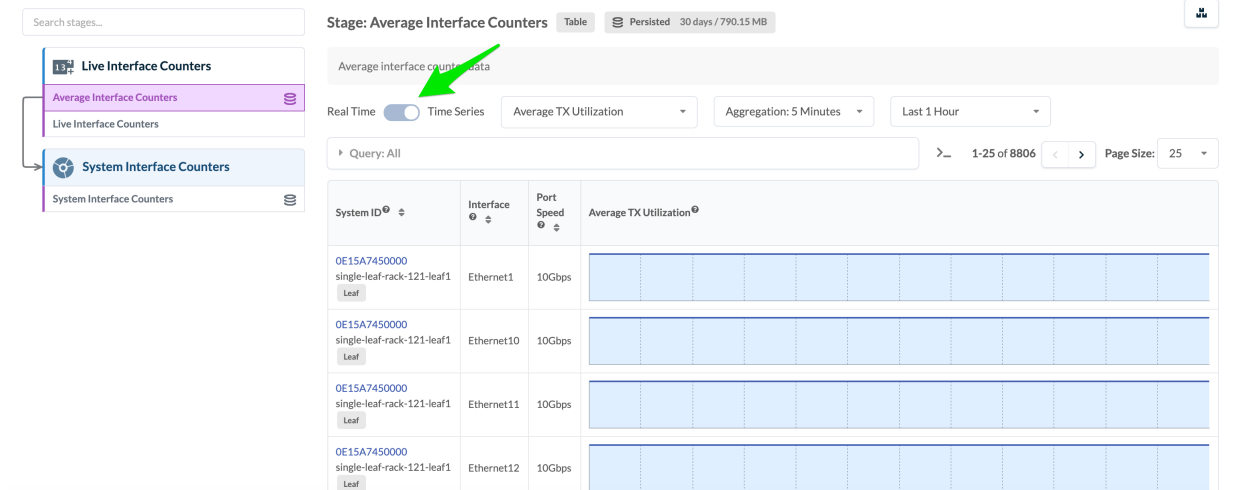
Real Time ☐ Time Series

Query: All

| System ID | Interface | Port Speed | Average TX Utilization | Average RX Utilization | Average Transmitted unicast packets per second | Average Transmitted broadcast packets per second | Average Transmitted multicast packets per second | Average Transmitted bits per second | Average Transmitted error packets per second | Average Transmitted discard packets per second | Average Received unicast packets per second |
|--|------------|------------|------------------------|------------------------|--|--|--|-------------------------------------|--|--|---|
| OE15A7450000 single-leaf-rack-121-leaf1 Leaf | Ethernet1 | 10Gbps | 29% | 29% | 731kpps | 731kpps | 731kpps | 2.99Gbps | 731kpps | 731kpps | 731kpps |
| OE15A7450000 single-leaf-rack-121-leaf1 Leaf | Ethernet10 | 10Gbps | 29% | 29% | 731kpps | 731kpps | 731kpps | 2.99Gbps | 731kpps | 731kpps | 731kpps |
| OE15A7450000 single-leaf-rack-121-leaf1 Leaf | Ethernet11 | 10Gbps | 29% | 29% | 731kpps | 731kpps | 731kpps | 2.99Gbps | 731kpps | 731kpps | 731kpps |

Note: Use **Query All** to filter outputs based on any value in the table.

Enable **Real Time** to filter real time outputs based on any value in the table for a particular aggregation time in history.



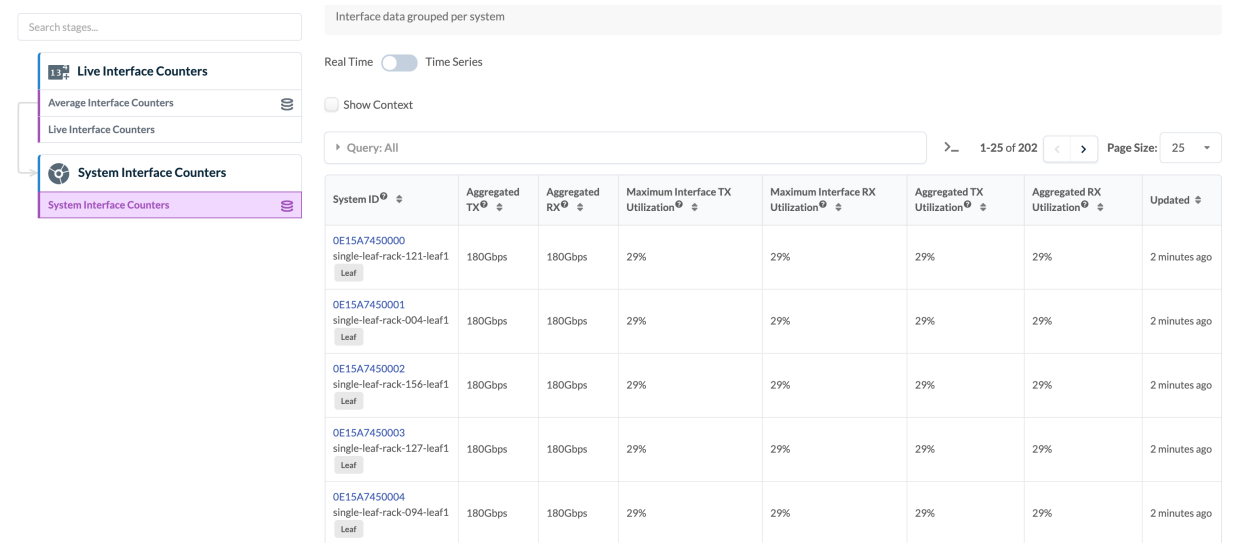
‘Live Interface Counters’: set of live interface counters samples (for each port of each managed device).

System Counters processor

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.

Outputs of “System Counters” processor

‘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 %.



Note: System Level RX/TX Calculation: It aggregate the Tx/RX of all the device interfaces that are “up”.

Note: Max Interface RX/TX Calculation: This is the device interface with the highest Rx and the device interface with highest Tx.

Traffic between a Source and Destination

To visualize traffic between a particular source and destination, first click on System Interface Counters, Check the Box **Show Context**, and Choose Source and Destination.

The screenshot shows the 'System Interface Counters' page in the Apstra interface. The 'Show Context' checkbox is highlighted with a green arrow. The table below shows the aggregated data for two leaf nodes.

| System ID | Aggregated TX | Aggregated RX | Maximum Interface TX Utilization | Maximum Interface RX Utilization | Aggregated TX Utilization | Aggregated RX Utilization | Updated |
|--|---------------|---------------|----------------------------------|----------------------------------|---------------------------|---------------------------|--------------|
| OE15A7450000 single-leaf-rack-121-leaf1 Leaf | 180Gbps | 180Gbps | 29% | 29% | 29% | 29% | a minute ago |
| OE15A7450001 single-leaf-rack-004-leaf1 Leaf | 180Gbps | 180Gbps | 29% | 29% | 29% | 29% | a minute ago |

The probe uses different colours to describe the remaining link capacity, where green means plenty of capacity and red means that our link is running out of capacity.

The following capture displays an example of a 100G with 70.1Gbps available between a leaf and another leaf showing two available paths via spine1 and spine2.

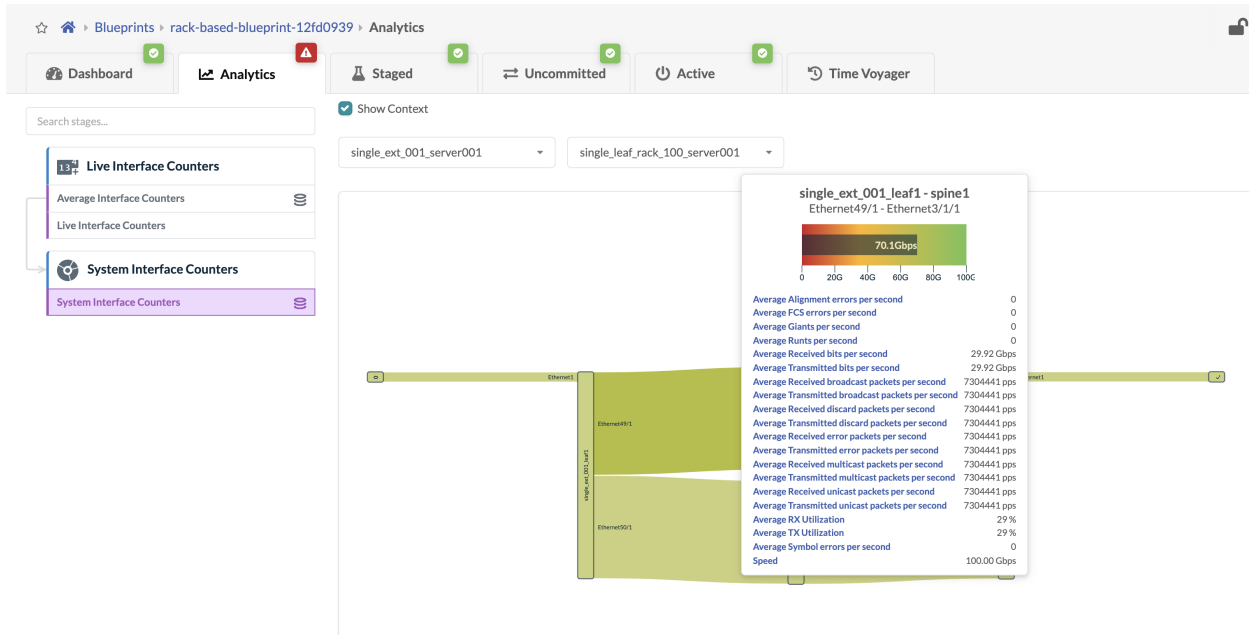
The screenshot shows the 'Analytics' page in the Apstra interface. The 'Show Context' checkbox is checked. The capture shows a traffic flow between two leaf nodes, with a 70.1Gbps link utilization. The capture details are as follows:

| Source | Destination | Link Utilization |
|----------------------|----------------------------|------------------|
| single_ext_001_leaf1 | single_leaf_rack_026_leaf1 | 70.1Gbps |

The capture also shows the following statistics:

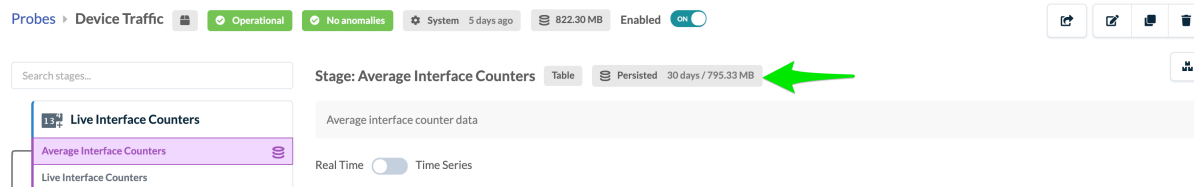
| Statistic | Value |
|--|-------------|
| Average Alignment errors per second | 0 |
| Average FCS errors per second | 0 |
| Average Giants per second | 0 |
| Average Runt errors per second | 0 |
| Average Received bits per second | 29.93 Gbps |
| Average Transmitted bits per second | 29.93 Gbps |
| Average Received broadcast packets per second | 7306951 pps |
| Average Transmitted broadcast packets per second | 7306951 pps |
| Average Received discard packets per second | 7306951 pps |
| Average Transmitted discard packets per second | 7306951 pps |
| Average Received error packets per second | 7306951 pps |
| Average Transmitted error packets per second | 7306951 pps |
| Average Received multicast packets per second | 7306951 pps |
| Average Transmitted multicast packets per second | 7306951 pps |
| Average Received unicast packets per second | 7306951 pps |
| Average Transmitted unicast packets per second | 7306951 pps |
| Average RX Utilization | 29% |
| Average TX Utilization | 29% |
| Average Symbol errors per second | 0 |
| Speed | 100.00 Gbps |

The next example displays capacity available between two endpoints in a network. It is showing an example of a 100G with 70.1Gbps available between two servers.



Probe Disk Consumption

The web interface displays the amount of disk space utilized by each probe. To view the disk space utilized by probe, go to specific processor output.



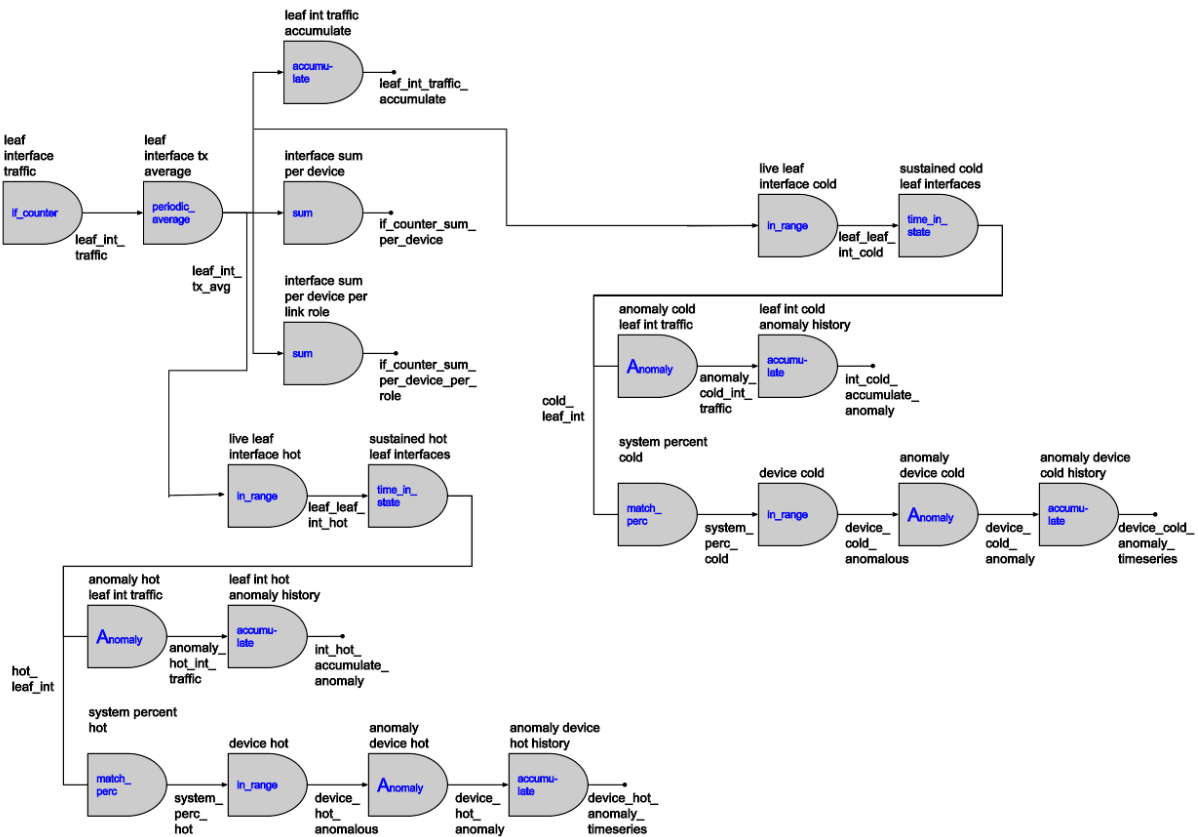
Warning: In order to ensure there is enough controller disk space to store the probe for the desired retention time, please work with the Support team. With insufficient disk space, older telemetry data files will automatically be deleted as a safety mechanism.

The Device Traffic probe first identifies all the links (across all ECMP paths) that may carry the traffic between the two nodes (source and target). It then calculates headroom for each of them. Headroom is defined as difference between the link capacity and the actual traffic. It then calculates minimum headroom per each path as the smallest headroom across all the links in the path. The semantic of this value is that if you send additional traffic from the source node (new app) which is smaller than path minimum headroom the path will be able to support it, otherwise there will be packet loss. But with ECMP you don't know which path the traffic will take. To address that, probe then calculates minimum headroom across all ECMP paths. The semantic of this value is that if you send additional traffic from the source which is less then this value, it will make it to the target without the packet loss regardless of which path is taken. Probe also calculates the maximum headroom across all the paths, which is the maximum value among all the minimum path headroom values, for each ECMP path. The semantic of this value is that if you add more traffic from the source (new app), there is NO path that can support it therefore you are guaranteed packet loss. And if you add

more traffic from the source which is between minimum and minimum headroom it may or may not be able to make it without a loss, i.e. the performance is unpredictable. The smaller the difference between these two values is, the more predictable the system is, which is why proper ECMP balancing is so important. And all of these calculations are up to date and in sync in presence of any topology change (addition/removal of leafs/links).

For more information, see [IBA](#).

Hot/Cold Interface Counters Probe



Found at `/predefined_probes/fabric_hotcold_ifcounter`

It first identifies all the fabric interfaces on all deployed and operational leafs. Fabric interfaces are defined as those facing spines, which also satisfy the constraint that they are deployed and operational. It then collects samples for each, generates time series and calculates average traffic across configurable time interval. It then raises hot/cold anomaly per interface if it has been cold/hot (defined as traffic being below/above a specified hot/cold threshold) longer then configurable period of time. It also creates time series showing these anomalies being raised and cleared. It then checks if the number of hot/cold interfaces per device is above specified threshold in which case it generates device hot/cold anomaly and then creates time series allowing user to inspect the recent history of these device level anomalies being raised/cleared. In addition, this probe provides the total traffic per device and per device and interface role.

“leaf interface traffic” processor

Purpose: wires in interface traffic samples (measured in bytes per second) from each spine facing interface on each leaf.

Outputs of “leaf interface traffic” processor

‘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’).

“leaf interface tx average” processor

Purpose: Calculate average traffic during period specified by average_period facade parameter. Unit is bytes per second.

Outputs of “leaf interface tx average” processor

‘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’).

“leaf int traffic accumulate” processor

Purpose: create recent history time series out of traffic samples from the leaf_int_traffic output. 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). Samples unit is bytes per second.

Outputs of “leaf int traffic accumulate” processor

leaf_int_traffic_accumulate: set of traffic samples 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) role (role of the interface, such as ‘fabric’). Samples unit is bytes per second.

“live leaf interface hot” processor

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).

‘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.

“live leaf interface cold” processor

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).

‘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.

“sustained hot leaf interface” processor

Purpose: 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.

Outputs of “sustained hot leaf interface” processor

‘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.

“sustained cold leaf interface” processor

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.

Outputs of “sustained cold leaf interface” processor

‘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.

“anomaly hot leaf int traffic” processor

Purpose: Export hot leaf interface status when true as an anomaly.

Outputs of “anomaly hot leaf int traffic” processor

‘anomaly_hot_int_traffic’: 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.

“anomaly cold leaf int traffic” processor

Purpose: Export cold leaf interface status when true as an anomaly.

Outputs of “anomaly cold leaf int traffic” processor

‘anomaly_cold_int_traffic’: 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.

“leaf int hot anomaly history” processor

Purpose: Create time series showing hot anomaly being raised and cleared for each interface under consideration. This time series may contain up to ‘anomaly_history_count’ anomaly state changes.

Outputs of “leaf int hot anomaly history” processor

‘int_hot_accumulate_anomaly’: Time series showing hot anomaly being raised and cleared for each interface under consideration. This time series may contain up to ‘anomaly_history_count’ anomaly state changes.

“leaf int cold anomaly history” processor

Purpose: Create time series showing cold anomaly being raised and cleared for each interface under consideration. This time series may contain up to ‘anomaly_history_count’ anomaly state changes.

Outputs of “leaf int cold anomaly history” processor

‘int_cold_accumulate_anomaly’: Time series showing cold anomaly being raised and cleared for each interface under consideration. This time series may contain up to ‘anomaly_history_count’ anomaly state changes.

“interface sum per device” processor

Purpose: Sum average traffic for all interface under consideration per device.

Outputs of “interface sum per device” processor

‘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” processor

Purpose: Sum average traffic for all interface under consideration per device, per interface role.

Outputs of “interface sum per device per link role” processor

‘if_counter_sum_per_device_per_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’).

“system percent hot” processor

Purpose: Calculate percentage of interfaces that are hot on any given device under consideration.

Outputs of “system percent hot” processor

‘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).

“system percent cold” processor

Purpose: Calculate percentage of interfaces that are cold on any given device under consideration.

Outputs of “system percent cold” processor

‘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).

“device hot” processor

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.

Outputs of “device hot” processor

‘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).

“device cold” processor

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.

Outputs of “device cold” processor

‘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).

“anomaly device hot” processor

Purpose: Export ‘device hot’ state as anomaly.

Outputs of “anomaly device hot” processor

‘device_hot_anomaly’: 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).

“anomaly device cold” processor

Purpose: Export ‘device cold’ state as anomaly.

Outputs of “anomaly device cold” processor

‘device_cold_anomaly’: 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).

“anomaly device hot history” processor

Purpose: Create time series showing device hot anomaly being raised and cleared for each device consideration. This time series may contain up to ‘anomaly_history_count’ anomaly state changes.

Outputs of “anomaly device hot history” processor

device_hot_anomaly_timeseries: Time series showing device hot anomaly being raised and cleared for each device under consideration. This time series may contain up to ‘anomaly_history_count’ anomaly state changes. Each set member has the following key to identify it: system_id (id of the leaf system, usually serial number).

“anomaly device cold history” processor

Purpose: Create time series showing device cold anomaly being raised and cleared for each device consideration. This time series may contain up to ‘anomaly_history_count’ anomaly state changes.

Outputs of “anomaly device cold history” processor

device_cold_anomaly_timeseries: Time series showing device cold anomaly being raised and cleared for each device under consideration. This time series may contain up to ‘anomaly_history_count’ anomaly state changes. Each set member has the following key to identify it: system_id (id of the leaf system, usually serial number).

Hypervisor MTU Mismatch Probe (Virtual Infra)

Purpose 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.

Hypervisor MTU Threshold Check Probe (Virtual Infra)

Purpose Detect virtual infra interfaces with maximum transmission units (MTU) below a specified threshold (default: 1600).

Source Processor

Interface MTU (*generic graph collector*) output stage: Interface MTU (number set) (generated from graph)

Additional Processor(s)

MTU below threshold (*range*) input stage: Interface MTU
output stage: MTU below threshold (discrete state set)

Example Usage NSX Integration - To carry VXLAN-encapsulated overlay traffic, an MTU greater than 1600 is recommended. NSX-T transport nodes connected to ToR leafs that are below the specified threshold are detected.

To support Geneve encapsulation, the MTU configuration on NSX-T nodes involved in an overlay transport zone must have a valid MTU setting on the ESXi host. The image below shows hypervisors with the MTU above the threshold.

Search stages...

Stage: MTU below threshold Discrete State Set

Search stage data... ☐ Anomalies Only

1-15 of 15 Page Size: 25

| Hypervisor | Pnic Name | Vtep | Anomaly | Value | Updated |
|---|-----------|-------------|------------|-------|-------------|
| 172.20.64.6 | vmnic1 | vmk10 | No anomaly | false | 10 days ago |
| demo-NSX-Node1 | eth1 | nsx-vtep0.0 | No anomaly | false | 10 days ago |
| demo-NSX-Node1 | eth2 | nsx-vtep0.0 | No anomaly | false | 10 days ago |
| demo-NSX-Node2 | eth1 | nsx-vtep0.0 | No anomaly | false | 10 days ago |
| demo-NSX-Node2 | eth2 | nsx-vtep0.0 | No anomaly | false | 10 days ago |
| zz-aaron-nxstveos.2485377892354-1782299148-TN-2 | eth1 | nsx-vtep0.0 | No anomaly | false | 10 days ago |
| zz-aaron-nxstveos.2485377892354-1782299148-TN-2 | eth2 | nsx-vtep0.0 | No anomaly | false | 10 days ago |
| zz-addbp-nxst.cvx.2485377892354-1122158190-TN-1 | eth1 | nsx-vtep0.0 | No anomaly | false | 2 days ago |
| zz-addbp-nxst.cvx.2485377892354-1122158190-TN-1 | eth2 | nsx-vtep0.0 | No anomaly | false | 2 days ago |
| zz-addbp-nxst.cvx.2485377892354-1122158190-TN-2 | eth1 | nsx-vtep0.0 | No anomaly | false | 2 days ago |

If any of the hypervisors were below the threshold, the expected value would change to **true** and an anomaly would be raised.

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 leafs connected to hypervisors.

Source Processor

Hypervisor NICs with LAG (*generic graph collector*) output stage: Hypervisor NICs LAG Intent Status (discrete state set) (generated from graph)

Additional Processor(s)

Hypervisor NIC LAG anomalies (*state*) input stage: Hypervisor NICs LAG Intent Status
output stage: Hypervisor NIC LAG Mismatch Anomaly (discrete state set)

Example Usage vSphere Integration - This probe detects inconsistent LAG configs between fabric LAG dual-leafs and ESXi hosts. LACP mode information is collected from the fabric LAG dual-leafs and also connects to vCenter API and collects LAG groups and members per hypervisor.

Note: Current validation is done on vCenter virtual Distributed Switches only, not on virtual Standard Switches. LLDP must be enabled on vCenter vDS switches.

Anomalies are raised if any of the following occurs:

- LAG member ports on ToR are connected to non-LAG physical ports on ESXi.
- Non-LAG member ports on ToR are connected to LAG physical ports on ESXi.

NSX Integration - Enabling this probe activates a continuous LAG validation between NSX-T transport nodes and data center fabric. It validate that LAGs are properly configured between fabric LAG dual-leafs and NSX-T transport nodes. The NSX-T uplink profile defines the network interface configuration facing the fabric in terms of LAG and LACP config. Network interface misconfiguration between the transport node and the ToR switch is validated and detected.

Anomalies are raised in the following circumstances:

- NSX-T transport nodes are not configured for LAG but ToR has LAG member ports in the fabric.
- ESXi hosts are dual-attached to ToR leafs but corresponding NSX-T transport nodes are “single-attached” or they are using “NIC-teaming” using active-standby or load-balanced config.

1. Add NSX-T API user as a Virtual Infra.
2. Add NSX-T Manager in the blueprint (External Systems > Virtual Infra Managers).
3. Enable this probe (Hypervisor and Fabric LAG config mismatch).

Let’s say in the NSX-T uplink profile, LAG is deleted but the fabric has LAG in terms of ToR leafs having LAG member ports. As a result in a blueprint after enabling this probe LAG mismatch anomalies are raised.

| Fabric Interface | Fabric Lag | Hypervisor | Leaf | Pnic | Pnic Lag | Anomaly | Value | Updated |
|------------------|------------|---|---------------------|------|----------|---|-------|-----------|
| swp3 | bond1 | zz-karun-nsxt.cnx.2485377892354-3839439666-TN-2 | leaf-2-52540005BE0B | eth1 | | Anomalous value: mismatch Actual value: mismatch | true | a day ago |
| swp4 | bond1 | zz-karun-nsxt.cnx.2485377892354-3839439666-TN-2 | leaf-2-52540005BE0B | eth2 | | Anomalous value: mismatch Actual value: mismatch | true | a day ago |

Since the LAG on the NSX-T transport nodes has been deleted, there is a mismatch between physical network adapter (pnic) on ESXi host LAG configuration and LAG configuration on ToR leafs.

Hypervisor and Fabric VLAN Config Mismatch Probe (Virtual Infra)

Purpose Calculate VLAN mismatch between configured virtual networks on leafs and VLANs needed by VMs running on the hypervisors attached to leafs. (Formerly known as Virtual Infra VLAN Match).

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 collector*) 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

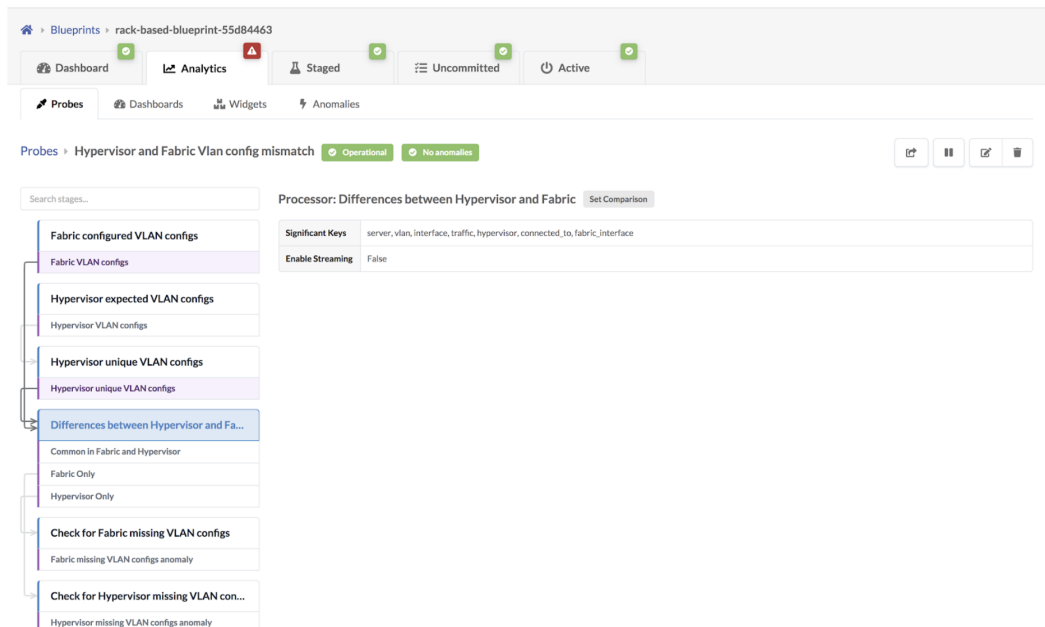
output stage: Fabric missing VLAN configs anomaly (discrete state set)

Check for Hypervisor missing VLAN configs (*range*) input stage: Fabric Only TimeSeries

output stage: Hypervisor missing VLAN configs anomaly (discrete state set)

Example Usage NSX-T Integration

If the VLAN on the data center fabric and NSX-T transport nodes match, then the probe looks similar to the image below:



The **Fabric VLAN Configs** stage shows the VLANs tagged towards NSX-T transport nodes on fabric ToR leafs as shown below:

Blueprints > rack-based-blueprint-55d84463

Dashboard Analytics Staged Uncommitted Active

Probes > Hypervisor and Fabric Vlan config mismatch Operational No anomalies

Search stages...

Stage: Fabric VLAN configs Number Set

Search stage data...

1-3 of 3 Page Size: 25

| Connected To | Fabric Interface | Hypervisor | Interface | Server | Traffic | Vlan |
|---------------------|------------------|---|--------------------------------------|---------------------|---------|------|
| leaf-2-52540005BE0B | bond1 | zz-karun-nsxt.cvx.2485377892354-3839439666-TN-2 | a5bb8183-90ef-497f-a988-3ef571066261 | rack2_001_server001 | tagged | 10 |
| leaf-2-52540005BE0B | bond1 | zz-karun-nsxt.cvx.2485377892354-3839439666-TN-2 | a5bb8183-90ef-497f-a988-3ef571066261 | rack2_001_server001 | tagged | 10 |
| leaf-2-52540005BE0B | bond1 | zz-karun-nsxt.cvx.2485377892354-3839439666-TN-2 | a5bb8183-90ef-497f-a988-3ef571066261 | rack2_001_server001 | tagged | 20 |

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

The **Common in Fabric and Hypervisor** stage shows 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.cvx.2485377892354-3839439666-TN-2 | a5bb8183-90ef-497f-a988-3ef571066261 | rack2_001_server001 | tagged | 100 | 1 |
| -52540005BE0B | bond1 | zz-karun-nsxt.cvx.2485377892354-3839439666-TN-2 | a5bb8183-90ef-497f-a988-3ef571066261 | rack2_001_server001 | tagged | 1000 | 1 |
| -52540005BE0B | bond1 | zz-karun-nsxt.cvx.2485377892354-3839439666-TN-2 | a5bb8183-90ef-497f-a988-3ef571066261 | rack2_001_server001 | tagged | 2000 | 1 |

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 on the NSX-T uplink profile is missing on the fabric, then VLAN mismatch anomalies are raised.

The screenshot shows the Apstra Probes interface. The top navigation bar includes 'Probes', 'Dashboards', 'Widgets', and 'Anomalies'. The main header shows 'Probes > Hypervisor and Fabric Vlan config mismatch' with a status of 'Operational' and '2 anomalies'. A sidebar on the left lists various stages for search, including 'Fabric configured VLAN configs', 'Hypervisor expected VLAN configs', and 'Check for Fabric missing VLAN configs'. The main content area displays the 'Stage: Fabric missing VLAN configs anomaly' with a 'Discrete State Set' button. Below this, there is an 'Anomaly Remediation' section with a 'Remediate Anomalies' button. A search bar and filters are present, showing '1 of 1' results. A table lists the details of the anomaly:

| Connected To | Fabric Interface | Hypervisor | Interface | Server | Traffic | Vlan | A |
|---------------------|------------------|---|-------------------------------------|---------------------|---------|------|---|
| leaf-2-52540005BEDB | bond1 | zz-karun-nxst.cvx.2485377892354-383943f666-TN-2 | a5bb183-90ef-497f-a988-3ef571066261 | rack2_001_server001 | tagged | 99 | A |

Some examples of mismatches 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.

In some scenarios, a VLAN mismatch anomaly can be remediated. If so, the **Remediate Anomalies** button will be shown on the probe details page. 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.

vCenter Integration

If VLAN config between vCenter and the fabric is mismatched, an anomaly is raised.

Some scenarios where a VLAN mismatch can be remediated include the following cases:

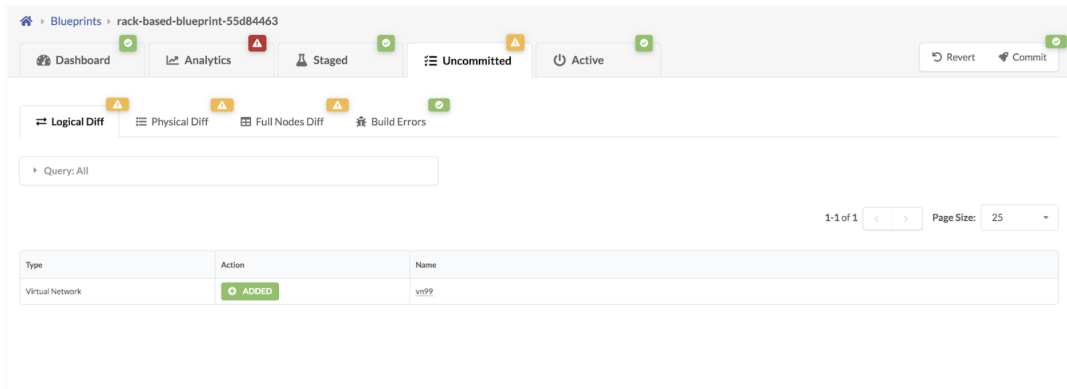
- If the vCenter Distributed Virtual Switch (vDS) port group does not have a corresponding rack-local VN (VLAN) for VLAN ID X. With one-click remediation, a new rack-local virtual network (VLAN) with the proper VLAN ID is created.
- If endpoint X in a rack-local VN with VLAN ID Y, does not have a corresponding 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.

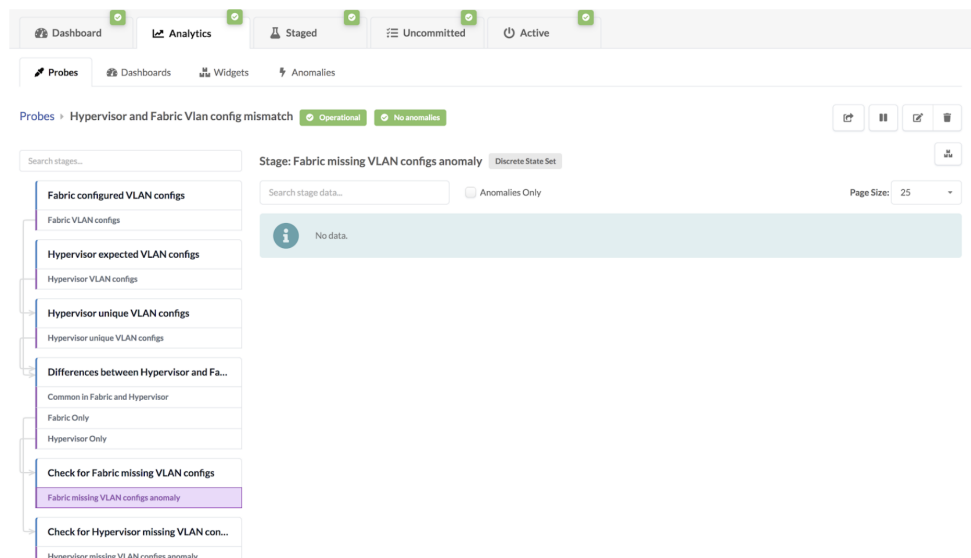
A VLAN-based rack-local virtual network is extending each VLAN segment defined on the vDS, across servers within the same rack. For example, vDS port group VLAN 10 = rack-local virtual network with VLAN 10.

Anomaly Remediation

To remediate anomalies, go to the details page of the **Hypervisor and Fabric Vlan config mismatch** probe, and click **Remediate Anomalies**. The necessary remediation for the anomalies is staged.



Review the staged configuration, add any necessary resources (such as IP subnet address, virtual gateway IP, etc.), then commit the configuration; the anomaly will be resolved.



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 collector*) 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.

Hypervisor Redundancy Checks Probe (Virtual Infra)

Purpose Detect hypervisor redundancy.

Source Processors

Hypervisor and connected leaf (*generic graph collector*) output stage: hypervisor_and_leaf (text set) (generated from graph)

Hypervisor pn timer and vnet (*generic graph collector*) output stage: hypervisor_pn timer_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 pn timer count (*set count*) input stage: hypervisors_pn timer_vnet
output stage: hypervisors_vnet_pn timer_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_pn timer_count
output stage: hypervisor_vnet_single_pn timer_anomaly (discrete state set)

Example Usage NSX-T Integration - an anomaly is raised in cases without redundancy or a single point of failure (SPOF) in hypervisor connectivity. Examples include:

- NSX-T transport nodes with a single non-LAG uplink towards ToR Leafs in the fabric can result in a single point of failure (SPOF) for overlay traffic.
- NSX-T transport nodes with a single LAG uplink with both members going to a single ToR leaf can result in a single point of failure (SPOF).
- Lack of redundancy between fabric LAG dual-leafs and ESXi hosts.

Detect single point of failures in hypervisor connectivity

Search stages...

Stage: **hypervisor_single_leaf_anomaly** Discrete State Set

Search stage data...

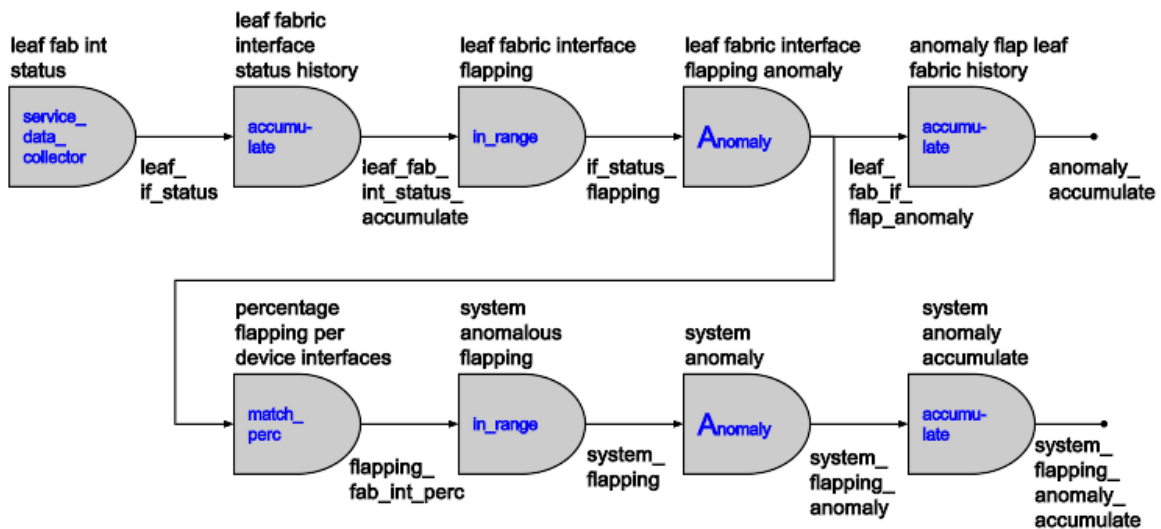
☐ false ☒ true

1 of 1 < > Page Size: 25

| Hypervisor | Hypervisor Id | Rack | Anomaly | Value | Updated |
|---|--------------------------------------|-----------|---|-------|----------------|
| zz-karun-nxst.cvx.2485377892354-3839439666-TN-2 | 85ff0779-c85a-4380-a256-5c2eb001f41c | rack2_001 | Anomalous value: s 1 Actual value: 1 | true | 39 minutes ago |

Hypervisor and connected leaf
 hypervisor_and_leaf
Hypervisor pnic and vnet
 hypervisor_pnic_vnet
Hypervisor and leaf count
 hypervisors_leaf_count
Hypervisor vnet pnic count
 hypervisors_vnet_pnic_count
Hypervisors without ToR Switch redu...
 hypervisor_single_leaf_anomaly ▲ 1

Interface Flapping Probe



Found at `/predefined_probes/fabric_interface_flapping`

It first identifies all the fabric interfaces on all the leaves. Fabric interfaces are defined as those facing spines. Then it collects operational status for each interface and creates a time series out of it. Then it checks if the number of state changes over configurable duration is within acceptable range (in this case less than specified upper bound) and generates anomaly if that condition is not satisfied. It then creates time series for this anomaly so that recent history of

existence and clearing of anomaly can be inspected. It also calculates the percentage of interfaces on any given device that has flapping anomaly and if that number is higher than the specified threshold it raises device level anomaly. It then creates time series for this anomaly so that recent history can be inspected.

“leaf fab int status” processor

Purpose: wires in interface status telemetry for all fabric interfaces on the leafs.

Outputs of “leaf fab int status” processor

‘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).

“leaf fabric interface status history” processor

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).

Outputs of “leaf fabric interface status history” processor

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” processor

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”.

Outputs of “leaf fabric interface flapping” processor

if_status_flapping: set of statuses (for each spine facing interface on each leaf), indicting 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).

“leaf fabric interface flapping anomaly” processor

Purpose: Export interface flapping as anomaly.

Outputs of “leaf fabric interface flapping anomaly” processor

leaf_fab_if_flap_anomaly: set of statuses (for each spine facing interface on each leaf), indicting 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).

“anomaly flap leaf fabric history” processor

Purpose: create recent history time series for each interface flapping anomaly. The time series will hold up to ‘anomaly_history_count’ samples (facade parameter).

Outputs of “anomaly flap leaf fabric history” processor

anomaly_accumulate: set of interface flapping anomaly 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).

“percentage flapping per device interfaces” processor

Purpose: Calculate percentage of interfaces that are flapping on any given device under consideration.

Outputs of “percentage flapping per device interfaces” processor

‘flapping_fab_int_perc’: Set of numbers, each indicating the the percentage of flapping 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 anomalous flapping” processor

Purpose: Evaluate if the percentage of flapping interface on any given device is within acceptable range (less than max_flapping_interfaces_percentage facade parameter) and generate system_flapping indication otherwise.

Outputs of “system anomalous flapping” processor

system_flapping: set of statuses for each leaf, indicting if the leaf has higher then acceptable percentage of flapping interfaces. Each set member has the following key to identify it: system_id (id of the leaf system, usually serial number).

“system anomaly” processor

Purpose: Export system anomalous flapping as anomaly.

Outputs of “system anomaly” processor

system_flapping_anomaly: set of statuses for each leaf, indicting if the leaf has higher then acceptable percentage of flapping interfaces. Each set member has the following key to identify it: system_id (id of the leaf system, usually serial number).

“system anomaly accumulate” processor

Purpose: create recent history time series for each leaf system flapping. In terms of the number of samples, the time series will hold anomaly_history_count (facade parameter).

Outputs of “system anomaly accumulate” processor

system_flapping_anomaly_accumulate: set of system flapping anomaly time series (for each spine facing interface on each leaf). Each set member has the following key to identify it: system_id (id of the leaf system, usually serial number).

Interface Policy 802.1x Probe

The Interface Policy predefined probe is used to monitor 802.1X supplicants and interface authentication.

You can use this probe to maintain 802.1X networks. From the Probes view of your Analytics blueprint, click Instantiate Predefined Probe button, then find Interface policy 802.1X in the list that pops up, then click Create. This probe is not activated by default.

The screenshot shows the Apstra Analytics interface. The top navigation bar includes 'Blueprints', 'L2V', and 'Analytics'. Below this, there are tabs for 'Dashboard', 'Analytics', 'Staged', 'Uncommitted', 'Active', and 'Time Voyager'. The 'Analytics' tab is selected, and the 'Probes' sub-tab is active. The main area displays a table of probes with columns: Name, Anomalies, State, Updated by, Tags, Enabled, and Actions. The table lists three probes: 'Counter Errors', 'Device system health', and 'Drain traffic anomaly'. All three are in an 'Operational' state with 'No anomalies'.

Below the table, the 'Instantiate Predefined Probe' dialog is open. It shows a search for 'Interface policy 802.1x' and a list of predefined probes. The 'Interface policy 802.1x' probe is selected. The dialog also contains a description of the probe and a 'Create' button.

| Name | Anomalies | State | Updated by | Tags | Enabled | Actions |
|-----------------------|--------------|-------------|-------------------|------|-------------------------------------|--|
| Counter Errors | No anomalies | Operational | System 7 days ago | | <input checked="" type="checkbox"/> | Edit Copy Delete |
| Device system health | No anomalies | Operational | admin 5 days ago | | <input checked="" type="checkbox"/> | Edit Copy Delete |
| Drain traffic anomaly | No anomalies | Operational | System 5 days ago | | <input checked="" type="checkbox"/> | Edit Copy Delete |

Instantiate Predefined Probe

Predefined Probe *

- Interface policy 802.1x
- Interface Flapping (Specific Interfaces)
- Interface Flapping (spine to superspine interfaces)
- Interface policy 802.1x**
- LAG Imbalance
- MLAG Imbalance
- OSPF Sessions

Generate a probe to extract system 802.1x telemetry status on authenticated MAC addresses and 802.1x enabled ports.

An anomaly is generated if the port is in a blocked state, which implies that 802.1x authentication failed, and there was no authorization fallback behavior to put the port into a forwarding state in a fallback VLAN.

Anomalies are raised for ports that are expected to be blocked, but are not actually blocked.

A processor is available for the end-user to obtain interface health, status, dynamic vlan, supplicant MACs, etc from active 802.1x authenticated sessions.

☐ Create Another? **Create**

The 802.1X hosts probe gives a fast view of network 802.1X MAC Addresses, authorization status, ports, and dynamic VLAN information.

[Home](#) > [Blueprints](#) > [L2V](#) > [Analytics](#)

Dashboard Analytics Staged Uncommitted Active Time Voyager

Probes > Interface policy 802.1x Operational No anomalies admin a few seconds ago Enabled

Obtain telemetry status for interfaces that are activated for interface policies defining 802.1x port control.

Search stages...

- 802.1x Authorization status
 - 802.1x Authorization status
 - 802.1x Authorized MACs
 - 802.1x Authorized MACs
 - 802.1x Interface status
 - 802.1x Interface status
 - 802.1x hosts
 - 802.1x hosts
 - 802.1x Expected authorization ...
 - 802.1x Expected authorization status
 - 802.1x dot1x_port_control aut...
 - Unexpected 802.1x authentication status
 - 802.1x port control expected s...

Stage: 802.1x Authorization status Text Set

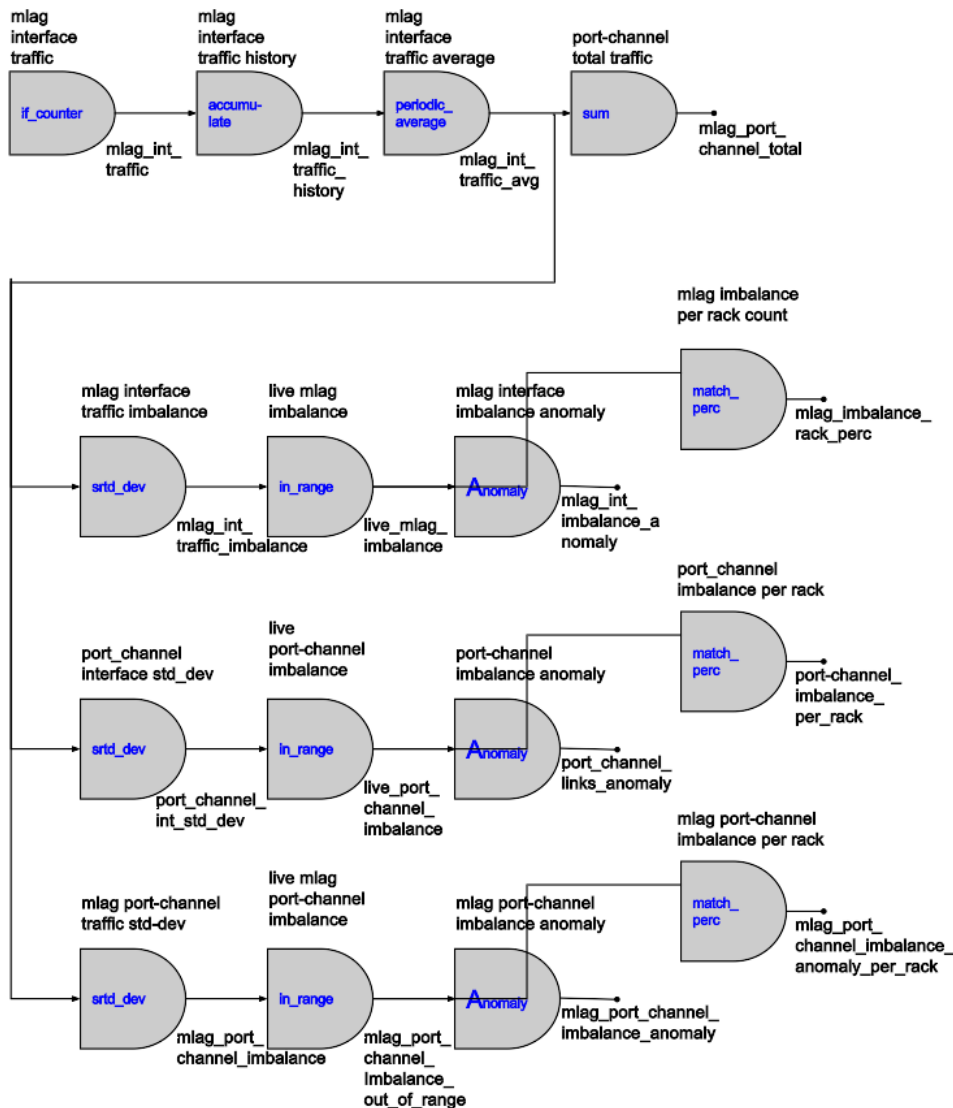
Collect 802.1x actual authorization status from interfaces that are dot1x_port_control 'auto' or 'force_unauthorized', for interfaces that use 802.1x PAE features.

Search stage data...

1-6 of 6 Page Size: 25

| System ID # | Dot1x Port Control # | Interface # | Type # | Value # | Telemetry Service Status | Updated # |
|--|----------------------|-------------|--------|------------|--------------------------|-------------------|
| 5054000BF029 I2-virtual-003-leaf1 Leaf | auto | Ethernet5 | Auth | blocked | No warnings | a few seconds ago |
| 5054000BF029 I2-virtual-003-leaf1 Leaf | auto | Ethernet6 | Auth | blocked | No warnings | a few seconds ago |
| 525400651BB3 I2-virtual-001-leaf1 Leaf | auto | swp5 | Auth | blocked | No warnings | a few seconds ago |
| 525400651BB3 I2-virtual-001-leaf1 Leaf | auto | swp6 | Auth | authorized | No warnings | a few seconds ago |
| 525400D591C4 I2-virtual-002-leaf1 Leaf | auto | Ethernet1/5 | Auth | No value | No warnings | |

MLAG Imbalance Probe



Found at /predefined_probes/mlag_imbalance

It first identifies all the interfaces across the data center that participate in server to leaf MLAGs. It then collects samples for each, generates time series and calculates average traffic across configurable time interval for each interface. It then calculates the following 3 types of imbalances by calculating standard deviation across the following sets: (1) set of all interfaces participating in MLAG (on both leafs), (2) set of all interfaces within a port channel on a given leaf, (3) set of port channels in an MLAG. It then generates anomaly if any of these imbalances is out of range. It then also calculates percentage of MLAGs per rack (leaf pair) that have imbalance as well as percentage of port channels on each leaf that have imbalance.

“mlag interface traffic” processor

Purpose: wires in interface traffic samples (measured in bytes per second) for all leaf interfaces that are part of an MLAG. Unit is bytes per second.

Outputs of “mlag interface traffic” processor

‘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).

“mlag interface traffic history” processor

Purpose: create recent history time series out of traffic samples from the mlag_int_traffic output. In terms of the number of samples, the time series will hold the smaller of: 100 samples or samples collected during the last ‘total_duration’ seconds (facade parameter). Samples unit is bytes per second.

Outputs of “mlag interface traffic history” processor

mlag_int_traffic_history: set of traffic samples time series (for each mlag interface on the leafs). 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). Samples unit is bytes per second.

“mlag interface traffic average” processor

Purpose: Calculate average traffic during period specified by average_period facade parameter. Unit is bytes per second.

Outputs of “mlag interface traffic average” processor

‘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” processor

Purpose: Calculate standard deviation between traffic averages on all interfaces belonging to a given MLAG. Unit is bytes per second.

Outputs of “mlag interface traffic imbalance” processor

‘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.

“live mlag imbalance” processor

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).

'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.

“mlag interface imbalance anomaly” processor

Purpose: Export 'live mlag imbalance' state as anomaly.

Outputs of “mlag interface imbalance anomaly” processor

'mlag_int_imbalance_anomaly': Set of boolean values, each indicating the presence or absence of live mlag imbalance. Each set member has the following keys to identify it: rack, mlag_id.

“mlag imbalance per rack percent” processor

Purpose: Calculate percentage of mlags on a given rack that have imbalance anomaly.

Outputs of “mlag imbalance per rack percent” processor

'mlag_imbalance_rack_perc': Set of numbers, each indicating the percentage of mlags with imbalance on each rack. Each set member has the following key to identify it: rack.

“port-channel interface std-dev” processor

Purpose: Calculate standard deviation between traffic averages on all interfaces belonging to a port channel. Unit is bytes per second.

Outputs of “port-channel interface std-dev” processor

'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.

“live port-channel imbalance” processor

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).

'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.

“port-channel imbalance anomaly” processor

Purpose: Export 'live_port_channel_imbalance' state as anomaly.

Outputs of “port-channel imbalance anomaly” processor

‘port_channel_links_anomaly’: Set of boolean values, each indicating the presence or absence of live port channel imbalance. Each set member has the following keys to identify it: rack, mlag_id, leaf.

“port-channel imbalance per rack” processor

Purpose: Calculate percentage of port channels on a given rack that have imbalance anomaly.

Outputs of “port-channel imbalance per rack” processor

‘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 traffic std-dev” processor

Purpose: Calculate standard deviation between traffic averages on both port channels belonging to a mlag. Unit is bytes per second.

Outputs of “mlag port-channel traffic std-dev” processor

‘mlag_port_channel_imbalance’: set of numbers, one for each mlag identified by mlag_id, rack pair. Each number each 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.

“live mlag port-channel imbalance” processor

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).

‘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_port-channel imbalance anomaly” processor

Purpose: Export ‘mlag_port_channel_imbalance_out_of_range’ state as anomaly.

Outputs of “mlag_port-channel imbalance anomaly” processor

‘mlag_port_channel_imbalance_anomaly’: Set of boolean values, each indicating the presence or absence of live mlag port-channel imbalance. Each set member has the following keys to identify it: rack, mlag_id.

“mlag port-channel imbalance per rack” processor

Purpose: Calculate percentage of mlags on a given rack that have port channel imbalance anomaly.

Outputs of “mlag port-channel imbalance per rack” processor

‘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.

“port-channel total traffic” processor

Purpose: Calculate total traffic per port channel. Unit is byte per second.

Outputs of “port-channel total traffic” processor

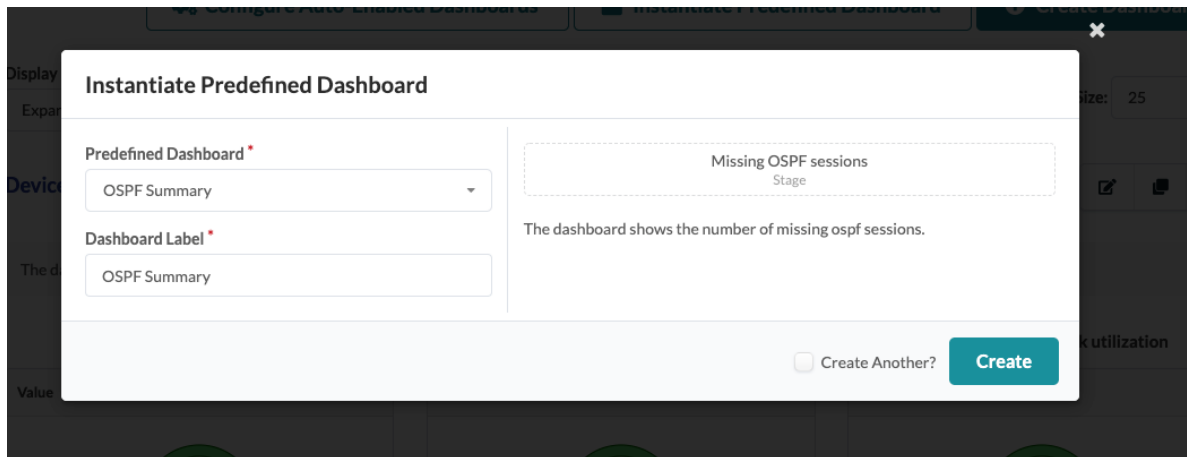
‘mlag_port_channel_total’: Set of numbers, each indicating total traffic for each port channel. Each set member has the following key to identify it: rack, mlag_id, leaf. Unit is byte per second.

OSPF Sessions Probe

Customers are alerted in case of intent violation wrt external OSPF peering status and route expectations. To troubleshoot external network OSPF connectivity problems, you can use the predefined IBA probe to obtain telemetry status for the OSPF sessions. This probe shows the number of missing ospf sessions.

To start troubleshooting OSPF neighbor state follow these steps:

1. From the blueprint, navigate to **Analytics > Probes**, then click **Create Probe** and select **Instantiate Predefined Probe**.
2. Select “OSPF Summary”



When you click “Create”, OSPF sessions and present neighbors are collected in a single stage output table (session status, expected sessions, active sessions and missing sessions).

OSPF session status

OSPF session status

OSPF expected sessions

OSPF expected sessions

Check active OSPF ses...

Check active OSPF sessions

OSPF missing sessions

OSPF missing sessions

Probes > OSPF Sessions

Operational

No anomalies

System 8 days ago

Obtain telemetry status for the OSPF sessions

Search stages...

Stage: OSPF session status

Text Set

Search stage data...

1-2 of 2

Page Size: 25

| System ID | Area | Interface | Neighbor Addr | Vrf | Value | Telemetry Service Status | Updated |
|--|------|-----------|---------------|------|---------|--------------------------|--------------|
| 525400AF5C1C racktype-1-002-leaf1 Leaf | 51 | Vlan60 | 10.61.60.254 | blue | FULL/DR | No warnings | a minute ago |
| 525400AF5C1C racktype-1-002-leaf1 Leaf | 51 | Vlan62 | 10.61.62.254 | red | FULL/DR | No warnings | a minute ago |

Dashboards

Anomalies

Widgets

Probes

Probes > OSPF Sessions

Operational

No anomalies

System 8 days ago

Obtain telemetry status for the OSPF sessions

Search stages...

Stage: OSPF missing sessions


Number Set

Search stage data...

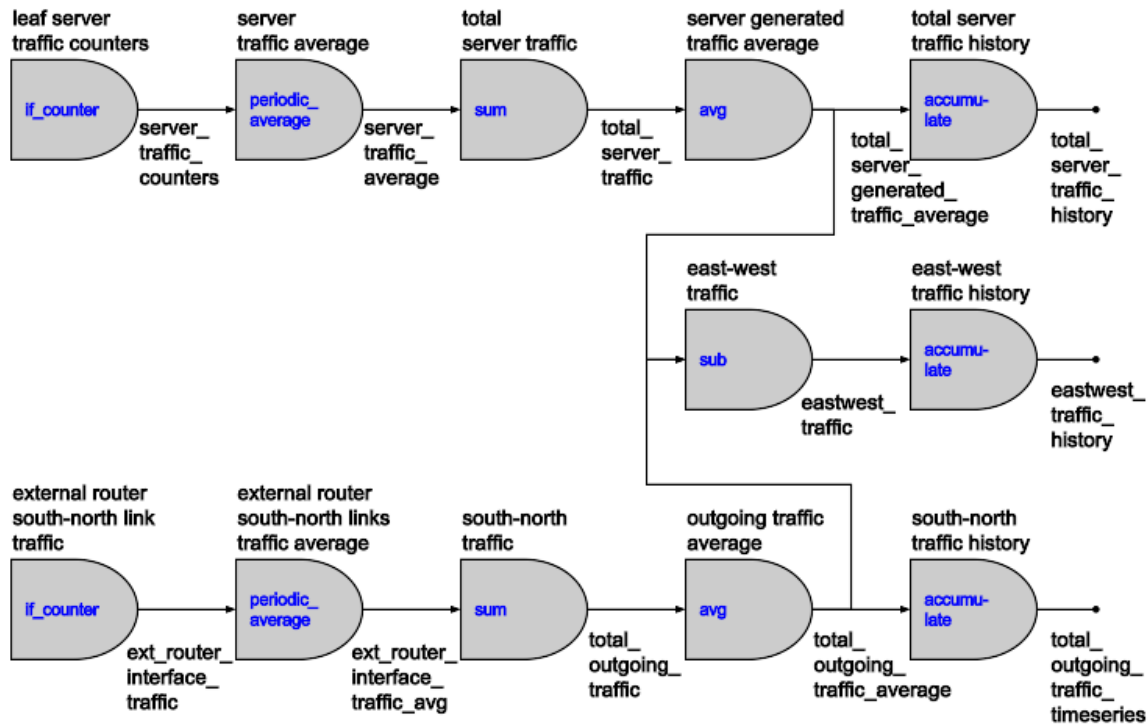
Show Context

1 of 1

Page Size: 25

| Total Count | Value | Updated |
|-------------|--|------------|
| 2 |  | 2 days ago |

Total East/West Traffic Probe



Found at `/predefined_probes/eastwest_traffic`

It first identifies all the server facing interfaces and all external router facing interfaces. It then collects samples for each, generates time series and calculates average traffic across configurable time interval. It then perform accumulation to calculate total server traffic, total external router traffic (south-north) and derives total east-west traffic as the difference between the two. It then provides time series showing how these three totals changed over time.

“leaf server traffic counters” processor

Purpose: wires in interface traffic samples (measured in bytes per second) for traffic received on leafs from the servers

Outputs of “leaf server traffic counters” processor

‘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).

“server traffic average” processor

Purpose: Calculate average server traffic during period specified by `average_period` facade parameter. Unit is bytes per second.

Outputs of “server traffic average” processor

`'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).

“external router south-north link traffic” processor

Purpose: wires in interface traffic samples (measured in bytes per second) for traffic sent to external routers

Outputs of “external router south-north link traffic” processor

`'external router south-north links traffic average'`: set of traffic samples (for each external router facing interface on each device). Each set member has the following keys to identify it: `system_id` (id of the system, usually serial number), `interface` (name of the interface).

“external router south-north links traffic average” processor

Purpose: Calculate average traffic for each interface facing external router traffic during period specified by `average_period` facade parameter. Unit is bytes per second.

Outputs of “external router south-north links traffic average” processor

`'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).

“total server traffic” processor

Purpose: Calculate total server traffic by summing average traffic on each interface attached to servers, in receive direction. Unit is bytes per second.

Outputs of “total server traffic” processor

`'total_server_traffic'`: total server traffic average in bytes per second.

“server generated traffic average” processor

Purpose: Calculate total average server traffic over `average_period` seconds, which is a facade parameter. Unit is bytes per second.

Outputs of “server generated traffic average” processor

‘total_server_generated_traffic_average’: total server traffic average in bytes per second.

“total server traffic history” processor

Purpose: create recent history time series out showing how total average server traffic changed over time. In terms of the number of samples, the time series will hold history_sample_count values (facade parameter). Unit is bytes per second.

Outputs of “total server traffic history” processor

total_server_traffic_history: time series showing total average server traffic over recent history. Unit is bytes per second.

“south-north traffic” processor

Purpose: Calculate total traffic by summing average traffic on each interface facing external router. Unit is bytes per second.

Outputs of “south-north traffic” processor

‘total_outgoing_traffic’: total south-north traffic average in bytes per second.

“outgoing_traffic_average” processor

Purpose: Calculate total south-north traffic over average_period seconds, which is a facade parameter. Unit is bytes per second.

Outputs of “outgoing_traffic_average” processor

‘total_outgoing_traffic_average’: total south-north traffic average in bytes per second.

“south-north traffic history” processor

Purpose: create recent history time series showing how total average south-north traffic changed over time. In terms of the number of samples, the time series will hold history_sample_count values (facade parameter). Unit is bytes per second.

Outputs of “south-north traffic history” processor

total_outgoing_traffic_timeseries: time series showing total average server traffic over recent history. Unit is bytes per second.

“east-west traffic” processor

Purpose: Subtract south-north traffic from total server traffic to obtain total east-west traffic average, in bytes per second.

Outputs of “east-west traffic” processor

‘eastwest_traffic’: Total average east-west traffic over last average_period seconds, in bytes per second.

“east-west traffic history” processor

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 will hold history_sample_count values (facade parameter). Unit is bytes per second.

Outputs of “east-west traffic history” processor

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 leafs connected to hypervisors. (formerly known as VMs without configured VLANs)

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 graph collector*) output stage: VMs on hypervisors connected to Fabric (number set)

Additional Processor(s)

Differences between Fabric and Hypervisor (*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*) input stage: VMs not backed by Fabric VLANs

output stage: Affected VM Anomalies (discrete state set)

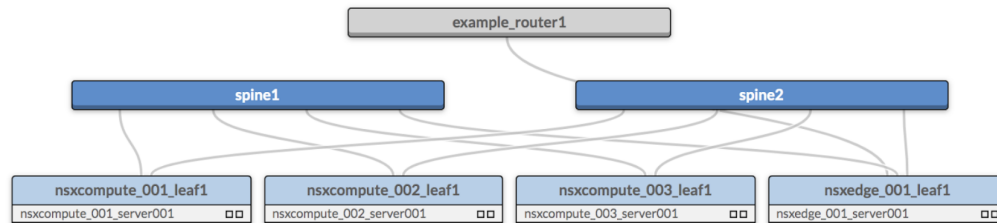
Example Usage NSX-T Integration - VMs participating in a particular network are attached to an NSX logical switch. In NSX transport zone controls to which hypervisors or ESXi host an NSX logical switch can span. To have VXLAN connectivity for these VMs they need to be part of the same transport zone. This predefined anomaly helps validate that all VLAN backend interfaces defined for NSX-T nodes are also configured on the ToR interfaces connecting that node to the fabric.

VLAN probe anomaly checks for VLAN specification in case of NSX-T via one of the two methods below:

Method One: When you have VMs that are connected to the NSX-T overlay, you can configure a bridge-backed logical switch to provide layer 2 connectivity with other devices or VMs. So via VLAN specification on NSX-T layer 2 bridges and fabric if respective VXLAN VN is not there, then an anomaly is raised.

Method Two: Edge uplinks go out through VLAN logical switches. So let's say if the uplink VLAN logical switch has a particular VLAN ID and respective VLAN on ToR port connected to the hypervisor host is not configured then also this VLAN probe will raise anomalies and help detect such misconfiguration.

The following is a simple topology where nsxcompute_001_server_001 and nsxedge_001_server001 are ESXi hosting VMs that are connected to the NSX-T overlay network.



There is one VM on each ESXi host that needs a VXLAN VN endpoint on each leaf, i.e. nsxcompute_001_leaf1 and nsxedge_001_leaf1 to communicate on the overlay network.

When VXLAN VNs assigned to ToR leafs are deleted, VLAN misconfig anomalies are raised as below under Fabric Health in the dashboard.

Critical services affected by VLAN misconfig

| Hypervisor | Virtual Machine | Virtual Machine Ip |
|----------------------------|-----------------|--------------------|
| nsxtcomputehost01 | webtier010 | 192.168.1.10 |
| nsxtcomputehost01 | webtier011 | 192.168.1.30 |
| nsxtedgehost01 | webtier020 | 192.168.1.20 |
| View stage | | |

VMs not backed by Fabric VLANs shows VMs with VLAN missing.

Probes > VMs without Fabric configured VLANs Operational 3 anomalies

Search stages...

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-aef5d8cc4c2 | webtier010 | 192.168.1.10 | No value | benefitswebtier | Network adapter 1 |
| nsxtcomputehost01 | 33c307a5-5895-4252-8e60-aef5d8cc4c2 | webtier011 | 192.168.1.30 | No value | benefitswebtier | Network adapter 1 |
| nsxtedgehost01 | f0286797-26d1-4e00-b8af-5260e3b671ac | webtier020 | 192.168.1.20 | No value | benefitswebtier | Network adapter 1 |

VMs backed by Fabric VLANs

VMs backed by Fabric VLANs

VMs on hypervisors connected t...

VMs on hypervisors connected to Fabric

Differences between Fabric and ...

VMs backed by VLANs on Fabric

VMs not backed by Fabric VLANs

NA

Affected VM Anomalies

Affected VM Anomalies 3

Affected VM Anomalies shows VLAN missing in the fabric.

Probes > VMs without Fabric configured VLANs Operational 3 anomalies

Search stages...

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-aef5d8cc4c2 | webtier010 | 192.168.1.10 | No value | benefitswebtier | Network adapter 1 | Expected value: 0 Actual value: 1 | true | 8 hours ago |
| 252-8e60-aef5d8cc4c2 | webtier011 | 192.168.1.30 | No value | benefitswebtier | Network adapter 1 | Expected value: 0 Actual value: 1 | true | 8 hours ago |
| e00-b8af-5260e3b671ac | webtier020 | 192.168.1.20 | No value | benefitswebtier | Network adapter 1 | Expected value: 0 Actual value: 1 | true | 8 hours ago |

VMs backed by Fabric VLANs

VMs backed by Fabric VLANs

VMs on hypervisors connected t...

VMs on hypervisors connected to Fabric

Differences between Fabric and ...

VMs backed by VLANs on Fabric

VMs not backed by Fabric VLANs

NA

Affected VM Anomalies

Affected VM Anomalies 3

Cloning Probes

Instead of entering all details for a new probe, you can clone an existing one, give it a new name and customize it.

4.3.5.5 Creating Probe

1. From the blueprint, navigate to **Analytics > Probes**, then click **Create Probe** and select **New Probe**.
2. Enter a name and (optional) description.
3. To be able to filter by your own defined categories, you can 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 the links below.

6. Customize inputs and properties as appropriate, or leave defaults as is.
7. Repeat the previous two steps until you've added all required processors for the new probe.
8. Click **Create** to create the probe and return to the list view.

Processor: Accumulate

The Accumulate processor used in IBA probes creates one (N/DS) time-series output for each input with the same properties; each time the input changes, it takes its timestamp and value and appends it 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 `total_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.

Input Types - Number-Set (NS), Discrete-State-Set (DSS)

Output Types - NSTS, DSSTS

Properties

Max Samples (`max_samples`) Limits the maximum number of samples or an expression that evaluates to number of samples (default:1024)

Total Duration (`total_duration`) Limits the number of samples by their total duration. (in seconds) or an expression that evaluates to number of seconds (default:0)

Graph Query (`graph_query`) One or more queries on graph specified as strings, or a list of such queries. (String will be deprecated in a future release.) Multiple queries should provide all the named nodes referenced by the expression fields (including `additional_properties`). Graph query is executed on the “operation” graph. Results of the queries can be accessed using the “`query_result`” variable with the appropriate index. For example, if querying property set nodes under name “ps”, the result will be available as “`query_result[0][“ps”]`”.

In collector processors (`*_collector`, `if_counter`) it is used to choose a set of nodes for further processing (for example, all leafs, or all interfaces between leaf and spines)

In other processors it is used for general parameterization and it is only supported as a list of queries.

Listing 1: Fabric Interfaces Example

```
graph_query: "node("system", role="leaf", name="system").
             out("hosted_interfaces").
             node("interface", name="iface").out("link").
             node("link", role="spine_leaf")"
```

Listing 2: Leafs and Spines using two queries Example

```
graph_query: ["node("system", role="leaf", name="system")",
             "node("system", role="spine", name="system)"]
```

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 (as of version 3.0). 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.

```
graph_query: [node("property_set", label="probe_propset", name="ps")]
duration: int(query_result[0]["ps"].values["accumulate_duration"])
```

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.

Accumulate Example

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.

Input Types - Number-Set (NS), NSTS

Output Types - Number-Set (NS)

Properties

Group by (group_by) 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.

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](#) example for how this works.

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.

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.

Average Example

See [standard deviation example](#) - it is the same except we calculate average instead of standard deviation.

Processor: Comparison

The Comparison processor takes two NS 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.

Input Types - Number-Set (NS)

Output Types - Discrete-State (DS): true or false

Properties

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.

Comparison Example

```
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: Detailed Interface Counters

The Detailed Interface Counters processor selects interfaces according to the configuration and outputs all available interface related counters (e.g tx_bits, rx_bits etc) and interface utilization.

Input Types - No inputs. This is a source processor.

Output Types

Properties

Graph Query (graph_query) One or more queries on graph specified as strings, or a list of such queries. (String will be deprecated in a future release.) Multiple queries should provide all the named nodes referenced by the expression fields (including additional_properties). Graph query is executed on the “operation” graph. Results of the queries can be accessed using the “query_result” variable with the appropriate index. For example, if querying property set nodes under name “ps”, the result will be available as “query_result[0][“ps”]”.

In collector processors (*_collector, if_counter) it is used to choose a set of nodes for further processing (for example, all leafs, or all interfaces between leaf and spines)

In other processors it is used for general parameterization and it is only supported as a list of queries.

Listing 3: Fabric Interfaces Example

```
graph_query: "node("system", role="leaf", name="system").
             out("hosted_interfaces").
             node("interface", name="iface").out("link").
             node("link", role="spine_leaf")"
```

Listing 4: Leafs and Spines using two queries Example

```
graph_query: ["node("system", role="leaf", name="system")",
             "node("system", role="spine", name="system)"]
```

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 Size of the averaging period. (seconds)

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: EVPN Type 3

The EVPN Type 3 processor generates a configuration containing expectations of EVPN type 3 routes.

Input Types - Number-Set (NS), Discrete-State-Set (DSS)

Output Types - NSTS, DSSTS

Properties

Execution count Number of times the data collection is done.

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".

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.

Input Types - Number-Set (NS), Discrete-State-Set (DSS)

Output Types - NSTS, DSSTS

Properties

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'.

Input Types - No inputs. This is a source processor.

Output Types - NSTS, DSSTS

Properties

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) 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”]”.

In collector processors (*_collector, if_counter) it is used to choose a set of nodes for further processing (for example, all leafs, or all interfaces between leaf and spines)

In other processors it is used for general parameterization and it is only supported as a list of queries.

Listing 5: Fabric Interfaces Example

```
graph_query: "node("system", role="leaf", name="system").
              out("hosted_interfaces").
              node("interface", name="iface").out("link").
              node("link", role="spine_leaf")"
```

Listing 6: Leafs and Spines using two queries Example

```
graph_query: ["node("system", role="leaf", name="system")",
              "node("system", role="spine", name="system)"]
```

Ingestion filter (ingestion_filter) New (reserved) key. Ingestion filter determines what metrics from the collector make it into the probe. As of version 3.0, 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.

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.

Keys available to express in the filter are same as the metric identity keys.

- 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 data_type 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 Group by (query_group_by) 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.

When grouping is active (query_group_by is not null), query results are d by the specified list of names, where one output item is created per each group. In this case, the expressions can only access matcher names specified in query_group_by and the query results for each group are accessed

using a new `group_items` variable. The `group_items` variable is a list of query results, where each result has named nodes/relationships, not present in `query_group_by`.

The following table describes the behavior for various values of this field:

| Value of <code>query_group_by</code> field | Semantics |
|---|--|
| Omitted or provided as <code>json null</code> (ala <code>None</code> in Python) | No grouping is done. This is equivalent to current behavior of <code>extensible_data_collector</code> . Using <code>'group_items'</code> in this case is not permitted and results in probe error state. |
| Empty list (<code>[]</code>) | 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. |

Value Map 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.

Listing 7: Sample value map for interface status

```
{
  "0": "unknown",
  "1": "down",
  "2": "up",
  "3": "missing"
}
```

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: Generic Graph Collector

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.

Input Types - No inputs. This is a source processor.

Output Types - Discrete-State-Set (DSS), Number-Set (NS), TS (based on data_type)

Properties

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) 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”]”.

In collector processors (*_collector, if_counter) it is used to choose a set of nodes for further processing (for example, all leafs, or all interfaces between leaf and spines)

In other processors it is used for general parameterization and it is only supported as a list of queries.

Listing 8: Fabric Interfaces Example

```
graph_query: "node("system", role="leaf", name="system").
             out("hosted_interfaces").
             node("interface", name="iface").out("link").
             node("link", role="spine_leaf")"
```

Listing 9: Leafs and Spines using two queries Example

```
graph_query: ["node("system", role="leaf", name="system")",
             "node("system", role="spine", name="system)"]
```

Value Map 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.

Listing 10: Sample value map for interface status

```
{
  "0": "unknown",
  "1": "down",
  "2": "up",
  "3": "missing"
}
```

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.

Generic Graph Collector Example

```
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.

Input Types - No inputs. This is a source processor.

Output Types - Discrete-State-Set (DSS), Number-Set (NS), TS (based on data_type)

Properties

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) 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”]”.

In collector processors (*_collector, if_counter) it is used to choose a set of nodes for further processing (for example, all leafs, or all interfaces between leaf and spines)

In other processors it is used for general parameterization and it is only supported as a list of queries.

Listing 11: Fabric Interfaces Example

```
graph_query: "node("system", role="leaf", name="system").
              out("hosted_interfaces").
              node("interface", name="iface").out("link").
              node("link", role="spine_leaf") "
```

Listing 12: Leafs and Spines using two queries Example

```
graph_query: ["node("system", role="leaf", name="system")",
             "node("system", role="spine", name="system)"]
```

Value Map 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.

Listing 13: Sample value map for interface status

```
{
  "0": "unknown",
  "1": "down",
  "2": "up",
  "3": "missing"
}
```

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

The Interface Counters processor selects interfaces according to the configuration and outputs counter stats of the specified types (such as 'tx_bytes').

Input Types - No inputs. This is a source processor.

Output Types - Number-Set (NS)

Properties

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) 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”]”.

In collector processors (*_collector, if_counter) it is used to choose a set of nodes for further processing (for example, all leafs, or all interfaces between leaf and spines)

In other processors it is used for general parameterization and it is only supported as a list of queries.

Listing 14: Fabric Interfaces Example

```
graph_query: "node("system", role="leaf", name="system").
             out("hosted_interfaces").
             node("interface", name="iface").out("link").
             node("link", role="spine_leaf") "
```

Listing 15: Leafs and Spines using two queries Example

```
graph_query: ["node("system", role="leaf", name="system")",
              "node("system", role="spine", name="system")"]
```

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.

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.

Interface Counter Example

```
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.

Listing 16: Sample Data

```
[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: Match Count

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.

Input Types - Discrete-State-Set (DSS), TS

Output Types - NS

Properties

Group by (group_by) 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.

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](#) example for how this works.

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.

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.

Match Count Example

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

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.

Input Types - Discrete-State-Set (DSS), TS

Output Types - Number-Set (NS)

Properties

Group by (group_by) 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.

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](#) example for how this works.

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.

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.

Match Percentage Example

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

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’.

Input Types - Time-Series (TS), TSTS

Output Types - Discrete-State-Set (DSS)

Properties

Graph Query (graph_query) One or more queries on graph specified as strings, or a list of such queries. (String will be deprecated in a future release.) Multiple queries should provide all the named nodes referenced by the expression fields (including additional_properties). Graph query is executed on the “operation” graph. Results of the queries can be accessed using the “query_result” variable with the appropriate index. For example, if querying property set nodes under name “ps”, the result will be available as “query_result[0][“ps”]”.

In collector processors (*_collector, if_counter) it is used to choose a set of nodes for further processing (for example, all leafs, or all interfaces between leaf and spines)

In other processors it is used for general parameterization and it is only supported as a list of queries.

Listing 17: Fabric Interfaces Example

```
graph_query: "node("system", role="leaf", name="system").
              out("hosted_interfaces").
              node("interface", name="iface").out("link").
              node("link", role="spine_leaf")"
```

Listing 18: Leafs and Spines using two queries Example

```
graph_query: ["node("system", role="leaf", name="system"),
              "node("system", role="spine", name="system)"]
```

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 (as of version 3.0). 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.

```
graph_query: [node("property_set", label="probe_propset", name="ps")]
duration: int(query_result[0]["ps"].values["accumulate_duration"])
```

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 (regex) 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.

Match String Example

```
regex: "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

The Max processor groups as described by **Group by**, then finds the maximum value and outputs it for each group.

Input Types - Number-Set (NS), NSTS

Output Types - Number-Set (NS)

Properties

Group by (group_by) 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.

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* example for how this works.

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.

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.

Max Example

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

The Min processor groups as described in **Group by**, then finds the minimum value and outputs it for each group.

Input Types - Number-Set (NS), NSTS

Output Types - Number-Set (NS)

Properties

Group by (group_by) 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.

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](#) example for how this works.

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.

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.

Min Example

Assume a configuration of:

```
group_by: ["system_id"]
```

Sample Input:

```
[system_id=leaf0,if_name=swp40] : 10
[system_id=leaf0,if_name=swp41] : 11
[system_id=leaf0,if_name=swp42] : 15
[system_id=spine0,if_name=eth15] : 32
[system_id=spine0,if_name=eth16] : 30
[system_id=spine0,if_name=eth17] : 36
```

Output "out":

```
[system_id=leaf0] : 10
[system_id=spine0] : 30
```

Processor: Periodic Average

One N is created on output per each input. Each `<period>` seconds, the output is set to the average of the input over the last `<period>` seconds. This is not a weighted average.

Input Types - Number-Set (NS)

Output Types - Number-Set (NS)

Properties

Period Size of the averaging period. (time in seconds, integer, or an expression that evaluates to time in seconds integer value)

Graph Query (`graph_query`) One or more queries on graph specified as strings, or a list of such queries. (String will be deprecated in a future release.) Multiple queries should provide all the named nodes referenced by the expression fields (including `additional_properties`). Graph query is executed on the "operation" graph. Results of the queries can be accessed using the "query_result" variable with the appropriate index. For example, if querying property set nodes under name "ps", the result will be available as "query_result[0]["ps"]".

In collector processors (`*_collector`, `if_counter`) it is used to choose a set of nodes for further processing (for example, all leafs, or all interfaces between leaf and spines)

In other processors it is used for general parameterization and it is only supported as a list of queries.

Listing 19: Fabric Interfaces Example

```
graph_query: "node("system", role="leaf", name="system").
    out("hosted_interfaces").
    node("interface", name="iface").out("link").
    node("link", role="spine_leaf")"
```

Listing 20: Leafs and Spines using two queries Example

```
graph_query: ["node("system", role="leaf", name="system")",
    "node("system", role="spine", name="system")"]
```

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 (as of version 3.0). 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.

```
graph_query: [node("property_set", label="probe_propset", name="ps")]
duration: int(query_result[0]["ps"].values["accumulate_duration"])
```

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.

Periodic Average Example

```
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

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’.)

As of version 3.1, the range processor was enhanced to generate the output of True when the input matches the specified criteria - this is the opposite of previous behavior.

Input Types - Number-Set (NS), NSTS

Output Types - Discrete-State-Set (DSS)

Properties

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.

Graph Query (graph_query) One or more queries on graph specified as strings, or a list of such queries. (String will be deprecated in a future release.) Multiple queries should provide all the named nodes referenced by the expression fields (including additional_properties). Graph query is executed on the “operation” graph. Results of the queries can be accessed using the “query_result” variable with the appropriate index. For example, if querying property set nodes under name “ps”, the result will be available as “query_result[0][“ps”]”.

In collector processors (*_collector, if_counter) it is used to choose a set of nodes for further processing (for example, all leafs, or all interfaces between leaf and spines)

In other processors it is used for general parameterization and it is only supported as a list of queries.

Listing 21: Fabric Interfaces Example

```
graph_query: "node("system", role="leaf", name="system").
             out("hosted_interfaces").
```

(continues on next page)

(continued from previous page)

```
node("interface", name="iface").out("link").
node("link", role="spine_leaf") "
```

Listing 22: Leafs and Spines using two queries Example

```
graph_query: ["node("system", role="leaf", name="system")",
              "node("system", role="spine", name="system")"]
```

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 (as of version 3.0). 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.

```
graph_query: [node("property_set", label="probe_propset", name="ps")]
duration: int(query_result[0]["ps"].values["accumulate_duration"])
```

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.

Range Example

```
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

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.

Input Types - Number-Set (NS)

Output Types - Number-Set (NS)

Properties

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.

Ratio Output Example

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

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.

Input Types - No inputs. This is a source processor.

Output Types - Discrete-State-Set (DSS), Number-Set (NS)

Properties

Graph Query (graph_query) One or more queries on graph specified as strings, or a list of such queries. (String will be deprecated in a future release.) Multiple queries should provide all the named nodes referenced by the expression fields (including additional_properties). Graph query is executed on the “operation” graph. Results of the queries can be accessed using the “query_result” variable with the appropriate index. For example, if querying property set nodes under name “ps”, the result will be available as “query_result[0][“ps”]”.

In collector processors (`*_collector`, `if_counter`) it is used to choose a set of nodes for further processing (for example, all leafs, or all interfaces between leaf and spines)

In other processors it is used for general parameterization and it is only supported as a list of queries.

Listing 23: Fabric Interfaces Example

```
graph_query: "node("system", role="leaf", name="system").
             out("hosted_interfaces").
             node("interface", name="iface").out("link").
             node("link", role="spine_leaf")"
```

Listing 24: Leafs and Spines using two queries Example

```
graph_query: ["node("system", role="leaf", name="system")",
             "node("system", role="spine", name="system)"]
```

Service name (service_name) Name of the custom collector service.

Keys 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.

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.

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.

Service Data Collector Example

```
service_name: "interface"
graph_query: "node("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.

Listing 25: Sample Data

```
[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

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.

Input Types - Discrete-Set (DSS), Number-Set (NS)

Output Types

Properties

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.

Set Comparison Example

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

The Set Count processor groups as described in **Group by**, then calculates the number of items in each group.

Input Types - Number-Set (NS), Discrete-State-Set (DSS), TS

Output Types - Number-Set (NS)

Properties

Group by (group_by) 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.

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](#) example for how this works.

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.

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.

Set Count Example

See [standard deviation example](#) - it is the same except we calculate the number of stage items.

Processor: Standard Deviation

The Standard Deviation processor groups as described by **Group by**, calculates the standard deviation, then outputs one standard deviation per group.

Input Types - Number-Set (NS), NSTS

Output Types - Number-Set (NS)

Properties

Group by (group_by) 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.

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](#) example for how this works.

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.

DDoF (ddof) Delta Degrees of Freedom, 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.

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.

Standard Deviation Example

```
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

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’). As of version 3.1 state processor can support multiple reference states and output is ‘true’ when input is in any of the specified states.

Input Types - Discrete-State-Set (DSS), DSTS

Output Types - Discrete-State-Set (DSS)

Properties

Graph Query (graph_query) One or more queries on graph specified as strings, or a list of such queries. (String will be deprecated in a future release.) Multiple queries should provide all the named nodes referenced by the expression fields (including additional_properties). Graph query is executed on the “operation” graph. Results of the queries can be accessed using the “query_result” variable with the appropriate index. For example, if querying property set nodes under name “ps”, the result will be available as “query_result[0][“ps”]”.

In collector processors (*_collector, if_counter) it is used to choose a set of nodes for further processing (for example, all leafs, or all interfaces between leaf and spines)

In other processors it is used for general parameterization and it is only supported as a list of queries.

Listing 26: Fabric Interfaces Example

```
graph_query: "node("system", role="leaf", name="system").
              out("hosted_interfaces").
              node("interface", name="iface").out("link").
              node("link", role="spine_leaf") "
```

Listing 27: Leafs and Spines using two queries Example

```
graph_query: ["node("system", role="leaf", name="system")",
              "node("system", role="spine", name="system")"]
```

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 (as of version 3.0). 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.

```
graph_query: [node("property_set", label="probe_propset", name="ps")]
duration: int(query_result[0]["ps"].values["accumulate_duration"])
```

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: “true” (expression evaluating to a string) or “[‘missing’, ‘unknown’, ‘down’]” (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.

State Example

```
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 N is created on output per each N 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.

Input Types - Number-Set (NS)

Output Types - Number-Set (NS)

Properties

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

The Sum processor groups as described by **Group by** property, then calculates sum and outputs one for each group.

Input Types - Number-Set (NS), NSTS

Output Types - Number-Set (NS)

Properties

Group by (`group_by`) 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.

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](#) example for how this works.

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.

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.

Sum Output Example

See [standard deviation example](#) - it is 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.

Input Types

Output Types

Properties

RX Property Name Property name from the parent stage having RX bits value

RX Utilization Property Name Property name from the parent stage having RX utilization value

TX Property Name Property name from the parent stage having TX bits value

TX Utilization Property Name Property name from the parent stage having TX utilization 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.

Processor: Time in State

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')

Input Types - Discrete-State(DS)

Output Types - Discrete-State (DS)

Properties

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.

Graph Query (`graph_query`) One or more queries on graph specified as strings, or a list of such queries. (String will be deprecated in a future release.) Multiple queries should provide all the named nodes referenced by the expression fields (including `additional_properties`). Graph query is executed on the "operation" graph. Results of the queries can be accessed using the "query_result" variable with the appropriate index. For example, if querying property set nodes under name "ps", the result will be available as "query_result[0]["ps"]".

In collector processors (`*_collector`, `if_counter`) it is used to choose a set of nodes for further processing (for example, all leafs, or all interfaces between leaf and spines)

In other processors it is used for general parameterization and it is only supported as a list of queries.

Listing 28: Fabric Interfaces Example

```
graph_query: "node("system", role="leaf", name="system").
  out("hosted_interfaces").
  node("interface", name="iface").out("link").
  node("link", role="spine_leaf")"
```

Listing 29: Leafs and Spines using two queries Example

```
graph_query: ["node("system", role="leaf", name="system")",
  "node("system", role="spine", name="system)"]
```

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 (as of version 3.0). 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.

```
graph_query: [node("property_set", label="probe_propset", name="ps")]
duration: int(query_result[0]["ps"].values["accumulate_duration"])
```

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 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.

Time in State Example

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: Union

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.

Input Types - Number-Set (NS), Discrete-State-Set (DSS), TS

Output Types - Number-Set (NS), Discrete-State-Set (DSS), TS

Properties

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.

Union Output Example

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.

Input Types

Output Types

Properties

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.

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.

4.3.5.6 Editing Probe

When you edit probes that are referenced in widgets you’ll get the message ‘Some widget(s) are currently using this probe. Editing this probe will affect those widget(s) and related dashboard(s)’.

1. From the list view (Analytics > Probes) or the details view, click the **Edit** button for the probe to edit.
2. Make your changes.
3. Click **Update** to stage the changes and return to the list view.

4.3.5.7 Deleting Probe

A probe that is used by a widget cannot be deleted.

1. From the list view (Analytics > Probes) or the details view, click the **Delete** button for the probe to delete.
2. Click **Delete Probe** to stage the deletion and return to the list view.

4.4 Staged

When resources and devices are assigned or changed in a blueprint they are visible in the staged view. Multiple changes can be viewed here before they are pushed to the active blueprint. This staging area allows you to validate that the pending changes are compliant with the intent, and that they work together with available resources and devices before you deploy the network.

4.4.1 Physical

You can build the network from any of the physical views (topology, nodes, links, racks, and on 5-stage topologies, pods). The topology view is shown by default.

4.4.1.1 Build (Physical)

Staging Resources

1. **Staged**

2. **Physical**

3. **Build**

4. **Resources**

5. **Reset resource group overrides**

6. **Update assignments**

When resources are staged, status indicator turns green

| Resource | Status |
|-------------------------------|--------|
| ASNs - Spines | 0/2 |
| ASNs - Leafs | 4/4 |
| Loopback IPs - Spines | 0/2 |
| Loopback IPs - Leafs | 0/4 |
| Link IPs - Spines<>Leafs | 0/16 |
| Link IPs - To External Router | 0/4 |

From the *blueprint*, navigate to **Staged / Physical / Build / Resources**. (The build panel is on the right side.)

Update Assignments

1. A red status indicator means that resources need to be assigned. Click the indicator, then click the **Update assignments** button.
2. Select a pool from which to pull the resources, then click the **Save** button. AOS assigns the required number of resources to the resource group automatically. When the red status indicator turns green, the resource assignment has been successfully staged.

You can also assign resources on a per-device basis (especially useful if you have a predefined resource mapping). Select the device from the topology view or nodes view, then assign the resource from the **Properties** section of the **Selection** panel (right-side). (This is also where you can see the specific resource that was assigned from a resource pool.)

Reset Resource Group Overrides

Certain blueprint operations require AOS to retain resource allocations even when a device has been removed from a blueprint. The resource groups are overridden, which mean that when a device is re-used, previously allocated resources are re-used as well. For example, if you've deleted a rack, then you rollback to a version with that rack, AOS must ensure that the same resources are used. Otherwise, the topology would change (for example, it might have different IP addresses). Situations like this can (but do not always) result in build errors. Some other examples of where resource group overrides are used include:

- Other time voyager rollbacks.
- *Revert* operations.
- Using the **Update Stated Cabling Map from LLDP** feature.

If you do not need to re-use the same resources, click the **Reset resource group overrides** button to reset the resource group.

Reverting Example

After reverting the changes in a blueprint, resources become “sticky” and even removing a pool or changing it to a new one doesn't affect the resulting config. In the blueprint below, we can see old resources still assigned.

The screenshot displays the Apstra UI interface. The top navigation bar includes tabs for Dashboard, Analytics, Staged, Uncommitted, and Active. A 'Revert' button is highlighted in the top right. Below the navigation bar, the 'Build Errors' section is visible, showing a list of errors related to interface name assignment and system ID settings. The bottom section of the image shows the 'Resource Pools' management interface. It includes a table with columns for Name, Role, Deploy Mode, Device Profile, S/N, Hostname, ASN, Loopback IPv4, and Loopback IPv6. The table lists three virtual leaf nodes, all of which are 'Not assigned' for the Loopback IPv4 and IPv6 addresses. A red box highlights the 'Loopback IPv4' column, and a red arrow points to a 'No pools assigned' message at the bottom right.

Build Errors:

- Node 6e8a7c7c-0d82-41f0-bbe7-377fecb149ca: Interface name assignment error: Interface with name "eth1", speed "10G" and role "leaf" not found in interface map
- Node 6e8a7c7c-0d82-41f0-bbe7-377fecb149ca: System ID must be set when deploy mode is "ready"/"deploy"/"drain"
- Node f2c1f599-dfb6-400d-be69-4f918bd6ebce: Interface name is required
- Node 4ae9065b-83b2-4f4b-a200-430273fc82f3: Interface name assignment error: Interface with name "eth1", speed "10G" and role "leaf" not found in interface map
- Node 4ae9065b-83b2-4f4b-a200-430273fc82f3: System ID must be set when deploy mode is "ready"/"deploy"/"drain"
- Node cf516910-c374-4af8-aff4-4764dfb3b93: Interface name is required
- Node 2c31b4a1-96a4-4d4f-aecc-cfba9991bfae: Interface name is required
- Node a71563cc-587e-4624-8a09-c98951ce4126: Interface name is required
- Node 3d32355e-23ca-4b7f-9b53-86e317c4f0c1: Interface name is required
- Node 64fc9307-b5ef-4a44-beb6-8f6f876067fd: Interface name assignment error: Interface with name "eth1", speed "10G" and role "leaf" not found in interface map
- Node 1e44f799-4e05-45f3-90d4-9fd2d1836b2b: Interface name assignment error: Interface with name "eth1", speed "10G" and role "leaf" not found in interface map
- Node 1e44f799-4e05-45f3-90d4-9fd2d1836b2b: System ID must be set when deploy mode is "ready"/"deploy"/"drain"
- Node 6a487056-e6a2-423b-b255-89310e46c11c: Interface name assignment error: Interface with name "eth1", speed "10G" and role "leaf" not found in interface map
- Node 96f276ad-6b96-44fc-9a85-c662ce3b672b: Interface name assignment error: Interface with name "eth1", speed "10G" and role "leaf" not found in interface map
- Node 3fc3ded2-ca83-412e-87df-656ee7b8c1b1: Interface name is required

Resource Pools Table:

| Name | Role | Deploy Mode | Device Profile | S/N | Hostname | ASN | Loopback IPv4 | Loopback IPv6 |
|----------------------|------|-------------|----------------|--------------|----------------------|-------|----------------|---------------|
| I3_virtual_001_leaf1 | Leaf | Deploy | Arista vEOS | 505400CFD38E | I3-virtual-001-leaf1 | 64514 | 192.168.0.2/32 | Not assigned |
| I3_virtual_002_leaf1 | Leaf | Deploy | Arista vEOS | 50540085C308 | I3-virtual-002-leaf1 | 64515 | 192.168.0.3/32 | Not assigned |
| I3_virtual_003_leaf1 | Leaf | Deploy | Arista vEOS | 505400AD7D45 | I3-virtual-003-leaf1 | 64516 | 192.168.0.4/32 | Not assigned |

Resource Pools List:

- 2/2 ASNs - Spines
- 3/3 ASNs - Leaves
- 6/6 ASNs - L3 servers
- 2/2 Loopback IPs - Spines
- 3/3 Loopback IPs - Leaves

Message: No pools assigned

Click the **Reset resource group overrides** button. Then resources can be unallocated, and new ones can be allocated, as applicable.

Manage Resource Pools

The resource assignment section has a convenient shortcut button, **Manage resource pools**, that takes you to resource pool management. There, you can monitor resource usage (new in AOS version 3.3.0) and create additional resource pools, as needed.

Staging Device Profiles

1. From the *blueprint*, navigate to **Staged / Physical / Build / Device Profiles**.

The screenshot shows the Apstra interface with the following elements and annotations:

- Top Navigation Bar:** Dashboard, Analytics, **Staged** (annotated with red arrow 1), Uncommitted, Active, Time Voyager.
- Left Sidebar:** Physical (annotated with red arrow 2), Virtual, Policies, Catalog, Settings, Tasks.
- Main Panel:**
 - Selection / Build:** (Annotated with red arrow 3)
 - Device Profiles:** (Annotated with red arrow 4)
 - Build: Device Profiles - AOS-7x10-Spine:** (Annotated with red arrow 5)
 - Change interface map assignments:** (Annotated with red arrow 6, pointing to a pencil icon)
- Table:**

| Node Name | Device Profile |
|-----------|----------------|
| spine1 | Not assigned |
| spine2 | Not assigned |
- Note:** When interface maps are staged, status indicator turns green (pointing to a green status indicator for 'AOS-7x10-Leaf').

2. Click a red status indicator, then click the **Change interface maps assignment** button (looks like an edit button). Device profiles are staged 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, you can select the ones that use the same interface map (or all of them with one click), then select the interface map from the drop-down list located above the selections, and click **Assign Selected**.

Update interface map for AOS-7x10-Leaf ✕

To assign more than one at a time...

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

1. Select multiple devices

Interface Map

Cumulus_VX_AOS-7x10-Leaf ✕

Assign Selected 3.

2. Select interface map

| <input type="checkbox"/> | Name | Interface Map | Device Profile |
|-------------------------------------|----------------------|---------------|----------------|
| 3 selected | | | |
| <input checked="" type="checkbox"/> | I2_virtual_001_leaf1 | Select... | N/A |
| <input checked="" type="checkbox"/> | I2_virtual_002_leaf1 | Select... | N/A |
| <input type="checkbox"/> | I2_virtual_003_leaf1 | Select... | N/A |
| <input checked="" type="checkbox"/> | I2_virtual_004_leaf1 | Select... | N/A |

Or select one at a time from these drop-down lists

Update Assignments

- Click **Update Assignments**. When the red status indicator turns green, the device profile assignments have been successfully staged.

Staging Devices

Devices must have interface maps associated with them before they can have system IDs assigned to them. See the previous section for details. When a device is assigned to a blueprint, it performs discovery configuration. During this phase all interfaces are changed to L3-only mode allowing interfaces to be *up*. There is no BGP configuration, no routing expectations, nothing that can influence the network. A device in *discovery* mode is benign; it does not participate in the datacenter fabric, and it does not forward any packets through it. You can then perform critical validations of network health including viewing statistics for cabling, LLDP, transceivers and more. Any issues, such as miscabling or physical link errors, cause a telemetry alarm. You can address and correct the anomalies *before* deploying the device.

Sanitize the Device

When resetting the system ID (serial number) Discovery-1 is re-applied. It is good practice to fully erase the device configuration and uninstall the AOS device agent before physically uninstalling the device.

Assigning System IDs and Deploy Modes to Multiple Devices

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 Apstra interface with the following components and annotations:

- Top Navigation:** Dashboard, Analytics, Staged (selected), Uncommitted, Active, Time Voyager.
- Left Sidebar:** Physical (selected), Virtual, Policies, Catalog, Settings, Tasks.
- Annotations:**
 - 1.** Points to the 'Staged' tab in the top navigation.
 - 2.** Points to the 'Physical' tab in the left sidebar.
 - 3.** Points to the 'Build' button in the top right of the main area.
 - 4.** Points to the 'Devices' button in the top right of the main area.
 - 5.** Points to the 'Assigned System IDs' button in the top right of the main area.
 - 6.** Points to the 'Change System IDs assignments' button in the top right of the main area.
- Main Content Area:**
 - Nodes:** pod1
 - Links:** All
 - Layer:** Build: System IDs
 - Topology:** Grouped (selected), Compact, Full
 - Selected Rack:** All
 - Selected Node:** All
 - Show Servers?** (checkbox)
 - Nodes List:**
 - spines: 2 yellow squares (Not Assigned)
 - evpn_mlag_001: 2 yellow squares (Not Assigned)
 - evpn_single_001: 1 green square (Assigned)
- Assigned System IDs Panel:**
 - Node:** spine1, spine2, evpn_mlag_001_leaf1, evpn_mlag_001_leaf2, evpn_single_001_leaf1, evpn_mlag_001_server001, evpn_mlag_001_server002, evpn_mlag_001_server003, evpn_single_001_server001
 - System ID:** Not assigned, Not assigned, Not assigned, Not assigned, 505400E8A335, Not assigned, Not assigned, Not assigned, 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.

Assign Systems

Query: All 1-9 of 9 < >

| Name | Role | Hostname | System ID | Deploy Mode |
|---------------------|-------|---------------------|---|---|
| spine1 | Spine | spine1 | 52540029EF4D (172.20.80.11) | <input checked="" type="radio"/> Deploy <input type="radio"/> Ready <input type="radio"/> Drain <input type="radio"/> Undeploy |
| spine2 | Spine | spine2 | <div> 525400E8ABA3 (172.20.80.12) 525400E8ABA3 (172.20.80.12) 525400DAC745 (172.20.80.15) 525400948215 (172.20.80.13) </div> | <input type="radio"/> Deploy <input type="radio"/> Ready <input type="radio"/> Drain <input type="radio"/> Undeploy |
| evpn_mlag_001_leaf1 | Leaf | evpn-mlag-001-leaf1 | <div> 525400E8ABA3 (172.20.80.12) 525400DAC745 (172.20.80.15) 525400948215 (172.20.80.13) </div> | <input type="radio"/> Deploy <input type="radio"/> Ready <input type="radio"/> Drain <input type="radio"/> Undeploy |
| evpn_mlag_001_leaf2 | Leaf | evpn-mlag-001-leaf2 | Select... | <input type="radio"/> Deploy <input type="radio"/> Ready <input type="radio"/> Drain <input type="radio"/> Undeploy |

Update Assignments

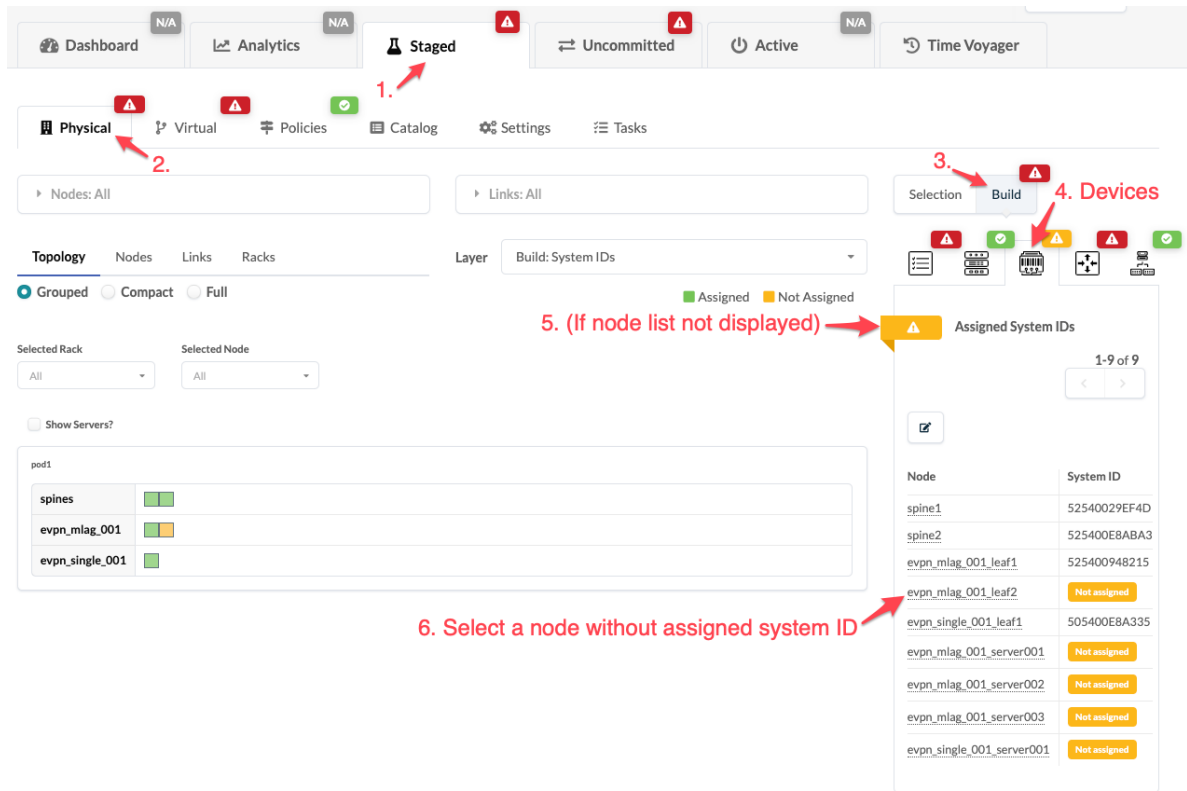
Tip: If you don't see an expected serial number (system ID), you may still need to *acknowledge the device* (Devices / Managed Devices).

- When a system ID is selected, the deploy mode changes to **Deploy** by default. If you don't want to stage the device to be deployed yet, change the deploy mode here. When you're ready to deploy the device, return here to set the deploy mode back to **Deploy**. See the next section for more information about deploy modes.
- 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.

Note: You can also use AOS CLI to bulk-assign system IDs to devices either with a CSV text file or the `blueprint set-serial-numbers` command.

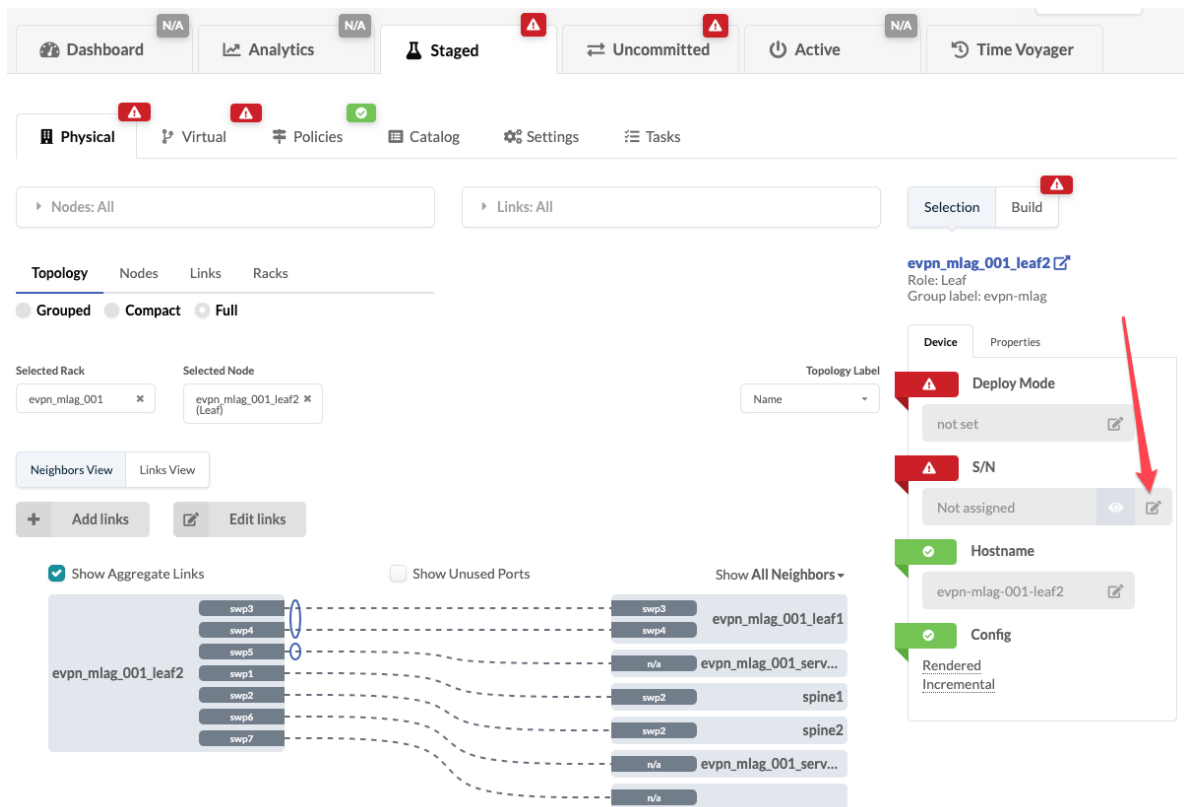
Assigning System IDs to One Device

- 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 (deploy mode, serial number, hostname rendered, incremental and pristine config, as applicable).

Tip: Another way to access these device attributes is to click **Selected Nodes** (left-middle), then select a node name in the drop-down list.



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.

Tip: If you don't see the expected serial number (system ID), you may still need to *acknowledge the device* (Devices / Managed Devices).

4. If you want 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.

Changing Deploy Mode on One Device

1. From the blueprint, navigate to **Staged / Physical / Build / Devices**; if you don't see the nodes list, click the status indicator for **Assigned System IDs**.
2. Click a node name to see device details.
3. Click the **Edit** button for **Deploy Mode** and select a deploy mode.
 - Deploy - Adds service configuration and puts the device fully in service.
 - Ready - Adds discovery 2 configuration (hostnames, interface descriptions, port speeds / breakouts). Changing from deploy to ready removes service configuration.
 - Drain - Takes a device out of service gracefully for maintenance. See *Draining Device Traffic* for more information.

Tip: 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 re-

solved. To ensure switches are completely drained before undeploying them, you could *instantiate* the drain validation dashboard to monitor progress.

- Undeploy - Removes AOS-rendered configuration. If a device is carrying traffic it is best to first put the device into drain mode (and commit the change). When the device is completely drained, proceed to undeploy the device.
4. Click the **Save** button to stage the change.
 5. When you are ready, *commit* the changes.

Tip: You can also change deploy modes from **Staged / Physical / Nodes**. See *Setting Deploy Mode for Multiple Nodes* for more information.

Changing Hostname on One Device

1. From the blueprint, navigate to **Staged / Physical / Build / Devices**; if you don't see the nodes list, click the status indicator for **Assigned System IDs**.
2. Click a node name to see device details.
3. Click the **Edit** button for **Hostname**, change the name, and click the **Save** button to stage the change.
4. *Commit* the changes.

Staging External Routers

1. Click the **Physical** tab.

2. Click the **Nodes: All** dropdown.

3. Click the **Build** tab.

4. Click the **External Routers** section.

5. Click the **Manage External Routers** button.

6. Click the **Import external router directly if not already imported from catalog** button.

7. Click the **Edit links to stage external router** button.

1. Make sure that the appropriate external routers have been *imported* into the blueprint catalog from the global catalog.

2. From the *blueprint*, navigate to **Staged / Physical / Build / External Routers**.
3. Click the red status indicator for **External Router Links**. If the external router has not been imported yet, you can click the **Import External Router** button, select the external router from the drop-down list, and click **Import External Router**.
4. Click the **Edit Links** button, then select the external router from the drop-down list.
5. Select **Connectivity Type**.
 - L3 (default) - for BGP loopback (eBGP multi-hop) peering
 - L2 - for BGP interface on L2 SVI peering. Overlay control protocol must be MP-EBGP EVPN and external router connections must be leaf. Spine external router connections are not supported.
6. Select external router links. By default, all available external router interface links are selected. If more than one external router is imported, you can add and remove links as needed for each external router. All defined external router interface links must be configured before the blueprint can be deployed.
7. Click **Update** to stage the external router and return to the list view.
8. L3 connectivity requires additional IP resources for **Link IPs - To External Router** for the default security zone. See **Staging Physical Resources** above for steps.

If you're connecting external routers in an EVPN blueprint, you must add external connectivity points to the security zone (as of version 3.0).

If you are connecting OSPF external routers in a blueprint, you must add OSPF external connectivity points to the security zone (as of version 3.2).

For more information, see [Adding Connectivity Point to Security Zone](#).

Staging Configlets

Configlets are vendor-specific. AOS automatically ensures that configlets of a specific vendor are not assigned to devices from a different vendor.

1. Make sure that the appropriate configlets have been *imported* 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. See *Importing Configlet* for details.
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**.

4.4.1.2 Topology

Topology Overview

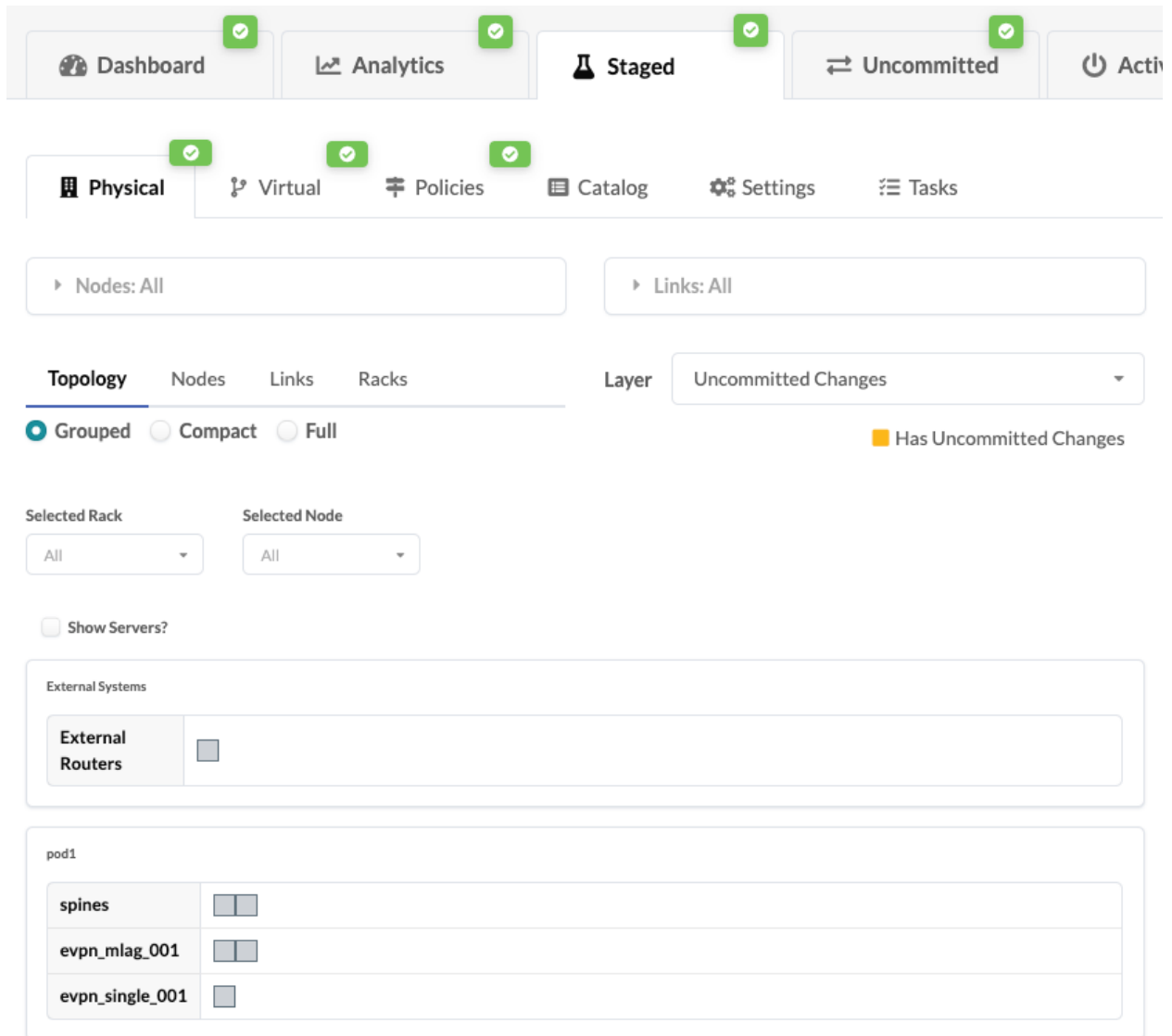
Topologies can be viewed at three levels of detail: grouped, compact, and full. (Grouped and compact are new in AOS version 3.3.0.)

When a node is selected from the topology view (by clicking its element in the topology, or by selecting it from the **Selected Nodes** drop-down list), the node topology is shown. The selected node can be viewed in two different ways: neighbors view and links view.

From the blueprint, navigate to **Staged / Physical / Topology**.

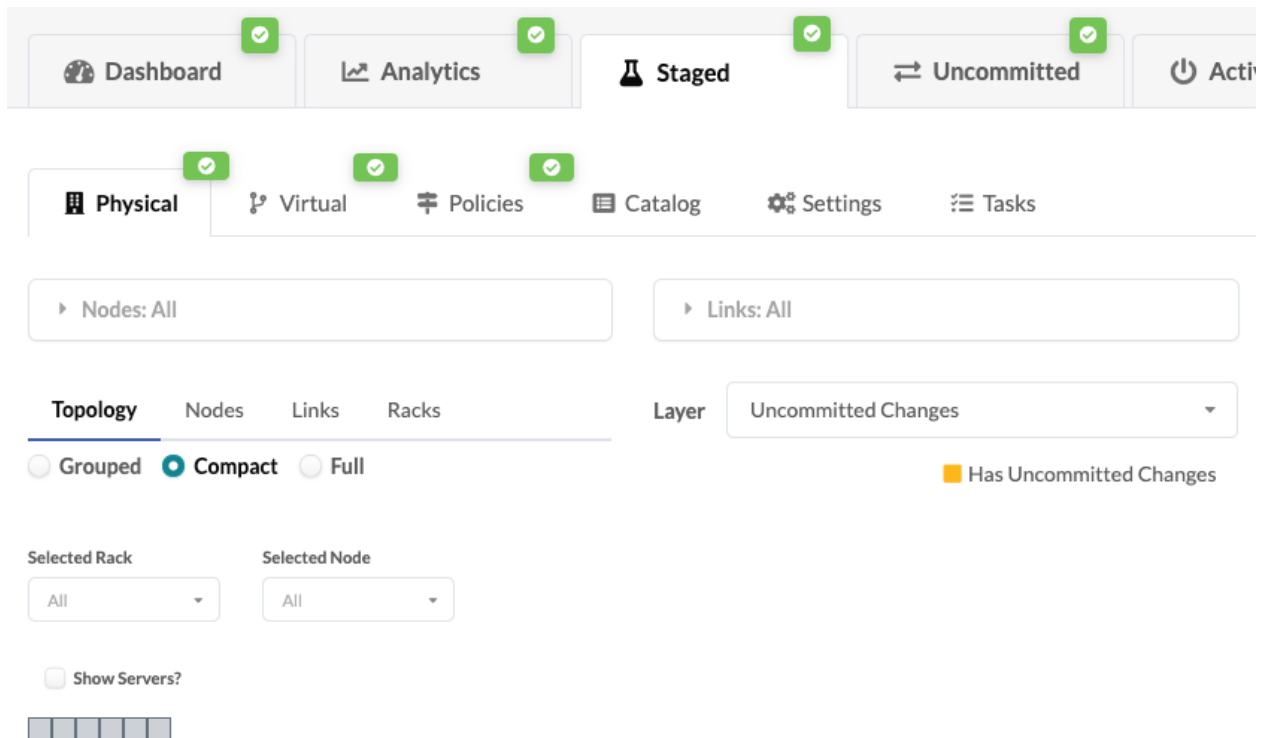
Grouped

- To show servers, click the **Show Servers** check box.
- To show node name (and hostname as applicable) hover over an element (square).
- To show node details, select the node by either clicking its element or by selecting it from the **Selected Node** drop-down list.



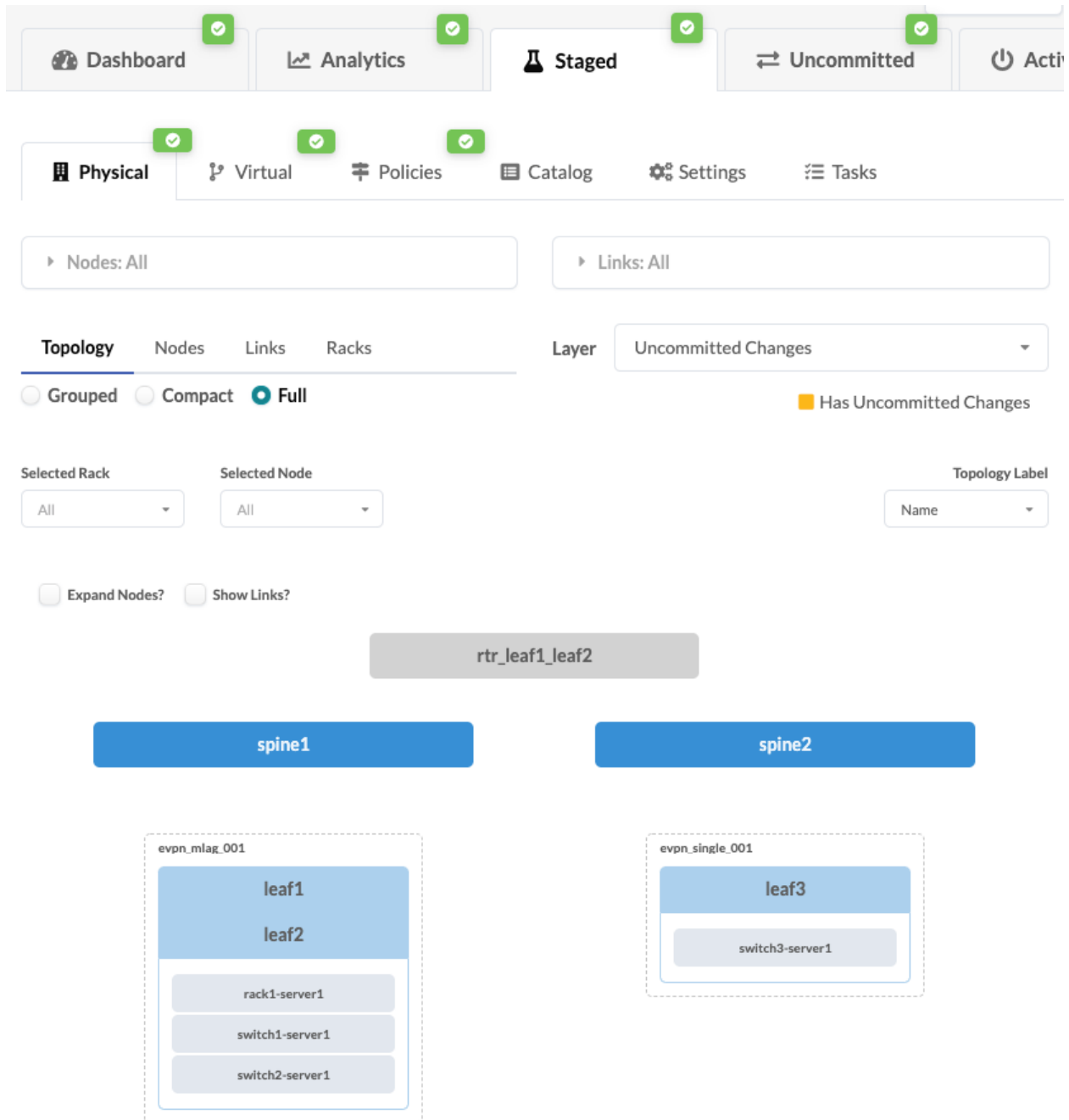
Compact

- To show servers, click the **Show Servers** check box.
- To show node name (and hostname as applicable) hover over an element (square).
- To show node details, select the node by either clicking its element or by selecting it from the **Selected Node** drop-down list.



Full

- 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 (and hostname as applicable) hover over an element.
- To change the labels (name, hostname, S/N) that are shown in the topology, select a different label from the **Topology Label** drop-down list.
- To show node details, select the node by either clicking its element or by selecting it from the **Selected Node** drop-down list.



Neighbors View

- Aggregated server-leaf links are encircled (new in AOS version 3.3.0).
- To show unused ports, click the **Show Unused Ports** check box (new in AOS version 3.3.0).
- To change the labels (name, hostname, S/N) that are shown in the topology, select a different label from the **Topology Label** drop-down list.
- Choose to show all neighbors or only specific types (leaf, spine, L2 server, access, etc.).

Dashboard

Analytics

Staged

Uncommitted

Active

Time Voyager

Physical

Virtual

Policies

Catalog

Settings

Tasks

Nodes: Role = L2 server

Links: All

Topology

Nodes

Links

Racks

Pods

Grouped

Compact

Full

Selected Plane

Selected Pod

Selected Rack

Selected Node

Topology Label

Neighbors View

Links View

Add links

Edit links

Show Aggregate Links

Show Unused Ports

Show All Neighbors

leaf001_014_1

Ethernet1

Ethernet2

Ethernet3

Ethernet4

Ethernet5

Ethernet6

Ethernet7

Ethernet8

Ethernet9

Ethernet10

Ethernet49/1

Ethernet50/1

Ethernet51/1

Ethernet52/1

Ethernet53/1

Ethernet54/1

leaf001_014_2

access001_014_1

spine001_001_1

spine001_001_2

spine001_001_3

Links View

Dashboard

Analytics

Staged

Uncommitted

Active

Time Voyager

Physical

Virtual

Policies

Catalog

Settings

Tasks

Nodes: All

Links: All

Topology

Nodes

Links

Racks

Grouped

Compact

Full

Selected Rack

Selected Node

Topology Label

Neighbors View

Links View

+ Add links

Edit links

Query: All

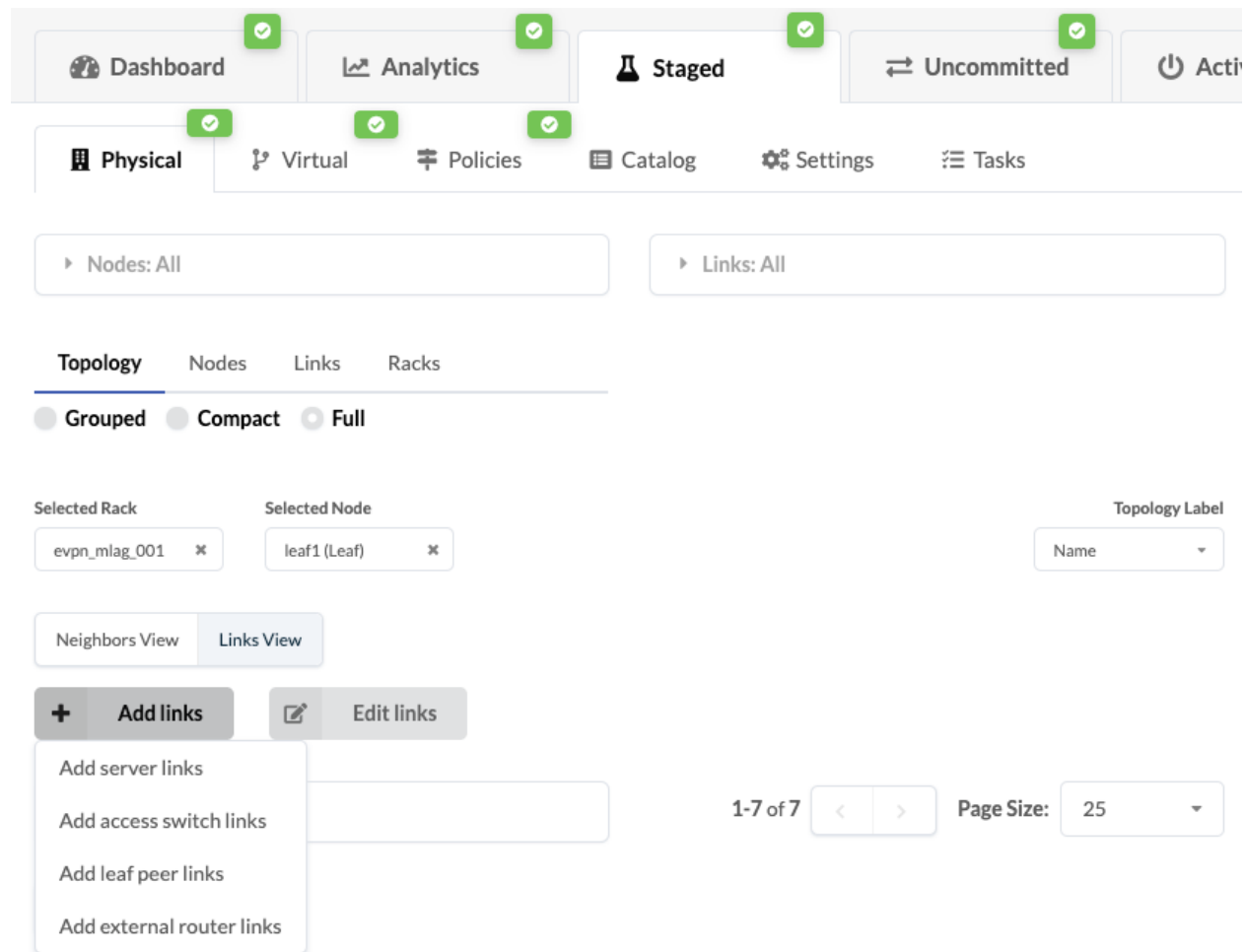
1-7 of 7

Page Size: 25

0 selected

| | Name | Role | Speed | Logical Link | Port Channel ID | Endpoint 1 | | | | Endpoint 2 | | | | Actions |
|--------------------------|---|-------------------|-------|--------------|-----------------|------------|------|-----------|---------------|-----------------|-----------|-----------|---------------|----------------------|
| | | | | | | Name | Role | Interface | Lag Mode | Name | Role | Interface | Lag Mode | |
| <input type="checkbox"/> | evpn_mlag_001_leaf1<->evpn_mlag_001_server001(dual-link)[1] | Leaf to L2 Server | 10G | dual-link | 1 | leaf1 | Leaf | swp4 | LACP (Active) | rack1-server1 | L2 server | n/a | LACP (Active) | <div>>> </div> |
| <input type="checkbox"/> | evpn_mlag_001_leaf1<->evpn_mlag_001_server002(single-link)[1] | Leaf to L2 Server | 10G | single-link | N/A | leaf1 | Leaf | swp7 | No LAG | switch1-server1 | L2 server | n/a | No LAG | <div>>> </div> |
| <input type="checkbox"/> | evpn_mlag_001_leaf1<->evpn_mlag_001_leaf2(l3_peer_link)[1] | Leaf L3 Peer Link | 10G | l3_peer_link | 3 | leaf1 | Leaf | swp5 | N/A | leaf2 | Leaf | swp5 | N/A | <div>>> </div> |
| <input type="checkbox"/> | evpn_mlag_001_leaf1<->evpn_mlag_001_leaf2[1] | Leaf Peer Link | 10G | N/A | 2 | leaf1 | Leaf | swp6 | N/A | leaf2 | Leaf | swp6 | N/A | <div>>></div> |

Adding Links



Tip: When adding a link, you can review the config changes in the *incremental config* to evaluate the changes.

Add Server Links

You can add L2 servers or L3 servers to a running blueprint by adding a server link (as of AOS version 3.2). You can link to a new server, or to an existing one to make it dual-homed. In an MLAG configuration, server links and external links must be symmetrical to both switches in the MLAG. When changing a server from single- to dual-homed, both links must be of the same speed.

1. From the blueprint, navigate to **Staged / Physical / Topology** and make a selection.
2. From the neighbors view or the links view of a leaf selection, click **Add links**, then select **Add server links**.
3. Click in the **Server** field.
 - To make an existing server dual-homed, select it from the drop-down list.
 - For a new server, click **Add new server**.
4. In the port layout, select an unused server port on the leaf logical device.

Add Leaf to Server Links ✕

Select devices and their interfaces to create a link:

Leaf: leaf3
Device profile: Arista vEOS

1 2 3 4 5 6 7

3. Select an available port

Add Link →

Server *

1. 2.

Add new server

switch3-server1 Device Profile: N/A

Links

Logical Link Name[Ⓢ] *

Select from dropdown or type a new name...

LAG Mode *

☐ LACP (Active)[Ⓢ]

☐ LACP (Passive)[Ⓢ]

☐ Static LAG (no LACP)[Ⓢ]

☒ No LAG[Ⓢ]

1-1 of 1 < >

| Type | Speed | leaf | | Logical Link | LAG Mode | server | | Actions |
|----------|-------|-------|-----------|--------------|----------|-----------------|-----------|---------|
| | | Name | Interface | | | Name | Interface | |
| Existing | 10G | leaf3 | Ethernet3 | single-link | No LAG | switch3-server1 | N/A | |

Add

5. Select the interface.
6. For **Select Source**, select **Logical Device**.
7. In the drop-down list, select the appropriate server logical device.
8. Click **Add Link**.

Add Leaf to Server Links ✕

Select devices and their interfaces to create a link:

Leaf: leaf3
Device profile: Arista vEOS

1 2 3 4 5 6 7

4.

Add Link →

Port #7 Tr. #1 (10 Gbps, default) Ethernet7

1.

Server *

Add new server

Select Source *

☐ Interface Map ☒ Logical Device

☐ Show whole catalog

2. 3.

AOS-1x1-1

AOS-1x1-1

AOS-1x10-1

AOS-1x10-1 [embedded]

Links

Logical Link Name[Ⓢ] *

Select from dropdown or type a new name...

LAG Mode *

☐ LACP (Active)[Ⓢ]

☐ LACP (Passive)[Ⓢ]

☐ Static LAG (no LACP)[Ⓢ]

☒ No LAG[Ⓢ]

1-1 of 1 < >

| Type | Speed | leaf | | Logical Link | LAG Mode | server | | Actions |
|----------|-------|-------|-----------|--------------|----------|-----------------|-----------|---------|
| | | Name | Interface | | | Name | Interface | |
| Existing | 10G | leaf3 | Ethernet3 | single-link | No LAG | switch3-server1 | N/A | |

Add

9. If you're adding a new server, enter a **System Group Name**.
10. Select/enter a **Logical Link Name** and **LAG Mode**.
11. Click **Add** to stage the addition.

Add Access Switch Links

Access switches have limited support. Apstra does not recommend using this feature in a production network. See *Adding Access Layer Switches* for more information.

Add Leaf Peer Links

You can add MLAG peer links as of AOS version 3.2.

Important: Do not use MLAG peer Links if the platform used does not support it. Currently, SONiC devices do not support MLAG L3 peers.

1. From the blueprint, navigate to **Staged / Physical / Topology**.
2. Select a leaf that is an MLAG member.
3. From the neighbors view or the links view of the selection, click **Add links**, then click **Add leaf peer links**.
4. Select the link type.
 - Peer Link - adds a link between the two leafs and adds a BGP session (between the two leafs) in the default vrf only.
 - L3 Peer Link - adds a link between the two leafs and adds a BGP session (between the two leafs) in the default vrf and in non-default vrfs.
5. Select an unused port for each MLAG member. (Only unused ports are selectable). When a port has been selected for each MLAG member, the **Add Link** button becomes clickable.
6. Click **Add Link**.
7. Click **Add** to stage the addition.

Add External Router Links

You can import external routers and assign external router links from the topology selection view (as of AOS version 3.2).

1. From the blueprint, navigate to **Staged / Physical / Topology** and make a selection.
2. From the neighbors view or the links view of a selection, click **Add links**, then click **Add external router links**.

Add External Router Links ✕

Select interface to create a link:
 Leaf: leaf3
 Device profile: Arista vEOS

1. Select port

Port #6 Tr. #1
 (10 Gbps, default)

Ethernet6

2. Click... →

3. Please select external router

4. Connectivity Type
☐ L2 ☒ L3

5. Add

Add Link →

1-1 of 1

| Label | Interface | Speed | Actions |
|-------|-----------|-------|---------|
| leaf3 | Ethernet7 | 10G | |

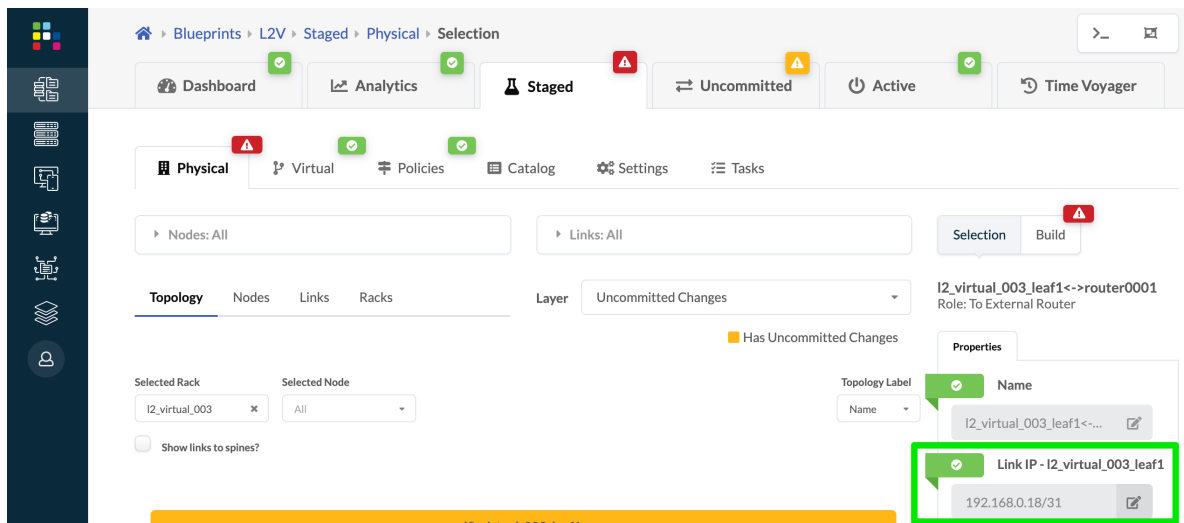
... to add to list →

3. Select an appropriate port on the leaf switch with the **External Router** role (defined in the respective interface map), then click **Add Link**.
4. If using an MLAG, select a port on the other MLAG member.
5. Add additional links, as needed.
6. Select the external router from the drop-down list. If the external router that you need has not been imported yet, check the **Show routers from Global Catalog** check box, then select the external router. It will be imported into the blueprint.
7. Choose Layer 2 or Layer 3 connectivity.
8. Click **Add** to add stage the addition.

Manually Overriding IP Addresses on External Links

When the remote end has already been configured and cannot be changed, you may want to manually override the external link IP addresses.

1. From the blueprint, navigate to **Staged / Physical / Topology**.
2. Select the link to modify.

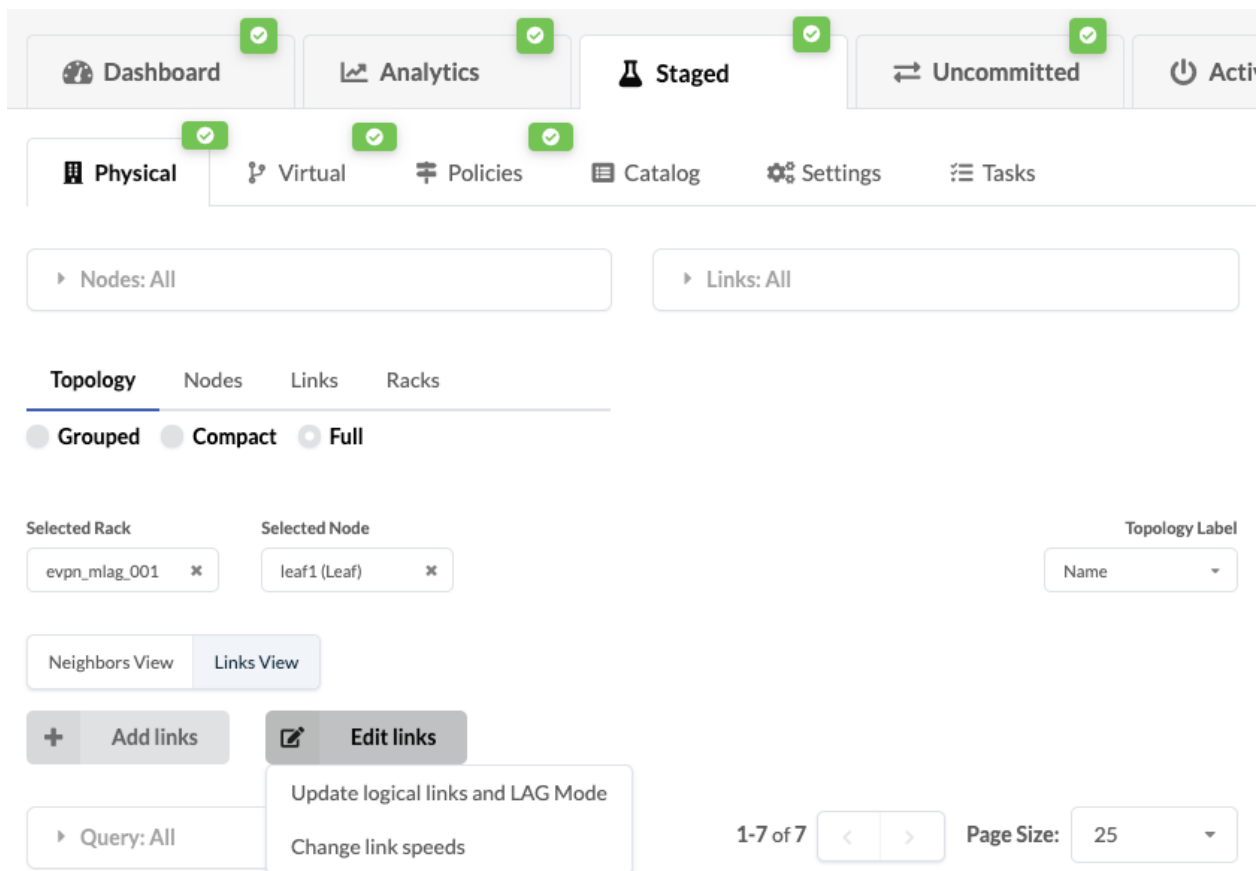


3. Click the **Edit** button and make your changes.

4. Click **Save** to stage the changes.

If you are using external routers in an EVPN blueprint, you must add external connectivity points (ECPs) in the security zone.

Editing Links



Update Logical Links

To edit the server logical link for existing leaf-server links follow the steps below.

1. From the blueprint, navigate to **Staged / Physical / Topology** and make a selection.
2. From the neighbors view or the links view of a selection, click **Edit links**, then click **Update logical links and LAG Mode**.
3. Change logical link, as needed.

Update Logical Links and LAG Mode

Specify the same logical link name for links to bond them. All links with the same label must share the same LAG setting.

Query: All

1-2 of 2

Page Size: 10

| | Name | Role | Speed | Logical Link | Port Channel ID | LAG Mode | Endpoint 1 | | | Endpoint 2 | | |
|--------------------------|---|-------------------|-------|--------------|-----------------|----------------|------------|------|-----------|-----------------|-----------|-----------|
| | | | | | | | Name | Role | Interface | Name | Role | Interface |
| <input type="checkbox"/> | evpn_mlag_001_leaf1<->evpn_mlag_001_server001(dual-link)[1] | Leaf to L2 Server | 10G | dual-link | 1 | LACP (Active) | leaf1 | Leaf | swp4 | rack1-server1 | L2 server | n/a |
| <input type="checkbox"/> | evpn_mlag_001_leaf1<->evpn_mlag_001_server002(single-link)[1] | Leaf to L2 Server | 10G | single-link | N/A | LACP (Passive) | leaf1 | Leaf | swp7 | switch1-server1 | L2 server | n/a |

Update

4. Click **Update** to stage the change.

Update LAG Mode

The LAG mode between a server and a leaf can be changed. A common example is if an individual dual-connected server was configured with LACP, and you want it to have separate, individually configured layer-2 links. If the server links are in a “LAG” mode (e.g. **LACP (Active)**), and you would like to change it to **No LAG**, follow the next steps:

1. From the blueprint, navigate to **Staged / Physical / Topology** and make a selection.
2. From the neighbors view or the links view of a selection, click **Edit links**, then click **Update logical links and LAG Mode**.
3. Select the involved links to be changed and choose the new LAG Mode per link based or from the **Apply to selection** option (see picture below). LAG Mode must be **No LAG**.
4. Change the **Logical Link** to different values across the links as it is highlighted in the example from the picture below. If any of the **Logical Link** values are the same when changing the LAG Mode to **No LAG**, you will receive an error.
5. When the **Logical Link** values are changed, all virtual network assignments for the port(s) are lost and will need to be reassigned. If changing a dual-connected server to LAG Mode **No LAG**, it is recommended to only change the second **Logical Link** value. Virtual network assignments will be retained on the first port.
6. Click **Update** and verify the new settings before committing the change.

Update Logical Links and LAG Mode

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

Query: All 1-2 of 2 Page Size: 10

| 2 selected | Name | Role | Speed | Logical Link | Port Channel ID | LAG Mode | Endpoint 1 | | | Endpoint 2 | | |
|-------------------------------------|--|-------------------|-------|--------------|-----------------|----------|----------------------|-----------|-----------|------------------|------|-----------|
| | | | | | | | Name | Role | Interface | Name | Role | Interface |
| <input checked="" type="checkbox"/> | rack_1_001_leaf1<->rack_1_001_server001(link)[1] | Leaf to L2 Server | 10G | link | 1 | No LAG | rack_1_001_server001 | L2 server | n/a | rack_1_001_leaf1 | Leaf | Ethernet4 |
| <input checked="" type="checkbox"/> | rack_1_001_leaf2<->rack_1_001_server001(link)[1] | Leaf to L2 Server | 10G | link2 | 1 | No LAG | rack_1_001_server001 | L2 server | n/a | rack_1_001_leaf2 | Leaf | Ethernet4 |

Update

If you would like to change from *no LAG* links to *LAG Mode*, follow the same steps, set the same **Logical Link** values in all the links belonging to the same group and select the desired *LAG mode*. Again, changing the **Logical Link** value will result in virtual network assignment to be lost. However, using an existing value will retain the virtual network assignment.

Change Link Speeds

1. From the blueprint, navigate to **Staged / Physical / Topology** and make a selection.
2. From the neighbors view or the links view of a selection, click **Edit links**, then click **Change link speeds**.

Change Link Speeds

Query: All 1-5 of 5 Page Size: 10

| 0 selected | Name | Role | Speed | Logical Link | Port Channel ID | Endpoint 1 | | | Endpoint 2 | | |
|--------------------------|---|--------------------|---------|--------------|-----------------|------------|------|-----------|-----------------|-----------------|-----------|
| | | | | | | Name | Role | Interface | Name | Role | Interface |
| <input type="checkbox"/> | evpn_mlag_001_leaf2<->evpn_mlag_001_server001(dual-link)[1] | Leaf to L2 Server | 10 Gbps | dual-link | 1 | leaf2 | Leaf | swp2 | rack1-server1 | L2 server | n/a |
| <input type="checkbox"/> | evpn_mlag_001_leaf2<->evpn_mlag_001_server003(single-link)[1] | Leaf to L2 Server | 10 Gbps | single-link | N/A | leaf2 | Leaf | swp3 | switch2-server1 | L2 server | n/a |
| <input type="checkbox"/> | evpn_mlag_001_leaf1<->evpn_mlag_001_leaf2(l3_peer_link)[1] | Leaf L3 Peer Link | 10 Gbps | l3_peer_link | 3 | leaf2 | Leaf | swp5 | leaf1 | Leaf | swp5 |
| <input type="checkbox"/> | evpn_mlag_001_leaf1<->evpn_mlag_001_leaf2[1] | Leaf Peer Link | 10 Gbps | N/A | 2 | leaf2 | Leaf | swp6 | leaf1 | Leaf | swp6 |
| <input type="checkbox"/> | evpn_mlag_001_leaf2<->router0002 | To External Router | 10 Gbps | N/A | 4 | leaf2 | Leaf | swp1 | rtr_leaf1_leaf2 | External Router | eth1 |

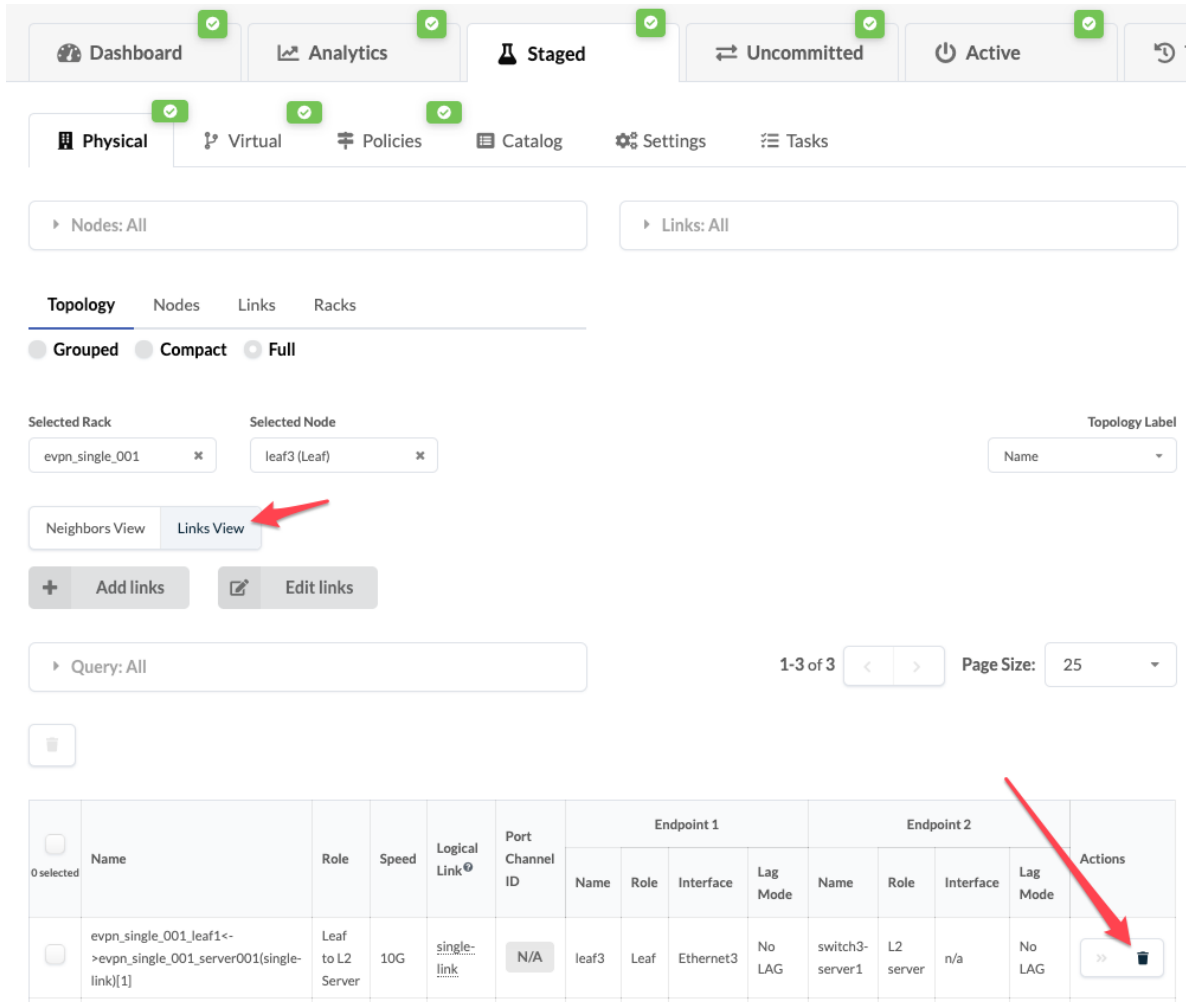
Update

3. Change link speeds, as needed.
4. Click **Update** to stage the change.

Deleting Links

External router links, server links, and MLAG peer links can be removed as of AOS version 3.2.

1. From the blueprint, navigate to **Staged / Physical / Topology** and make a selection.
2. From the links view of a selection, click the **Delete** button for the link to remove.



The screenshot shows the Apstra interface for the 'Staged' view. The top navigation bar includes 'Dashboard', 'Analytics', 'Staged', 'Uncommitted', and 'Active'. The left sidebar shows 'Physical', 'Virtual', 'Policies', 'Catalog', 'Settings', and 'Tasks'. The main content area is titled 'Nodes: All' and 'Links: All'. Below this, there are tabs for 'Topology', 'Nodes', 'Links', and 'Racks'. The 'Topology' tab is active, and the 'Full' view is selected. The 'Selected Rack' is 'evpn_single_001' and the 'Selected Node' is 'leaf3 (Leaf)'. The 'Topology Label' is 'Name'. The 'Neighbors View' and 'Links View' tabs are shown, with 'Links View' selected and highlighted by a red arrow. Below the tabs are 'Add links' and 'Edit links' buttons. A 'Query: All' search bar is present. The table below shows a single link with the following details:

| | Name | Role | Speed | Logical Link | Port Channel ID | Endpoint 1 | | | | Endpoint 2 | | | | Actions |
|------------|---|-------------------|-------|--------------|-----------------|------------|------|-----------|----------|-----------------|-----------|-----------|----------|-------------|
| | | | | | | Name | Role | Interface | Lag Mode | Name | Role | Interface | Lag Mode | |
| 0 selected | evpn_single_001_leaf1<->evpn_single_001_server001(single-link)[1] | Leaf to L2 Server | 10G | single-link | N/A | leaf3 | Leaf | Ethernet3 | No LAG | switch3-server1 | L2 server | n/a | No LAG | >> [Delete] |

3. Click **Delete** to stage the deletion.

4.4.1.3 Nodes

From the staged nodes view you can edit various node properties and device details, and access device configuration (rendered, incremental, pristine).

The screenshot shows the Apstra interface with the following components and annotations:

- Top Navigation:** Dashboard, Analytics, **Staged** (highlighted with a red arrow and '1.'), Uncommitted, Active, Time Voyager.
- Left Sidebar:** Physical (highlighted with a red arrow and '2.'), Virtual, Policies, Catalog, Settings, Tasks.
- Filters:** Nodes: All (highlighted with a red arrow and '3.'), Links: All.
- View Tabs:** Topology, **Nodes** (selected), Links, Racks.
- Annotations:**
 - Edit server names and hostnames:** Points to the edit icon in the table header.
 - Select node(s) to set deploy mode:** Points to the checkbox in the first column of the table.
 - This is the table view:** Points to the table view icon in the view selector.
 - Select name... ...to see properties:** Points to the 'evpn_mlag_001_leaf_pair1' entry in the table and its corresponding properties panel on the right.
- Table Data:**

| Name | Role | Group Label | Deploy Mode | Device Profile | S/N | Hostname | ASN | Loopback IPv4 | Loopback IPv6 |
|--------------------------|-----------|-------------|-------------|----------------|--------------|----------|-------|---------------|---------------------|
| evpn_mlag_001_leaf_pair1 | Leaf Pair | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| leaf1 | Leaf | evpn-mlag | Deploy | Cumulus VX | 525400CD0630 | leaf1 | 64514 | 10.0.0.2/32 | fc01:a05:fab::2/128 |

Changing Server Names and Hostnames

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

1. From the blueprint, navigate to **Staged / Physical / Nodes**.
2. Click the **Edit server names and hostnames** button above the nodes (table view / card view) and make your changes.
 - To change names, select a name and enter a different 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 staged physical view.

Any associated link names do not automatically update to match the new server names and/or hostnames. As of AOS version 3.3.0 you can manually *change the link names* to match so when you are reviewing an updated cabling map the names will align.

Setting Deploy Mode for Multiple Nodes

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**.
2. Check the boxes for one or more nodes that are to be changed to the same deploy mode, then click the **Set Deploy Modes** button.
3. Select the new deploy mode.
4. To filter the node selections before changing deploy mode, use the query.
5. Click **Set Deploy Mode** to stage the change and return to the list view.

Editing Node Properties

Selecting a node gives you access to change various attributes such as name, interface map, ASN, and loopback IP, depending on the node chosen. The attributes that can be edited have an **Edit** button associated with them.

New in version 3.3.0: Leaf pair names can be changed. If the names of leafs that are used in a leaf pair have been changed, the leaf pair name does not automatically change to match the new leaf names. As of AOS version 3.3.0, you can manually change the leaf pair name to correspond with the new leaf names. This is especially useful when assigning them during virtual network creation.

1. From the blueprint, navigate to **Staged / Physical / Nodes**.
2. Select a node from the table view or card view. In the right panel, click **Properties**, if not already selected.

The screenshot shows the Apstra interface with the 'Staged / Physical / Nodes' view selected. The top navigation bar includes tabs for Dashboard, Analytics, Staged, Uncommitted, Active, and Time Voyager. The sub-navigation bar shows Physical, Virtual, Policies, Catalog, Settings, and Tasks. The main area displays a list of nodes in a card view. Red arrows point to 'spine1' and 'spine2' nodes, with text labels: 'Click a node name...', '...to access its properties', and 'This is the card view'. A red arrow also points to the 'Properties' tab in the right panel, with the label 'Edit'.

3. Change properties, as applicable.
4. Click the **Save** button to stage the changes.

Editing Device Details

In addition to editing node properties, you can change device details such as deploy mode, system ID (serial number), and hostname. You can also access device configuration files from the device tab.

Example: You can suppress L2 server cabling anomalies by setting the **Deploy Mode** to **Undeploy**. This is useful when you want to pre-provision virtual networks on all L2 servers in a blueprint, even if some L2 servers are not actually part of the network yet. (You can also get to the editing panel by navigating to Staged / Physical / Topology and selecting the server from there.)

1. From the blueprint, navigate to **Staged / Physical / Nodes**.
2. Select a node from the table view or card view. In the right panel, click **Device**, if not already selected.

Click a node name... ...to access device details

1-11 of 11

Page Size: 25

| Name | Role | Group Label | Deploy Mode | Device Profile | S/N | Hostname | ASN | Loopback IPv4 | Loopback IPv6 |
|--------------------------|-----------------|-------------|--------------|----------------|--------------|---------------|-------|---------------|---------------------|
| spine1 | Spine | N/A | Deploy | Cumulus VX | 52540040AC6D | spine1 | 64512 | 10.0.0.0/32 | fc01:a05:fab::1/128 |
| spine2 | Spine | N/A | Deploy | Cumulus VX | 525400259DB6 | spine2 | 64513 | 10.0.0.1/32 | fc01:a05:fab::1/128 |
| rtr_leaf1_leaf2 | External Router | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| evpn_mlag_001_leaf_pair1 | Leaf Pair | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| leaf1 | Leaf | evpn-mlag | Deploy | Cumulus VX | 525400CD0630 | leaf1 | 64514 | 10.0.0.2/32 | fc01:a05:fab::2/128 |
| leaf2 | Leaf | evpn-mlag | Deploy | Cumulus VX | 525400D47C50 | leaf2 | 64515 | 10.0.0.3/32 | fc01:a05:fab::3/128 |
| leaf3 | Leaf | evpn-single | Deploy | Cumulus VX | 525400EB10B9 | leaf3 | 64516 | 10.0.0.4/32 | fc01:a05:fab::4/128 |
| rack1-server1 | L2 server | dual-server | Not assigned | Not assigned | Not assigned | rack1-server1 | N/A | N/A | N/A |

Device Info

Management IP: 172.20.83.11

OS: Cumulus 3.7.11

Operation Mode: FULL CONTROL

Hostname: spine1

Config: Rendered, Incremental, Pristine

See config files

3. Change device details, as applicable.
4. Click the **Save** button to stage the changes.

4.4.1.4 Links

Note: This page describes tasks that can be performed from the **Links** view of the staged blueprint. You can perform additional links-related tasks, such as adding, editing and deleting links, from the *Topology* view of the staged blueprint.

When you've built the blueprint, review the AOS-calculated cabling map and cable up your devices according to the map. If you don't want to use the prescribed cabling from AOS you can have AOS discover existing cabling in the network. You can also change link speeds of existing links.

Links Example

In this example, each link is assigned a unique /31 subnet from the IP Pool, the smaller /31 IP is assigned to the spine interface, the larger /31 IP is assigned to the leaf interface. Subnets are assigned in increasing order in a spine-major order, that is, the links between spine1 and all leafs (in ascending order) are assigned subnets first, followed by links between spine2 and all leafs, and so on.

Dashboard

Analytics

Staged

Uncommitted

Active

Physical

Virtual

Policies

Catalog

Settings

Nodes: All

Links: All

Topology

Nodes

Links

Layer

Build: Link IPs - Spines > Leaves

Assigned

Not Assigned

1-18 of 18

Page Size: 25

| Name | Role | Speed | Endpoint 1 | | | | Endpoint 2 | | | | | |
|--|--------------------|-------|--------------------------|-----------------|--------------|-----------------|------------|----------------------|-------|--------------|-----------------|------|
| | | | Name | Role | Interface | IPv4 | IPv6 | Name | Role | Interface | IPv4 | IPv6 |
| i2_virtual_001_leaf1<->i2_virtual_001_server001link[1] | Leaf to L2 Server | 10G | i2_virtual_001_server001 | L2 server | Not assigned | N/A | N/A | i2_virtual_001_leaf1 | Leaf | Not assigned | N/A | N/A |
| i2_virtual_001_leaf1<->i2_virtual_001_server002link[1] | Leaf to L2 Server | 10G | i2_virtual_001_server002 | L2 server | Not assigned | N/A | N/A | i2_virtual_001_leaf1 | Leaf | Not assigned | N/A | N/A |
| i2_virtual_002_leaf1<->i2_virtual_002_server001link[1] | Leaf to L2 Server | 10G | i2_virtual_002_server001 | L2 server | Not assigned | N/A | N/A | i2_virtual_002_leaf1 | Leaf | Not assigned | N/A | N/A |
| i2_virtual_002_leaf1<->i2_virtual_002_server002link[1] | Leaf to L2 Server | 10G | i2_virtual_002_server002 | L2 server | Not assigned | N/A | N/A | i2_virtual_002_leaf1 | Leaf | Not assigned | N/A | N/A |
| i2_virtual_003_leaf1<->i2_virtual_003_server001link[1] | Leaf to L2 Server | 10G | i2_virtual_003_server001 | L2 server | Not assigned | N/A | N/A | i2_virtual_003_leaf1 | Leaf | Not assigned | N/A | N/A |
| i2_virtual_003_leaf1<->i2_virtual_003_server002link[1] | Leaf to L2 Server | 10G | i2_virtual_003_server002 | L2 server | Not assigned | N/A | N/A | i2_virtual_003_leaf1 | Leaf | Not assigned | N/A | N/A |
| i2_virtual_004_leaf1<->i2_virtual_004_server001link[1] | Leaf to L2 Server | 10G | i2_virtual_004_server001 | L2 server | Not assigned | N/A | N/A | i2_virtual_004_leaf1 | Leaf | Not assigned | N/A | N/A |
| i2_virtual_004_leaf1<->i2_virtual_004_server002link[1] | Leaf to L2 Server | 10G | i2_virtual_004_server002 | L2 server | Not assigned | N/A | N/A | i2_virtual_004_leaf1 | Leaf | Not assigned | N/A | N/A |
| spine1<->i2_virtual_001_leaf1[1] | Spine to Leaf | 10G | i2_virtual_001_leaf1 | Leaf | Not assigned | 192.168.0.1/31 | N/A | spine1 | Spine | Not assigned | 192.168.0.3/31 | N/A |
| spine1<->i2_virtual_002_leaf1[1] | Spine to Leaf | 10G | i2_virtual_002_leaf1 | Leaf | Not assigned | 192.168.0.3/31 | N/A | spine1 | Spine | Not assigned | 192.168.0.2/31 | N/A |
| spine1<->i2_virtual_003_leaf1[1] | Spine to Leaf | 10G | i2_virtual_003_leaf1 | Leaf | Not assigned | 192.168.0.5/31 | N/A | spine1 | Spine | Not assigned | 192.168.0.4/31 | N/A |
| spine1<->i2_virtual_004_leaf1[1] | Spine to Leaf | 10G | i2_virtual_004_leaf1 | Leaf | Not assigned | 192.168.0.7/31 | N/A | spine1 | Spine | Not assigned | 192.168.0.6/31 | N/A |
| spine2<->i2_virtual_001_leaf1[1] | Spine to Leaf | 10G | i2_virtual_001_leaf1 | Leaf | Not assigned | 192.168.0.9/31 | N/A | spine2 | Spine | Not assigned | 192.168.0.8/31 | N/A |
| spine2<->i2_virtual_002_leaf1[1] | Spine to Leaf | 10G | i2_virtual_002_leaf1 | Leaf | Not assigned | 192.168.0.11/31 | N/A | spine2 | Spine | Not assigned | 192.168.0.10/31 | N/A |
| spine2<->i2_virtual_003_leaf1[1] | Spine to Leaf | 10G | i2_virtual_003_leaf1 | Leaf | Not assigned | 192.168.0.13/31 | N/A | spine2 | Spine | Not assigned | 192.168.0.12/31 | N/A |
| spine2<->i2_virtual_004_leaf1[1] | Spine to Leaf | 10G | i2_virtual_004_leaf1 | Leaf | Not assigned | 192.168.0.15/31 | N/A | spine2 | Spine | Not assigned | 192.168.0.14/31 | N/A |
| spine1<->router0001 | To External Router | 10G | Not assigned | External Router | N/A | Not assigned | N/A | spine1 | Spine | Not assigned | Not assigned | N/A |
| spine2<->router0002 | To External Router | 10G | Not assigned | External Router | N/A | Not assigned | N/A | spine2 | Spine | Not assigned | Not assigned | N/A |

Importing Cabling Map

1. From the links view (Staged / Physical / Links) click the **Import cabling map** button to see the dialog for importing a cabling map.
2. Either click **Choose File** and navigate to the file on your computer, or drag and drop the file onto the dialog window.
3. Click **Import**.

Exporting Cabling Map

Datacenter technicians may find a printed cabling map useful when wiring in switches, or remote network operators may find it useful for viewing IP assignments. It's available in CSV or JSON format, and you can copy the contents or download the file to your local computer.

1. From the links view (Staged / 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.

Cabling maps can also be exported from the **Active / Physical / Links** view.

Editing Cabling Map

Some of the applications where you might want to override existing cabling include:

- having AOS use existing network cabling to avoid recabling
- changing interface names or IP addresses in the existing network cabling map
- specifying a different port from the one that the AOS cabling algorithm selected
- avoiding the use of a defective interface

Warning: Overriding AOS-generated cabling can be disruptive to the network. Use with extreme caution. Please contact [Apstra Support](#) for assistance with production networks.

Overriding Cabling Using Web Interface

Device Profiles must already be assigned to Blueprint nodes.

Drop the Override

To drop the override for either an interface name or IPv4/IPv6 address, submit an empty value in the corresponding field.

1. From the links view (Staged / Physical / Links) click the **Edit cabling map** button to see the dialog for editing a cabling map.
2. Change interface names and/or IP addresses, as applicable.

3. **Batch clear override** can be used to clear all Interface and IPv4/IPv6 values for a specific device type.

Cabling Map Editor

Fields to be cleared: ☒ Interface ☐ IPv4 ☐ IPv6

System roles: ☐ Spine ☒ Leaf

Query: Role = Spine to Leaf

1-6 of 6 Page Size: 25

| 6 selected | Role | Logical Link | Port Channel ID | 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

4. Click **Update** to stage the changes.
5. Click **Uncommitted** to see the diffs between **Staged** and **Active**.
6. Click the **Commit** button to save the changes to the **Active** Blueprint.

Overriding Cabling Using JSON

To change the cabling map with a JSON file, you'll export the JSON file, edit the file, then import it back into AOS.

1. From the links view (**Staged / Physical / Links**) 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**.
7. Click **Uncommitted** to see the diffs between **Staged** and **Active**.
8. Click the **Commit** button to save the changes to the **Active** Blueprint, or click the **Revert** button to discard changes and return to the previous cabling map.

Changing Link Speeds

Alternate Method

You can also change link speeds from the **Topology** view.

From the **Links** view of the **Staged** blueprint, you can change link speeds on leaf-server, external routers and MLAG peer links (as of AOS 3.2). To change link speeds on spine-leaf and superspine-spine you must *change the Rack*.

1. From the links view (Staged / Physical / Links) click the **Change link speeds** button to see the dialog for changing link speeds.
2. To search for specific links, click the query box, enter search criteria and click **Apply** to see results.
3. From the **Speed** drop-down list corresponding to the link to be changed, select the new speed.
4. Click **Update** to update the link and return to the cabling map.

Fetching Discovered LLDP Data

Before AOS can discover existing cabling, all system nodes in the Blueprint must have system IDs assigned to them.

Warning: This is a disruptive operation. All links can potentially be renumbered.

1. From the links view (Staged / Physical / Links) click the **Fetch discovered LLDP data**.
2. If staged data is *identical* to LLDP discovery results, you will see a message with that statement. Your actual cabling matches the AOS 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 Staged Cabling Map from LLDP**. You might also need to *Reset Resource Group Overrides*.

Changing Link Name

If you have *changes 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. New in AOS version 3.3.0, you can change link names to match your other name changes.

1. From the blueprint, navigate to **Staged / Physical / Links**, then click the name of the link to change.
2. In **Properties** (right panel) click the **Edit** button for the link name.
3. Change the name and click the **Save** button to stage the change.

4.4.1.5 Racks

AOS provides full control over the growth of your network by enabling you to add, export, edit, and delete complete racks in a running blueprint. This flexible fabric expansion (FFE) feature is supported on 3-stage and 5-stage Clos networks. These capabilities apply to racks in 5-stage topologies, but not to the pods themselves. It is not possible to add or remove pods, superspine planes or superspine devices in a running blueprint, so ensure that the day 0 template is correct.

Annotations in the screenshot:

- Click to search (points to the Query: All search bar)
- Add rack (points to the + icon)
- Click to sort (points to the sort icons in the table headers)
- Click rack name to select and see rack properties (points to the rack name 'evpn_mlag_001' in the table)
- Change rack name (points to the edit icon in the Rack Properties panel)
- Export rack type to global catalog (points to the export icon in the Actions column)

| Name | Rack Type | Connectivity Type | IP Version | Leaf Count | External Router Links Count | Servers Capacity | Actions | | | | | | | | | | | | |
|-----------------|-----------|-------------------|------------|-------------|-----------------------------|---|---------|------|-----------|-------------|---|---|-----------------|---|---|-----------------|---|---|--|
| evpn_mlag_001 | evpn-mlag | L2 | IPv4 | 1 MLAG pair | 2 | <table border="1"> <thead> <tr> <th>Name</th> <th>Used</th> <th>Available</th> </tr> </thead> <tbody> <tr> <td>dual-server</td> <td>1</td> <td>0</td> </tr> <tr> <td>single-server-1</td> <td>1</td> <td>0</td> </tr> <tr> <td>single-server-2</td> <td>1</td> <td>0</td> </tr> </tbody> </table> | Name | Used | Available | dual-server | 1 | 0 | single-server-1 | 1 | 0 | single-server-2 | 1 | 0 | <div>Export rack type to global catalog</div> <div>Edit Delete</div> |
| Name | Used | Available | | | | | | | | | | | | | | | | | |
| dual-server | 1 | 0 | | | | | | | | | | | | | | | | | |
| single-server-1 | 1 | 0 | | | | | | | | | | | | | | | | | |
| single-server-2 | 1 | 0 | | | | | | | | | | | | | | | | | |

Embedded Rack Types

Rack Types have timestamps in the Blueprints. When the Blueprints are modified via “Flexible Fabric Expansion (FFE)” operations such as Changing Link Speeds, Adding Servers, Adding & Removing Links, etc., AOS will modify the Embedded Rack Types accordingly then update the timestamps.

In the example below, both Rack Types have the timestamp “2021-01-12 10:20”.

| Name | Pod Name | Rack Type | Connectivity Type | IP Version | Leaf Count | External Router Links Count | Servers Capacity | Actions | | | | | | |
|-------------------------|----------|-------------------------------------|-------------------|------------|-------------|-----------------------------|--|---------|------|-----------|--------|---|---|------------------------|
| _my_rack_dbl_ex_001_001 | pod1 | _my_rack_dbl_ex 2021-01-12 10:20 | L2 | IPv4 | 1 ESI group | 1 | <table border="1"> <thead> <tr> <th>Name</th> <th>Used</th> <th>Available</th> </tr> </thead> <tbody> <tr> <td>server</td> <td>2</td> <td>2</td> </tr> </tbody> </table> | Name | Used | Available | server | 2 | 2 | <div>Edit Delete</div> |
| Name | Used | Available | | | | | | | | | | | | |
| server | 2 | 2 | | | | | | | | | | | | |
| _my_rack_double_002_001 | pod2 | _my_rack_double 2021-01-12 10:20 | L2 | IPv4 | 1 ESI group | 0 | <table border="1"> <thead> <tr> <th>Name</th> <th>Used</th> <th>Available</th> </tr> </thead> <tbody> <tr> <td>server</td> <td>2</td> <td>3</td> </tr> </tbody> </table> | Name | Used | Available | server | 2 | 3 | <div>Edit Delete</div> |
| Name | Used | Available | | | | | | | | | | | | |
| server | 2 | 3 | | | | | | | | | | | | |

And the next screenshot is taken after the “Add Leaf to Server Links” FFE operation. Although the Rack Type name is the same, the contents of the Rack Type has been modified by AOS.

| 0 selected | Name | Pod Name | Rack Type | Connectivity Type | IP Version | Leaf Count | External Router Links Count | Servers Capacity | | | Actions |
|--------------------------|------------------------|----------|------------------------------------|-------------------|------------|-------------|-----------------------------|------------------|------|-----------|--|
| <input type="checkbox"/> | my_rack_dbl_ex_001_001 | pod1 | my_rack_dbl_ex 2021-01-12 10:20 | L2 | IPv4 | 1 ESI group | 1 | Name | Used | Available | <div><div></div><div></div><div></div></div> |
| | | | | | | | | server | 2 | 2 | |
| <input type="checkbox"/> | my_rack_double_002_001 | pod2 | my_rack_double 2021-01-12 11:45 | L2 | IPv4 | 1 ESI group | 0 | Name | Used | Available | <div><div></div><div></div><div></div></div> |
| | | | | | | | | server | 3 | 2 | |

Changing 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 (as of AOS version 3.3.0).

1. From the blueprint, navigate to **Staged / Physical / Racks**.
2. Choose the rack that needs a name change.
3. In **Rack Properties** (right panel) click the **Edit** button for the rack name.
4. Change the name and click the **Save** button to stage the change.

Adding Rack

The easiest and fastest way to expand your network is to add a rack.

1. From the blueprint, navigate to **Staged / Physical / Racks**.
2. Click the **Add Racks** button (+), and 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 view 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**, AOS re-calculates port assignments. This may result in the need for re-cabling. When in doubt, leave the box checked.
6. Click **Add** to stage the rack addition and return to the list view.
7. *Assign device profiles and system IDs (serial numbers) to the new rack(s).*
8. Commit the changes to your blueprint to configure the rack(s) and complete the fabric expansion.

Exporting Rack

Some changes cannot be made directly in the blueprint rack type (for example, changing port channel id ranges). In these cases, you can export the rack type to the global catalog, update it there, then from the blueprint, edit the rack type to use the modified one from the global catalog.

1. From the blueprint, navigate to **Staged / Physical / Racks**.
2. Click the **Export rack to global catalog** button.
3. Give the **Rack Type** a unique name.
4. Click **Export** to export the rack type to the global catalog.
5. Navigate to **Design / Rack Types** and edit the rack type from there.

Editing Rack

Existing running racks can be changed while preserving many of their characteristics (as of AOS version 3.2.0). Depending on the change, editing a rack may require re-assigning device profiles and system IDs (serial numbers).

Warning:

- (Re)assigning device profiles and system IDs results in a full config push to those devices when committing the blueprint. This is service impacting.
- When editing a rack it is not always possible to retain virtual network endpoints. If not, you will have to re-assign the endpoints. When in doubt, you can always review your changes before committing. If there are undesired changes you can revert them from the **Uncommitted** tab.

Important: Editing a rack is not a simple ‘Delete & re-add’ operation: it is not a complete replacement of a rack. Instead, when editing a rack, AOS tries to retain as much of the original data (such as virtual network endpoints, leaf / server / link names, etc) as possible. If such preservation is not needed, it is recommended perform a ‘Delete & Add’ instead of an Edit.

Typically, a rack edit operation involves the following steps:

1. *Ensure that a Rack exists in AOS* that meets your design requirements.
2. From the blueprint, navigate to **Staged / Physical / Racks**.
3. Click the **Edit** button for the rack to edit.
4. From the **New Rack Type** drop-down list, select the required rack type.
5. If new devices were added, *assign device profiles and system IDs (serial numbers)* to them.
6. Optionally, review the **Incremental Config** to see the changes AOS will push to the device(s). If devices were assigned, a full config push is performed.
7. Commit the changes to the blueprint. AOS will push all required configuration changes to the devices in the modified Rack.

Note: Virtual network (VN) endpoints will remain as long as the server and link labels between the old and new rack type are the same.

Deleting Rack

Before deleting a rack that has live traffic on it, you may want to take its devices out-of-service gracefully by draining them. See the *device draining guide* for more information.

1. From the blueprint, navigate to **Staged / Physical / Racks**.
2. Click the **Delete** button (trash can) for the rack to delete.
 - 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.

3. Click **Delete Rack** to stage the deletion and return to the list view.
4. Commit the changes to the blueprint. Configuration on any running devices will be erased and the devices will be ready to be decommissioned.

4.4.1.6 Pods

From the blueprint, navigate to **Staged / Physical / Pods** to see details about the staged pods in a **5-stage Clos network**. (The pod view only applies to pod-based blueprints.) You can search for specific nodes or links and select a layer to see build details, deploy modes, or uncommitted changes.

1. Click **Staged** to stage the deletion and return to the list view.

2. Click **Physical** to stage the deletion and return to the list view.

3. Click **Pods** to stage the deletion and return to the list view.

Click to search

Select layer to see build details, deploy modes, or uncommitted changes

Click rack name to see rack type preview

pod1

Capacity:

Query: All

6-7 of 7

| Name | Type | Used | Available |
|------------------|--------|------|-----------|
| L2 Leaf External | global | 0 | 22 |
| L2 One Leaf | global | 0 | 22 |

pod2

Capacity:

Query: All

1-5 of 14

| Name | Type | Used | Available |
|---------------|----------|------|-----------|
| border | global | 0 | 4 |
| border | embedded | 0 | 4 |
| compute-2pair | global | 0 | 4 |
| compute-2pair | embedded | 3 | 4 |
| compute-10g | global | 0 | 39 |

Changing Pod Name

1. From the blueprint, navigate to **Staged / Physical / Pods**.
2. Click the name of the pod to change.
3. In **Pod Properties** (right panel) click the **Edit** button for the pod name.
4. Change the name and click the **Save** button to stage the change.

Click pod name... to change pod name

4.4.2 Virtual

4.4.2.1 Build - Virtual

1. Staged
2. Virtual
3. Virtual Networks
4. Build
5. VNI Virtual Network IDs
6. Update assignments

When resources are staged, status indicator turns green

Reset resource group overrides

Manage resource pools

| Name | Security Zone | Type | VN ID | Assigned to | DHCP Service | L3 Connectivity | IPv4 Connectivity | IPv4 Subnet | IPv6 Connectivity | IPv6 Subnet | Actions |
|---------------------------|---------------|----------|--------------|-------------|--------------|-----------------|-------------------|------------------|-------------------|--------------------|---------|
| blue_120_vxpn_mlag_001_v4 | blue | VXLAN | Not assigned | 1 nodes | Enabled | Enabled | Enabled | 192.168.120.0/24 | Disabled | N/A | |
| blue_121_leaf3_v4 | blue | VXLAN | Not assigned | 1 nodes | Enabled | Enabled | Enabled | 192.168.121.0/24 | Disabled | N/A | |
| blue_vxlan_100_v4 | blue | VXLAN | Not assigned | 2 nodes | Enabled | Enabled | Enabled | 192.168.100.0/24 | Disabled | N/A | |
| L2_60_blue | blue | External | 60 | 1 nodes | Disabled | Enabled | Enabled | 10.60.60.0/24 | Enabled | a05:198:60:60::/64 | |
| L2_61_default | default | External | 61 | 1 nodes | Disabled | Enabled | Enabled | 10.60.61.0/24 | Disabled | N/A | |
| L2_62_red | red | External | 62 | 1 nodes | Disabled | Enabled | Enabled | 10.60.62.0/24 | Enabled | a05:198:60:62::/64 | |

From the *blueprint*, navigate to **Staged / Virtual / Virtual Networks / Build**. (The build panel is on the right side.)

Update Assignments

1. A red status indicator means that resources need to be assigned. Resources may include virtual network SVI subnets for security zones, SVI subnets for MLAG domain, SVI subnet for virtual networks, VNI Virtual Network IDs, and VTEP IPs. Click the indicator, then click the **Update assignments** button.
2. Select a pool from which to pull the resources, then click the **Save** button. AOS assigns the required number of resources to the resource group automatically. When the red status indicator turns green, the resource assignment has been successfully staged.

Reset Resource Group Overrides

Certain blueprint operations require AOS to retain resource allocations even when a device has been removed from a blueprint. The resource groups are overridden, which mean that when a device is re-used, previously allocated resources are re-used as well. Situations like this can (but do not always) result in build errors. If you do not need to re-use the same resources, click the **Reset resource group overrides** button to reset the resource group.

Manage Resource Pools

The resource assignment section has a convenient shortcut button, **Manage resource pools**, that takes you to resource pool management. There, you can monitor resource usage (new in AOS version 3.3.0) and create additional resource pools, as needed.

4.4.2.2 Virtual Networks

Virtual Networks (VN) are collections of L2 forwarding domains. In AOS managed fabric, a Virtual Network can be constructed using either VLANs or VXLANs.

VLAN (Rack-local VN)

- A scope of a single rack
- Can be either a single leaf or leaf pair
- Can be deployed in L2-only mode or with a L3-gateway (SVI) IP address hosted on the rack leaf

VXLAN (Inter-rack VN)

- Fabric-wide scope for ubiquitous L2
- Can be combination of single rack leaf or leaf pair (MLAG)
- Deployed in L2-only mode
- L3-gateway functionality

AOS API Samples

For AOS API samples for creating AOS VNs with server endpoints, see <https://github.com/Apstra/aos-api-samples>

Refer to the the *AOS Feature Matrix* for complete AOS Virtual Networks feature compatibility for supported Network Operating Systems (NOS).

Virtual Networks are managed on a per Blueprint basis.

To access virtual networks - from the blueprint, navigate to **Staged / Virtual / Virtual Networks**.

Blueprints > evpn-nxosv-virtual

Dashboard Analytics Staged Uncommitted Active

Physical Virtual Policies Catalog Settings

Virtual Networks Security Zones Virtual Infra Internal Endpoints External Endpoints Enforcement Points Endpoint Groups

Create Virtual Networks

Query: All

1-4 of 4 Page Size: 25

| Name | Security Zone | Type | VN ID | Assigned to | DHCP Service | L3 Connectivity | IPv4 Connectivity | IPv4 Subnet | IPv6 Connectivity |
|---------------------|---------------|-------|-------|-------------|--------------|-----------------|-------------------|------------------|-------------------|
| blue_vxlan_100_v4 | blue | VXLAN | 10001 | 2 nodes | Enabled | Enabled | Enabled | 192.168.100.0/24 | Disabled |
| blue_vxlan_105_dual | blue | VXLAN | 10002 | 2 nodes | Enabled | Enabled | Enabled | 192.168.105.0/24 | Disabled |
| red_125_leaf3_v4 | red | VXLAN | 10003 | 2 nodes | Enabled | Enabled | Enabled | 192.168.125.0/24 | Disabled |
| red_vxlan_102_v4 | red | VXLAN | 10000 | 2 nodes | Enabled | Enabled | Enabled | 192.168.106.0/24 | Disabled |

Build

- 1/1 SVI Subnets - MLAG domain
- 4/4 VNI Virtual Network IDs
- 2/2 VTEP IPs

Creating Virtual Networks

A VLAN requires a name, VLAN ID, and an optional specification of Layer3 intent. A VLAN based virtual network can be deployed in a pure L2-mode (eg, isolated cluster networks for database replication), or with L3 (SVI provisioned) with first-hop redundancy. AOS also supports DHCP Relay which is configured under [Security Zones](#).

A VXLAN requires a name and a VNID (defined below) and an optional specification of Layer3 intent.

Create Single Rack-local VLAN Based Virtual Network

1. From the blueprint, navigate to **Staged / Virtual / Virtual Networks**, then click **Create Virtual Networks**.

Create Virtual Network

Virtual Network Parameters

Type
☒ VLAN ☐ VXLAN

Will create a VLAN per selected node

Name Security Zone default

Default VLAN ID DHCP Service ☒ Disabled ☐ Enabled IPv4 Connectivity ☒ Disabled ☐ Enabled IPv6 Connectivity ☒ Disabled ☐ Enabled

Default Endpoint Types

Link Label Tag Type

☐ Create Another? **Create**

2. Fill in the required information

- **Type** **Rack-local** for VLAN Based Virtual Networks, **Inter-rack** for VXLAN/EVPN based Virtual Network.
- **Name** 32 characters or fewer. Underscore, dash and alphanumeric characters only.
- **Security Zone** VLAN Based “rack-local” Virtual Networks are currently only allowed in the **Default** Security Zone.
- **Default VLAN ID** The L2 VLAN ID on the switch the virtual network is assigned. This can either be explicitly assigned or left blank to have AOS auto-assign from a static pool with the range of 2-4094.

Note: Using VLAN ID 1 is supported, but will need to be explicitly assigned. However, best practices do not recommend using VLAN ID 1 (default) for any active virtual network.

Note: Different network device vendors have varying requirements for “reserved” VLAN ID ranges. Cumulus VLAN-aware Bridge Mode reserves a VLAN ID range by default from 3000 to 3999. Cisco NXOS reserves VLAN ID range from 3968 to 4094. Arista, by default, will use a VLAN ID range from 1006 to 4094 for internal VLANs for routed ports.

Note: For Arista EOS devices, the user can modify the “reserved” VLAN ID range with the EOS `vlan internal allocation policy configuration` command.

```
l2-virtual-ext-002-leaf1(config)#vlan internal allocation policy
↪ascending range 3001 3999
l2-virtual-ext-002-leaf1(config)#exit
l2-virtual-ext-002-leaf1#show vlan internal allocation policy
Internal VLAN Allocation Policy: ascending
```

(continues on next page)

(continued from previous page)

```
Internal VLAN Allocation Range: 3001-3999
l2-virtual-ext-002-leaf1#
```

This EOS configuration can be applied to all EOS devices using AOS System Configlets **prior** to the configuration and deployment of Virtual Networks.

- **Enable DHCP Service** If set to 'Enabled', and the blueprint template has DHCP enabled, a DHCP relay forwarder will be configured on the SVI. This option also implies L3 routing on this SVI.
- **IPv4 Connectivity** Enables SVI IPv4 routing features for this Virtual Network.
- **IPv4 Subnet**

If **IPv4 Connectivity** is enabled, this is the IP address range used on this Virtual Network. This can be explicitly set with an IPv4 subnet (e.g. 192.168.100.0/24), an IPv4 CIDR length (e.g. /26) for auto-assignment of a subnet with the specified length or left blank for auto-assignment of a /24 subnet network. For auto-assignment, the IP is automatically derived from the assigned Virtual Networks SVI Pool (more on this described below).

Note: If creating multiple "batch" VLAN networks, you can only leave **IPv4 Subnet** blank or specify a IPv4 CIDR length (e.g. /26).

- **IPv6 Connectivity** Enables SVI IPv6 routing features for this Virtual Network.

Note: This option is only available if support for IPv6 Applications is enabled. Also, if the Template used for the Blueprint used Spine to Leaf Links Type of "IPv4" only, then IPv6 cannot be used in the Default Security Zone and VLAN Virtual Networks.

- **IPv6 Subnet**

If **IPv6 Connectivity** is enabled, this is the IP address range used on this Virtual Network. This can be explicitly set with an IPv4 subnet (e.g. 2001:4de0::/64), an IPv6 CIDR length (e.g. /56) for auto-assignment of a subnet with the specified length or left blank for auto-assignment of a /64 subnet network. For auto-assignment, the IP is automatically derived from the assigned Virtual Networks SVI Pool (more on this described below).

Note: If creating multiple “batch” VLAN networks, you can only leave **IPv6 Subnet** blank or specify a IPv6 CIDR length (e.g. /56).

- **Default Endpoint Types** This allows you to set all endpoints (servers) to a common VLAN type (**Unassigned**, **Untagged**, **VLAN Tagged**) when the VN is created. This configuration is not persistent and can be changed by editing the VN endpoint configuration.
- **Assigned To** Select the rack the VN is to be assigned to. For 5-stage Clos networks, the PODs are shown, and the Leaf devices within each POD can be selected to extend the VN to those devices. For MLAG racks, the leaf pair is shown. If you select more than one rack, AOS automatically creates multiple “Rack-local” VLAN Based Virtual Networks via the “Batch” process described below. Each rack has option to specify the **VLAN ID** (used for “Batch VN Creation”) and **IPv4** field used for racks to set the first-hop-redundancy IP address for the SVI (VRRP, VARP, etc). If left blank AOS auto-assigns the MLAG SVI from the SVI pool selected:

Create Virtual Network

Assigned To

| Query: All | | | | | 1-3 of 3 | < | > | Page Size: 25 |
|-------------------------------------|----------|--------------------|-------------|--------------------|----------|---|---|---------------|
| <input checked="" type="checkbox"/> | Pod Name | Bound To | Link Labels | VLAN ID | | | | |
| <input checked="" type="checkbox"/> | pod1 | leaf_pair001_001_1 | SL1 | From resource pool | | | | |
| <input checked="" type="checkbox"/> | pod2 | leaf002_001_1 | SL1 | From resource pool | | | | |
| <input checked="" type="checkbox"/> | pod2 | leaf002_001_2 | SL2 | From resource pool | | | | |

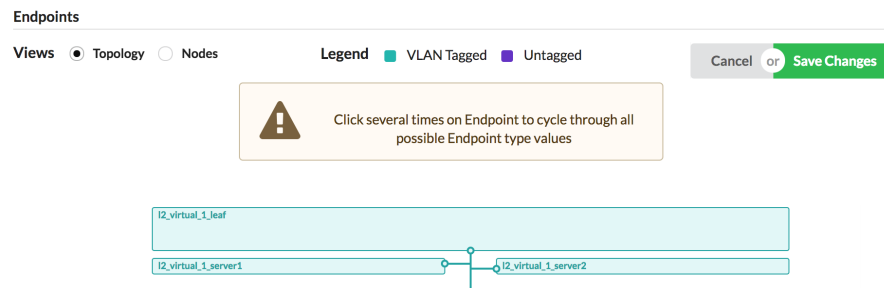
3. Click **Create** to stage the new virtual network and return to the list view.
4. If IPv4 or IPv6 resources are required for Network SVI Subnets you must assign resources, from the virtual

networks build section, before the staged blueprint can be committed. (Prior to AOS version 3.3.0, SVI subnets were assigned from the security zone build section.)

VLAN Network Endpoints

To modify Endpoints (Servers), click the ‘Manage Endpoints’ button. You can edit the Endpoints in either a “Topology” or “Node” mode. “Topology” mode shows a graphical representation of the VN. “Nodes” mode shows a list of the node in the rack.

In the “Topology” mode you can click on the Endpoint to cycle through the available Endpoint modes **Unassigned** (grey), **VLAN Tagged** (blue-green) or **Untagged** (purple).



In the “Nodes” mode you can click on the Endpoint mode to cycle through the available Endpoint modes **Unassigned** (grey), **VLAN Tagged** (blue-green) or **Untagged** (purple).

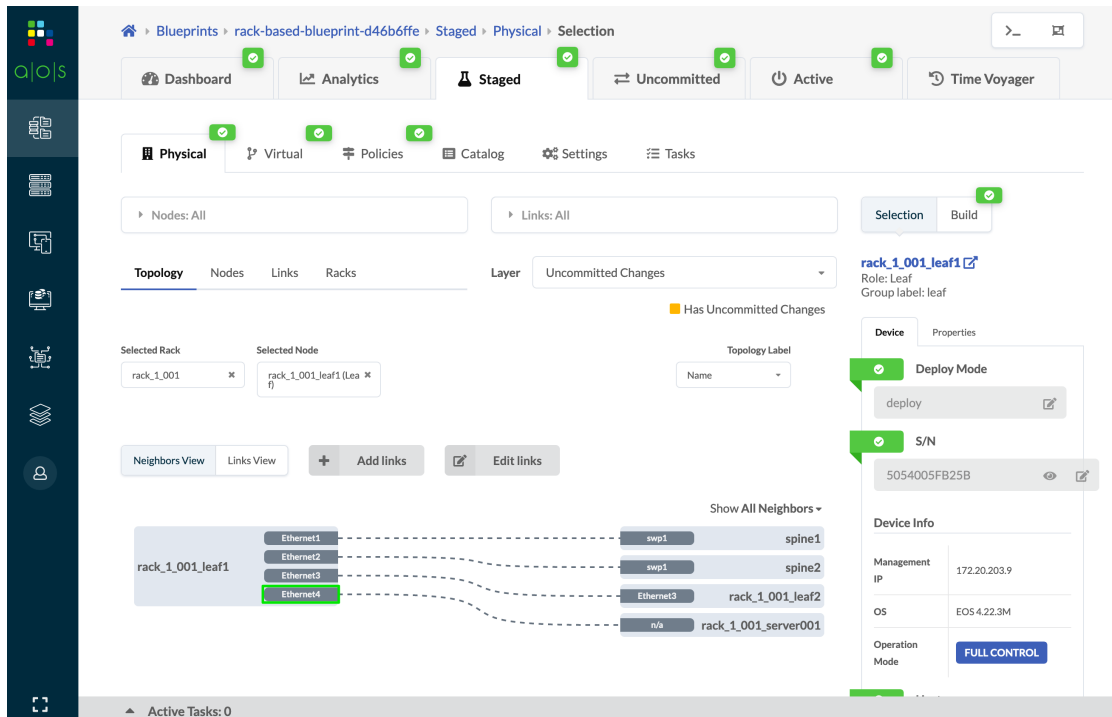
| | Server Name | Assigned to | Endpoint |
|--------------------------|----------------------|-------------------|-------------|
| <input type="checkbox"/> | I2_virtual_1_server1 | I2_virtual_1_leaf | VLAN Tagged |
| <input type="checkbox"/> | I2_virtual_1_server2 | I2_virtual_1_leaf | VLAN Tagged |

For VLAN based Virtual Networks only one “untagged” VLAN may be assigned per endpoint, since a switch port cannot have two native VLANs. If the server is dual-attached, a port-channel (bond) and MLAG configuration will be generated automatically. A server facing port-channel (bond) configuration will not be created until a Virtual Network is created for the endpoint.

All “tagged” VLANs are 802.1q encapsulated towards the server.

Add or remove a Virtual Network from an interface

To add or remove a Virtual Network (new or existing) to an interface or port-channel we need to identify in a first instance the interfaces (physical switch ports in case of port-channels too) from the Topology tab.



Then, from Staged → Virtual → Virtual Networks, we select the VN that we want to add to the port-channel or we create a new one using the “Create Virtual Networks” option.

If this is a new VLAN we specify where this VLAN will be configured, in what switch/rack and if we desire, we can assign it to the server facing ports too by selecting VLAN Tagged (if it is a trunk) or Untagged (if it is an access port) in the “Server Logical Link” section (see capture below), or if we prefer to this later and select specific ports we can leave it as “Unassigned”.

in this case, we will configured in the rack_1_001 leaf pair, and the VLAN ID is 60. We will leave it as unassigned so we can apply it to our port-channel in a different step:

Create Virtual Network

1

DHCP Service
☒ Disabled
☐ Enabled

IPv4 Connectivity
☒ Disabled
☐ Enabled

Default Endpoint Types

| Server Logical Link | Tag Type |
|---------------------|--|
| link | <input checked="" type="radio"/> Unassigned <input type="radio"/> Untagged <input type="radio"/> VLAN Tagged |

Assigned To

Query: All 1-2 of 2 Page Size: 25

| | Bound To | Link Labels | VLAN ID |
|-------------------------------------|-----------------------|-------------|-----------------|
| <input checked="" type="checkbox"/> | rack_1_001_leaf_pair1 | link | 60 |
| <input type="checkbox"/> | rack_2_001_leaf1 | link | Default VLAN ID |

Create

After the VLAN has been created, click the VLAN name (the same applies if this VLAN was already defined before):

Now apply it to the desired port:

- click on the switch port as in the capture
- select the leaf pair
- Click on VLAN Tagged to assign it to a port-channel with more VLANs (trunk)

Click on ports to select endpoints and make them:

Unassigned or Untagged **VLAN Tagged** 3

Select All Reset

Selected Endpoints:

Query: All 1 of 1 Page Size: 5

| Leaf | Interface Name(s) | Server | Server Group | Logical Link | Endpoint | Actions |
|--|--------------------------------------|------------------------|--------------------------------|--------------|-------------|---------|
| 1 selected | | | | | | |
| 2 <input checked="" type="checkbox"/> | rack_1_001_leaf1 rack_1_001_leaf2 | Ethernet4 Ethernet4 | rack_1_001_server001 server | link | VLAN Tagged | |

Port Maps:

Query: All

Ports: Unassigned Untagged **VLAN Tagged** Partial 1 of 1 Page Size: 5

rack_1_001_leaf_pair1 rack_1_001_leaf1 + rack_1_001_leaf2

1 2 3 **4** 5 6 7

Active Tasks: 0

Finally, commit the change.

You can use “Revert” rather than “Commit” if you are not happy with the change, or once committed, you can restore the previous setup from the “Time Voyager” tab (this will restore your blueprint to a previous state).

To DELETE the VLAN from that port, proceed in the same way by clicking on “Unassigned” instead of “VLAN tagged” as in the example above on the actual ports.

Create Inter-rack VXLAN/EVPN based Virtual Networks

Depending on the control plane selected when configuring the Template, either Static VXLAN Routing or MP-EBGP EVPN, you can configure Inter-rack Virtual Networks. MP-EBGP EVPN provides a control plane for VXLAN routing. References to VXLAN in this documentation refers to either type of control plane.

Note: The VXLAN/EVPN capabilities for Inter-rack Virtual Networks are highly dependent on the Make and Model of the Network Devices being used. Refer to the [AOS Feature Matrix](#) and [AOS EVPN Support Addendum](#) for information. Contact your Network Device Vendor or Apstra Support if you need detailed capability information for your device.

1. From the blueprint, navigate to **Staged / Virtual / Virtual Networks**, then click **Create Virtual Networks**.

The screenshot shows the 'Create Virtual Network' modal window. Under 'Virtual Network Parameters', the 'Type' is set to 'VXLAN'. A message states 'Will create single VXLAN for all selected nodes'. The 'Name' field is empty, and 'Security Zone' is set to 'Select...'. For 'VNI ID', the option 'From resource pool' is selected. 'DHCP Service' is set to 'Disabled', and both 'IPv4 Connectivity' and 'IPv6 Connectivity' are also set to 'Disabled'. The 'Default Endpoint Types' section shows 'Link Label' and 'Tag Type' as options. At the bottom right, there is a 'Create' button and a 'Create Another?' checkbox.

2. Fill in the required information

- **Type** **Rack-local** for VLAN Based Virtual Networks, **Inter-rack** for VXLAN/EVPN based Virtual Network.
- **Name** 32 characters or fewer. Underscore, dash and alphanumeric characters only.
- **Security Zone** This is the Security Zone the VN is associated with. All the VNs need to be associated to a Security Zone different than the default, which is reserved for the underlay network and Single Rack-local VLAN Based Virtual Network.

- **VNI ID** The L2 VXLAN ID on the switch the virtual network is assigned. This can be explicitly assigned or you may leave this blank and let AOS auto-assign a VNID from a VNID pool created in the **Resources** tab. You can batch create up to 40 virtual networks by entering their VNI IDs in this field (ranges or individual VNI IDs separated by commas, e.g. 5555-5560, 7777). If more than 40 virtual networks are required, we recommend committing the first 40 VNs before creating additional ones.
- **Enable DHCP Service** If set to 'Enabled', and the blueprint template has DHCP enabled, a DHCP relay forwarder will be configured on the SVI. This option also implies L3 routing on this SVI.
- **IPv4 Connectivity** Enables SVI IPv4 routing features for this Virtual Network.
- **IPv4 Subnet** If **IPv4 Connectivity** is enabled, this is the IP address range used on this Virtual Network. This can be explicitly set with an IPv4 subnet (e.g. 192.168.100.0/24), a IPv4 CIDR length (e.g. /26) for auto-assignment of a subnet with the specified length or left blank for auto-assignment of a /24 subnet network. For auto-assignment, the IP is automatically derived from the assigned Virtual Networks SVI Pool (more on this described down below).
- **IPv6 Connectivity** Enables SVI IPv6 routing features for this Virtual Network.

Note: This option is only available if support for IPv6 Applications is enabled. Also, if the Template used for the Blueprint used Spine to Leaf Links Type of "IPv4" only, then IPv6 cannot be used in the Default Security Zone.

- **IPv6 Subnet** If **IPv6 Connectivity** is enabled, this is the IP address range used on this Virtual Network. This can be explicitly set with an IPv4 subnet (e.g. 2001:4de0::/64), an IPv6 CIDR length (e.g. /56) for auto-assignment of a subnet with the specified length or left blank for auto-assignment of a /64 subnet network. For auto-assignment, the IP is automatically derived from the assigned Virtual Networks SVI Pool (more on this described down below).

The screenshot shows the 'Create Virtual Network' dialog box in the Apstra GUI. The dialog is titled 'Create Virtual Network' and contains the following fields and options:

- Type:** Radio buttons for VLAN and VXLAN. VXLAN is selected.
- Name:** Text field containing 'Compute'.
- Security Zone:** Text field containing 'red'.
- VNI ID:** Text field containing 'From resource pool'.
- DHCP Service:** Radio buttons for Disabled and Enabled. Disabled is selected.
- IPv4 Connectivity:** Radio buttons for Disabled and Enabled. Enabled is selected.
- IPv4 Subnet:** Text field containing 'From resource pool'.
- Virtual Gateway IPv4:** Text field containing 'From resource pool'.
- IPv6 Connectivity:** Radio buttons for Disabled and Enabled. Enabled is selected.
- IPv6 Subnet:** Text field containing 'From resource pool'.
- Virtual Gateway IPv6:** Text field containing 'From resource pool'.
- Default Endpoint Types:** A section with two fields: 'Link Label' and 'Tag Type'.
- Create Another?:** A checkbox.
- Create:** A blue button at the bottom right.

- **Default Endpoint Types** This allows you to set all endpoints (servers) to a common VLAN type (**Unassigned**, **Untagged**, **VLAN Tagged**) when the VN is created. This configuration is not persistent and can be changed by editing the VN endpoint configuration.
- **Assigned To** Select the rack the VXLAN VN is to be assigned to. For MLAG racks, the leaf pair is shown.

Each rack has option to specify the **VLAN ID** (used for “Batch VN Creation”) and **IPv4** field used for racks to set the first-hop-redundancy IP address for the SVI (VRRP, VARP, etc). If left blank AOS auto-assigns the MLAG SVI from the SVI pool selected.

Create Virtual Network

Default Endpoint Types

| Link Label | Tag Type |
|------------|--|
| link | <input type="radio"/> Unassigned <input type="radio"/> Untagged <input checked="" type="radio"/> VLAN Tagged |

Assigned To

| <input checked="" type="checkbox"/> | Bound To | Link Labels | VLAN ID | IPv4 Address | IPv6 Address |
|-------------------------------------|--------------------------|-------------|--------------------|--------------------|--------------------|
| <input checked="" type="checkbox"/> | I2_virtual_ext_001_leaf1 | link | From resource pool | From resource pool | From resource pool |
| <input checked="" type="checkbox"/> | I2_virtual_ext_002_leaf1 | link | From resource pool | From resource pool | From resource pool |
| <input checked="" type="checkbox"/> | I2_virtual_ext_003_leaf1 | link | From resource pool | From resource pool | From resource pool |

☐ Create Another? **Create**

3. Click **Create** to stage the new virtual network and return to the list view.
4. If IPv4 or IPv6 resources are required for Network SVI Subnets you must assign resources, from the virtual networks build section, before the staged blueprint can be committed. (Prior to AOS version 3.3.0, SVI subnets were assigned from the security zone build section.)

Query: All 1-3 of 3 Page Size: 25

| Name | Security Zone | Type | VN ID | Assigned to | DHCP Service | L3 Connectivity | IPv4 Connectivity | IPv4 Subnet | IPv6 Connectivity | IPv6 Subnet |
|---------|---------------|-------|-------|-------------|--------------|-----------------|-------------------|----------------|-------------------|-------------|
| Compute | red | VXLAN | 5000 | 1 nodes | Disabled | Enabled | Enabled | 192.168.1.0/24 | Disabled | N/A |
| VLAN2 | default | VLAN | 2 | 1 nodes | Disabled | Enabled | Enabled | 10.0.0.0/24 | Disabled | N/A |
| VLAN3 | default | VLAN | 3 | 1 nodes | Disabled | Enabled | Enabled | 172.30.16.0/24 | Disabled | N/A |

Build

- 2/2 SVI Subnets - Virtual Networks
- 1/1 VNI Virtual Network IDs
- 0/1 VTEP IPs

No pools assigned

- After creating the VXLAN virtual network you must assign the VTEP IP from one of the IP pools created in the **Build** tab. Each single leaf Node needs one VTEP IP and an “anycast” VTEP IP for all switches in the Virtual Network. For MLAG leaf-pair Nodes, a common VTEP IP is assigned for the leaf-pair as well as the “anycast” VTEP IP for all switches in the Virtual Network.

Server Per-port Virtual Networks

New in version 2.3: Racks and Servers can be configured to allow for Server Ports to have connections to different Virtual Networks (VN). Racks can be designed with multiple leafs, or a single leaf with different logical links. Server links can then be defined how they connect to these leafs. When creating or editing VNs, the VN endpoints can be modified on a per-port basis, associating the port to the VN in an **Unassigned**, **Untagged** and **VLAN Tagged** mode.

Create Per-port Rack Types

Refer to [Create Rack Types](#) for instructions on how to create Rack Types which support Servers with multiple connections allowing for per-port VN.

When creating an L2 Rack Type to be used, you can create a Rack Type with either multiple Leafs or a single leaf with different logical links.

Create Rack Type

Topology
Summary
Leaves
Servers

Leaf1

Label *

Leaf1

Logical device *

AOS-7x10-Leaf

2 x 10 Gbps Spine 2 x 10 Gbps Peer 2 x 10 Gbps L2 Server • L3 Server 1 x 10 Gbps External Router

☐ MLAG pair

☐ External facing

Links per spine * 1 10 Gbps

Leaf2

Label *

Leaf2

Logical device *

AOS-7x10-Leaf

2 x 10 Gbps Spine 2 x 10 Gbps Peer 2 x 10 Gbps L2 Server • L3 Server 1 x 10 Gbps External Router

☐ MLAG pair

☐ External facing

Links per spine * 1 10 Gbps

+ Add New Leaf

☐ Create Another? **Create**

Next, define a Server Group in the Rack Type which has multiple Links to the Leafs that were created. The Link Labels will be used with creating VN. If different Server Per-port VN configurations will be used, you must create unique Link Labels.

Create Rack Type

Topology

Summary

Leaves

Servers

Label *
Servers

Server count *
2

Logical device *
AOS-2x10-1

2 x 10 Gbps
Leaf

Logical links *

Link1

Attached to leaf *
Leaf1

Attachment type *
☒ single ☐ dual

Physical link count to leaf *
1

Speed *
10 Gbps

Link2

Attached to leaf *
Leaf2

Attachment type *
☒ single ☐ dual

Physical link count to leaf *
1

Speed *
10 Gbps

+ Add New Link

+ Add New Server Group

The resulting Rack Type shows the Leafs and how the Servers are connected to these leafs.

Create Server Per-port Virtual Networks

When Per-port Rack Type(s) have been added to a Template and a Blueprint has been created and deployed from the Template, you can create Virtual Networks. If any Per-port Rack Types are associated with the Template and Blueprint, the **Default Endpoint Types** field lists the defined Link Labels. You can select **Unassigned**, **Untagged** and **VLAN Tagged** as the default mode for each Link Label. The same options are available for both VLAN (leaf-local) and VXLAN (inter-leaf) VN.

/ Blueprints / Pod

Create Virtual Network

Virtual Network Parameters

Type
☒ VLAN ☐ VXLAN

Will create a VLAN per selected node

Name

Security Zone
default

Default VLAN ID
From resource pool

DHCP Service
☒ Disabled
☐ Enabled

L3 Connectivity
☒ Disabled
☐ Enabled

Default Endpoint Types

| Link Label | Tag Type |
|------------|--|
| link1 | <input checked="" type="radio"/> Unassigned <input type="radio"/> Untagged <input type="radio"/> VLAN Tagged |
| link2 | <input checked="" type="radio"/> Unassigned <input type="radio"/> Untagged <input type="radio"/> VLAN Tagged |

Assigned To

| <input type="checkbox"/> | Bound To | Link Labels | VLAN ID |
|--------------------------|---------------------------|-------------|-----------------|
| <input type="checkbox"/> | I2_virtual_dual_001_leaf1 | link1 | Default VLAN ID |
| <input type="checkbox"/> | I2_virtual_dual_001_leaf2 | link2 | Default VLAN ID |
| <input type="checkbox"/> | I2_virtual_dual_002_leaf1 | link1 | Default VLAN ID |
| <input type="checkbox"/> | I2_virtual_dual_002_leaf2 | link2 | Default VLAN ID |
| <input type="checkbox"/> | I2_virtual_dual_003_leaf1 | link1 | Default VLAN ID |
| <input type="checkbox"/> | I2_virtual_dual_003_leaf2 | link2 | Default VLAN ID |

☐ Create Another?

To edit the endpoint after the VN has been created, select the VN from the **Virtual Networks** list and go to **Endpoints**. By default the associated devices and ports are displayed in the **Topology View**. The VN Endpoints are displayed on a leaf per-port basis. Click on the appropriate leaf port then select the endpoint mode to use between **Unassigned**, **Untagged** and **VLAN Tagged**.

| | |
|---------------------------|---|
| I2_virtual_dual_003_leaf1 | 3 |
| I2_virtual_dual_003_leaf2 | 3 |

Endpoints

Topology View

Links View

I2_virtual_dual_001_leaf1

I2_virtual_dual_001_leaf2

I2_virtual_dual_001_server001

I2_virtual_dual_001_server002

I2_virtual_dual_002_leaf1

I2_virtual_dual_002_leaf2

I2_virtual_dual_002_server001

I2_virtual_dual_002_server002

I2_virtual_dual_003_leaf1

I2_virtual_dual_003_leaf2

I2_virtual_dual_003_server001

I2_virtual_dual_003_server002

vn3

Click on ports to select endpoints and make them:

Unassigned

or

Untagged

or

VLAN Tagged

Clear selection

Search endpoints by interface name, server or link label

I2_virtual_dual_001_leaf1

1 2 3 4 5 6 7

I2_virtual_dual_001_leaf2

1 2 3 4 5 6 7

I2_virtual_dual_002_leaf1

1 2 3 4 5 6 7

I2_virtual_dual_002_leaf2

1 2 3 4 5 6 7

I2_virtual_dual_003_leaf1

1 2 3 4 5 6 7

I2_virtual_dual_003_leaf2

1 2 3 4 5 6 7

The **Topology View** has a search field to allow you to search for specific endpoints.

Endpoints

Topology View

Links View

I2_virtual_dual_001_leaf1

I2_virtual_dual_001_leaf2

I2_virtual_dual_001_server001

I2_virtual_dual_001_server002

I2_virtual_dual_002_leaf1

I2_virtual_dual_002_leaf2

I2_virtual_dual_002_server001

I2_virtual_dual_002_server002

I2_virtual_dual_003_leaf1

I2_virtual_dual_003_leaf2

I2_virtual_dual_003_server001

I2_virtual_dual_003_server002

vn3

Click on ports to select endpoints and make them:

Unassigned

or

Untagged

or

VLAN Tagged

server002

I2_virtual_dual_001_leaf1

1

2

3

4

5

6

7

I2_virtual_dual_001_leaf2

1

2

3

4

5

6

7

I2_virtual_dual_002_leaf1

1

2

3

4

5

6

7

I2_virtual_dual_002_leaf2

1

2

3

4

5

6

7

I2_virtual_dual_003_leaf1

1

2

3

4

5

6

7

I2_virtual_dual_003_leaf2

1

2

3

4

5

6

7

Editing VLAN Virtual Network

1. From the blueprint, navigate to **Staged / Virtual / Virtual Networks**, then click the name of the VN to edit.

Blueprints > L2V > Staged > Virtual > Virtual Networks

Dashboard Analytics Staged Uncommitted Active Time Voyager

Physical Virtual Policies Catalog Settings Tasks

Virtual Networks Security Zones Remote EVPN Gateways Virtual Infra Endpoints

Parameters

| | |
|-------------------|-------------|
| Name | VLAN2 |
| Type | VLAN |
| Security Zone | default |
| VLAN ID | 2 |
| DHCP Service | Disabled |
| IPv4 Connectivity | Enabled |
| IPv4 Subnet | 10.0.0.0/24 |

Assigned To

Active Tasks: 0

2. Click the **Edit** button, and make your changes.
3. Click **Update** to stage the changes and return to the list view.

Deleting Virtual Network

1. From the blueprint, navigate to **Staged / Virtual / Virtual Networks**, then either from the list view or the details view, click the **Delete** button (trash can) for the VN to delete.

Blueprints > evpn-nxosv-virtual

Dashboard Analytics Staged Uncommitted Active

Physical Virtual Policies Catalog Settings

Virtual Networks Security Zones Virtual Infra Internal Endpoints External Endpoints Enforcement Points Endpoint Groups

Parameters

| | |
|----------------------|------------------|
| Name | red_vxlan_102_v4 |
| Type | VXLAN |
| Security Zone | red |
| VNI ID | 10000 |
| DHCP Service | Enabled |
| IPv4 Connectivity | Enabled |
| IPv4 Subnet | 192.168.106.0/24 |
| Virtual Gateway IPv4 | 192.168.106.1 |

Assigned To

2. Click **Delete** to stage the deletion and return to the list view.

Moving Endpoints Assignments

To overwrite an existing untagged (server) port by a new virtual network, follow these steps:

1. From the blueprint, navigate to **Staged / Virtual / Virtual Networks**, then click the name of the virtual network, and select the tagged port to change.
2. Set the port to “Untagged”. As this port was already assigned to a different virtual network, you will see a warning before forcing the move. Click “Force Move” and commit.

As an example, we have these two virtual networks named VLAN2 and VLAN3:

Blueprints > L2V > Staged > Virtual > Virtual Networks

Dashboard Analytics Staged Uncommitted Active Time Voyager

Physical Virtual Policies Catalog Settings Tasks

Virtual Networks Security Zones Remote EVPN Gateways Virtual Infra Endpoints

Create Virtual Networks

Query: All 1-2 of 2 Page Size: 25

| Name | Security Zone | Type | VN ID | Assigned to | DHCP Service | L3 Connectivity | IPv4 Connectivity | IPv4 Subnet | IPv6 Connectivity | IPv6 Subnet |
|-------|---------------|------|-------|-------------|--------------|-----------------|-------------------|----------------|-------------------|-------------|
| VLAN2 | default | VLAN | 2 | 1 nodes | Disabled | Enabled | Enabled | 10.0.0.0/24 | Disabled | N/A |
| VLAN3 | default | VLAN | 3 | 1 nodes | Disabled | Enabled | Enabled | 172.30.16.0/24 | Disabled | N/A |

Build 2/2 SVI Subnets - Virtual Networks

Active Tasks: 0

Currently, our server is attached to port swp5, where VLAN3 has been configured as untagged.

The screenshot shows the Apstra Virtual Networks interface. The breadcrumb navigation is: Blueprints > L2V > Staged > Virtual > Virtual Networks. The 'Staged' tab is active, with other tabs being Dashboard, Analytics, Uncommitted, Active, and Time Voyager. Below the tabs are filters: Unassigned, Untagged (selected), and VLAN Tagged. There are also buttons for Select All and Reset.

Selected Endpoints:

Query: All 1 of 1 Page Size: 5

| Leaf | Interface Name(s) | Server | Server Group | Logical Link | Endpoint | Actions |
|----------------------|-------------------|--------------------------|--------------|--------------|----------|--------------|
| I2_virtual_001_leaf1 | swp5 | I2_virtual_001_server001 | server | link | Untagged | [Trash Icon] |

Port Maps:

Query: All

Ports: Unassigned Untagged (selected) VLAN Tagged Partial 1 of 1 Page Size: 5

I2_virtual_001_leaf1

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

Port #5 Tr. #1 (10 Gbps) swp5

Active Tasks: 0

If you would like to change this port to VLAN2 as untagged, proceed as described above. Click on “VLAN2” in the **Virtual Networks** tab, and select server port swp5, which is currently configured as “Tagged”. Click “Untagged” as displayed in the following capture:

The screenshot displays the Apstra Virtual Networks management interface. The breadcrumb navigation at the top reads: **Blueprints > L2V > Staged > Virtual > Virtual Networks**. Below this, a series of tabs are visible: **Dashboard**, **Analytics**, **Staged** (active), **Uncommitted**, **Active**, and **Time Voyager**. Each tab has a green checkmark icon.

Below the tabs, a message states: "Click on ports to select endpoints and make them:". Below this message are three buttons: **Unassigned** (grey), **Untagged** (purple, highlighted with a green box), and **VLAN Tagged** (teal). To the right of these buttons are **Select All** and **Reset** buttons.

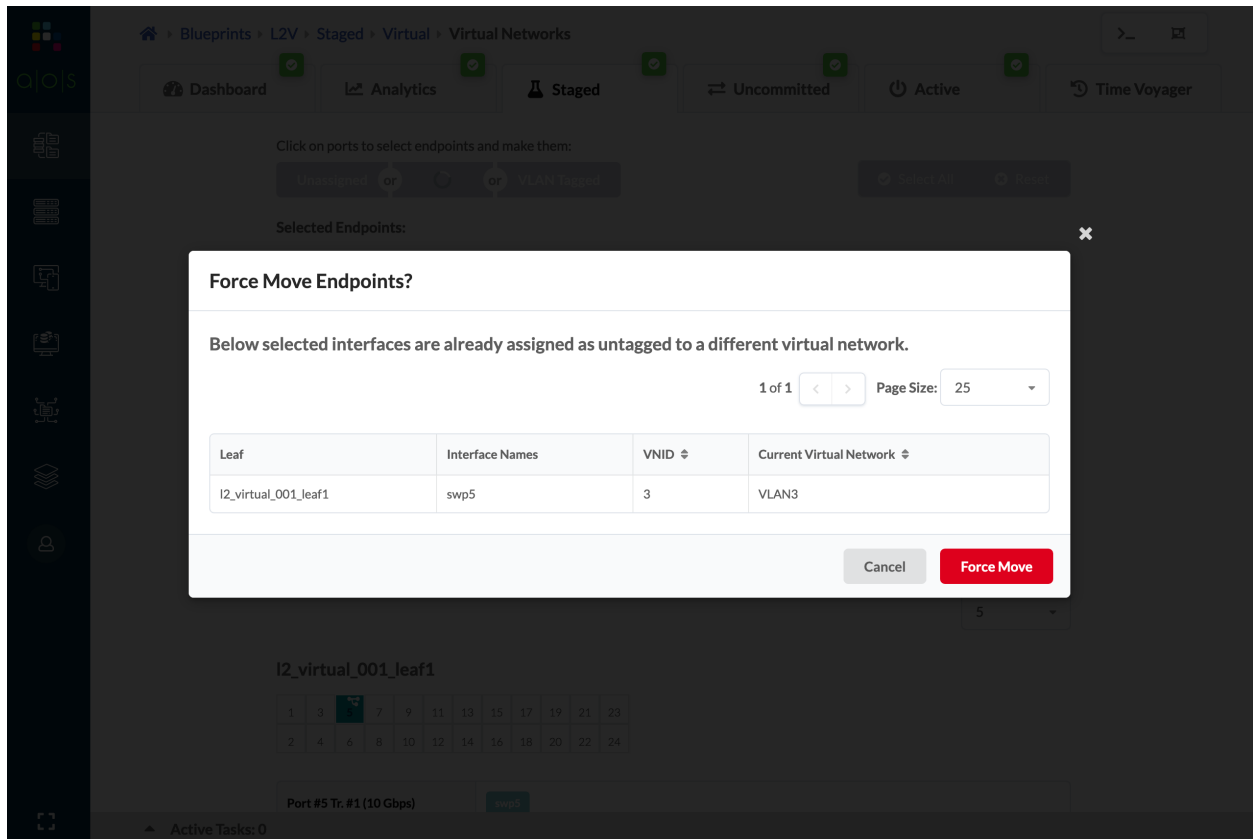
The **Selected Endpoints:** section contains a search bar with "Query: All", pagination showing "1 of 1", and a "Page Size" dropdown set to "5". Below this is a table with the following columns: **Leaf**, **Interface Name(s)**, **Server**, **Server Group**, **Logical Link**, **Endpoint**, and **Actions**. The table contains one row for **I2_virtual_001_leaf1** with interface **swp5** and server **I2_virtual_001_server001**. The **Endpoint** column shows a teal **VLAN Tagged** button, and the **Actions** column shows a trash icon.

The **Port Maps:** section has a search bar with "Query: All". Below it, a legend for **Ports** shows: **Unassigned** (grey), **Untagged** (purple), **VLAN Tagged** (teal), and **Partial** (blue). To the right is pagination showing "1 of 1" and a "Page Size" dropdown set to "5".

The **I2_virtual_001_leaf1** section shows a 2x24 grid of ports. Port 5 is highlighted with a green box and a blue icon.

At the bottom, a status bar shows **Active Tasks: 0**.

And confirm by clicking on “Force Move” to overwrite the previous Virtual Network VLAN3 assignment.



To complete the change you will need to confirm it from the **Uncommitted Tab**

Updating Blueprint

Any virtual network changes made to the Staged Blueprint can be examined by clicking the **Uncommitted** ribbon button. Here you have opportunity to view and verify any intended changes prior to deploying to the **Active** Blueprint.

In this example, we can verify the virtual network “VLAN101” was removed from the staged blueprint and can be deployed by clicking on the rocket icon. If you do not want to deploy the change and want to revert the staged changes to the original blueprint, click the revert “undo” icon.

| Type | Action | Name |
|-----------------|---------|---------------------|
| Virtual Network | REMOVED | red_vxlan_102_v4 |
| Virtual Network | ADDED | blue_131_leaf3_dual |

VLANs can be added and deleted, and endpoints can be added and removed while the blueprint is deployed.

Virtual Network Telemetry

When virtual network endpoints are created on servers, expectations are set for the leaf interfaces facing those servers.

4.4.2.3 Security Zones

A security zone is an L3 domain, the unit of tenancy in multi-tenant networks. You create security zones to isolate tenants' IP traffic from each other, thus enabling tenants to re-use IP subnets.

Some of the characteristics and requirements of security zones include:

- Each security zone is its own Virtual Routing and Forwarding (VRF) instance.
- You can stretch Layer 2 applications across multiple racks within a security zone by creating one or more virtual networks within that security zone.
- A default security zone is assigned to all blueprints. Any SVIs on virtual networks in the default security zone are in the default VRF. This is the same VRF used for the underlay (fabric network) routing between network devices.
- For virtual networks with Layer 3 SVI, the SVI is associated with a VRF instance for each security zone isolating the virtual network SVI from other virtual network SVIs in other security zones.
- You can override the default routing policy when assigning connectivity points.
- You can assign each security zone its own DHCP relay server and external router connections.
- The number of security zones is limited only by the network devices that you're using.
- To route between security zones you must use external routers that are connected to leafs.

Security zones include the following details:

VRF Name 64 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 router connections. Leave this field blank to have it automatically assigned from a static pool (2-4094), or enter a specific value.

VNI VxLAN VNI associated with the security zone. Enter a value, or leave this field blank to pull one from a resource pool later.

Routing Policies This default routing policy may be overridden when configuring external router links.

Import Policy

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 route is permitted.

Extra Only Only user-defined extra import routes are permitted.

Extra Import Routes User-defined import routes

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.

Export Policies

Spine Leaf Links Exports all spine-leaf (fabric) links within a VRF. EVPN security zones do not have spine-leaf addressing, so this generated list may be empty. For security zones of type Virtual L3 Fabric, sub-interfaces between spine-leaf are included.

L3 Edge Server Links Exports all leaf to L3 server links within a security 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 security zone (VRF).

Loopbacks Exports all loopbacks within a security zone (VRF) across spine, leaf, and L3 servers.

Extra Export Routes User-defined export routes. These policies are additive. To advertise extra routes only, unselect all export policies.

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.

Note: To enable default route for EVPN host routes, go to **Staged > Settings > Virtual Network Policy** and enable the "Generate EVPN host routes" option.

Aggregate Prefixes Aggregate routes to be imported into a routing zone (VRF) on all border switches. This option can only be set on routing policies associated with security zones, and cannot be set on per-connectivity point policies. The aggregated routes are sent to all external router peers in a security zone (VRF). Aggregate Prefixes must be specified with an exact network/prefixlen.

To see security zones in the blueprint, navigate to **Staged > Virtual > Security Zones**.

The screenshot shows the Apstra UI navigation. The top bar has tabs: Dashboard, Analytics, Staged, Uncommitted, Active, and Time Voyager. The left sidebar has categories: Physical, Virtual, Policies, Catalog, Settings, and Tasks. Under 'Virtual', there are sub-items: Virtual Networks, Security Zones, Remote EVPN Gateways, Virtual Infra, and Endpoints. The 'Security Zones' sub-item is highlighted. The main content area shows a 'Create Security Zone' button and a table of security zones.

| 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 |

Creating Security Zone

Only blueprints that were created from a *template* with **MP-EBGP EVPN** overlay control protocol can use non-default security zones. If you're using a blueprint that was created from a template with **Static VXLAN** overlay control protocol you cannot create security zones. You must use the default security zone.

Make sure that any external routers are connected to racks. Connecting external routers to spines is not supported when overlay control protocol is **MP-EBGP EVPN**.

1. From the blueprint, navigate to **Staged > Virtual > Security Zones** and click **Create Security Zone**.
2. Enter a VRF name.
3. You can leave the remaining fields as is to use default values and have resources assigned from pools or you can configure it manually. See the security zone overview for details.
4. Click **Create** to stage the security zone and return to the list view.

The new security zone requires resources (leaf loopback IPs, leaf L3 peer links, EVPN L3 VNIs, as applicable). See the next section for how to assign them.

Assigning Resources to Security Zone

You must assign a loopback IP address for each leaf network device in each security zone. If IPv6 is enabled on the blueprint, you must also assign IPv6 addresses to the security zone.

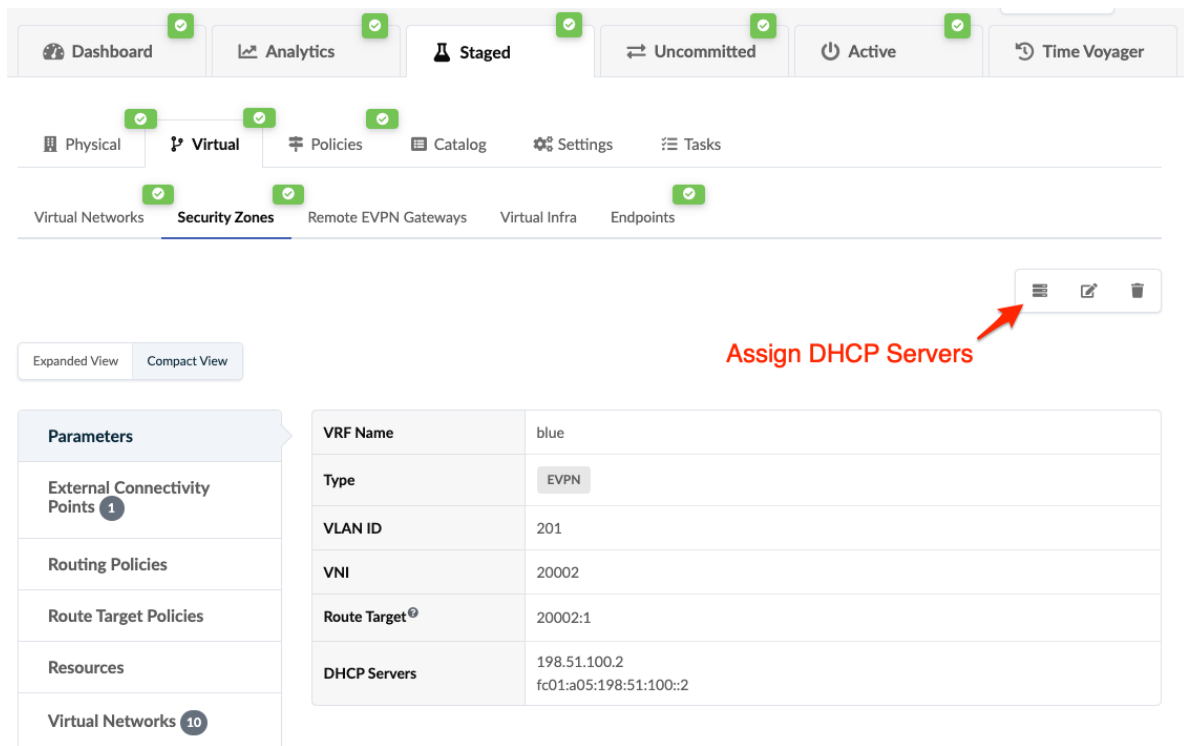
Resource Pools

For information about working with resource pools, see [Resources](#).

1. From the blueprint, navigate to **Staged > Virtual > Security Zones**.
2. Red status indicators in the **Build** panel (on the right) mean that resources need to be assigned. Click a red indicator and click the **Update assignments** button.
3. Select a pool from which to pull the resources, then click the **Save** button. When the red status indicator turns green, it means that the required resources were successfully assigned.
4. Repeat the steps to assign resources from pools until all required resources have been assigned.

Assigning DHCP to Security Zone

1. From the blueprint, navigate to **Staged > Virtual > Security Zones** and click the name of the security zone to assign a DHCP server to.
2. Click the **Assign DHCP Servers** button (upper-right).



The screenshot shows the Apstra interface with the following elements:

- Navigation Bar:** Dashboard, Analytics, Staged, Uncommitted, Active, Time Voyager.
- Left Sidebar:** Physical, Virtual, Policies, Catalog, Settings, Tasks.
- Sub-Menu:** Virtual Networks, Security Zones (selected), Remote EVPN Gateways, Virtual Infra, Endpoints.
- View Toggle:** Expanded View, Compact View.
- Parameters Panel:**
 - External Connectivity Points 1
 - Routing Policies
 - Route Target Policies
 - Resources
 - Virtual Networks 10
- Table:**

| | |
|--------------|--|
| VRF Name | blue |
| Type | EVPN |
| VLAN ID | 201 |
| VNI | 20002 |
| Route Target | 20002:1 |
| DHCP Servers | 198.51.100.2 fc01:a05:198:51:100::2 |
- Action Buttons:** Assign DHCP Servers (indicated by a red arrow), Edit, Delete.

3. Enter the IPv4 address (or IPv6 address) for one or more DHCP servers.
4. Click **Update** to stage the assignment and return to the security zone.

Adding Connectivity Point to Security Zone

If you're connecting external routers in an EVPN blueprint, you must add external connectivity points to the security zone (as of version 3.0).

If you are connecting OSPF external routers in a blueprint (because of limitations running BGP between the fabric and external routers), you must add OSPF external connectivity points to the security zone (as of version 3.2).

1. From the blueprint, navigate to **Staged > Virtual > Security Zones** and click the VRF name for the security zone to add connectivity points to.
2. In the **External Connectivity Points** section, click **Add Connectivity Point** and configure the external connectivity point as described below:

Parameters

Connectivity Type L2 or L3 - must match the type defined when you *assigned external routers to the blueprint*.

Routing Protocol BGP or OSPF

Peering Type (for BGP routing protocol) Loopback (EBGP multi hop loopback) or Interface

VLAN ID The tagged VLAN ID when the external connectivity point is an L2 SVI interface.

IPv4 Subnet You must explicitly assign the IPv4 subnet used for the external connectivity point. It cannot be assigned from a resource pool.

Enable IPv6 If IPv6 is enabled on the blueprint (as of version 3.0), select whether IPv6 must be enabled for the external connectivity point.

IPv6 Subnet If IPv6 is enabled for the external connectivity point, you must explicitly assign the IPv6 subnet for the external connectivity point. It cannot be assigned from a resource pool.

OSPF Settings (for OSPF routing protocol)

OSPF Area ID 32-bit integer or IP address that external router and this router will participate in. All connectivity points of type OSPF in a VRF must be in same **OSPF Area ID** and if they are not, an API error is raised.

Enable advanced OSPF options

MTU Ignore All NOS defaults to performing an MTU check between neighbors. Check MTU Ignore if you want to disable MTU check between neighbors.

Bidirectional Forwarding Detection (BFD) Disabled by default. Check BFD to enable BFD for faster OSPF failure detection mechanisms.

Hello/Dead Interval Timer Must match between neighbors. Define custom interval timers if required.

MDS key ID/key We discourage clear-text passwords. Therefore you can specify the MD5 password. A key ID, used for gracefully migrating MD5 keys, must match between neighbors. When MD5 authentication is chosen, you must specify both the key and the associated key-ID.

Network type Broadcast or Point-to-Point.

Note: OSPF specific timers other than Hello/Dead interval such as SPF throttling, LSA throttling, LSA group pacing, LSA arrival can be customized with *configlets*.

We do not support changing the default Router Priority value.

Links Select the links to use for the external connectivity point.

Resources For L3 external connectivity points, an IP is automatically assigned from the IPv4 subnet for each external connectivity point device.

For L2 external connectivity points, you must manually assign IP addresses for each node. If IPv6 is enabled for the external connectivity point, enter the IPv6 addresses for each node.

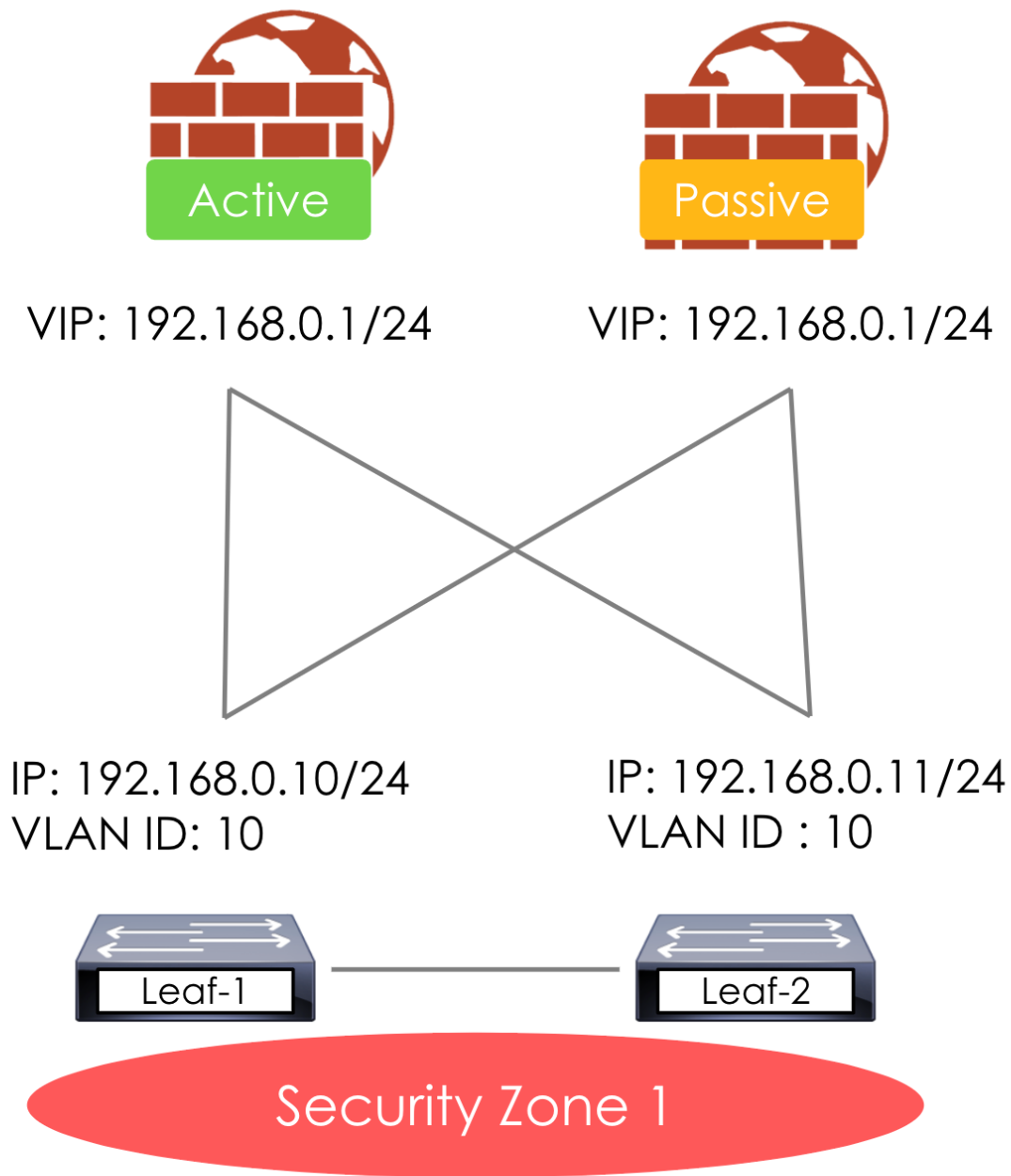
Routing Policies / Enable Routing Policies Overrides If checked, you can override the default routing policies that you defined when creating the security zone.

External Connectivity Point Scenario - Shared ECP

In this scenario, a shared upstream gateway (e.g. Active/Passive Firewalls) is connected to a leaf or an MLAG leaf-pair. The default or any new security zone is connected to two external firewalls using a single IP connectivity point (the shared virtual IP address). Each security zone uses a different IP connectivity point (with a different SVI interface / VLAN ID and different VIP). On each border leaf, 1 EBGP session is established between its SVI and the shared virtual IP of the firewall pair.

Custom export/import policies per security zone can be defined:

- Enable/Disable export of l2edge_subnets
- Import specific prefix-list (new)
- Export aggregated subnets (new)



To configure this when adding the external connectivity point, you must add both “links”.

Type

EVPN

VLAN ID

2

×

Add External Connectivity Point

Parameters

Connectivity Type *

☒ L2
☐ L3

Peering Type *

☐ Loopback
☒ Interface

VLAN ID *

201

IPv4 Subnet *

172.16.0.32/29

Links

I2_mlag_ext_001_leaf_pair1<->FW-CORP-A
I2_mlag_ext_001_leaf1: port-channel3, I2_mlag_ext_001_leaf2: port-channel3 (bond) <-> FW-CORP-A

I2_mlag_ext_001_leaf_pair1<->FW-CORP-B
I2_mlag_ext_001_leaf1: port-channel4, I2_mlag_ext_001_leaf2: port-channel4 (bond) <-> FW-CORP-B

Select...

Add Link

Add

Loopbacks *

yes

Extra Export Routes ®

not provided

Under “Resources” there will be an “IPv4 Address” field for each link node. Place the same “VIP” IP address in the link node “IPv4 Address” field.

Type

EVPN

VLAN ID

2

×

Add External Connectivity Point

I2_mlag_ext_001_leaf_pair1<->FW-CORP-A
I2_mlag_ext_001_leaf1: port-channel3, I2_mlag_ext_001_leaf2: port-channel3 (bond) <-> FW-CORP-A

I2_mlag_ext_001_leaf_pair1<->FW-CORP-B
I2_mlag_ext_001_leaf1: port-channel4, I2_mlag_ext_001_leaf2: port-channel4 (bond) <-> FW-CORP-B

Select...

Add Link

Resources

| Node | IPv4 Address |
|-----------------------|----------------|
| I2_mlag_ext_001_leaf1 | 172.16.0.34/29 |
| I2_mlag_ext_001_leaf2 | 172.16.0.35/29 |
| FW-CORP-A | 172.16.0.33/29 |
| FW-CORP-B | 172.16.0.33/29 |

Add

Loopbacks *

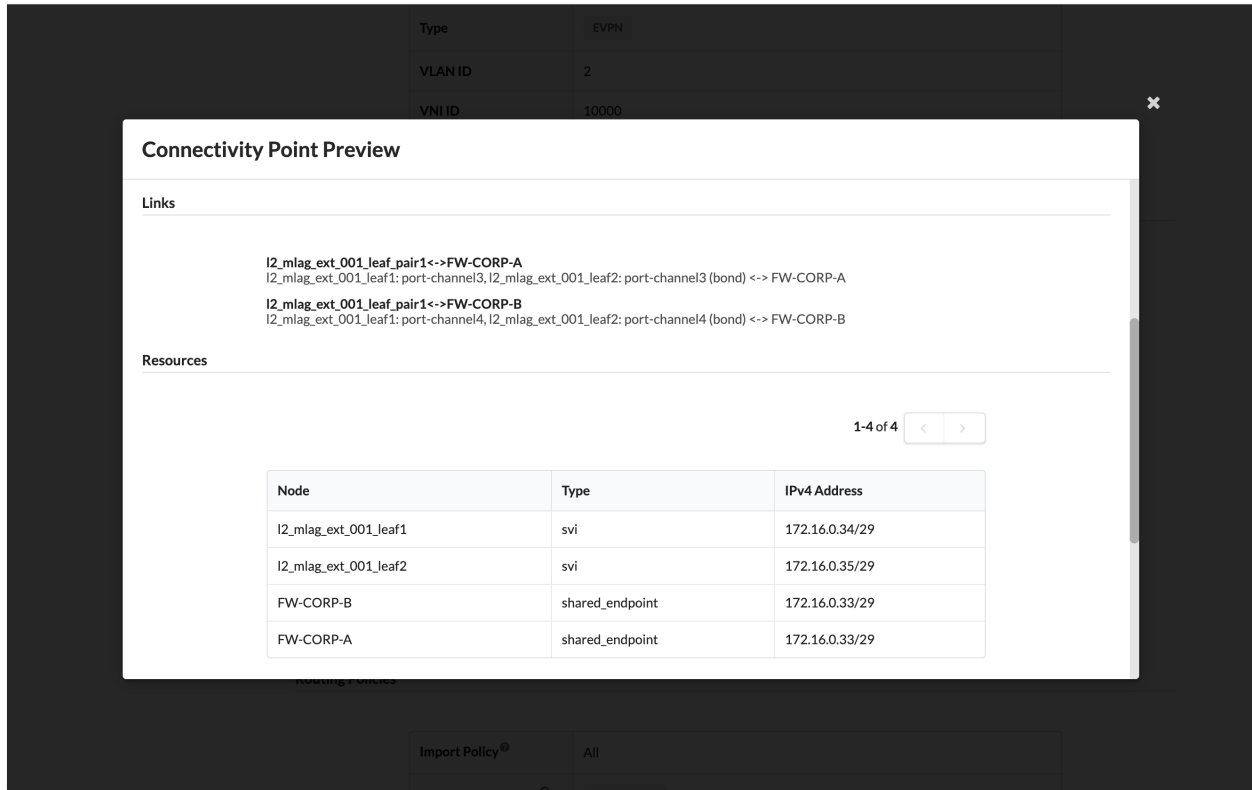
yes

Extra Export Routes ®

not provided

When you view the “Details” of the ECP, you will see the (2) link nodes as “shared_endpoint” along with the External

EBGP sessions.

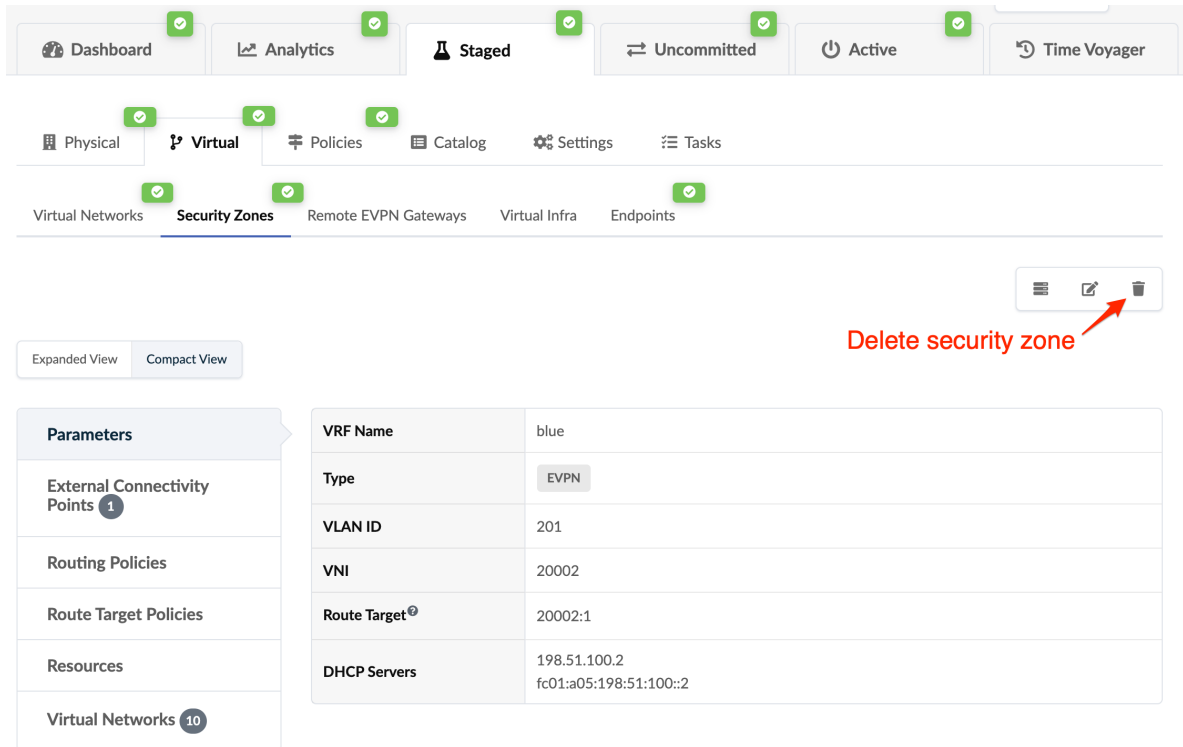


Editing Security Zone

1. From the blueprint, navigate to **Staged > Virtual > Security Zones** and click the name of the security zone to edit.
2. Click the **Edit** button (upper-right) and make your changes.
3. Click **Confirm** to stage the change and return to the security zone.

Deleting Security Zone

1. From the blueprint, navigate to **Staged > Virtual > Security Zones** and click the name of the security zone to delete.
2. Click the **Delete** button (upper-right). All virtual networks that were created under that security zone will also be deleted.



3. Click **Confirm** to stage the deletion and return to the list view.

4.4.2.4 Data Center Interconnect / EVPN Gateways

Historically, enterprises have leveraged Data Center Interconnect (DCI) technology as a building block for business continuity, disaster recovery (DR), or Continuity of Operations (COOP). These service availability use cases primarily relied on the need to connect geographically separated data centers with Layer-2 connectivity for application availability and performance.

With the rise of highly virtualized Software-Defined Data Centers (SDDC), cloud computing, and more recently, edge computing, additional use cases have emerged:

- **Colocation Expansion:** Share compute and storage resources to colocation data center facilities.
- **Resource Pooling:** Share and shift applications between data centers to increase efficiency or improved end-user experience.
- **Rapid Scalability:** Expand capacity from a resource-limited location to another facility or data center.
- **Legacy Migration:** Gracefully move applications and data off older and inefficient equipment and architecture to more efficient, higher-performing, and cost-effective architecture.

Users can use Apstra AOS to deploy and manage a vendor inclusive DCI solution that is simple, flexible, and Intent-Based. AOS utilizes the standards-based MP-BGP EVPN with VXLAN, which has achieved broad software and hardware adoption in the networking industry. Customers 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 L2 broadcast domains between hosts as well as Layer-3 (L3) routed domains in spine-leaf networks. Relying on a pure L3 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 (L2) between hosts, the DCI feature enables L2 connectivity between sites. The AOS 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 DC to another.

AOS DCI Options

Apstra AOS implemented the DCI feature to be open and flexible, with three different use cases.

1. **DCI using Over the Top**
2. **DCI using Gateways (GW)**
3. **DCI using Autonomous System Border Router (ASBR)**

There are two important details to note, regardless of which deployment option you choose.

1. You can extend AOS DCI to other AOS managed data centers, non-AOS managed data centers, or even to legacy non-Spine-Leaf devices.
2. AOS implementation and behavior is exactly the same thing in all three cases.

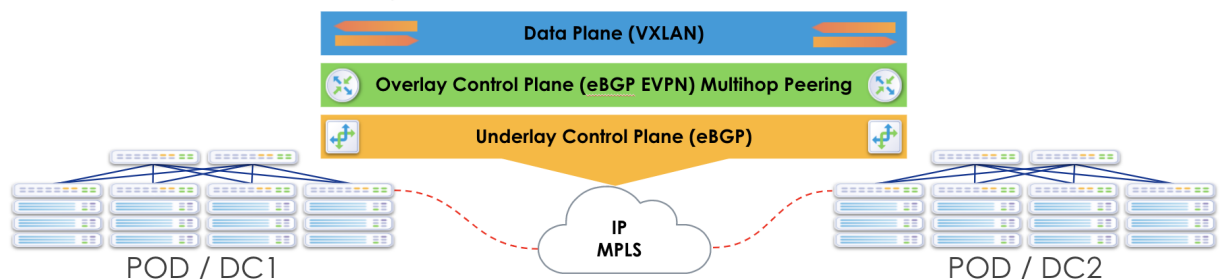
Whether the remote end is another DCI GW or an ASBR, it is transparent to AOS. AOS manages neither the GWs nor ASBRs. Should you have any questions, consult your Apstra Solutions Architect (SA) or Systems Engineer (SE) to determine which option is best for your organization.

DCI Using Over the Top

DCI “Over the Top” is a transparent solution, meaning EVPN routes are encapsulated into standard IP and hidden from the underlying transport. This makes the extension of services simple and flexible and is often chosen because it can be implemented by data center teams with little to no coordination with WAN or Service Provider groups. This reduces the implementation times and internal company friction. However, the tradeoff is scalability and resilience.

DCI - Over the Top

Similar to RFC 4364 - InterAS option C

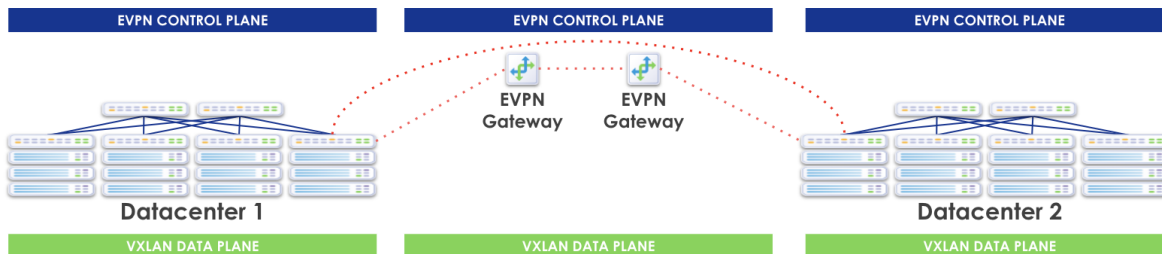


DCI Using Gateway

Building upon the AOS **Remote EVPN Gateway** capability, one can optionally specify the **Remote EVPN Gateway** to be an external router (GW) in the same site, thus extending the EVPN attributes to said GW. 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 an external router GW. WAN QoS and security can also be implemented, along with optimizations made available by the transport technology (e.g., MPLS TE). However, this solution is more operationally complex, requiring additional hardware and cost.

DCI using Gateway

Independent Control Planes - described in RFC 8365, section-10.1



DCI Using ASBR

Using the AOS **Remote EVPN Gateway** capability, one can optionally specify the **Remote EVPN Gateway** to be an ASBR WAN Edge Device. It is an end-to-end EVPN, that enables uniform encapsulation, removes the dedicated GW requirement. It is also 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)



AOS EVPN Gateways DCI Implementation

New in version 3.2: You can extend Security Zones and Virtual Networks (VN) to span across AOS-managed Blueprints (across pods) or to remote networks that are not managed by AOS (across datacenters). This feature introduces the EVPN Gateway (GW) role, which could be a switch that participates in the fabric or RouteServer(s) on a server that is connected to the fabric.

EVPN Gateways Use Cases

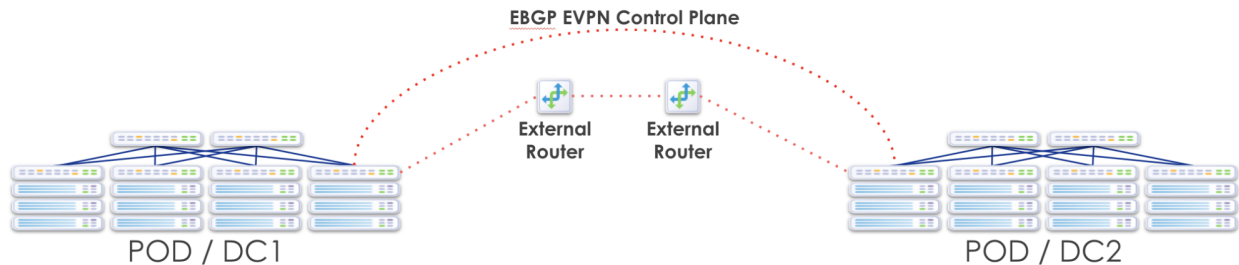
- Span L3 isolation domains (VRFs / Security Zones) to multiple AOS-managed PODs(blueprints) or extend to remote EVPN domains.
- Provide L2 domain extensions for L2VNI/Virtual networks.
- Help extend EVPN domain from AOS to AOS-managed and AOS to unmanaged PODs.
- No VXLAN traffic termination on the spines. This can be achieved by connecting external routers on spines. This is to support IPv4 (underlay) external connectivity. Here Spines don't need to terminate VXLAN traffic unlike border leafs when connected to external routers. This new concept is fundamentally different from Connectivity Points for External Routers. In a nutshell, using this can exchange IPv4 routes to remote VTEPs (in the default Security Zone/VRF) and only L3 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 shall be established between gateways in different DCs.

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 DCs 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 L3 hops between sites over a WAN, otherwise the default TTL will decrement to 0 and packets will be discarded and won't make it to the remote router. AOS 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 L2 and L3 (VRF) services between sites, but also renders the sites as a single fault domain. So a failure in one site will necessarily be propagated. Also, anytime one stretches L2 across the WAN between sites, you are also extending the flood domain and along with it, all broadcast traffic over your costly WAN links. At this time, this solution does not offer any filtering or QoS.

Note: When individual sites are managed by separate AOS Blueprints, or only one site is AOS managed, then extended Security Zones (VRFs) and Virtual Networks (L2 and/or L3 defined VLANs/subnets) have to be created and managed independently in each site. This creates administrative overhead and requires manual mapping of VRFs and VNs between sites.

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,

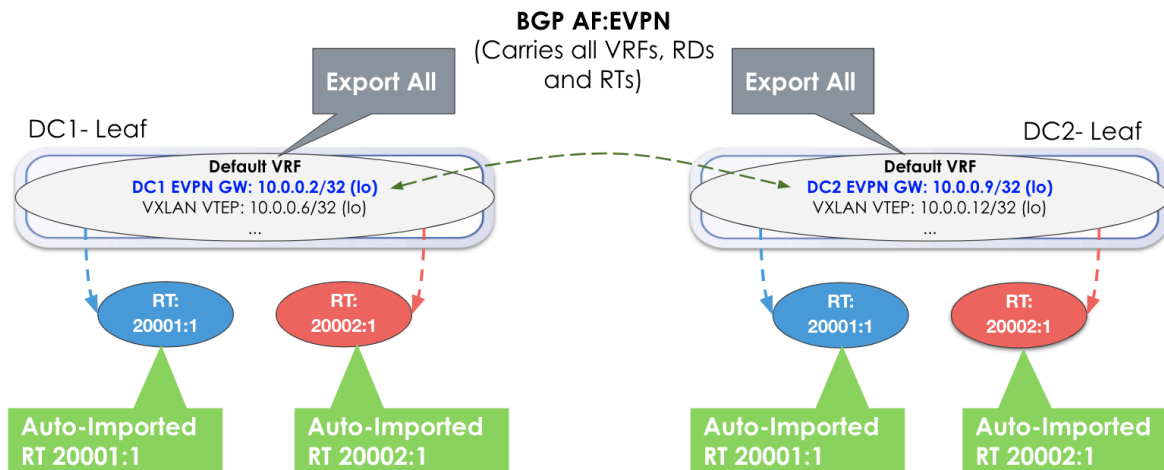
the downside is that there is a single EVPN Control plane and a routing anomaly in one site will affect convergence and reachability in the other site(s). As previously mentioned, the extension of L2 flood domains also implies that a broadcast storm in one site would extend to the other site(s).

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) have to be imported. This is all covered in detail later in this document.

Data Plane Extension: L3 Extension

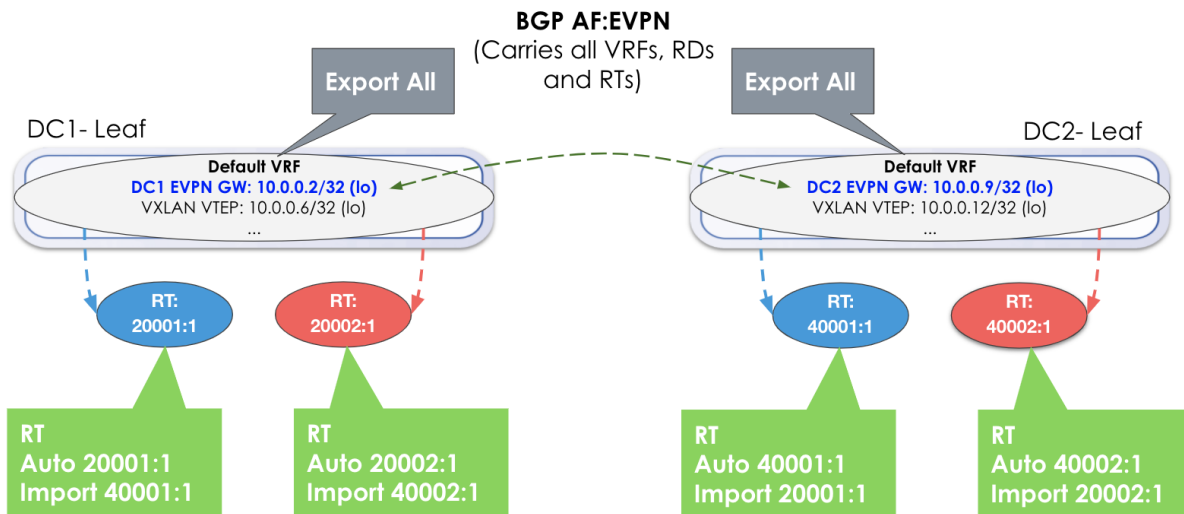
VXLAN Network IDs (VNIDs) are a part of the VXLAN header that identify unique VXLAN tunnels, each of which are isolated from the other VXLAN tunnels in an IP network. L3 packets can be encapsulated into a VXLAN packet or L2 MAC frames can be encapsulated directly into a VXLAN packet. In both cases, a unique VNID will be associated with either the L3 subnet, or the L2 domain. When extending either L3 or L2 services between sites, one is 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 SZ in AOS terminology). VNIDs exist within a VRF. They are tied to a VRF. For L3 services, the stitching, or extending, of each VNID is done with the export and import of RTs within a Security Zone (VRF). L3 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 will automatically be imported at the EVPN gateway (edge) coming into the fabric. So if one coordinates the L3 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 one must import the RTs respectively. Each respective gateway will still automatically import 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: L2 Extension

In AOS, when creating a VN, one can create the VN to be a pure L2 service, meaning no L3 anycast gateway is instantiated. This is for either a rack local L2 (VLAN on server facing ports contained within a rack), or one can extend the L2 flood and broadcast domain between racks by selecting VXLAN and then selecting the racks you want the L2 domain extended to. This L2 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 L2 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 Security Zone level, but instead at the VN itself.

AOS Workflow

Control Plane Extension: The EVPN Gateway

AOS 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 AOS, 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 a AOS EVPN blueprint, this will always be 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 AOS terminology. The Local EVPN Gateway is assumed to be one of the AOS managed devices in the blueprint, and is selected during the creation of the “Remote EVPN Gateway”. The Local EVPN Gateway will be the border-leaf switch with one or more external routing connections for traffic in and out of the EVPN Clos fabric.

Due to this capability, a Local EVPN Gateway (always an AOS managed switch) can be configured to peer with a non AOS 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 AOS, each DC is represented by an autonomous blueprint and if two or more sites are under AOS management, you still have to configure each and every site to point to the “Remote EVPN Gateway(s)” in other sites. Apstra recommends that the user creates multiple, redundant EVPN

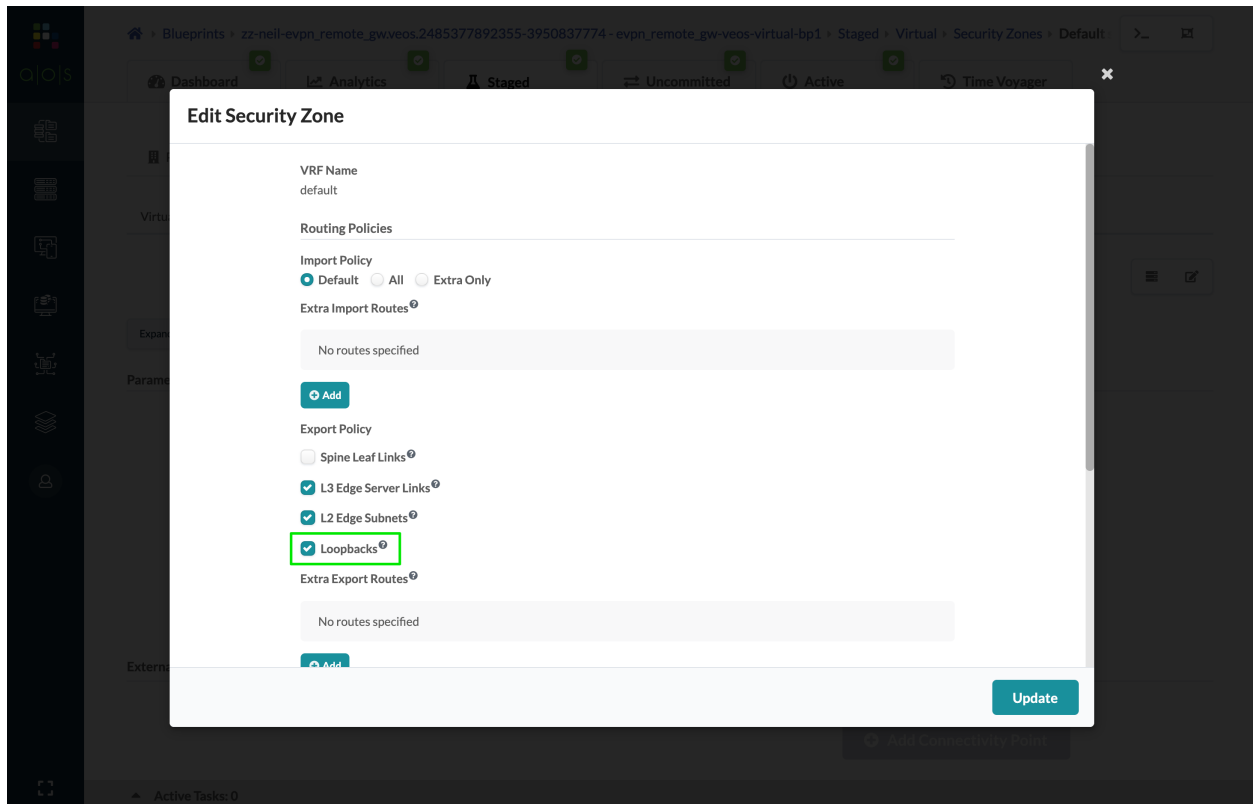
Gateways for each DC. There is also currently a full mesh requirement between EVPN gateways, although in future releases this requirement will be removed.

Underlay VTEP Route Advertisements

It is required that the underlay reachability to VTEP IP addresses or an equivalent summary route, is established reciprocally. Each site must advertise these VTEP loopbacks from within the “Default” Security Zone (SZ) into the exported BGP (IPv4) underlay advertisements.

In AOS, this is done in the user’s blueprint, in the “Staged”, “Virtual”, and “Security Zone” tabs. Open the “default” Security Zone VRF and click the “Edit” icon to edit the External Connectivity Point for the “default” Security Zone VRF.

Under “Routing Policies”, verify the “Export Policy” has “Loopbacks” checked.

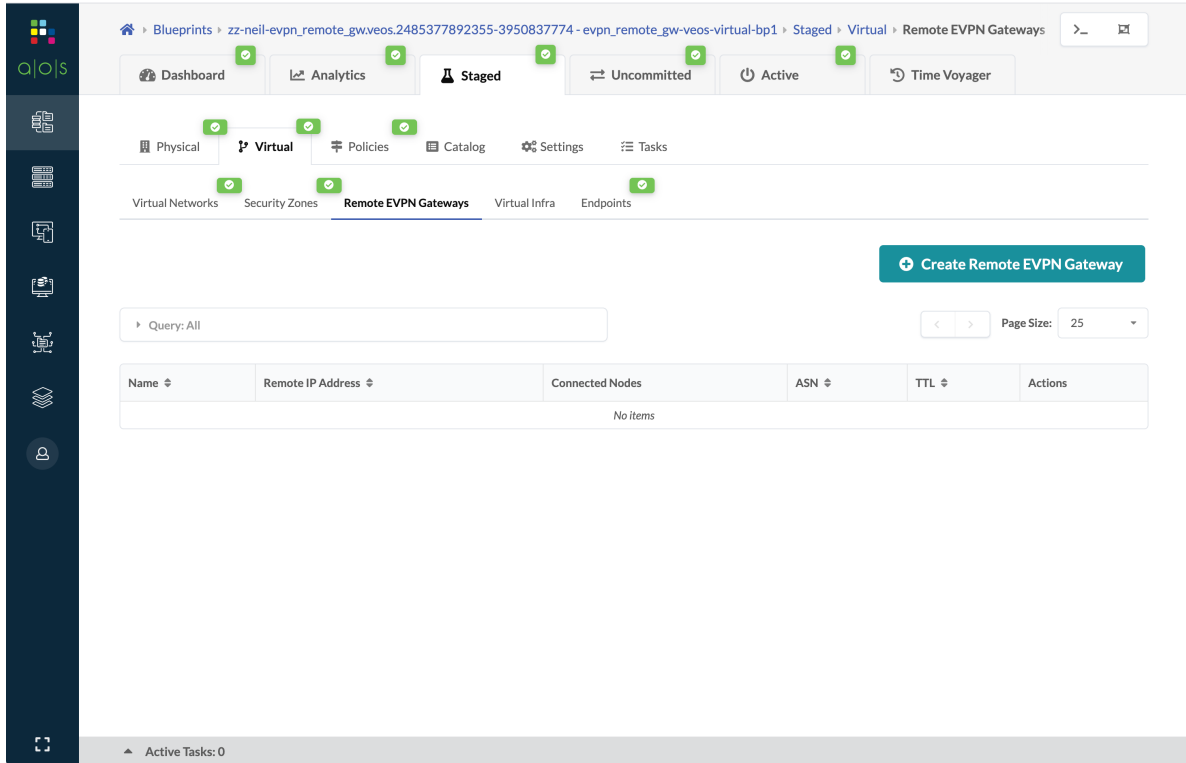


Creating AOS EVPN Gateways

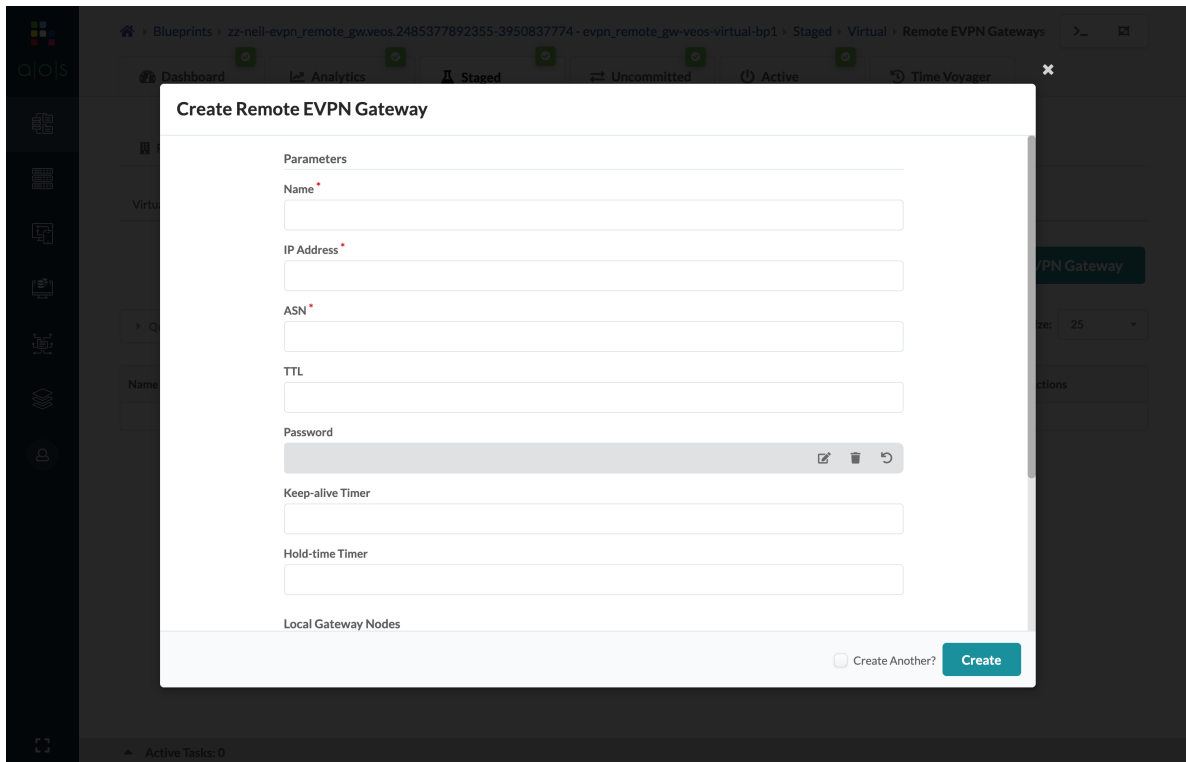
Remote EVPN Gateway is a logical function that could be instantiated anywhere and on any device, and requires support for BGP in general, L2VPN/EVPN AFI/SAFI specifically. To establish a BGP session with an EVPN gateway, IP connectivity, as well as connectivity to TCP port 179 (BGP TCP port allocated by IANA), should be available.

Important: For resilience purposes, we recommended having at least two remote gateways for the same remote EVPN domain.

1. In AOS, EVPN Gateways are defined in the user’s blueprint, in the “Staged”, “Virtual”, and “Remote EVPN Gateways” tabs.



- To add an EVPN Gateway, click **Create Remote EVPN Gateway**.



- Fill in the following information for the Remote EVPN Gateway.

Create Remote EVPN Gateway

Parameters

- Name** *
BP2 EVPN Gateway
- IP Address** *
10.0.0.9
- ASN** *
64517
- TTL**
15
- Password**

- Keep-alive Timer**
5
- Hold-time Timer**
20

Local Gateway Nodes

☐ Create Another? **Create**

1. **Name** - An arbitrary name for the Remote EVPN Gateway
 2. **IP Address** - The IP Address for the Remote EVPN Gateway
 3. **ASN** - The BGP Autonomous System Number (ASN) for the Remote EVPN Gateway
 4. **TTL** - The BGP multi-hop “Time to Live” (maximum number of L3 hops) value for the BGP session with the Remote EVPN Gateway. This is optional and if omitted, AOS will not configure a TTL for this BGP session on the device and the device will use its default.
 5. **Password** - BGP TCP authentication password. This is optional and if omitted, AOS will not configure a password for this BGP session on the device.
 6. **Keep-alive Timer** - The BGP keep-alive timer. This is optional and if omitted, AOS will not configure a keep-alive for this BGP session on the device and the device will use its default.
 7. **Hold-time Timer** - The BGP hold-time timer. This is optional and if omitted, AOS will not configure a hold-time for this BGP session on the device and the device will use its default.
4. Select the **Local Gateway Nodes**. These are the devices in the AOS blueprint that will be configured with a Local EVPN Gateway. You can select one or more devices here to peer with the configured “Remote EVPN Gateway”.

Create Remote EVPN Gateway

TTL: 15

Password: [masked]

Keep-alive Timer: 5

Hold-time Timer: 20

Local Gateway Nodes

Query: All 1-3 of 3 Page Size: 25

| | Label | Role | Group Label | ASN | Hostname |
|-------------------------------------|-------------|------------|-------------|-------|-------------|
| <input type="checkbox"/> | bp1-sspine1 | Superspine | N/A | 64512 | bp1-sspine1 |
| <input type="checkbox"/> | bp1-spine1 | Spine | N/A | 64513 | bp1-spine1 |
| <input checked="" type="checkbox"/> | bp1-leaf1 | Leaf | evpn-single | 64514 | bp1-leaf1 |

☐ Create Another? **Create**

Apstra recommends using multiple border-leafs which have direct connections to External Routers.

- The new Remote EVPN Gateways will be listed on the page.

Create Remote EVPN Gateway

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

| Name | Remote IP Address | Connected Nodes | ASN | TTL | Actions |
|------------------|-------------------|-----------------|-------|-----|-----------------|
| BP2 EVPN Gateway | 10.0.0.9 | 1 nodes | 64517 | 15 | [edit] [delete] |

Apstra recommends using multiple Remote EVPN Gateways. To configure additional Remote EVPN Gateways, the user can click **Create Remote EVPN Gateway** to add another Remote EVPN Gateways.

- From the blueprint, “Staging” tab, the use can commit changes to deploy them to the devices in the AOS blueprint.

The screenshot displays the Apstra AOS interface. The top navigation bar shows the breadcrumb: Blueprints > zz-neil-evpn_remote_gw.veos.2485377892355-3950837774 - evpn_remote_gw-veos-virtual-bp1 > Uncommitted > Logical Diff. The 'Uncommitted' tab is active, and the 'Logical Diff' view is selected. The table below shows a single entry:

| Type | Action | Name |
|---------------------|--------|------------------|
| Remote EVPN Gateway | ADDED | BP2 EVPN Gateway |

Once the change is deployed, AOS will monitor the BGP session for the Remote EVPN Gateways. Any anomalies for this will be shown as a “External Routing” service anomaly.

Blueprints > zz-neil-evpn_remote_gw.veos.2485377892355-3950837774 - evpn_remote_gw-veos-virtual-bp1 > Active > Anomalies

Dashboard Analytics Staged Uncommitted Active Time Voyager

Physical Virtual Policies Settings Query **Anomalies** Root Causes

Query: Service = External Routing and Anomaly Type = bgp 1-1 of 1 Page Size: 25

| Node | Hostname | Service | Anomaly Type | Role | Anomaly Extra Details | Expected | Actual | Time Updated |
|-----------|-----------|------------------|--------------|-------------------|--|------------------------------|--------------------------------|---------------|
| bp1-leaf1 | bp1-leaf1 | External Routing | bgp | To Remote Gateway | Property Value Address Family "evpn" Destination ASN "64517" Destination IP "10.0.0.9" Destination Name "bp2-evpn-gateway" Source ASN "64514" Source IP "10.0.0.2" VRF Name "default" | Property Value value "up" | Property Value value "down" | 2 minutes ago |

Active Tasks: 0

If the user is configuring the Remote EVPN Gateway(s) to another AOS blueprint, the user will need to configure and deploy the Remote EVPN Gateway(s) separately in that AOS blueprint.

Blueprints > zz-neil-evpn_remote_gw.veos.2485377892355-3950837774 - evpn_remote_gw-veos-virtual-bp2 > Staged > Virtual > Remote EVPN Gateways

Dashboard Analytics Staged Uncommitted Active Time Voyager

Create Remote EVPN Gateway

Parameters

Name BP1 EVPN Gateway

IP Address 10.0.0.2

ASN 64514

TTL 15

Password

Keep-alive Timer 5

Hold-time Timer 20

Local Gateway Nodes

☐ Create Another? **Create**

Active Tasks: 0

Enhanced Security Zone

The installation of EVPN routes is mainly governed by RT (route-target) import/export policies on devices which are part of extended service. As per this feature can add additional import and export route-targets used by AOS for Security Zones/VRFs. Please find below related screenshot:

Note: The default route-target generated by AOS for security zones is **<L3 VNI>:1**. This can't be altered.

To make sure correct routes are received at VTEP please make sure L3VNIs and route target are identical between the AOS blueprint and remote EVPN domains.

Enhanced Virtual Networks

You can now add additional import and export route-targets used by AOS for Virtual Networks(VN).

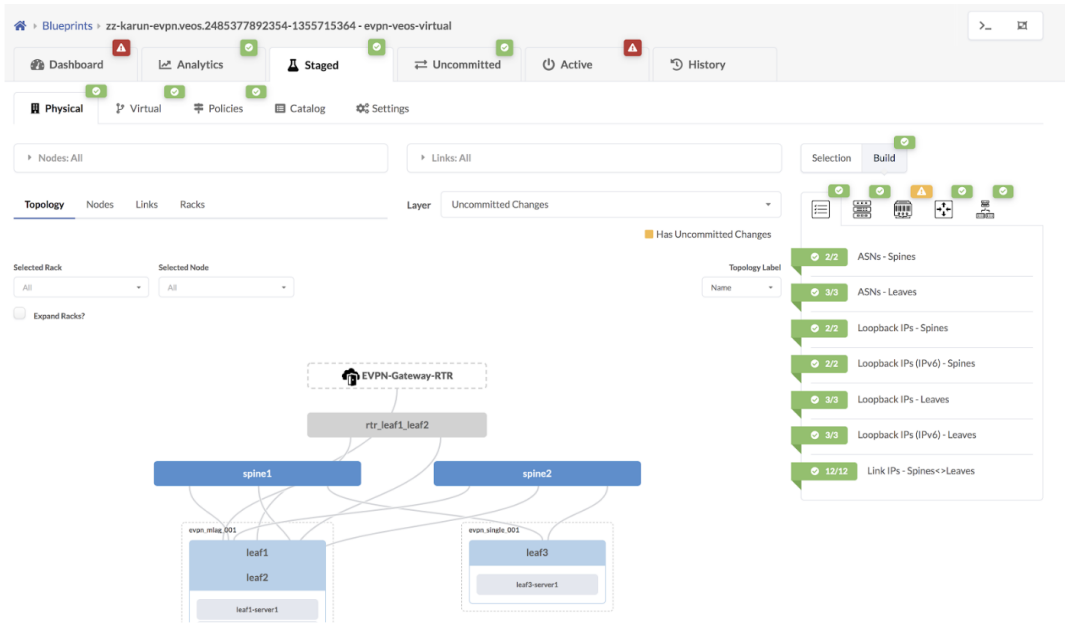
Note: The default route target generated by AOS for AOS for Virtual Networks is **<L2 VNI>:1**. You can't alter this.

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. L2 VNIs need to be the same between the blueprint and remote domains for connectivity to be established.

You also need to ensure that the SVI subnets are 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. Please find the related screenshot as below:



4.4.2.5 Virtual Infra

See the *Integration Guides* for information about how **Virtual Infra** is used.

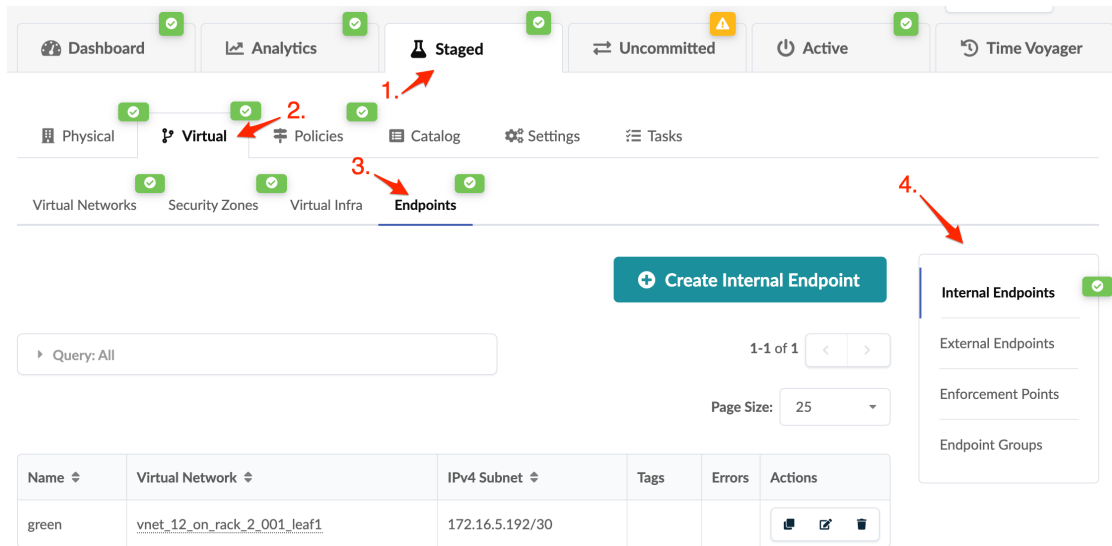
4.4.2.6 Endpoints

Endpoint Overview

When you want more granularity in your security policies than virtual networks and security zones can provide, you'll use endpoints. Endpoints can be internal or external to the fabric. You can also combine endpoints into groups.

Endpoints and security policies can be applied to Layer 2 IPv4 blueprints. (Blueprints with IPv6 applications enabled are not supported.) Endpoints are supported on Cisco NX-OS and Arista EOS physical network devices only. For more information about working with security policies, see *Security Policies*.

You can create, clone, edit and delete endpoints as described in the sections below. Then, when you create a security policy you'll select the endpoints that you've created. To see endpoints from the blueprint, navigate to **Staged > Virtual > Endpoints**, then click the name of a section to go to its list view.



Internal Endpoints

Cloning Internal Endpoint

Instead of entering all details for a new endpoint, you can clone an existing one, give it a unique name and customize it.

Creating Internal Endpoint

1. From the blueprint, navigate to **Staged > Virtual > Endpoints > Internal Endpoints** and click **Create Internal Endpoint**.
2. Configure the endpoint as described below:

Name 32 characters or fewer. Alphanumeric characters, underscores and dashes only.

Virtual Network Select the virtual network where the endpoint is located.

IPv4 Subnet Enter the IPv4 Subnet/CIDR.

Tags (optional) You can add tags for filtering or grouping beyond membership custom groups or virtual networks (for example “web server”, “db” and so on).
3. Click **Create** to stage the endpoint addition and return to the list view. Validation is performed to ensure that the IP address is within the L2 subnet of the virtual network and that no endpoint with the same IP address is within the same security zone.

Editing Internal Endpoint

1. From the blueprint, navigate to **Staged > Virtual > Endpoints > Internal Endpoints** and click the **Edit** button for the endpoint to edit.
2. Make your changes.

3. Click **Update** to stage the endpoint change and return to the list view.

Deleting Internal Endpoint

1. From the blueprint, navigate to **Staged > Virtual > Endpoints > Internal Endpoints** and click the **Delete** button for the endpoint to delete.
2. Click **Delete** to stage the endpoint removal and return to the list view.

External Endpoints

Cloning External Endpoint

Instead of entering all details for a new endpoint, you can clone an existing one, give it a unique name and customize it.

Creating External Endpoint

1. From the blueprint, navigate to **Staged > Virtual > Endpoints > External Endpoints** and click **Create External Endpoint**.
2. Configure the endpoint as described below:
 - Name** 32 characters or fewer. Alphanumeric characters, underscores and dashes only.
 - 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).
 - Enforcement Points (optional)** Enforcement points are supported on external-facing interfaces on border leaf devices only. They are external-facing points where access lists that involve external endpoints are applied. Any external generic, external connectivity points and enforcement groups can be added.
3. Click **Create** to stage the endpoint addition and return to the list view.

Editing External Endpoint

1. From the blueprint, navigate to **Staged > Virtual > Endpoints > External Endpoints** and click the **Edit** button for the endpoint to edit.
2. Make your changes.
3. Click **Update** to stage the endpoint change and return to the list view.

Deleting External Endpoint

1. From the blueprint, navigate to **Staged > Virtual > Endpoints > External Endpoints** and click the **Delete** button for the endpoint to delete.
2. Click **Delete** to stage the endpoint removal and return to the list view.

Enforcement Points

Enforcement points are supported on external-facing interfaces on border leaf devices only. They are automatically created when you add external generics or external connectivity points to a blueprint.

To see enforcement points in the blueprint, navigate to **Staged > Virtual > Endpoints > Enforcement Points**.

Endpoint Groups

Cloning Endpoint Group

Instead of entering all details for a new endpoint group, you can clone an existing one, give it a unique name and customize it.

Creating Endpoint Group

1. From the blueprint, navigate to **Staged > Virtual > Endpoints > Endpoint Groups** and click **Create Endpoint Group**.
2. Configure the endpoint group as described below:
 - Name** 32 characters or fewer. Alphanumeric characters, underscores and dashes only
 - Type** Select the type of endpoint group to create: **Internal Endpoint Group**, **External Endpoint Group**, or **Enforcement Point Group**.
 - Members** Depending on the type of endpoint group you are creating, options for selecting members are presented.
 - 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 list view.

Editing Endpoint Group

1. From the blueprint, navigate to **Staged > Virtual > Endpoints > Endpoint Groups** and click the **Edit** button for the endpoint group to edit.
2. Make your changes.
3. Click **Update** to stage the endpoint group change and return to the list view.

Deleting Endpoint Group

1. From the blueprint, navigate to **Staged > Virtual > Endpoints > Endpoint Groups** and click the **Delete** button for the endpoint group to delete.
2. Click **Delete** to stage the endpoint group removal and return to the list view.

4.4.3 Policies

4.4.3.1 Security Policies

Security Policy Overview

Security policies can be used on Layer 2 IPv4-enabled blueprints with Cisco NX-OS and Arista EOS physical network devices only. Ipv6-enabled blueprints cannot use security policies.

Apstra provides policy constructs to express security/segmentation intent with different levels of granularity. Granularity level is a function of reference design, but policy constructs are able to express a wide range of granularity levels.

With security policies you can simplify security management when using multiple vendors. You can also isolate virtual machines and control the allowed application flows between virtual networks.

To add granularity to your security policies you add endpoints. Endpoint connectivity is determined by reachability (the correct forwarding state in the network) and security (the policy must allow connectivity). Specifically, you must specify security policies between L2 and L3 domains (already modeled in Apstra) and between more granular L2/L3 IP endpoints.

You can specify a wide range of granularity levels.

Endpoints/groups that can be subject to security policies are as follows:

- Virtual networks (contain subnet)
- Internal IP endpoints associated with virtual networks (contain IP /32 address)
- Internal IP endpoint groups
- External IP endpoints (contain /32 or subnet)
- External IP endpoint groups
- Security zone (logical collection of all virtual networks and internal IP endpoints)

You can associate one or more security policy with endpoints/groups. You can indicate if you want to log actions taken by specific rules.

Policy controls traffic in and out of the endpoint/group to which it is applied, in this case, the subnet or IP endpoint. Policy is expressed via “policy” node type, which contains the description of the policy and its association with endpoints/groups using relationships “from” and “to”, indicating the directionality. For a bi-directional security policy, two instances of the policy must be created, one for each direction. A policy contains a list of rules. An actual rule consists of 5-tuple (Source IP, Destination IP, Source Port, Destination Port, Protocol) and an action (deny, deny+log, allow, allow+log, log_only). Each subnet/IP endpoint can have one or more security policies applied to it.

Source IP and Destination IP are derived from properties of endpoints/groups attached to the policy.

Actions that include “logging” merely indicate that ACL is configured to log matches using whatever mechanism is supported on the device. Parsing these logs is outside scope of this document.

Policies consist of a set of rules that are stateless, meaning responses to allowed inbound traffic are subject to the rules for outbound traffic (and vice versa).

Given that any endpoint can have more than one policy applied to it, it is important to deal with composition of these policies as ordering of rules has an impact on the behavior. In addition, there is implicit hierarchy between IPE, VN and SZ and policies applied at different levels of hierarchy also need to be considered from the composition perspective.

Composition (in this case ordering of rules) matters when applicable policies have “conflicting” rules which are defined as rules that have:

- Overlapping match sets (there is a packet whose 5-tuple matches both rules)

- Different action (for example “allow” vs “deny”)

Match sets are non-overlapping when the condition in at least one field in the 5-tuple is disjoint (non-overlapping) between the rules. Or to put it differently, match sets are overlapping when the conditions set in all the fields in the 5-tuple are overlapping.

When conflicting rules are present, there are two situations. First one is when one rule’s match set contains the other’s match set (full containment case). In this situation the ordering can be driven by a policy knob such as:

- More specific rule is executed first (“exception” focus/mode)
- Less specific executes first (“override” focus/mode)

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 less specific rule and more specific rule becomes what is called a “shadow” rule. Apstra alerts you at the time conflicting rules are detected and what is the resolution.

With its built-in hierarchy and the nature/semantics of endpoints/groups, Apstra helps to identify the presence of conflicting rules. A few cases are described below:

- Rules in policies between different pairs of IPEs (even if one IPE is common to both pairs) are non-overlapping given that the pairs of IP addresses are different. This will cause disjoint match set from the sIP/dIP perspective (different “IP signature”).
- Rules in policies between the same IPEs can overlap (such as Destination Port) fields and Apstra checks for this.
- Rules in policies between different pairs of VNs (even if one VN is common to both pairs) are non-overlapping given that the pairs of subnets are different. This will cause disjoint match set from the Source IP / Destination IP perspective (different “IP signature”).
- Rules in policies between the same VNs can overlap (such as Destination Port) fields and Apstra checks for this.
- When IPECG are used they are essentially resulting in a set of IPE pairs so the above discussion related to IPE pairs applies.
- Rules in policies between a pair of IPEs and pair of parent VNs have containment from IP signature perspective. Apstra analyzes Destination Port / Protocol overlap and classifies it as full-containment or non-full-containment conflict.
- Rules in policies between pair of IPEs and pair of VNs where at least one VN is not parent are non conflicting (different IP signature).
- Rules in policies between pair of IPEs and IPE-VN pair where VN is a parent have full containment from IP signature perspective so Apstra analyzes the remaining fields.
- Rules in policies that contain EIPE or EIPECG have to be analyzed from IP signature perspective as external points are not bound by any hierarchical assumptions.
- SZ is simply a set of VNs and IPE endpoints so the above discussions apply.

In order 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.

Apstra implements this feature by doing the following:

- Automated creation and deployment of ACLs to Leaf switches (enforcement points).
- Inter VLAN/VXLAN filtering.
- Automated rule ordering and conflict resolution.
- Config validation - useful for compliance.
- Adding a new VXLAN Endpoint (ex. Add rack or add leaf to VN) automatically places the ACL on the VN interface.

- Adding a new External Router External Connectivity Point (ECP) (enforcement point) automatically places ACL for external endpoint groups.

New in version 3.1: Apstra adds support for shadow relationship, user-defined conflict resolution, selective policy enablement, searching for networks impacted by policy, security policy search, per blueprint settings and views of policy changes between staged and active blueprints.

Note: Controls inter virtual network traffic (ACLs on SVIs) or external to internal traffic (ACLs in border leafs).

Note: For External Endpoints, ACL enforcement only on border leafs.

Screen Changes in Version 3.2

The Policies screen changed in version 3.2. Interface Policies was added as a new policy type, and Security Policy Search, Conflicts and Settings moved to a sidebar on the right.

Recommended Workflow

Several tasks, as listed below, must be completed before creating security policies. Access Control Lists (ACL) are rendered in the appropriate device syntax. ACLs will be applied on enforcement points.

1. Create *security zones*.
2. Create *virtual networks*.
3. Create *endpoints* (internal/external). (new in version 3.0)
4. Create *endpoint groups* (internal, external).
5. Create security policies (as described in the sections below).
6. *Commit* changes.

Policies

To go to security policies in the blueprint, navigate to **Staged > Policies > Security Policies > Policies**. (Image below is for version 3.1 which is slightly different from more recent versions.)

1. Click **Staged**

2. Click **Policies**

3. Click **Security Policies**

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"/> | | Copy Edit Delete |
| Permit External Clients | External Endpoint External Clients | Virtual Network red_125_leaf3 v4 | 1 | | | <input checked="" type="checkbox"/> | | Copy Edit Delete |

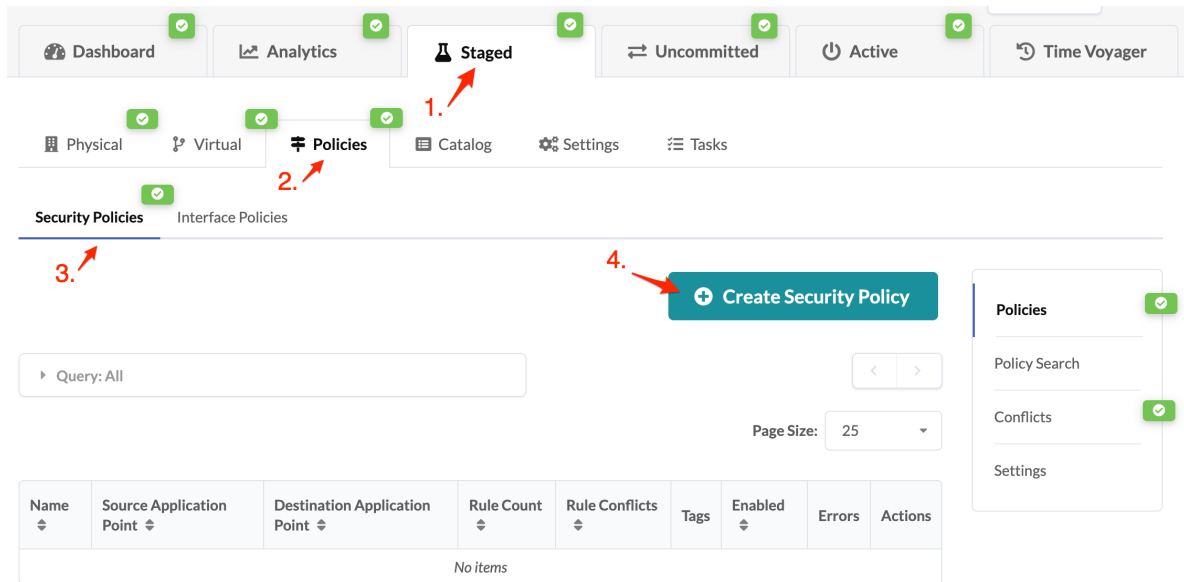
To see endpoints and subnets (if applicable), click on a **Source Application Point** or **Destination Application Point**.

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> |

Creating Security Policy

1. From the blueprint, navigate to **Staged > Policies > Security Policies > Policies** and click **Create Security Policy**.



2. Enter **Common Parameters**.

Name 32 characters or fewer, underscore, dash and alphanumeric characters only

Description optional

Enabled (as of version 3.1) **ON** to enable the policy (default)

OFF to disable the policy

Tags optional

3. Enter **Application Points**.

Source Point Type Internal Endpoint, External Endpoint, External Endpoint Group, Internal Endpoint Group, Virtual Network, Security Zone

Source Point Source point (that was previously created) from the drop-down list

Destination Point Type Internal Endpoint, External Endpoint, External Endpoint Group, Internal Endpoint Group, Virtual Network, Security Zone

Destination Point Destination (that was previously created) from the drop-down list

4. To create an allowlist-type policy in the **Rules** section, click **Permit All** to automatically create the rule.

5. To create a blocklist-type policy in the **Rules** section, click **Deny All** to automatically create the rule.

6. To manually create other rules, click **Add Rule** to open the dialog for adding a rule.

1. Enter a name (32 characters or fewer, underscore, dash and alphanumeric characters only) and (optional) description.

2. Select an action from the drop-down list: **Deny**, **Deny & Log**, **Permit** or **Permit & Log**

3. Select a protocol from the drop-down list: **TCP**, **UDP**, **IP**, or **ICMP**.

4. If you select **TCP** or **UDP**, fields appear for **Source port** and **Destination port**. You can enter specific numeric ports, and/or port-ranges (such as, 8000-8080,9000,9092). If you have previously created *TCP/UDP port aliases*, they will appear in the drop-down list for selection.

5. To add an additional rule, click **Add Rule** and enter rule details as above.

6. You can adjust the order of the rules by clicking the **Move up** or **Move Down** buttons in each rule.

- When you're finished adding and ordering the rules, click **Create** to stage the addition and return to the list view.

Note: Log configuration is local to the network device. It is on the device and not on the Apstra server.

Cloning Security Policy

Instead of entering all details for a new policy, you can clone an existing one, give it a unique name and customize it.

Errors

After creating a security policy, check the list view for errors. Errors are highlighted in red. (Image below is for version 3.1 which is slightly different from more recent versions.)

The screenshot shows the Apstra interface with the 'Staged' tab selected. The 'Security Policies' section is active, and a table lists a policy named 'External to Compute'. The 'Show errors' column for this policy has a red error icon, indicating an error. A tooltip 'Show errors' is visible over the icon.

| Name | Source Application Point | Destination Application Point | Rule Count | Rule Conflicts | Tags | Enabled | Show errors | Actions |
|---------------------|-------------------------------------|---|------------|----------------|------|---------|-------------|--------------------|
| External to Compute | External Endpoint Group External | Internal Endpoint Group Webservers and Databases | 2 | | | Enabled | Show errors | Copy, Edit, Delete |

Click the **Show errors** button to see details.

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

After you resolve any errors, the policy is no longer highlighted red and the **Errors** field is blank. (Image below is for version 3.1 which is slightly different from more recent versions.)

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"/> | | Edit Delete |

Editing Security Policy

1. Either from the list view (Staged > Policies > Security Policies > Policies) or the details view, click the **Edit** button for the policy to edit.
2. Make your changes.
3. Click **Edit** to stage the changes and return to the list view.

Deleting Security Policy

1. Either from the list view (Staged > Policies > Security Policies > Policies) or the details view, click the **Delete** button for the policy to delete.
2. Click **Delete** to stage the deletion and return to the list view.

Policy Search

You can search for security policies by their source points and destination points (as of version 3.1).

1. To search security policies from the blueprint, navigate to **Staged > Policies > Security Policies > Policy Search**.
2. Select a source point type and enter the exact IPv4 network subnet that was configured in the policy.
3. Select a destination point type and enter the exact IPv4 network subnet that was configured in the policy.
4. From the **Protocol** drop-down list, select a protocol (TCP, UDP, IP, ICMP).
5. Click **Search**.

Conflicts

Apstra detects and resolves security policy rule conflicts whenever possible. More specific policies are applied before less specific ones by default, but you can change this setting. (See the [Settings](#) section below).

Any conflicts appear on the security policy list view (Staged > Policies > Security Policies > Policies) in the **Rule Conflicts** column. (Image below is for version 3.1 which is slightly different from more recent versions.)

Navigation: Dashboard, Analytics, Staged, Uncommitted, Active, Physical, Virtual, Policies, Catalog, Settings, Security Policies, Security Policy Search, Conflicts, Settings.

Buttons: 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"/> | | |

To see details of a conflict, navigate to **Staged > Policies > Security Policies > Conflicts**. When Apstra automatically resolves a conflict, **Resolved by AOS** appears in the **Status** column. (Image below is for version 3.1 which is slightly different from more recent versions.)

Navigation: Dashboard, Analytics, Staged, Uncommitted, Active, Physical, Virtual, Policies, Catalog, Settings, Security Policies, Security Policy Search, Conflicts, Settings.

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

| Status | Policy / Rule #1 | Policy / Rule #2 |
|-----------------|---|---|
| Resolved by AOS | External to Compute / Permit HTTPS External Endpoint Group External any → Virtual Network red_125_leaf3_v4 443 | Permit External Clients / Deny External Endpoint External Clients any → Virtual Network red_125_leaf3_v4 any |

Settings

You can configure security policy settings in the following categories (as of version 3.1).

Conflicts 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 all traffic (default)

Permit & Log - permits and logs all traffic

Deny - denies all traffic

Deny & Log - denies and logs all traffic

Updating Security Policy Settings

1. From the blueprint, navigate to **Staged > Policies > Security Policies > Settings**.
2. Select one option in each category.
3. Click **Save Changes** to stage the changes.
4. When you are ready to activate the staged changes, *commit* them to the blueprint.

4.4.3.2 Interface Policies

802.1X Server Port Authentication

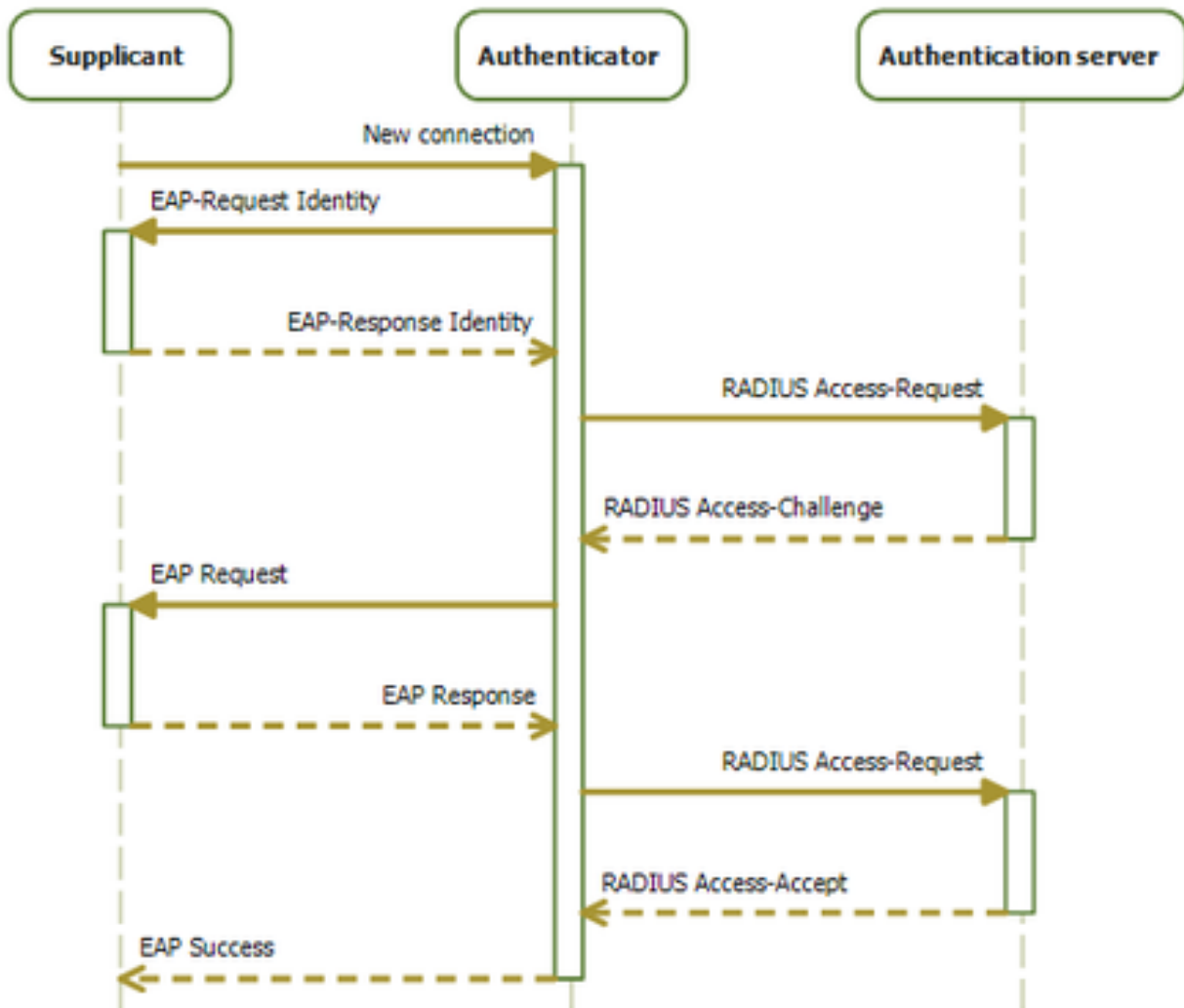
IEEE 802.1X is an IEEE Standard for network port-based Network Access Control. It is part of the IEEE 802.1 group of networking protocols. It provides an authentication mechanism to devices wishing to attach to a LAN.

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)

New in version 3.2: 802.1X server port authentication is introduced as a collection of interface policy settings. This feature allows AOS to manage 802.1X configuration on network devices.

Warning: As of AOS 3.3.0, 802.1X interface policy is only supported on Junos and Arista EOS physical network devices.

This policy setting enables the network to require L2 servers in an AOS 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 AOS 802.1X server port authentication and dynamic VLAN allocation.

Refer to *Interface Policies* for more details on 802.1X Server Port Authentication concept.

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

The following are the summary of 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. Edit 802.1X interface policy to assign ports and fallback VLANs

Creating Virtual Networks

From the Virtual view of your Staged Blueprint, click Virtual Network to see the list of existing virtual networks. Click Create Virtual Networks button to create virtual networks.

Data VLAN (assigned to ports) :

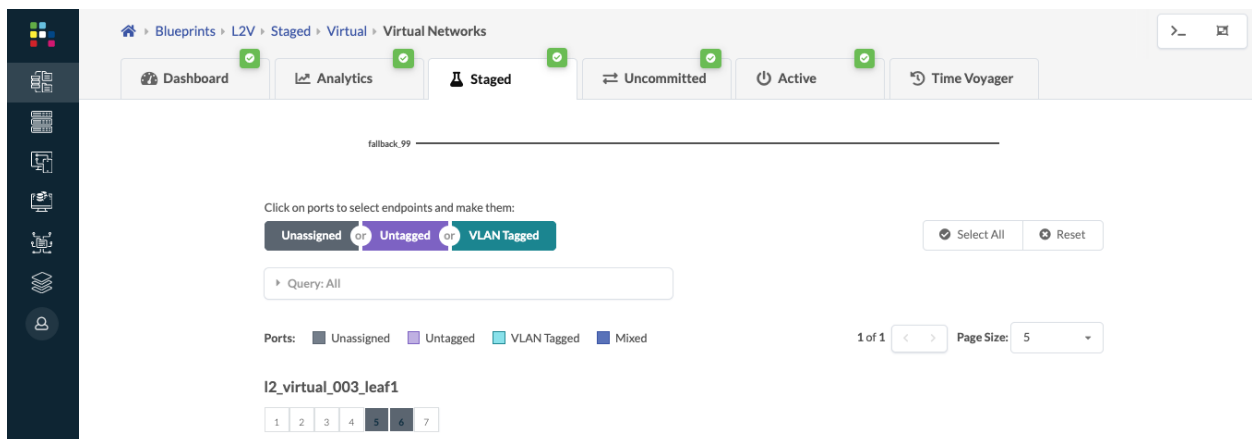
Interfaces will have 802.1X configuration if at least one VLAN is assigned to the port. If a port does not have any VLANs assigned, 802.1X configuration will not be rendered on the interface. The interface will be configured as a routed port.

Dynamic VLAN (optional, assigned to leafs, not ports) :

The RADIUS server itself optionally chooses the VLAN ID dynamically when the user (supplicant) is authenticated and authorized. AOS does not have control over Dynamic VLAN assignment. This decision is made by RADIUS configuration, not by the switch configuration.

Fallback VLAN (optional, assigned to leafs, not ports) :

Fallback VLAN can be assigned to the user (supplicant) in case of authentication failure. For fallback, the VLAN is controlled by the switch configuration.



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.

Note: Apstra suggests creating these virtual networks with a consistent VLAN ID amongst all leafs instead of using a resource pool.

Creating AAA Servers

Create the AAA server. See [Creating AAA Server](#) for more information.

Creating 802.1X Interface Policy

From the Policies view of your Staged Blueprint, click Interface Policies then click Create Interface Policy button.

| Label | 802.1x | | | | Actions |
|-------|---------------|------------|------------------|-----------------|-----------------|
| | Port Control | Host Mode | MAC Auth Bypass? | Re-auth Timeout | |
| dot1x | dot1x enabled | Multi-host | no | 30 | [Edit] [Delete] |

An interface policy name allows you to describe it to other administrators.

Note: Assigning interfaces or fallback VLANs to the policy is only possible after it is created.

Creating an interface policy will have a number of 802.1X options.

Port Control :

- Dot1x enabled - requires ports to authenticate EAPOL before given access to the network.
- Deny access - completely blocks the port, and no network access is permitted. An example use case of this is a quarantine configuration to quickly deactivate ports that may be infected.

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 deauthenticated.
- Single-host mode permits only a single host to authenticate and all other MACs are not permitted.

MAC Auth Bypass:

Optionally 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 alongside with 802.1X port control.

Warning: MAC auth bypass failure behavior may be different between switch vendors and major switch models.

Re-auth Timeout :

Optionally configured as a time period (seconds). Reauthentication 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.

Note: If reauthentication 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, AOS will enable 802.1X reauthentication on the port, and configure a time value.

Create Interface Policy

Common Parameters

Label *

Type

802.1x

802.1x

Port Control *

☒ dot1x enabled [?] ☐ Deny access [?]

Host Mode *

☒ Multi-host [?] ☐ Single-host [?]

MAC Auth Bypass *

☐ Enabled? (Note: Arista EOS only) [?]

Re-auth Timeout [?]

☐ Create Another? **Create**

Assigning Ports and Fallback VNs to the 802.1X Interface Policy

After the policy is created, you must add interfaces or dynamic VLANs to the current policy.

From the Policies view of your Staged Blueprint, click Interface Policies then scroll down to the Assign To section.

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.

Assign fallback VN :

Assigning the fallback virtual network is leaf-specific. To re-use the fallback on multiple leafs, you have to assign it to each leaf. Any VN that is assigned to the leaf may be used as a fallback virtual network - there are no restrictions.

Assigned To

Query: All
1-5 of 6
Page Size: 5

| Name | Hostname | S/N |
|---|---------------------------|--------------|
| l2_virtual_mlag_001_leaf1 | l2-virtual-mlag-001-leaf1 | 52540057E344 |
| <div> Ports: <input checked="" type="checkbox"/> Assigned to the current policy <input type="checkbox"/> Assigned to another policy <input type="checkbox"/> Unavailable for assignment <input type="checkbox"/> Not assigned to any policy <input type="checkbox"/> Partial </div> <div> Select port and choose interfaces you want to assign to the current policy Assign All Unassign </div> <div> 1 2 3 4 5 6 7 </div> <div> Port #5 Tr. #1 (10G, default) swp5 </div> <div> Fallback VN <div> fallback_99 x blue-2 fallback_99 red-1 </div> Save Change </div> | | |
| l2_virtual_mlag_001_leaf2 | l2-virtual-mlag-001-leaf2 | 5254005B9A65 |

Viewing the interface policy :

After the policy is configured, the settings are now visible, including interfaces those settings apply to.

Note: AAA, Dot1x, and Dot1x interface configurations are now pushed to the leafs. The following is a part of sample config rendered by AOS 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..
```

(continues on next page)

(continued from previous page)

```
!
dot1x system-auth-control
dot1x dynamic-authorization
```

4.4.4 Catalog

4.4.4.1 Logical Devices

1. Click to search

2. Click to sort

3. Export to global catalog

| Name | Capabilities | Panels Count | Ports Count | Ports Summary | Actions |
|------------|--------------|--------------|-------------|--|--------------------------|
| AOS-1x10-1 | 1 × 10 Gbps | 1 | 1 | AOS-1x10-1 1 x 10 Gbps Leaf • Access | Export to global catalog |

From the *blueprint*, navigate to **Staged / Catalog / Logical Devices**.

Logical devices in a blueprint catalog come from the template that was used to create the blueprint.

Exporting Logical Device

1. From the blueprint, navigate to **Staged / Catalog / Logical Devices**, then click the **Export to global catalog** button for the logical device to export.
2. Click **Export** to export the logical device and return to the list view.

4.4.4.2 Interface Maps

Click to search

Query: All

1-14 of 14

Page Size: 25

| Name | Device Profile | Logical Device | Actions |
|---------------------------|----------------|----------------|---------|
| Arista_vEOS_AOS-7x10-Leaf | Arista vEOS | AOS-7x10-Leaf | Delete |

Importing Interface Map

1. Make sure that the *interface map* to import has been created in the global catalog.
2. From the blueprint, navigate to **Staged / Catalog / Interface Maps**, then click **Import Interface Map**.
3. Select a logical device and an interface map from the drop-down lists to see a preview of your selection.
4. Click **Import Selected Interface Map** to import the interface map and return to the list view.

Deleting Interface Map from Blueprint Catalog

1. From the blueprint, navigate to **Staged / Catalog / Interface Maps**, then click the **Delete** button for the interface map to delete.
2. Click **Delete** to delete the interface map and return to the list view.

4.4.4.3 External Routers

The screenshot shows the Apstra interface with the following components and annotations:

- 1.** Points to the **Staged** tab in the top navigation bar.
- 2.** Points to the **Catalog** tab in the top navigation bar.
- 3.** Points to the **External Routers** sub-tab in the left sidebar.
- 4.** Points to the **Import External Router** button in the top right.
- Click to search**: Points to the search bar containing "Query: All".
- Click to sort**: Points to the column headers of the table: **Name**, **IPv4 Loopback Address**, **IPv6 Loopback Address**, and **ASN**.

| Name | IPv4 Loopback Address | IPv6 Loopback Address | ASN | Actions |
|-----------------|-----------------------|-----------------------|-------|---|
| example_router1 | 198.51.100.1 | Not assigned | 65534 | Edit Delete |

Importing External Router

1. Make sure that the *external router* to import has been created in the global catalog.
2. From the blueprint, navigate to **Staged / Catalog / External Routers**, then click **Import External Router**.
3. From the drop-down list, select an external router from the global catalog, then click **Import External Router** to import the router and return to the list view.

Editing External Router in Blueprint Catalog

Editing an external router in the blueprint does not affect the same-named one in the global catalog, just as editing the one in the global catalog does not affect the same-named one in the blueprint catalog.

1. From the blueprint, navigate to **Staged / Catalog / External Routers**, then click the **Edit** button for the external router to edit.
2. Make your changes, then click **Update** to change the router and return to the list view.

Deleting External Router from Blueprint Catalog

Deleting an external router in the blueprint catalog does not affect the same-named one in the global catalog, just as deleting the one in the global catalog does not affect the same-named one in the blueprint catalog.

1. From the blueprint, navigate to **Staged / Catalog / External Routers**, then click the **Delete** button for the external router to delete.
2. Click **Delete** to delete the external router and return to the list view.

4.4.4.4 Property Sets

Click to search

Click for details

| Name | Keys | Stale? | Actions |
|-------------|---------|----------------------|---------|
| my_prop_set | {{ntp}} | As in global catalog | Delete |

Importing Property Set

1. Make sure that the *property set* to import has been created in the global catalog.
2. From the blueprint, navigate to **Staged / Catalog / Property Sets**, then click **Import Property Set**.
3. From the drop-down list, select a property set from the global catalog, then click **Import Property Set** to import it and return to the list view.

Re-importing Property Set

If a property set that is used in a blueprint is updated in the global catalog, a message appears in the blueprint catalog stating that the property set in the blueprint catalog is **Different from global catalog**. You can re-import the property set from the global catalog by clicking the **Re-import** button, then clicking **Re-import Property Set** to stage the update.

Deleting Property Set from Blueprint Catalog

As long as a property set is not used in a configlet, it can be unassigned 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 would resolve that build error.

1. From the blueprint, navigate to **Staged / Catalog / Property Sets**, then click the **Delete** button for the property set to delete.
2. Click **Delete** to delete the property set and return to the list view.

4.4.4.5 Configlets

Click to search

Click for details

1. 2. 3. 4.

Query: All

1-2 of 2

Page Size: 25

| Name | Condition | Actions |
|--------------------------------|---|-------------|
| Cumulus Hostname | role in ["leaf", "spine", "superspine"] | Edit Delete |
| Cumulus LLDP Hostname override | role in ["leaf", "spine", "superspine"] | Edit Delete |

Importing Configlet

1. Make sure that the *configlet* to import has been created in the global catalog.
2. From the blueprint, navigate to **Staged / Catalog / Configlets**, then click **Import Configlet**.
3. From the drop-down list, select a configlet from the global catalog.
4. For OSPF configlets only - select security zone VRF names. If no VRFs are checked, the OSPF configlet will apply to all VRFs.
5. Enter/select details to specify the configlet scope: roles and/or individual nodes. (The individual nodes selection depends on the role and predicate settings.) As of AOS version 3.2.1 you can filter for specific individual nodes using regex (such as l2_virtual_ext_00[1-2]).
6. Importing an interface configlet includes additional options for defining the interface scope.
7. Click **Import Configlet** to stage the configlet and return to the list view.

Editing Configlet Scope in Blueprint Catalog

To change the roles and/or nodes that a configlet applies to directly in the blueprint catalog, follow the steps below:

1. From the blueprint, navigate to **Staged / Catalog / Configlets**, then click the **Edit** button for the configlet to edit.
2. Make your changes to the configlet scope.
3. Click **Update** to stage the update and return to the list view.

Note: To change configlet generators (template text, negation template text, filename, as applicable) you must change them in the global catalog, then re-import the configlet into the blueprint catalog. See [Editing Configlet Generators in Blueprint Catalog](#) for details.

Editing Configlet Generators in Blueprint Catalog

Configlet generators (template text, negation template text, filename, as applicable) cannot be changed directly in blueprints. If an existing configlet is no longer relevant, you can delete it and import a new or revised one. If you are changing a configlet in a blueprint catalog because of a configuration deviation, see also [Configuration Deviation and Configlets](#).

1. [Edit](#) or [create](#) a configlet in the global catalog.
2. [Delete the configlet](#) from the blueprint catalog.
3. [Import the configlet](#) from the global catalog into the blueprint catalog.
4. [Commit](#) the changes.

Deleting Configlet from Blueprint Catalog

1. From the blueprint, navigate to **Staged / Catalog / Configlets** and click the **Delete** button for the configlet to delete.
2. Click **Delete** to stage the deletion and return to the list view. AOS will remove the configlet from all devices that are within its scope.

4.4.4.6 AAA Servers

Click to search

Query: All

1-1 of 1

Page Size: 25

Click to sort

| Label | Server Type | Hostname | Auth Port | Accounting Port | Actions |
|------------|---------------|--------------|-----------|-----------------|---|
| freeradius | RADIUS 802.1x | 172.20.191.5 | 1812 | 1813 | Edit Delete |

AAA servers are used when configuring [interface policies](#).

To access AAA servers - from the blueprint, navigate to **Staged / Catalog / AAA servers**. AAA servers include the following details:

Label To identify the AAA server

Server Type

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 dynamic VLAN assignment after authentication instead of during auth. All RADIUS COA implementations are hard-coded to auth port 3799.

Hostname

Auth Ports

Accounting Port Optional

AAA RADIUS Server configuration Tasks

AAA RADIUS server configuration tasks are external to AOS. For example, the following files are to be configured in the case of *FreeRADIUS*.

/etc/freeradius/clients.conf – has credentials for each switch

```
client Arista-7280SR-48C6-1 {
    shortname = Arista-7280SR-48C6-1
    ipaddr    = 172.20.191.10
    secret    = testing123
    nastype   = other
}
```

/etc/freeradius/users – has users and MAC addresses to authenticate. Tunnel-Private-Group-Id shows a dynamic VLAN ID, which is optional.

```
leaf1-server1 ClearText-Password := "password"

"52:54:00:37:d5:e1" Cleartext-Password := "52:54:00:37:d5:e1"
    Tunnel-Type = VLAN,
    Tunnel-Medium-Type = IEEE-802,
    Tunnel-Private-Group-Id = "50"
```

Although this example shows a simple credential, actual implementations may use any EAP method that both the client and RADIUS server support.

Client Supplicant Configuration Tasks

Client supplicant configuration tasks are external to AOS. 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
```

(continues on next page)

(continued from previous page)

```
# 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
}
```

Creating AAA Server

1. From the blueprint, navigate to **Staged / Catalog / AAA Servers**, then click **Create AAA Server**.
2. Enter a label, select the server type (RADIUS 802.1x, RADIUS COA), enter a hostname, key, auth port, and (optional) accounting port.
3. Click **Create** to stage the server and return to the list view.

Editing AAA Server

1. From the blueprint, navigate to **Staged / Catalog / AAA Servers**, then click the **Edit** button for the AAA server to edit.
2. Make your changes, then click **Update** to stage the update and return to the list view.

Deleting AAA Server

1. From the blueprint, navigate to **Staged / Catalog / AAA Servers** and click the **Delete** button for the AAA server to delete.
2. Click **Delete** to stage the deletion and return to the list view.

4.4.5 Settings

4.4.5.1 Fabric Addressing Policy

Enabling IPv6 Applications

Enabling support for IPv6 virtual networks and IPv6 external connectivity points on EVPN L2 deployments or L3 deployments adds resource requirements and device configurations, including IPv6 loopback addresses on leafs and spines, IPv6 addresses for MLAG SVI subnets and IPv6 addresses for leaf L3 peer links. This feature (New in version 3.0) does not include IPv6 support in the fabric.

Caveats:

- Security policy functionality is not available on EVPN L2 deployments when IPv6 is enabled.
- IPv6 support is not available on non-EVPN L2 networks.

- IPv6 support is not available on 5-stage Clos networks.

Important: After IPv6 support is enabled, it cannot be disabled in the current blueprint revision. One way to disable it is to use the Time Voyager to rollback to a revision before IPv6 was enabled.

1. From the blueprint, navigate to **Staged > Settings > Fabric Addressing Policy** and click **Modify Settings**.
2. Click the toggle on to enable IPv6 applications.
3. Click **Save Changes** to enable IPv6 support.
4. See [Staging Resources](#) for information about assigning the IPv6 IPs.

Documentation for IPv6 configuration for Virtual Networks can be found in [Virtual Networks](#).

Documentation for IPv6 configuration for External Connectivity Points can be found in [Adding Connectivity Point to Security Zone](#).

ESI MAC MSB

Warning: Updating the Most Significant Byte (MSB) value results in regeneration of all existing ESI MACs in the blueprint. Hence it is recommended to leave the default value.

ESI MAC addresses are internally auto-generated using Most Significant Byte (MSB) values (default: 2). This must be an even number (up to 254) to ensure that multicast MACs are not generated. The config for ESI is rendered as 10 octets (first octet being 0) value. The second octet in the ESI value is derived from the MSB value. Here 6 octets after the first octet of 0 are significant and used as LACP system-id. Please find below an example 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
```

To change the ESI MAC MSB value from the web interface:

1. From the blueprint, navigate to **Staged > Settings > Fabric Addressing Policy** and click **Modify Settings**.
2. Make your changes and click **Save Changes**.

4.4.5.2 L3 Edge IP Connectivity

This setting applies only to blueprints with L3 servers.

From the blueprint, navigate to **Staged > Settings > L3 Edge IP Connectivity**.

4.4.5.3 Virtual Network Policy

Virtual Network Policy include the following details:

External Router MTU Specify fabric-wide external router interface IP MTU. Larger MTU may be required to provide EVPN DCI functionality or to support fabric-wide Jumbo frame functionality. We recommend an MTU of 9050 for EVPN-DCI. A null or empty (default) value implies that a user-overridden MTU will not be rendered.

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 only 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 only 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. For EVPN blueprints, this should be combined with `max_evpn_routes` to permit routes across the I3 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 routes to accept between spine and leaf in the fabric, and spine-superspine. This includes the default VRF. Setting this option may be required in the event of leaking EVPN routes from a security zone into the default security 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 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 fabric routes (effectively unlimited).

EVPN Type 5 Routes Default disabled. When enabled all EVPN vteps in the fabric redistribute ARP/IPV6 ND (when possible on NOS type) as EVPN type 5 /32 routes in the routing table. Currently, this option is only certified for Juniper Junos. FRR (SONiC/Cumulus) does this implicitly and cannot be disabled. This setting will be ignored. On Arista and Cisco, no configuration is rendered and will result in a blueprint warning that it is not supported. This value is disabled by default, as it generates a very large number of routes in the BGP routing table and takes large amounts of TCAM allocation space. When these /32 and /128 routes are generated, it assists in direct unicast routing to host destinations on VNIs that are not stretched to the ingress vtep, and avoids a route lookup to a subnet (such as /24) that may be hosted on many leafs. The directed host route prevents a double lookup to one of many vteps may hosts the /24 and instead routes the destination directly to the correct vtep.

Generate EVPN host routes from ARP/IPV6 ND ARP Setting “Generate EVPN host routes from ARP/IPV6 ND ARP” adds a policy-statement to the export policy used within the fabric.

Physical Virtual Policies Catalog Settings Tasks

Fabric Addressing Policy L3 Edge IP Connectivity **Virtual Network Policy**

 Modify Settings

| | |
|--|-----------|
| External router 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 |

Modifying Virtual Network Policy

1. From the blueprint, navigate to **Staged > Settings > Virtual Network Policy** and click **Modify Settings**.
2. Make your changes.
3. Click **Save Changes** to change the settings.

4.4.6 Tasks

Blueprint task history is available on the **Tasks** tab of the **Staged** Blueprint (as of version 3.2). 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.

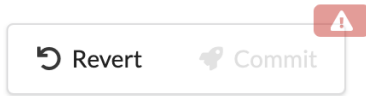
4.5 Uncommitted

4.5.1 Uncommitted Overview

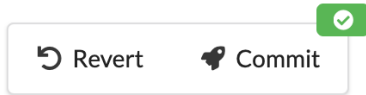
When a blueprint has pending changes, the **Uncommitted** tab shows a yellow status indicator, or if there are build errors, a red status indicator. From **Uncommitted**, you can review **Logical Diff**, **Full Nodes Diff**, **Build Errors**, and **Warnings**.

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 node requires a fair amount of resources and time to generate.

If any build errors exist, they must be resolved before you can commit changes. When they have been resolved, the status indicator on the **Build Errors** tab changes from red to green, and the **Commit** button turns from gray to black.



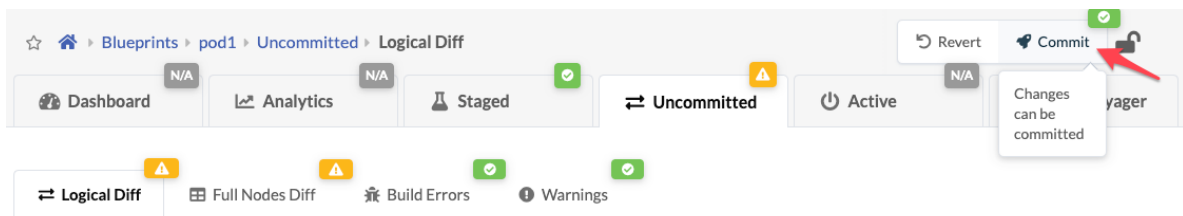
becomes...



4.5.2 Committing Staged Changes

Any build errors must be resolved before changes to a blueprint can be committed.

1. From the blueprint, navigate to **Uncommitted** and review changes as needed.
2. Click **Commit** to go to the dialog where you can add a description and commit changes.



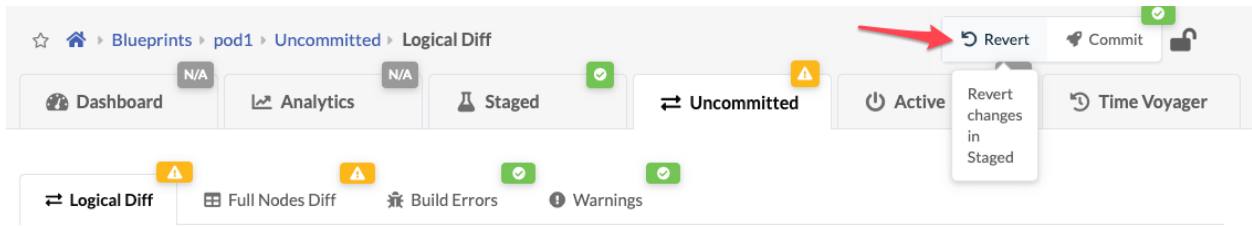
3. We recommend that you enter a revision description to identify the changes. These descriptions are displayed in the **Revisions** section of *Time Voyager*. If you don't add a description now you can always add one later. If you need to roll back to a previous revision, this description helps to determine the appropriate revision. Currently, specific diffs between revisions are not displayed, so the description is the only change information available for that revision.
4. Click **Commit** to push the staged changes to the active blueprint and create a revision.
5. 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 AOS's approach of *the network as a single entity* this is extremely easy; AOS generates all required device configurations and pushes them to the devices when the change is committed. We call this *Flexible Fabric Expansion* (FFE).

4.5.3 Reverting Staged Changes

If you decide not to commit staged changes to a blueprint, you can discard them.

From the blueprint, navigate to **Uncommitted**, then click **Revert**. In some cases, you might also need to [Reset Resource Group Overrides](#).



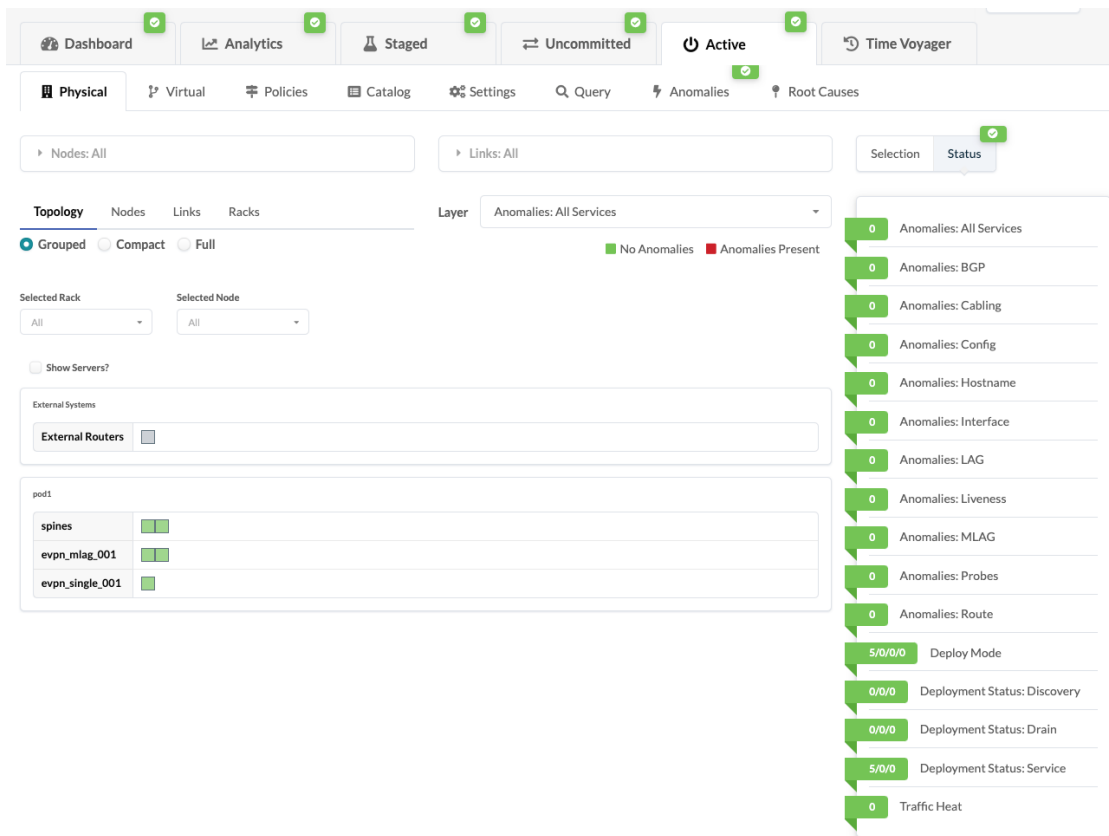
4.6 Active

The details of a deployed network are shown in the active view. From here, you can view telemetry status in a topology-centric view. Alerts and anomalies can be filtered by different layers to conduct root cause analysis of any problems.

4.6.1 Physical

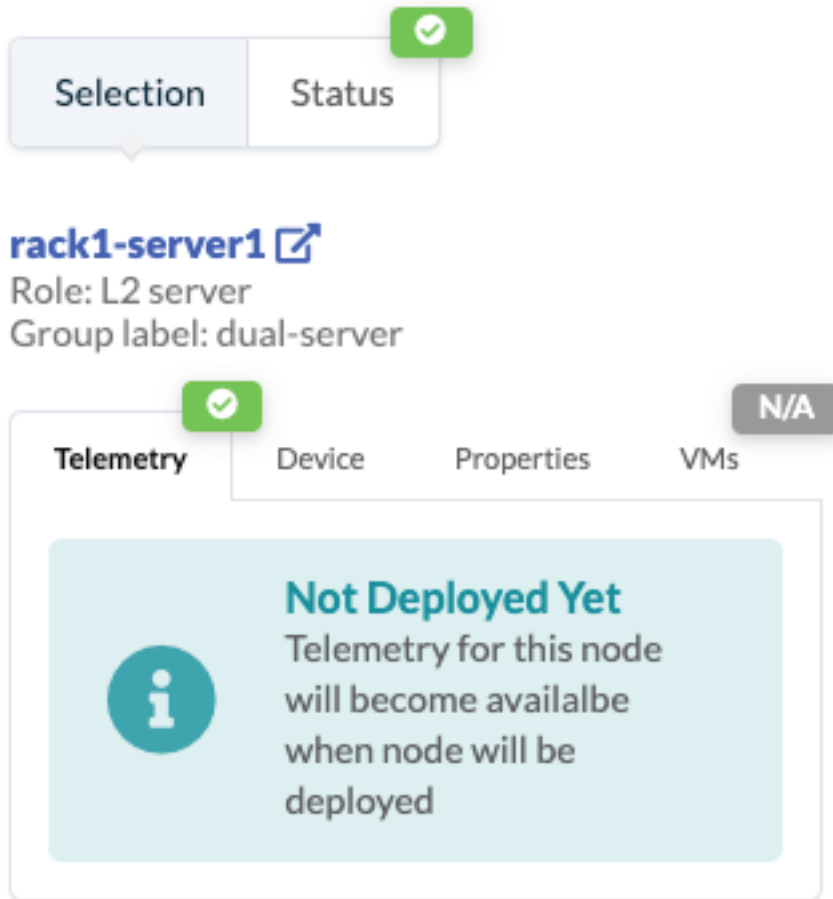
4.6.1.1 Status

The physical view of the active blueprint shows the statuses for services and deploy modes, deployment statuses for discovery, drain (as of AOS version 3.3.0) and service, as well as traffic heat.



4.6.1.2 Selection

Selecting an element provides information about the element. From the selection you can see information about telemetry, device, properties, and if a hypervisor is hosted on the node, VMs.



4.6.1.3 Topology

To access the active topology, from the blueprint, navigate to **Active / Physical / Topology**.

Topology Views

Topologies can be viewed at three levels of detail: grouped, compact, and full. (Grouped and compact are new in AOS version 3.3.0.) Green elements in the topology indicate no anomalies. Red elements indicate anomalies, which must be resolved before committing changes.

Grouped

- To show servers, click the **Show Servers** check box.
- To show node name (and hostname as applicable) hover over an element (square).

- To show node details, select the node by either clicking its element or by selecting it from the **Selected Node** drop-down list.

Dashboard Analytics Staged Uncommitted Active

Physical Virtual Policies Catalog Settings Query Anomalies Root

Nodes: All Links: All

Topology Nodes Links Racks Layer Anomalies: All Services

Grouped Compact Full No Anomalies Anomalies Present

Selected Rack: All Selected Node: All

Show Servers?

External Systems

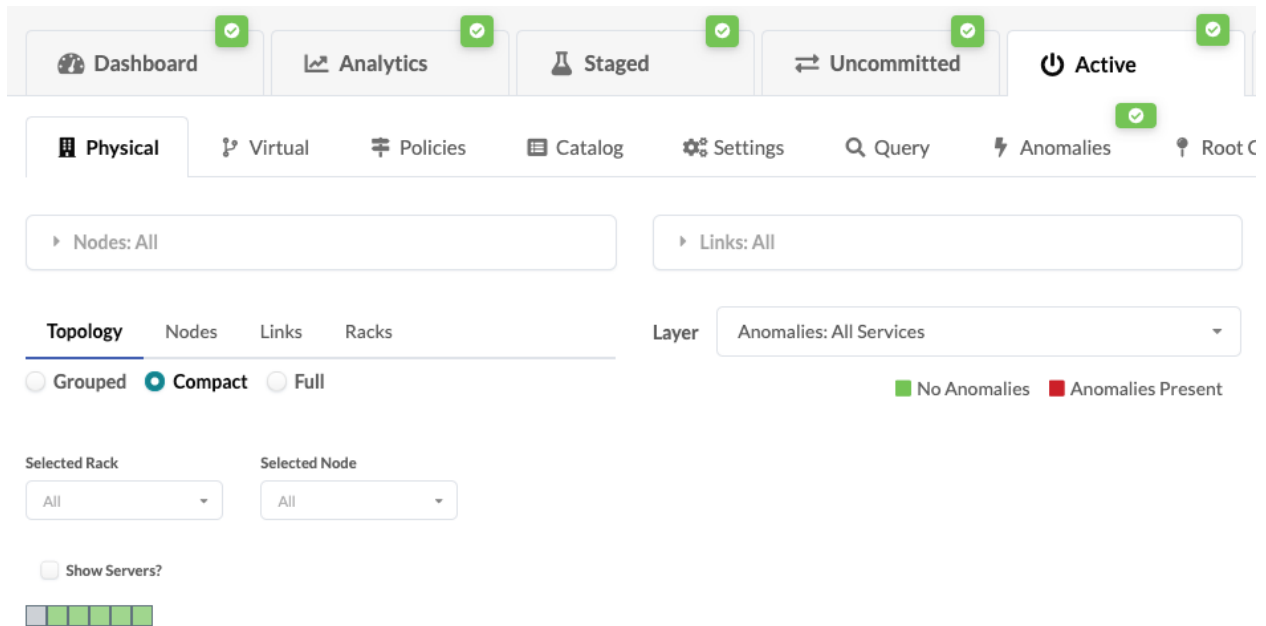
| External Routers | |
|------------------|--|
| | |

pod1

| | |
|-----------------|--|
| spines | |
| evpn_mlag_001 | |
| evpn_single_001 | |

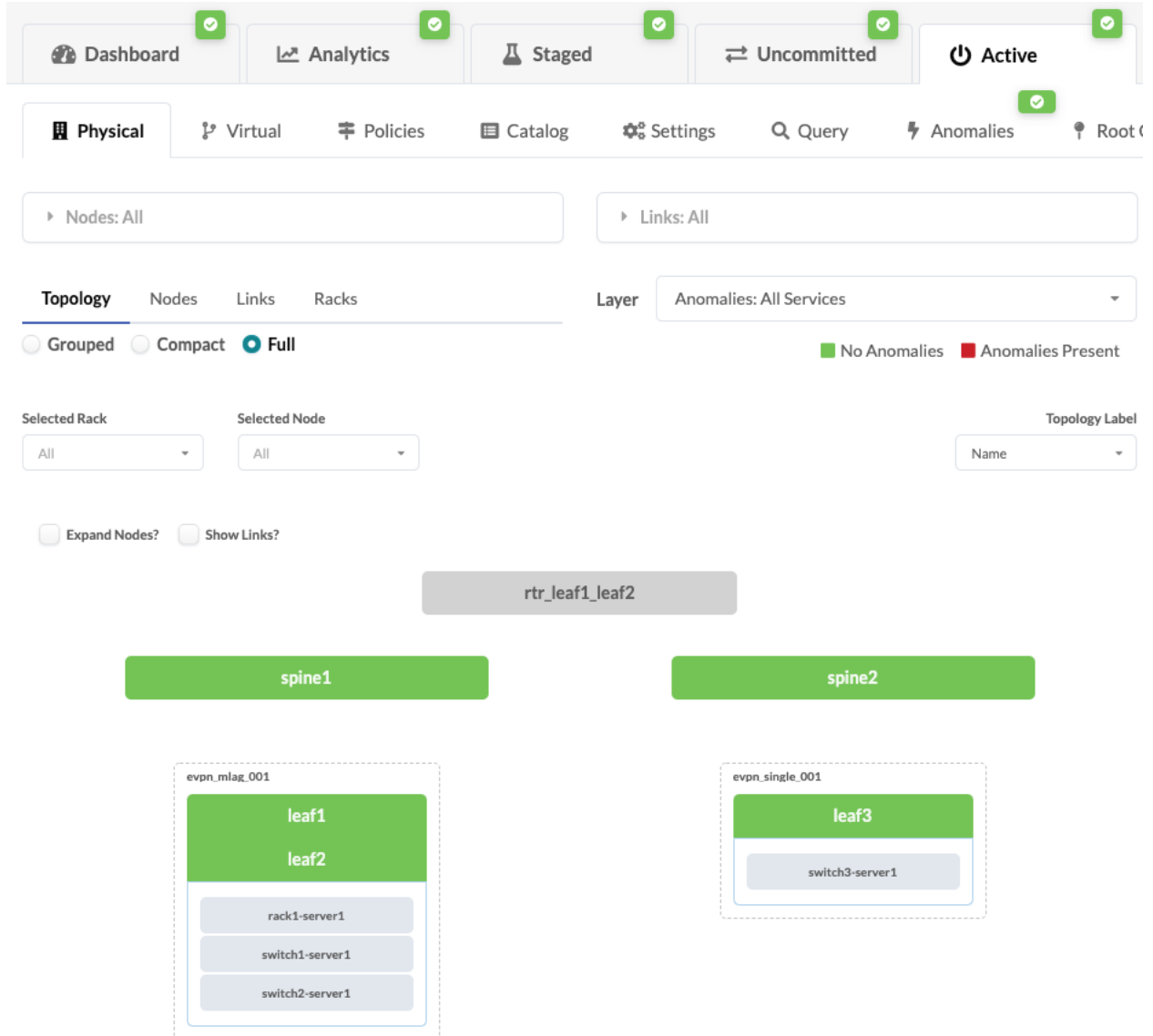
Compact

- To show servers, click the **Show Servers** check box.
- To show node name (and hostname as applicable) hover over an element (square).
- To show node details, select the node by either clicking its element or by selecting it from the **Selected Node** drop-down list.



Full

- 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, and hostname as applicable, hover over an element.
- To change the labels (name, hostname, S/N) that are shown in the topology, select a different label from the **Topology Label** 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.



Topology Selection Views

When a node is selected from the topology view (by clicking its element in the topology, or by selecting it from the **Selected Nodes** drop-down list), the node topology is shown. The selected node can be viewed in three different ways: neighbors view, links view, or (new in AOS version 3.3.0) headroom view. Telemetry and other device properties are shown in the *selection panel* on the right.

Neighbors View

- Aggregated server-leaf links are encircled (new in AOS version 3.3.0).
- To show unused ports, click the **Show Unused Ports** check box (new in AOS version 3.3.0).
- To change the labels (name, hostname, S/N) that are shown in the topology, select a different label from the **Topology Label** drop-down list.

- To change the layer that is shown in the topology, select a different layer (intent, traffic heat) from the **Layer** drop-down list.
- Choose to show all neighbors or only specific ones.

The screenshot displays the Apstra web interface's topology view. At the top, a navigation bar contains tabs for Dashboard, Analytics, Staged, Uncommitted, and Active, each with a green checkmark. Below this is a secondary navigation bar with Physical, Virtual, Policies, Catalog, Settings, Query, Anomalies, and Root. The main content area features filters for Nodes (All) and Links (All). View options include Topology, Nodes, Links, and Racks. Display modes are Grouped, Compact, and Full. The Selected Rack is evpn_mlag_001 and the Selected Node is rack1-server1 (L2 server). View buttons for Neighbors View, Links View, and Headroom View are present. A Layer dropdown is set to Intent. A legend for Color (ok, violating intent, unintended), Link (present, absent), and Interface (up, down) is shown. The topology diagram displays rack1-server1 connected to leaf1 and leaf2 via swp4 and swp2 interfaces. A 'Show All Neighbors' dropdown is visible.

Links View

Dashboard
Analytics
Staged
Uncommitted
Active

Physical
Virtual
Policies
Catalog
Settings
Query
Anomalies
Root C

Nodes: All
Links: Name =~ "leaf1"

Topology
Nodes
Links
Racks

Grouped
Compact
Full

Selected Rack
evpn_single_001

Selected Node
leaf3 (Leaf)

Topology Label
Name

Neighbors View
Links View
Headroom View

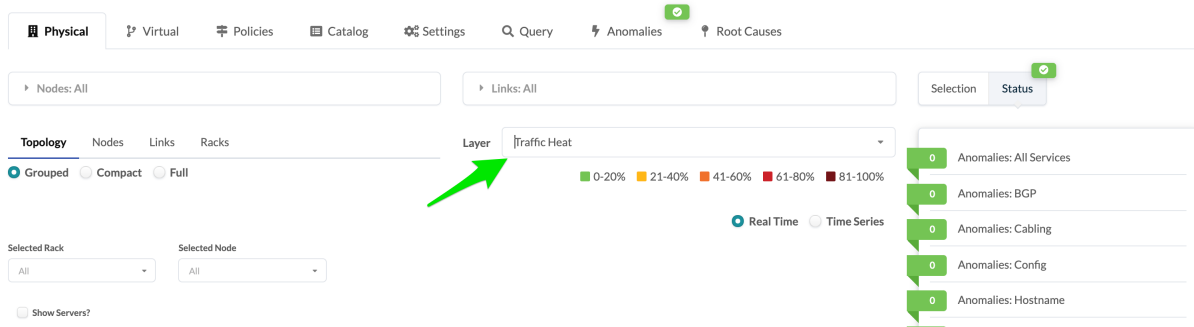
Query: All
1-3 of 3
Page Size: 25

| Name | Role | Speed | Logical Link | Port Channel ID | Endpoint 1 | | | | Endpoint 2 | | | |
|---|-------------------|-------|--------------|-----------------|------------|------|-----------|----------|-----------------|-----------|-----------|----------|
| | | | | | Name | Role | Interface | Lag Mode | Name | Role | Interface | Lag Mode |
| evpn_single_001_leaf1<->evpn_single_001_server001(single-link)[1] | Leaf to L2 Server | 10G | single-link | N/A | leaf3 | Leaf | swp3 | No LAG | switch3-server1 | L2 server | n/a | No LAG |
| spine1<->evpn_single_001_leaf1[1] | Spine to Leaf | 10G | N/A | N/A | leaf3 | Leaf | swp1 | N/A | spine1 | Spine | swp1 | N/A |
| spine2<->evpn_single_001_leaf1[1] | Spine to Leaf | 10G | N/A | N/A | leaf3 | Leaf | swp2 | N/A | spine2 | Spine | swp1 | N/A |

Headroom View

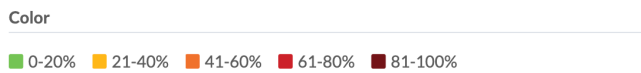
New in version 3.3.0: Headroom View is available in the AOS Blueprint Active Topology view.

To view the Headroom View in the AOS Blueprint Active Topology, go to Active → Physical → Topology, select the layer **Traffic Heat** to view Traffic Heat view on top of Physical topology.



Note: **Device Traffic probe** must be enabled for Headroom View layer. If user disables or deletes the probe, the Layer Traffic Heat” under the Active topology will disappear. See [Device Traffic Probe](#) for more info.

The view uses different colors to describe the capacity, where different color means different available/used capacity.



The node colors are based on the current System level TX/RX, averaged to 2min by default. If the aggregated TX or RX across all the device interfaces is < 20%: color **Green** . If between 21-40%, color **Yellow** etc. For each 20% difference, capacity is shown with a different color.

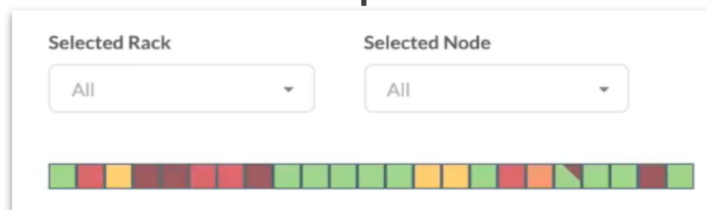
Note: The server coloring is calculated based on the interface counters of the leaf ports facing that server.

Traffic Heat view is available at all three levels of detail: grouped, compact, and full. See [Topology Views](#) to know more about levels.

Grouped



Compact



If any of a device deployed ports is > 81% of its capacity in either RX or TX, a new “Alert” icon is shown on the device.



Mousing over the node will show the exact aggregated values.

Selected Rack

All

Selected Node

All

☐ Expand Nodes?

☐ Show Lin

spine1

Hostname: spine1

Total TX: 5.95 Tbps (29%)

Total RX: 5.95 Tbps (29%)

Max Link TX: 29%

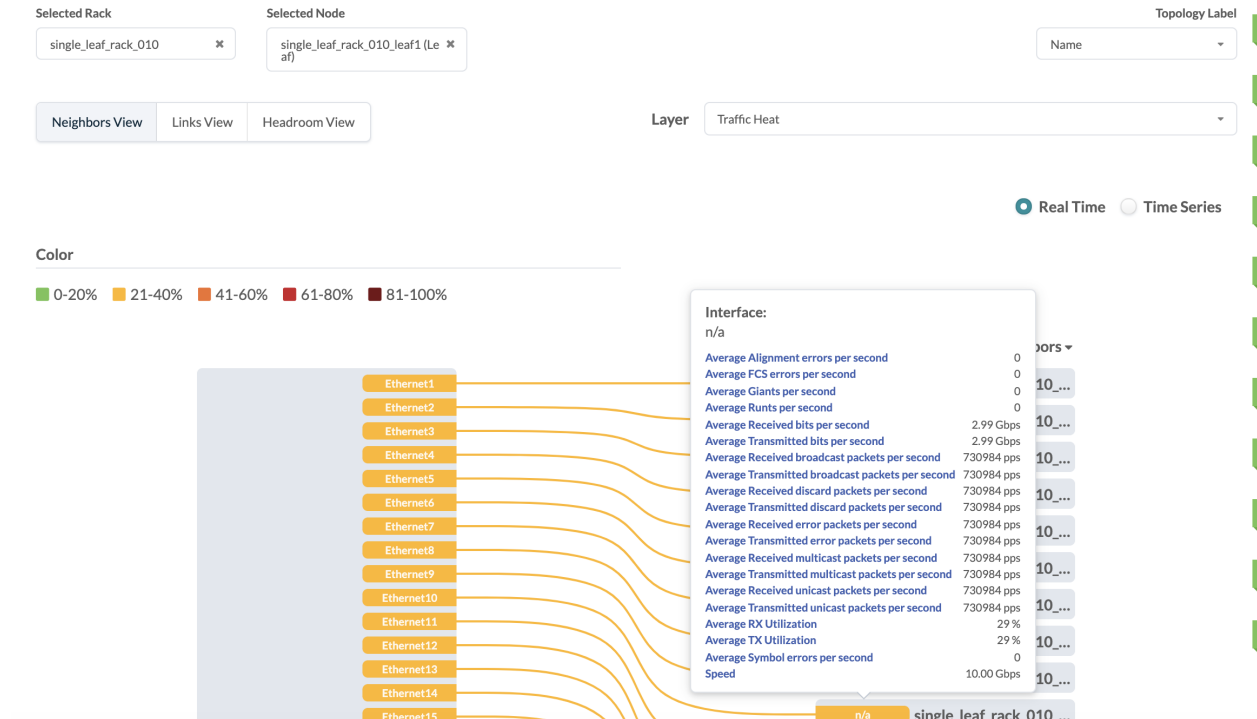
Max Link RX: 29%

ext_router_63cb5943

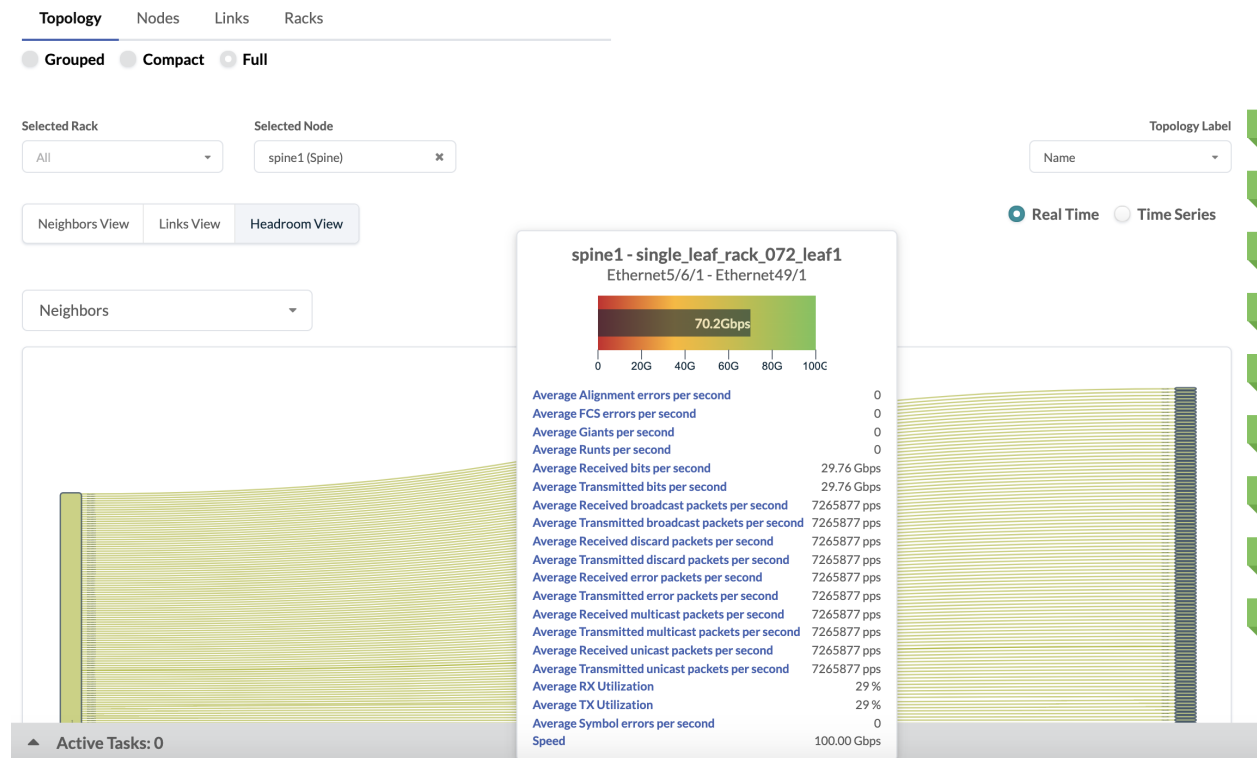
spine1

Note: On the grouped/compact view, we show this alert by coloring the right top corner of the square.

To view RX/TX per interface for a single node, click on the node.



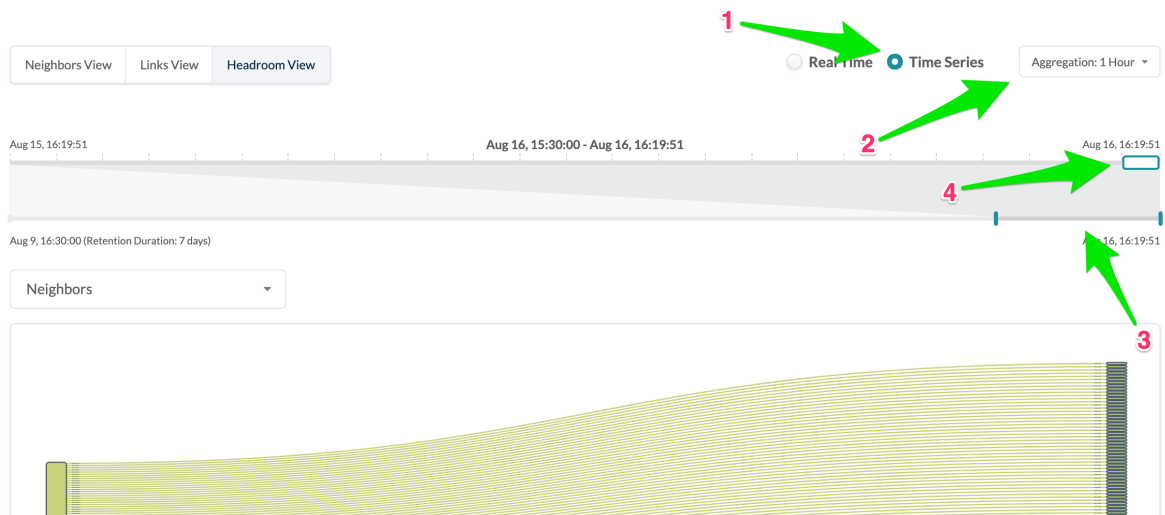
To view Headroom View layer, click on the node and select **Headroom View**. Hovering over to a link will display available Headroom capacity for that link.



Traffic History

To view traffic history on top of physical topology.

1. Select **Time Series**.
2. Select **Aggregation**: Users can select 2, 5, 10, 20 or 30 minutes to check traffic at specific time in the past but can also select larger aggregation such as 1 hour or 1 day to see traffic average across longer period of time.
3. Use Bottom Slider to select a past period of time to look at. The bottom left date+time is how far back user can go, based on data retention time. The bottom right date+time is when the user open the time series.
4. Use top slider to navigate across time in the selected period.
5. See Node Colors changing based on System level Traffic Heat.



4.6.1.4 Nodes

1. Active

2. Physical

3. Nodes

To see more details, select a name...

... to see telemetry, device, and properties info

1-25 of 1833

Page Size: 25

| Name | Role | Group Label | Deploy Mode | Device Profile | S/N | Hostname | ASN | Loopback IPv4 | Loopback IPv6 | Deploy Status | Hypervisor | VMs Count |
|---------------|------|-------------|-------------|-----------------------|---------------|---------------|-------|----------------|---------------|---------------------------------------|------------|-----------|
| leaf001_001_1 | Leaf | leaf | Deploy | Juniper_QFX1000 2-72Q | 0E229CC30004 | leaf001-001-1 | 64609 | 172.17.0.16/32 | | Service Config config apply succeeded | N/A | N/A |
| leaf001_001_2 | Leaf | leaf | Deploy | Juniper_QFX1000 2-72Q | 0E229CC30000 | leaf001-001-2 | 64610 | 172.17.0.17/32 | | Service Config config apply succeeded | N/A | N/A |
| leaf002_001_1 | Leaf | leaf-40g-up | Deploy | Juniper_QFX5110 -48S | 0E229CC3000A | leaf002-001-1 | 64611 | 172.17.0.18/32 | | Service Config config apply succeeded | N/A | N/A |
| leaf002_001_2 | Leaf | leaf-40g-up | Deploy | Juniper_QFX5110 -48S | 0EECB8B06000A | leaf002-001-2 | 64612 | 172.17.0.19/32 | | Service Config config apply succeeded | N/A | N/A |
| leaf002_001_3 | Leaf | leaf-10g-up | Deploy | Juniper_QFX5100 -48S | 0EEE6CBD0008 | leaf002-001-3 | 64613 | 172.17.0.20/32 | | Service Config config apply succeeded | N/A | N/A |

leaf001_001_1

Role: Leaf
Group label: leaf

Telemetry Device Properties

- Probes
- All Services
- Liveness
- Config
- Interface
- Cabling
- BGP
- LAG
- MLAG
- Route
- Hostname

From the blueprint, navigate to **Active / Physical / Nodes** to see the nodes in the active topology, either in table view or card view. You can search for specific nodes or links and see more details about a specific node by clicking its name.

Applying Full Config

See [Configuration Deviation](#) for more information about when to apply a full config.

Warning: Applying a full config is a disruptive operation and will result in a temporary loss of service to the device.

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**.

4.6.1.5 Links

To access active links, from the blueprint, navigate to **Active / Physical / Links**. To search for specific nodes or links, click its query box, enter your criteria and click **Apply** to see results. To see properties of a particular link (in the right panel), click its name.

1. Active

2. Physical

3. Links

Search

Links: All

Selection Status

Topology Nodes **Links** Racks Pods

Layer Anomalies: All Services

1-25 of 4286

Page Size: 25

Export cabling map

Click a name... to see properties

| Name | Role | Speed | Logical Link | Port Channel ID | Endpoint 1 | | | | Endpoint 2 | | | | | |
|--|-------------------|-------|--------------|-----------------|---------------|------|-----------|------|------------|-------------------|-----------|--------------|------|------|
| | | | | | Name | Role | Interface | IPv4 | IPv6 | Name | Role | Interface | IPv4 | IPv6 |
| leaf001_001_1<->server001_001(link) [1] | Leaf to L2 Server | 40G | link | N/A | leaf001_001_1 | Leaf | et-0/0/4 | N/A | N/A | server001_001_001 | L2 server | Not assigned | N/A | N/A |
| leaf001_001_1<->server001_002(link) [1] | Leaf to L2 Server | 40G | link | N/A | leaf001_001_1 | Leaf | et-0/0/5 | N/A | N/A | server001_001_002 | L2 server | Not assigned | N/A | N/A |

Properties

leaf001_001_1<->server001_001
Role: Leaf to L2 Server
Logical link: link

Exporting Cabling Map

1. From the links view (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.

Cabling maps can also be exported from the **Staged / Physical / Links** view.

4.6.1.6 Racks

The screenshot shows the Apstra Active / Physical / Racks view. Red arrows and numbers indicate the following steps:

- 1. Click the **Active** tab in the top navigation bar.
- 2. Click the **Physical** tab in the left sidebar.
- 3. Click the **Racks** tab in the left sidebar.
- 4. Click the **Name** column header in the Racks table to sort.
- 5. Click the **Edit** button (pencil icon) in the **Rack Properties** panel to change the rack name.

The Racks table displays the following data:

| Name | Rack Type | Connectivity Type | IP Version | Leaf Count | External Router Links Count | Servers Capacity | | | | | | | | | | | | |
|-----------------|-------------|-------------------|------------|---------------|-----------------------------|---|------|------|-----------|---------------|---|---|-----------------|---|---|-----------------|---|---|
| evpn_mlag_001 | evpn-mlag | L2 | IPv4 | 1 MLAG pair | 2 | <table border="1"> <thead> <tr> <th>Name</th> <th>Used</th> <th>Available</th> </tr> </thead> <tbody> <tr> <td>dual-server</td> <td>1</td> <td>0</td> </tr> <tr> <td>single-server-1</td> <td>1</td> <td>0</td> </tr> <tr> <td>single-server-2</td> <td>1</td> <td>0</td> </tr> </tbody> </table> | Name | Used | Available | dual-server | 1 | 0 | single-server-1 | 1 | 0 | single-server-2 | 1 | 0 |
| Name | Used | Available | | | | | | | | | | | | | | | | |
| dual-server | 1 | 0 | | | | | | | | | | | | | | | | |
| single-server-1 | 1 | 0 | | | | | | | | | | | | | | | | |
| single-server-2 | 1 | 0 | | | | | | | | | | | | | | | | |
| evpn_single_001 | evpn-single | L2 | IPv4 | 1 single leaf | 0 | <table border="1"> <thead> <tr> <th>Name</th> <th>Used</th> <th>Available</th> </tr> </thead> <tbody> <tr> <td>single-server</td> <td>1</td> <td>4</td> </tr> </tbody> </table> | Name | Used | Available | single-server | 1 | 4 | | | | | | |
| Name | Used | Available | | | | | | | | | | | | | | | | |
| single-server | 1 | 4 | | | | | | | | | | | | | | | | |

From the blueprint, navigate to **Active / Physical / Racks** to see details about the racks in the active blueprint. You can search for specific nodes or links and select a layer to see anomalies, deploy modes, deployment status, traffic heat and more. Click a rack type to see a preview.

Changing 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 (as of AOS version 3.3.0).

1. From the blueprint, navigate to **Active / Physical / Racks**.
2. Choose the rack that needs a name change.
3. In **Rack Properties** (right panel) click the **Edit** button for the rack name.
4. Change the name and click the **Save** button to stage the change.

Rack names can also be changed from the staged view of the blueprint.

4.6.1.7 Pods

1. Active

2. Physical

3. Pods

Click to search

Select layer to see anomalies, deployment status and more

Click rack name for rack type preview

pod1

Capacity:

Query: All

6-7 of 7

| Name | Type | Used | Available |
|------------------|--------|------|-----------|
| L2 Leaf External | global | 0 | 22 |
| L2 One Leaf | global | 0 | 22 |

pod2

Capacity:

Query: All

1-5 of 14

| Name | Type | Used | Available |
|---------------|----------|------|-----------|
| border | global | 0 | 4 |
| border | embedded | 0 | 4 |
| compute-2pair | global | 0 | 4 |
| compute-2pair | embedded | 3 | 4 |
| compute-10g | global | 0 | 39 |

From the blueprint, navigate to **Active / Physical / Pods** to see details about the active pods in a 5-stage Clos network. (The pod view only applies to pod-based blueprints.) You can search for specific nodes or links and select a layer to see anomalies, deploy modes, deployment status and more. Click a rack name in a pod to see its rack type preview.

4.6.2 Query

You can search for MAC addresses, IP addresses and VMs by using the query feature in the active blueprint.

The screenshot shows the Apstra Active/Query interface. The top navigation bar includes tabs for Dashboard, Analytics, Staged, Uncommitted, Active, and Time Voyager. Below this, there are filters for Physical, Virtual, Policies, Catalog, Settings, Query, Anomalies, and Root Causes. The 'Query' tab is selected, and the 'MAC' filter is chosen. The search results table shows two entries for 'leaf3' with S/N 525400EB10B9, Hostname leaf3, Type Static, VLAN 0, and MAC addresses 2e:58:a6:78:a6:8b and 3e:c1:b3:3a:7f:b6. The interface also shows a 'Query: All' dropdown and a 'Page Size' of 25.

| Node Name | S/N | Hostname | Type | VLAN | VXLAN | MAC | Interface |
|-----------|--------------|----------|--------|------|-------|-------------------|------------|
| leaf3 | 525400EB10B9 | leaf3 | Static | 0 | | 2e:58:a6:78:a6:8b | vxlان25003 |
| leaf3 | 525400EB10B9 | leaf3 | Static | 0 | | 3e:c1:b3:3a:7f:b6 | vxlان10001 |

4.6.2.1 Searching the Active Blueprint

1. From the blueprint, navigate to **Active / Query**.
2. Click **MAC**, **ARP**, or **VMs** depending on your query.
3. Click **Query:All**, enter search criteria, and click **Apply** to see results.

4.6.3 Anomalies (Service)

This section covers service Anomalies. For analytics anomalies see [IBA Anomalies](#).

4.6.3.1 Discovery Anomalies

To demonstrate anomalies during the discovery phase, cabling errors have been deliberately configured in the example below so that alarms are triggered.

The screenshot shows the Apstra Dashboard for the 'L2 Virtual EVPN' blueprint. The 'Deployment Status' section shows 'Service config deployment' with 0 SUCCEEDED, 0 PENDING, and 0 FAILED, and 'Discovery config deployment' with 9 SUCCEEDED, 0 PENDING, and 0 FAILED. The 'Anomalies' section shows 'All Probes' with 0 anomalies and 'IP Fabric' with 0 anomalies for BGP, 10 anomalies for Cabling, 0 anomalies for Interface, and 0 anomalies for Hostname. The 'Cabling' gauge is highlighted in red.

To see the list of ten 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 "evp3" | Property Value neighbor interface "evp1" neighbor name "12-virtual-evpn-001-leaf2" | Property Value neighbor interface "Ethernet1/1" neighbor name "12-virtual-evpn-002-leaf1" | 15 minutes ago |
| _l2_virtual_evpn_001_leaf2 | l2-virtual-evpn-001-leaf2 | IP Fabric | cabling | Spine to Leaf | Property Value Interface "evp1" | Property Value neighbor interface "evp3" neighbor name "spine1" | Property Value neighbor interface "evp2" neighbor name "spine1" | 15 minutes ago |
| _l2_virtual_evpn_002_leaf1 | l2-virtual-evpn-002-leaf1 | IP Fabric | cabling | Spine to Leaf | Property Value Interface "Ethernet1/1" | Property Value neighbor interface "evp1" neighbor name "spine1" | Property Value neighbor interface "evp3" neighbor name "spine1" | 15 minutes ago |
| spine1 | spine1 | IP Fabric | cabling | Spine to Leaf | Property Value Interface "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

Legend: No Anomalies (Green), Anomalies Present (Red)

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
0/9/0 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 displays the Apstra GUI interface for managing network blueprints. The breadcrumb trail at the top indicates the path: Blueprints > _L2 Virtual EVPN > Active > Physical > Selection > Node. The main navigation bar includes tabs for Dashboard, Analytics, Staged, Uncommitted, Active, and Time Voyager. Below this, a secondary navigation bar shows Physical, Virtual, Policies, Settings, Query, Anomalies, and Root Causes. The Physical tab is active, and the Anomalies section is highlighted.

The interface shows a topology view for a node named 'spine1'. The 'Nodes: All' and 'Links: All' filters are applied. The 'Topology' tab is selected, showing a diagram of the node's interfaces and their connections. The 'Selected Rack' is 'All' and the 'Selected Node' is 'spine1 (Spine)'. The 'Topology Label' is set to 'Name'. The 'Neighbors View' is selected, showing a list of neighbors for the selected node.

The 'Neighbors View' table shows the following connections:

| Color | Link | Interface |
|-------|------|-------------------------------------|
| ok | swp1 | Ethernet1/1 _l2_virtual_evpn_002... |
| ok | swp2 | swp1 _l2_virtual_evpn_001... |
| ok | swp3 | swp1 _l2_virtual_evpn_001... |
| ok | swp4 | Ethernet1/1 _l2_virtual_evpn_002... |
| ok | swp5 | Ethernet1 _l2_virtual_evpn_003... |
| ok | swp6 | Ethernet1 _l2_virtual_evpn_003... |

The 'Color' column indicates the status of the link: 'ok' (green), 'violating intent' (red), and 'unintended' (grey). The 'Link' column shows the link type: 'present' (solid line) and 'absent' (dashed line). The 'Interface' column shows the interface name and its role.

On the right side, the 'spine1' node is selected, and its 'Telemetry' tab is active. The 'Probes' section shows a list of probes with their status indicators. The 'All Services' probe has a red status indicator, indicating a violation. The 'Cabling' probe also has a red status indicator, indicating a violation. The 'BGP', 'Route', and 'Hostname' probes have green status indicators, indicating they are healthy.

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 and actual. If other anomalies existed in addition to the cabling anomalies, they would be shown in the list below as well.

[Home](#) › [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 | "swp3" | neighbor interface | "swp1" | 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 | "swp6" | 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 | "swp5" | 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 | "swp1" | neighbor interface | "Ethernet1/1" | neighbor interface | "swp1" | 4 hours ago |
| | | | | | neighbor name | "12-virtual-evpn-002-leaf1" | neighbor name | "12-virtual-evpn-001-leaf1" | |
| IP Fabric | cabling | Spine to Leaf | Interface | "swp2" | neighbor interface | "swp1" | neighbor interface | "swp1" | 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.

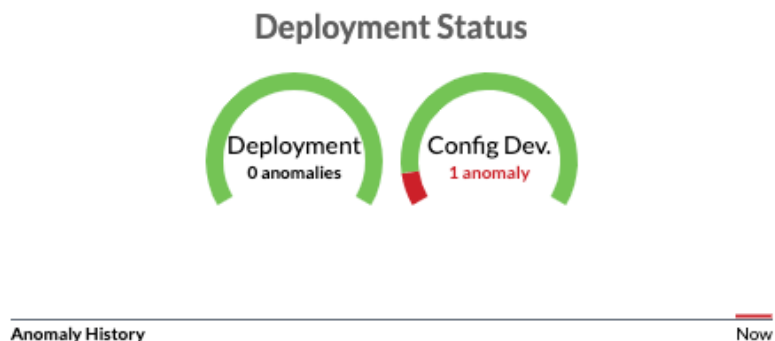
| Interface | Neighbor node | Neighbor interface | Actual Neighbor node | Actual Neighbor interface | Intent status | Neighbor system | Last fetched | Last modified |
|-----------|---------------------------|--------------------|---------------------------|---------------------------|---------------|--|-------------------|---------------|
| 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 [Cabling Discovery](#).

4.6.3.2 Configuration Deviation

AOS continuously compares device running configurations with the [Golden Config](#). If a config deviation is found a configuration anomaly is raised. Typically such deviations are seen when changes were made outside of AOS (by using the device CLI), or attempting to deploy configuration on a switch that is not able to take the change. These anomalies remain active until either the anomalous configuration is removed from the device or the anomaly is suppressed.

1. From the Blueprint dashboard, any configuration deviations are displayed in the **Deployment Status** section.



2. Click **Config Dev.** to see the list of node(s) with anomalies.

Blueprints > L2V > Active > Anomalies

Dashboard Analytics Staged Uncommitted Active Time Voyager

Physical Virtual Policies Settings Query Anomalies Root Causes

Query: Service = Deployment Status and Anomaly Type = config 1-1 of 1 Page Size: 25

| 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 Value config Show in modal | Property Value config 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.)

Blueprints > L2V > System Nodes > l2_virtual_003_leaf1 > Active > Telemetry > Config

Staged Active

Physical Virtual Telemetry

Anomalies Config Interface MAC LLDP BGP Route Hostname Counters ARP Transceivers Utilization

Apply Full Config Accept Changes

Actual config deviated from golden config

| 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. If you would like to keep the configuration difference, click **Accept Changes**. Note This **suppresses** the configuration anomaly, and does not affect “Intended” or AOS rendered config. the primary purpose of “Accept Changes” is to mitigate *cosmetic* configuration anomalies.

Important: Accepting the actual configuration cannot be used as a backdoor into AOS to implement out-of-band changes to the network (manual CLI). **AOS does not support OOB changes.** For custom changes, use *Configlets*.

Warning:

- Depending on the change, Out of Band changes can be overwritten by AOS. There is no way to avoid this. As such, OOB changes should always be avoided.
- Using *Accept Changes* does **not** make the OOB change persistent. In the event of a full config push or AOS writing to the same config, all OOB changes are discarded.

Warning: Do not use **Accept Changes** on Cumulus Linux, unless suppressing a cosmetic anomaly. On Cumulus, AOS maintains device configuration using updates to various files, contents of which are overwritten. Any Blueprint commit requiring a change to a file that has out-of-band (OOB) changes results in those changes being discarded.

- 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.

Warning: Applying a full config is a disruptive operation and will result in a temporary loss of service to the device.

Warning: Never directly modify any AOS rendered config that affects routing and connectivity. Doing so can potentially impact the network's operation. When in doubt, contact Apstra Support.

- After resolving the config deviation anomaly, either by accepting changes or applying a full config, the actual config matches the golden config and the anomaly is cleared.

Blueprints > L2V > System Nodes > l2_virtual_003_leaf1 > Active > Telemetry > Config

Staged Active

Physical Virtual Telemetry

Anomalies Config Interface MAC LLDP BGP Route Hostname Counters ARP Transceivers Utilization

Apply Full Config Accept Changes

Everything is OK!
Actual config matches golden config

```

1 ! Command: show running-config
2 ! device: l2-virtual-003-leaf1 (vEOS, EOS-4.22.3M)
3 !
4 ! boot system flash:/boot-image.sui

```

Config Deviation and Configlets

If an improperly-configured configlet causes AOS deployment errors (when the command is rejected by the device), a **service config deployment** failure occurs. In this case, follow the steps below to resolve the anomaly.

- From the blueprint, navigate to **Staged / Catalog / Configlets** and delete the configlet.
- Click **Uncommitted** and commit the change. The configuration deviation remains because the golden config is empty. The golden config is the running config of the device after *successful* deployment of AOS-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 AOS that the failure can be ignored.

4.6.4 Root Causes

4.6.4.1 Root Cause Overview

Root Cause Identification (RCI) is a technology integrated into AOS that automatically determines root causes of complex network issues. RCI leverages the AOS datastore for realtime network status, and automatically correlates telemetry with each active blueprint intent. Root cause use cases include the following:

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.

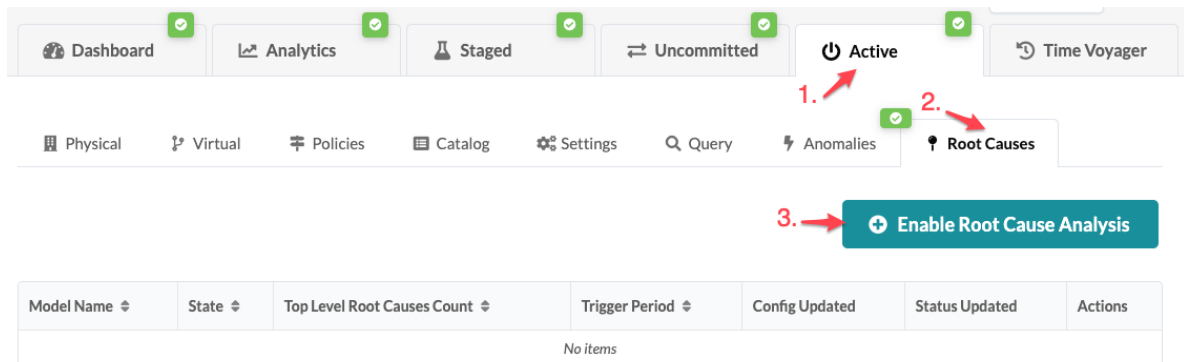
Operator shut interface Symptoms: Both interfaces on the link are operationally down; the interface in question is administratively down; LLDP missing on both sides, BGP peered across that link is operationally down.

Disconnection between 2 devices Symptoms: Union of symptoms for link broken / link miscabled / operator shut interface for all constituent links between a spine and a leaf

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.

4.6.4.2 Enabling Root Cause Analysis

1. From the blueprint, navigate to **Active > Root Causes**.



2. Click **Enable Root Cause Analysis**.
3. Enter a **Trigger Period** or leave the default, and click **Create** to enable root cause analysis and return to the list view.

4.6.4.3 Viewing Root Cause Analysis

From the blueprint, navigate to **Active > Root Causes** and click the model name **connectivity** in the list.

| | |
|----------------|-------------------|
| Model Name | connectivity |
| State | OPERATIONAL |
| Trigger Period | 20s |
| Config Updated | 5 hours ago |
| States Updated | a few seconds ago |

Root Causes

Topology View Tabular View

1-1 of 1 < > Page Size: 25 ▾

| Description | Timestamp | Symptoms |
|---|-------------|---|
| Cable broken between spine1:Ethernet3/1/1 and ted-rack-001-leaf1:Ethernet49/1 | 4 hours ago | <ul style="list-style-type: none"> Interface Ethernet3/1/1 is down on host spine1 (link) Interface Ethernet49/1 is down on host ted-rack-001-leaf1 (link) BGP session down from ted-rack-001-leaf1 (ASN: 500 IP: 172.10.0.1) to spine1 (ASN: 504 IP: 172.10.0.0) (link) BGP session down from spine1 (ASN: 504 IP: 172.10.0.0) to ted-rack-001-leaf1 (ASN: 500 IP: 172.10.0.1) (link) |

Root cause analysis runs periodically. Each time it runs, it produces zero or more root causes. Each root cause has associated detection timestamp, context, a human-readable description, and a list of symptoms caused by the root cause. Each symptom has associated context and a human-readable description.

AOS version 3.0 added RCI Connectivity Fault Model, which identifies any miscabled leaf/spine links. AOS correlates faults between neighbor down (with UP/UP interfaces) and intended LLDP.

4.7 Time Voyager

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 (new in version 3.2) to automatically restore previous revisions of a blueprint.

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); backup/restore does not. Effectively, the backup/restore is an out-of-band change from a device configuration standpoint. If a backup is restored, you would need to perform a full config apply to make sure the device configuration reflects what you restored from the database backup. This would most likely be disruptive.

A blueprint can be jumped back to any retained revision. The five 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 revisions, effectively having thirty 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. Currently, specific diffs 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.

Isn't Time Voyager just an UNDO function?

When using Time Voyager you roll back to a previous commit. This means that anything deleted on the last commit will be 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 the 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 version, refer to [Restoring Database](#) that was taken prior to upgrading, although this may 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 device operating system (DOS) versions are different between revisions. You could downgrade the DOS 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.

4.7.1 Listing Blueprint Revisions

From the web interface, navigate to the **Time Voyager** view of the blueprint. From the **Revisions** tab, you'll see the list of retained blueprint revisions. The first revision in the list is the currently active one. Successive revisions are ordered by date from most recent to oldest.

4.7.2 Keeping a Saved Blueprint Revision

1. From the **Time Voyager** view of the blueprint, click the **Keep this revision** button corresponding to the revision to keep. (It looks like a floppy disk.)
2. Click **Save** to confirm and proceed. The button turns gray indicating that the revision has been saved indefinitely. It will not be deleted until you manually delete it.

4.7.3 Deleting a Kept Blueprint Revision

1. From the **Time Voyager** view of the blueprint, click the **Delete** button corresponding to the revision to delete. You cannot delete a revision if there are five or fewer of them in the list.
2. Click **Delete** to confirm.

4.7.4 Jumping to a Previous Revision in a Blueprint

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 **Time Voyager** view of the blueprint, click the **Jump to this revision** button corresponding to the revision to jump to. You'll see its dialog.
2. Any uncommitted changes in the staged area will be 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 will not. 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. Currently, specific diffs 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](#).
9. If you click **Time Voyager** you'll see the revision as the current one.

Please see our *device guides* for more information on working with devices in AOS.

5.1 Managed Devices

Devices with installed *AOS device agents* appear in the managed devices list. From the AOS web interface, navigate to **Devices / Managed Devices**.

| Device Key | Device Profile | Operation Mode | Management IP | AOS Version | Hostname | Location | OS | Acknowledged? | State | Blueprint | Comms |
|--------------|----------------|----------------|---------------|------------------|-----------------------|----------|----------------|---------------|-----------|-----------|-------|
| 5254006C26F6 | Cumulus VX | FULL CONTROL | 172.20.131.11 | AOS_3.3.0_OB.730 | spline1 | | Cumulus 3.7.11 | ✓ | IS-ACTIVE | pod1 | ✓ |
| 52540001525B | Cumulus VX | FULL CONTROL | 172.20.131.12 | AOS_3.3.0_OB.730 | spline2 | | Cumulus 3.7.11 | ✓ | IS-ACTIVE | pod1 | ✓ |
| 52540094B762 | Cumulus VX | FULL CONTROL | 172.20.131.13 | AOS_3.3.0_OB.730 | evpn-mlag-001-leaf1 | | Cumulus 3.7.11 | ✓ | IS-ACTIVE | pod1 | ✓ |
| 50540013B75B | Arista vEOS | FULL CONTROL | 172.20.131.14 | AOS_3.3.0_OB.730 | evpn-single-001-leaf1 | | EOS 4.22.3M | ✓ | IS-ACTIVE | pod1 | ✓ |
| 52540045AA32 | Cumulus VX | FULL CONTROL | 172.20.131.15 | AOS_3.3.0_OB.730 | evpn-mlag-001-leaf2 | | Cumulus 3.7.11 | ✓ | IS-ACTIVE | pod1 | ✓ |

From **Managed Devices**, you can perform tasks, such as acknowledging devices (to bring them under AOS management), setting admin states, updating user configuration, changing associated device profiles or admin states, and deleting devices. See the *device guides* for detailed information about device management in AOS.

5.1.1 Acknowledge Device(s)

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

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

5.1.2 Setting Admin State on Devices

1. From the AOS web interface, navigate to **Devices / Managed Devices** and select the device(s) to update.

2. Click the button for the state to change the selection(s) to (MAINT, NORMAL, DECOMM).
3. Click **Confirm** to set the admin state and return to the list view.

Important: If you are decommissioning a device, after setting the admin state to **DECOMM**, you must uninstall the device agent. See the device guide for [removing a device](#). If you would subsequently like AOS to manage the device you can [add the device](#) back to AOS.

5.1.3 Updating User Config

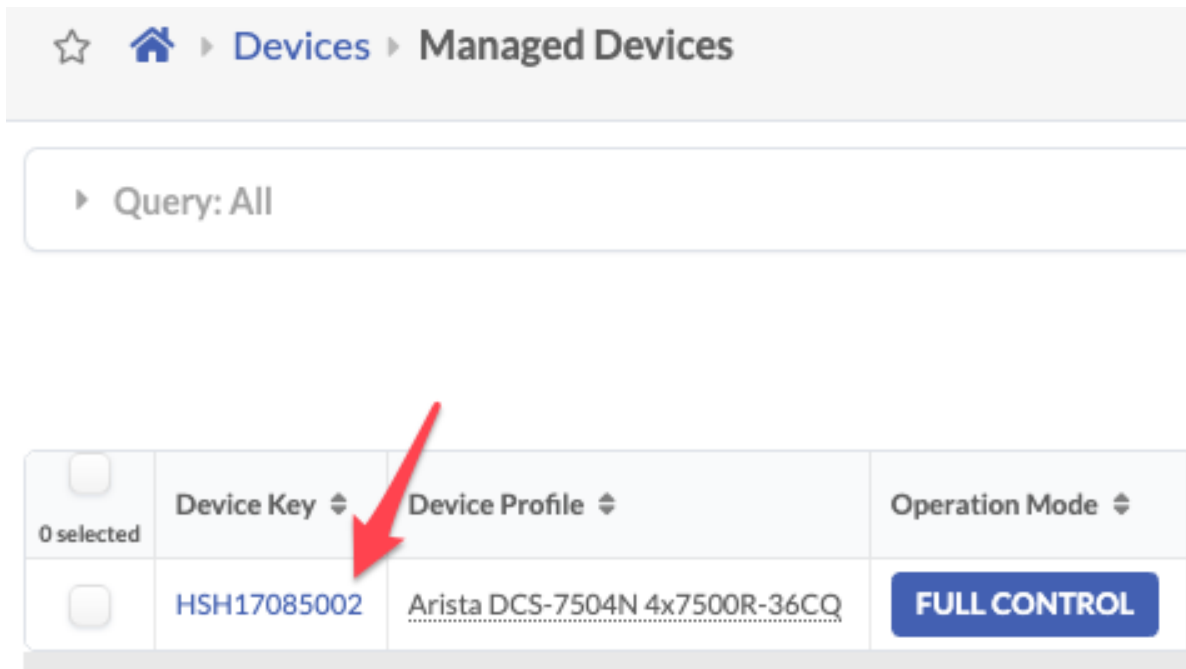
1. From the AOS web interface, navigate to **Devices / Managed Devices** and select the device(s) to update.
2. Click the **Update user config** button, then to override device profile, admin state, and/or location, check the appropriate boxes.
3. Click **Confirm** to update the user config and return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

5.1.4 Editing System

There may be cases where you'll need to change managed device details. For two examples, see the [Modular Devices and Device Profiles](#) and [Hyperloop Interface](#) sections below.

To edit a managed device:

1. From the AOS web interface, navigate to **Devices / Managed Devices** and select the device key for the device to update.



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

Edit System

Device Profile *

Arista DCS-7504N 4x7500R-36CQ

Admin State *

☐ DECOMM
☐ MAINT
☐ NORMAL

Location

Update

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

5.1.4.1 Modular Devices and Device Profiles

Multiple *device profiles* may exist for one modular device model to represent different line card configurations.

[☆](#)
[🏠](#)
[Devices](#)
[Device Profiles](#)

Create Device Profile

Query: Name = "7504"

1-2 of 2

Page Size: 25

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

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

☆
🏠
>
Devices
>
Managed Devices

+

Stock Device

Query: All

1-1 of 1

Page Size: 25

| 0 selected | Device Key ↕ | Device Profile ↕ | Operation Mode ↕ | Management IP ↕ | AOS Version ↕ | Hostname ↕ | Location ↕ | OS ↕ | Acknowledged? ↕ | State ↕ | Blueprint ↕ | Comms ↕ |
|--------------------------|--------------|-------------------------------|------------------|-----------------|-------------------|------------|------------|-------------|-----------------|-----------------|--------------|---------|
| <input type="checkbox"/> | HS17085002 | Arista DCS-7504N 4x7500R-36CQ | FULL CONTROL | 172.20.140.6 | AOS_3.3.0.1_OB.88 | localhost | | EOS 4.22.3M | | OOS-QUARANTINED | Not assigned | |

AOS currently does not match device profiles based on line card configuration.

When using a modular device in your network, check that the correct device profile is associated with it. If necessary change it by following the steps above for [editing a system](#). This must be done prior to acknowledging the managed device and assigning it to an AOS blueprint.

Edit System

Device Profile *

7504

Arista DCS-7504N 4x7500R-36CQ
Arista DCS-7504N 4x7500R-36Q

Location

Update

5.1.4.2 Hyperloop Interface

If you want to set up a hyperloop interface to [enable VXLAN Routing on a Cumulus device equipped with Tomahawk](#) so it supports Routing In and Out of Tunnels (RIOT) you must change the device profile used in the managed device.

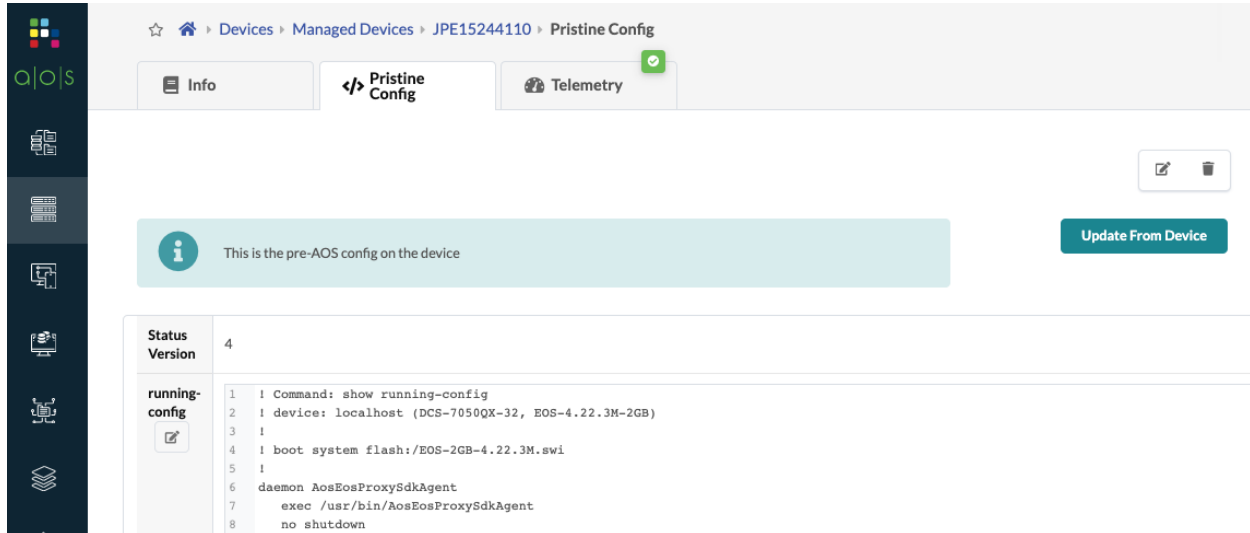
5.1.5 Editing Pristine Config

Warning: The Pristine Config should *never* be modified directly unless there is no alternative. Changes are not validated. Contact [Apstra Global Support](#) for assistance as needed.

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. Persistent changes to a configuration are best applied with [configlets](#).

1. From the AOS web interface, navigate to **Devices / Managed Devices** and select the **Device Key** of the device.

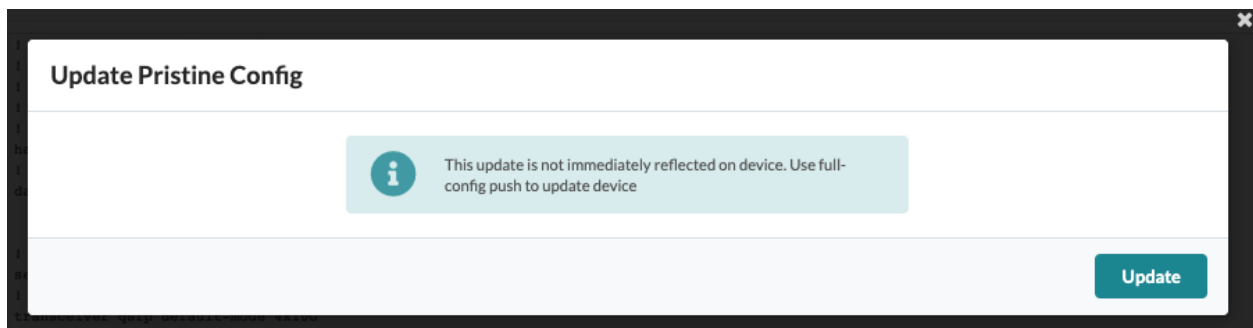
2. Click the **Pristine Config** tab (top-left), then click the **Edit pristine config** button (under running-config on the left).
3. Make your changes.
4. Click **Update** to apply the changes.



5.1.6 Updating Pristine Config from Device

It is also possible to update (customize) the pristine config by copying the running device configuration. To use this function, the device should be out of service (OOS-READY or OOS-MAINT). Once you have your device in OOS state, please make sure that all the necessary changes are done manually via CLI on the device before updating pristine config from the device.

1. Unassign the device from the blueprint.
2. Make any necessary changes to the running device configuration via CLI.
3. From the AOS web interface, navigate to **Devices / Managed Devices** and select the **Device Key** of the device.
4. Click the **Pristine Config** tab (top-left), then click the **Update From Device** button (top-right).
5. Click **Update** to update Pristine Config from the device.



Note: After the “Update From Device” operation, please verify the Pristine Config again. As you have copied the running config of the device in OOS state which should be the device “Discovery-1” config, it may include additional

configuration such as interface “speed” commands. You can edit Pristine Config again and delete those additional configuration manually. Contact [Apstra Global Support](#) for assistance as needed.

5.1.7 Deleting System(s)

Important: Devices that have been acknowledged cannot simply be deleted - as there is still an active agent on the device talking to the AOS server, they would re-appear within seconds. For details on removing a device from AOS, see the [device guides](#).

After decommissioning a device and uninstalling its agent, remove it from the managed devices list.

1. From the AOS web interface, navigate to **Devices / Managed Devices** and select the device(s) to remove.
2. Click the **Delete system(s)** button, then click **Confirm** to remove the device(s) and return to the list view.

5.1.8 Managed Device Telemetry

1. From the AOS web interface, navigate to **Devices / Managed Devices** and select the **Device Key** of the device.
2. Click the **Telemetry** tab (top-left), then click a tab to see the relevant telemetry.

From the **Config** tab, you can **Apply Full Config** or **Accept Changes**, as applicable. You can also [apply full config from the blueprint that it's used in](#). See [Configuration Deviation](#) for more information about when to apply a full config.

5.2 Telemetry

5.2.1 Telemetry Overview

5.2.1.1 Telemetry in the Web Interface

To go to a summary of the number of devices running the different telemetry services from the web interface, navigate to **Devices > Telemetry**. To go to collection statistics for a specific service for all relevant devices, click a service name.

Click a service to go to collection stats for that service.

Or click devices...

... then one of the devices to go to all telemetry services for that device

1. Click a service to go to collection stats for that service.

2. Click a device name to go to all telemetry services for that device

The screenshot shows the Apstra interface with the left sidebar containing 'Blueprints', 'Devices', 'Managed Devices', 'Telemetry' (highlighted), 'System Agents', 'Agents', 'Agent Profiles', 'Packages', 'OS Images', and 'ZTP Logs'. The main content area is titled 'Devices > Telemetry' and displays statistics for various services: ARP, BGP, DISK UTIL, HOSTNAME, INTERFACE, and INTERFACE COUNTERS. Each service card shows 'Configured on: 1 devices', 'Errors during enabling: 0 devices', and 'Last collection cycle errors: 0 devices'. A red arrow points from the 'Telemetry' link in the sidebar to the service cards. Another red arrow points from the '1 devices' link in the ARP card to the 'HOSTNAME' card. A third red arrow points from the '1 devices' link in the BGP card to the 'INTERFACE' card.

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 to see details about any errors

Click a device name to go to all telemetry services for that device

The screenshot shows the Apstra interface with the left sidebar containing 'Blueprints', 'Devices', 'Managed Devices', 'Telemetry' (highlighted), 'System Agents', 'Agents', 'Agent Profiles', 'Packages', 'OS Images', and 'ZTP Logs'. The main content area is titled 'Devices > Telemetry > INTERFACE' and displays a table of telemetry services for a specific device. The table has columns: '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', and 'Error message'. The table shows three rows of data. The first row is for device '525400B46A1E (spine1, 172.20.4.7)' with 'Service Started? yes', 'Interval, s 120', 'Run Count 1710', 'Success Count 1710', 'Failure Count 0', 'Max Run Count', 'Execution Time, ms 0.03', 'Waiting Time, ms 0.00', 'Last Run Timestamp 2021-04-30, 15:53:45', 'Last Error Timestamp', and 'Error message N/A'. The second row is for device 'MT2007J03912 (I2-virtual-ext-001-leaf1, 172.20.4.6)' with 'Service Started? yes', 'Interval, s 120', 'Run Count 1712', 'Success Count 45', 'Failure Count 1667', 'Max Run Count', 'Execution Time, ms 102.03', 'Waiting Time, ms 0.00', 'Last Run Timestamp 2021-04-30, 15:53:44', 'Last Error Timestamp 2021-04-30, 15:53:45', and 'Error message Show error'. The third row is for device 'MT2034J00075 (100.81.150.214)' with 'Service Started? no', 'Interval, s 120', 'Run Count', 'Success Count', 'Failure Count', 'Max Run Count', 'Execution Time, ms 0.00', 'Waiting Time, ms 0.00', 'Last Run Timestamp', 'Last Error Timestamp', and 'Error message N/A'. A red arrow points from the 'Show error' link in the second row to the 'Error message' column. Another red arrow points from the 'Device' column header to the first row. A third red arrow points from the 'Show error' link in the second row to the 'Error message' column.

To go to collection statistics for that device for all services, click **Collection Statistics**.

| <input type="checkbox"/> | 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 ↕ |
|--------------------------|--------------------|--------------------|----------------|---------|-------------|-----------------|-----------------|-----------------|----------------------|--------------------|----------------------|------------------------|-----------------|
| <input type="checkbox"/> | ARP | yes | 120 | | 253 | 253 | 0 | | 833.34 | 1.12 | 2021-04-19, 18:35:25 | | N/A |
| <input type="checkbox"/> | INTERFACE | yes | 120 | | 253 | 253 | 0 | | 1020.85 | 829.35 | 2021-04-19, 18:35:26 | | N/A |
| <input type="checkbox"/> | DISK UTIL | yes | 120 | | 251 | 251 | 0 | | 823.04 | 0.71 | 2021-04-19, 18:33:37 | | N/A |
| <input type="checkbox"/> | LLDP | yes | 10 | | 3019 | 3019 | 0 | | 301.13 | 0.68 | 2021-04-19, 18:35:18 | | N/A |
| <input type="checkbox"/> | HOSTNAME | yes | 120 | | 253 | 253 | 0 | | 931.78 | 0.64 | 2021-04-19, 18:35:25 | | N/A |
| <input type="checkbox"/> | ROUTE | yes | 120 | | 253 | 253 | 0 | | 928.86 | 875.84 | 2021-04-19, 18:35:26 | | N/A |
| <input type="checkbox"/> | XCVR | yes | 120 | | 253 | 253 | 0 | | 1760.83 | 1.89 | 2021-04-19, 18:35:25 | | N/A |
| <input type="checkbox"/> | BGP | yes | 120 | | 253 | 253 | 0 | | 679.94 | 925.42 | 2021-04-19, 18:35:26 | | N/A |
| <input type="checkbox"/> | RESOURCE UTIL | yes | 10 | | 3011 | 3011 | 0 | | 2419.77 | 0.98 | 2021-04-19, 18:35:22 | | N/A |
| <input type="checkbox"/> | INTERFACE COUNTERS | yes | 5 | | 5353 | 5353 | 0 | | 5082.44 | 0.70 | 2021-04-19, 18:35:19 | | N/A |
| <input type="checkbox"/> | MAC | yes | 120 | | 253 | 253 | 0 | | 875.60 | 1.06 | 2021-04-19, 18:35:25 | | N/A |

5.2.1.2 Telemetry Services

Telemetry services include the following:

ARP ARP telemetry shows an ARP table. This information can be queried via API. No anomalies are generated.

BGP BGP telemetry shows role(s), VRF name, address family, source and destination information, expected and actual states, intent status, last fetched/modified, and (as of version 3.3.0) BGP peer state.

Config 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.

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* instead.

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 **Discovery 2 Config** stage. Hostname telemetry is collected that validates the device hostname against intent. Mismatches result in anomalies.

Interface When you assign a device with deploy mode **Ready** to a blueprint, the device enters the **Discovery 2 Config** stage. 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 **Discovery 2 Config** stage. Every node is part of intent. On each link, there are expected neighbor hostnames, interfaces and connections. Physical cabling and links must match the specified intent. Any deviations result in anomalies that must be corrected by either recabling to match the blueprint or by modifying the blueprint to match cabling already in place.

Apstra knows what should be connected on each link, what its neighbor hostname should be, and what its neighbor interface should be.

MAC MAC Address-table telemetry shows which MAC addresses appear on which interfaces, and which VLANs.

MLAG 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.

If an MLAG-attached server is not fully connected, the state changes from 'active_full' to 'active_partial'.

Note: Cisco MLAG (VPC) commands cannot derive the status of the LAG on the VPC peer switch. Accordingly, the state *dual-active* cannot actually gather the command. This is a limitation from Cisco.

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 Utilization telemetry allows the network operator to view some vital statistics on the device - CPU and Memory utilization. No anomalies are generated.

5.2.1.3 Telemetry Collection Statistics

Telemetry collection statistics include the following details:

Device The device key

Service Started? Has the service started?

Interval How frequently the service is configured to run on the device (in seconds)

Input The input that is provided to the service for its processing

Run Count The number of times the collector is scheduled to run

Success Count The number of times the collector successfully executed

Failure Count The number of times the collector failed execution

Max Run Count User-specified maximum number of times for the collector to run

Execution Time The time it took for collection during the last iteration (in milliseconds)

Waiting Time A device runs multiple collectors. If some collectors monopolize CPU, other collector executions are deferred. Waiting time is the amount of time that the collector was deferred (in milliseconds).

Last Run Timestamp Timestamp at which the collector was scheduled to run

Last Error Timestamp Timestamp at which the collector last reported an error

Error message Error message from the last collector iteration.

5.2.2 L2 Server Telemetry

You can collect telemetry data from L2 servers with onbox agents. For agent installation instructions, see [Agents](#).

The following telemetry data is available on L2 servers by default. Data is collected using Subprocess over ssh. This is Unintended data, meaning since the data is not matched against Intent, anomalies are not raised for these data.

LLDP Displays LLDP neighbor information

CPU Collects five-second average CPU utilization every five seconds for individual Apstra processes as well as for the total

RAM Collects five-second average memory footprint every five seconds for individual Apstra processes as well as for the total

Interface Status Lists Interface names and status

With [IBA Probes](#), you can vastly expand on the collected data and define exact conditions in which anomalies are raised.

5.2.3 External Router Telemetry

Apstra includes telemetry expectations for external routing - a default route is expected to be received by each external router in the blueprint. These routes are load-balanced with ECMP, and each switch (and server, if L3) expects to receive multiple routes.

5.2.4 Telemetry Streaming

The term *telemetry streaming* refers to the Apstra server transmitting the following content to user-defined end-hosts so that you can further process the data and use them within your own internal systems:

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 to in order to troubleshoot your network. The best reference for example event data is syslog. You need a general amount of event history so that you can perform troubleshooting activities over a period of time. While this is an undefined amount of time, you generally want as much time as possible, because you don't get to troubleshoot a problem the instant that it occurs.

Alert Data Alert data is a collection of information that requires your attention to resolve an issue. In the best cases, alerts tell you what is wrong relative to the network service, and provide the necessary data to allow you to identify root-cause and resolve the issue as fast as possible.

The data streams themselves are implemented with Google Protocol Buffers (GPB). The format of the data streams is defined and also implemented using GPBs. GPBs allow software developers to use a language-agnostic definition of events and data types.

GPB offers support for C++, Python, Go, and possibly more languages in the future. Apstra has example Python code named *AOSOM Streaming* that is available for Google Protocol Buffers. The AOSOM Streaming demo software is open source and can be downloaded from github: <https://github.com/Apstra/aosom-streaming>.

The developer has options of different languages: C++, Python, and Go are all available. This means it integrates nicely with our C++ infrastructure. And then Infrastructure Engineers can use Python or Go for the client.

5.2.5 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": "547bcb9c9-963f-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"
      },

```

(continues on next page)

(continued from previous page)

```

    },
    "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": {

```

(continues on next page)

(continued from previous page)

```

    "value": "up"
  },
  "id": "ab9f4273-e86f-456c-8cc7-7115f3aafa45",
  "identity": {
    "anomaly_type": "bgp",
    "destination_asn": "1",
    "destination_ip": "1.1.1.1",
    "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
}

```

5.2.6 Telemetry Command Reference

The purpose of this section is to assist network administrators in understanding why telemetry alarms exist, and how they are generated. This is not an exhaustive list of interface commands.

5.2.6.1 Cisco

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.

Cisco Telemetry

Interface counters show interface counters | json

Interface error counters show interface counters errors | json

Interface status show interface status | json

LLDP neighbors show lldp neighbors detail | json

BGP Sessions show bgp session | json

Hostname show hostname | json and show hosts | json

ARP show ip arp vrf default | json

MAC Table show mac address-table | json

Routing table show ip route | json

Port-channel show port-channel summary | json

MLAG show vpc | json

5.2.6.2 Arista

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.

Warning: Event-based subscription requires the EOSProxySDK agent. For details please refer to [Arista Device Agent](#).

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

Arista telemetry commands

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

5.2.6.3 Cumulus

Cumulus switches use a combination of the Cumulus netshow command and standard Linux sockets.

Cumulus Telemetry commands

Interface counters ethtool -m

Interface error counters ethtool -m

Interface status Interface status is collected using the netlink api (AF_INET)

LLDP neighbors lldpctl -f json

BGP Sessions vtysh -c 'show ip bgp summary json'

Hostname hostname

ARP ip -4 neigh

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 and the AF_INET linux socket

Port-channel netshow bondmems --json

MLAG clagctl -j

5.2.6.4 Linux Servers

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

Interface counters `ethtool -m`

Interface error counters `ethtool -m`

Interface status Interface status is collected using the netlink api (AF_INET)

LLDP neighbors `lldpctl -f xml`

BGP Sessions `vttysh -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`

5.2.7 Extensible Telemetry

To collect additional telemetry, you need Apstra device drivers and telemetry collectors. You can use collected telemetry information in *IBA probes*.

5.2.7.1 AOS Device Drivers

AOS device drivers enable Apstra to connect to a Device Operating System (DOS) and collect telemetry. Apstra ships with drivers for EOS, Cumulus, NX-OS, Ubuntu, and CentOS. To add a driver for an operating system not listed here, contact *Juniper JTAC Apstra Support*.

5.2.7.2 Telemetry Collectors

Telemetry collectors are Python modules that help collect extended telemetry information. The following sections describe the pipeline for creating telemetry collectors and extending Apstra with the new collectors. You are expected to be familiar with Python for collector development.

Setting Up The Development Environment

Please contact *Juniper JTAC Apstra Support* for access to the telemetry collectors, which are housed in the *aos_developer_sdk* repository. Please contribute new collectors 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-fabric-conductor>. 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 setup for developing and testing the collectors. Apstra SDK packages, such as device drivers and REST client, are also installed in the environment.

Developing a Telemetry 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. **Device Operating System (DOS) for which the collector is developed** The collector plugins are DOS-specific. Before writing a collector, identify the DOS(s) for which collector(s) are required.
4. **How the required data can be obtained from the device** Identify the commands that can be used in the device to retrieve the required information. For example, 'show interfaces' command gives received and transmitted bytes from an Arista EOS device.
5. **Storage Schema Path** The storage schema path is determined by the type of key and value in each item. The type of collector selected determines the storage schema for the application. The storage schema defines

the high level structure of the data returned by the service. The storage schema path for your collector can be determined using the following table

Table 1: 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 2: 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": "1.1.1.1", "dest_ip": "1.1.1.2", }, "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 3: Sample Application Schema

| Sample Item | Application Schema |
|---|---|
| <pre>{ "identity": "eth0", "value": "up", }</pre> | <pre>{ "type": "object", "properties": { "identity": { "type": "string", }, "value": { "type": "string", } } }</pre> |
| <pre>{ "key": { "source_ip": "1.1.1.1", "dest_ip": "1.1.1.2", }, "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`

Writing A Collector

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 the data from the device A device driver instance is available inside the collector. The device driver 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 already installed. The methods available to collect the data can be explored. 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',
 u'bootupTimestamp': 1548302664.0,
 u'hardwareRevision': u'',
 u'internalBuildId': u'68f3ae78-65cb-4ed3-8675-0ff2219bf118',
 u'internalVersion': u'4.20.10M-10040268.42010M',
 u'isIntlVersion': False,
 u'memFree': 3003648,
 u'memTotal': 4011060,
 u'modelName': u'vEOS',
 u'serialNumber': u'',
 u'systemMacAddress': u'52:54:00:ce:87:37',
 u'uptime': 62620.55,
 u'version': u'4.20.10M'}
>>> dir(device)
['AOS_VERSION_FILE', '__class__', '__delattr__', '__dict__', '__doc__',
 '__format__', '__getattr__', '__hash__', '__init__', '__module__',
 '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__',
 '__sizeof__', '__str__', '__subclasshook__', '__weakref__', 'close',
 'device_info', 'driver', 'execute', 'get_aos_server_ip',
 'get_aos_version_related_info', 'get_device_aos_version',
 'get_device_aos_version_number', 'get_device_info', 'get_json',
 'get_text', 'ip_address', 'onbox', 'open', 'open_options', 'password',
 'probe', 'set_device_info', 'upload_file', 'username']
```

Parse the data The collected data needs to be parsed and re-formatted as 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>,
  },
  {
```

(continues on next page)

(continued from previous page)

```

        "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_<DOS>.txt. The packages installed by the dependency do not conflict with packages used by Apstra. The Apstra-installed packages are available at */etc/aos/python_dependency.txt* in the development environment.

Post the data to the framework When the data is collected and parsed as per the required schema, post the data to the framework. You can use the *post_data* method that is 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 DOS. Add your collector to the folder that matches the DOS. 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',

```

(continues on next page)

(continued from previous page)

```

        },
        '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 Testing The Collector

The folder `aos_developer_sdk/aosstdcollectors/test` in the repository contains folders based on the DOS. Add your test to the folder that matches the DOS. 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 AOS
    # 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,
```

(continues on next page)

(continued from previous page)

```

    }},
  },
  {
    '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
```

This command executes all the tests in the repository.

Packaging A Collector

All the collectors are packaged based on the DOS. To generate all packages, execute `make` at `aos_develop_sdk`. The build packages can be found at `aos_developer_sdk/dist`. The packages build can be broadly classified as:

Built-In Collector Packages These packages have the prefix `aosstdcollectors_builtin_`. To collect telemetry from a device per the reference design, Apstra requires services as listed in the [Device Telemetry](#) section. Built-In collector packages contain collectors for these services. The packages are generated on a per DOS basis.

Custom Collector Packages These package have the prefix `aosstdcollectors_custom_` in their names. The packages are generated on a per DOS basis. The package named `aosstdcollectors_custom_<DOS>-0.1.0-py2-none-any.whl` contains the developed collector.

AOS SDK Device Driver Packages These packages have a prefix `apstra_devicedriver_`. These packages are generated on a per DOS basis. Packages are generated for DOS that are not available by default in Apstra.

Uploading Packages

If Apstra did not ship with the built-in collector packages and the [AOS SDK Device Driver](#) for your Device Operating System (DOS), you must upload them to Apstra.

If you are using an offbox solution and your DOS is not EOS or Cumulus, you must upload the built-in collector package.

Upload the package containing your collector(s) and assign them to a System Agent or System Agent Profile.

Using The Telemetry Collector

To use the collector, set up the telemetry service registry.

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`. The collector for a service cannot be enabled without adding a registry entry for the particular service. The registry entry for a service cannot be modified while the service is in use.

Note: When executing *make*, all application schemas are packaged together to a tar file (json_schemas.tgz) in the dist folder. With AOS CLI, you have the option of importing all the schemas in the .tgz file.

Starting A Collector You can start a service using the POST API `/api/systems/<system_id>/services` with the following three 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 [AOS CLI](#).

Deleting a Collector You can delete a service with the DELETE API `/api/systems/<system_id>/services/<service_name>`.

Getting the Collected Data You can retrieve collected data with 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 all Running Collector Services You can retrieve the list of services enabled on a device with the GET API `/api/systems/<system_id>/services`.

5.2.8 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.

5.3 Agents

5.3.1 Agents Overview

System agents manage the devices that are connected to the Apstra server. They handle configuration management, device-to-server communication, and telemetry collection.

Important: On some platforms (Junos for example) you can configure rate-limiting for management traffic (SSH for example). The Apstra server interacts with devices differently than from a regular user; it can be of a more bursty nature. Rate-limiting configurations that are used for hardening security can impact device management, and lead to deployment failures and other agent-related issues.

The integrated agent installer automates device installation and verification. Use this installer in environments without a *ZTP Server* or for one-off agent installations.

5.3.1.1 On-box Agents

On-box agents are installed on the device itself and are supported on the following devices:

- Arista EOS
- Cisco NX-OS
- Cumulus Linux
- Enterprise SONiC

On-box agents require a user with **full admin / root privileges** which is used to configure the box. We recommended that you create a dedicated user on the device using [ZTP](#) or other means.

Important: You cannot change to a different user without completely re-onboarding the device. Changing the password involves several steps (password change on the device, device agents and ‘Pristine config’). This is not a straightforward process. If a password change is required, we recommend that you contact [Juniper JTAC Apstra Support](#).

5.3.1.2 Off-box Agents

Off-box agents are not installed on the device; they run server-side, and communicate with devices through their API for configuration management and [telemetry collection](#). Off-box agents are supported on the following devices:

- Arista EOS (eos)
- Cisco NX-OS (nxos)
- Juniper Junos (junos).

Before enabling devices for off-box deployment, you must configure them a specific way, and upload required packages. See sections below for details.

5.3.1.3 Agent Details

Agents include the following details:

Device Addresses (25 max) Management IP(s) of the device(s) to add

Agent Type

Onbox Installed on the switch

Platforms: Arista EOS, Cisco NX-OS, Cumulus Linux, and Enterprise SONiC.

Offbox Runs on the Apstra server and communicates with the switch through the device API

Platforms: Arista EOS (eos), Cisco NX-OS (nxos), Juniper Junos (junos)

Operation Mode

Full Control Deploys configuration and collects telemetry

Telemetry Only Only telemetry is collected

Platform For off-box agents only: Enter `eos`, `nxos`, or `junos` exactly, as applicable.

Username and Password If not provided by an [agent profile](#), check these boxes and add credentials.

Enabled? (version 3.1 and earlier) Check this box to activate. When you want to create the agent, but don't want to run it yet, leave this field blank.

Agent Profile (optional) Useful when you have several devices with the same profile. [Create system agent profiles](#) before creating agents.

Job to run after creation (version 3.2 and later) To install the on-box agent on the device, select **Install** (default). To skip the install of the on-box agent on the device, select **Check**. The device will be listed and the on-box agent can be installed later.

Install Requirements (version 3.1 and earlier)

Unchecked (default) To successfully create the agent, you must manually install any dependencies.

Checked Dependencies are automatically installed. Internet access is required per pip or apt package managers.

Intent (version 3.1 and earlier) Install

Uninstall

Install Policy (version 3.1 and earlier) Automatic

Once

Packages If you have previously installed packages, they will be available here for selection. If you selected an [agent profile](#) that has packages associated with it, they will be listed here as well.

Open Options (Off-box agents only) This passes configured parameters to the off-box agent. An example, may be to use HTTPS as the API connection from the off-box agent to the device API. In this case, use the following key-values:

| key | value |
|-------|-------|
| proto | https |
| port | 443 |

To go to the agents list view from the web interface, navigate to **Devices > Agents**.

The screenshot displays the Apstra web interface's 'Agents' list view. The sidebar on the left contains navigation links, with 'Agents' highlighted under 'System Agents'. The main content area shows a table of agents with columns: Device Address, Operation Mode, Platform, Platform Version, AOS Version, Job State, System ID, Hostname, and Device State. A 'Create Onbox Agent(s)' button is located at the top right. Red annotations highlight key features: '1.' points to the 'Devices' menu item, '2.' points to the 'Agents' menu item, '3.' points to the 'Create Onbox Agent(s)' button, 'Check' points to the 'Job State' column, 'OS Upgrade' points to the 'OS Upgrade' button, 'Show Log' points to the 'Show Log' button, 'Delete' points to the 'Delete' button, 'Install' points to the 'Install' button, and 'Uninstall' points to the 'Uninstall' button.

5.3.2 Before Creating Agent

Before creating and installing an agent, make sure of the following:

- The management network has IP connectivity between the devices and the Apstra server.
- Login credentials exist on the devices.
- Any required device packages are uploaded to the AOS controller.
- For off-box agents on Juniper devices, add Junos license configuration. (This is not the preferred method for adding license configuration. For more information, see *Juniper Device Agent*.)

5.3.2.1 On-Box Agent Minimum Configuration

Arista EOS and Cisco NX-OS devices require the below minimal configuration before installing on-box device system agents. Other than Management Network and privileged user access, Cumulus Linux and SONiC have no specific configuration requirements. On-box device system agents are not supported on Juniper Junos devices.

Listing 1: Arista EOS On-box Agent Minimum Configuration

```
!  
service routing protocols model multi-agent  
!  
aaa authorization exec default local  
!  
username admin privilege 15 role network-admin secret <admin-password>  
!  
interface Management1  
    ip address <address>/<cidr>  
!  
ip route vrf management 0.0.0.0/0 <management-default-gateway>  
!
```

Listing 2: Cisco NX-OS On-box Agent Minimum Configuration

```
!  
copp profile strict  
!  
username admin password <admin-password> role network-admin  
!  
vrf context management  
    ip route 0.0.0.0/0 <management-default-gateway>  
!  
interface mgmt0  
    ip address <address>/<cidr>  
!
```

5.3.2.2 Off-Box Agent Minimum Configuration

Juniper Junos, Arista EOS and Cisco NX-OS devices require the below minimal configuration before installing off-box device system agents. Off-box device system agents are not supported on Cumulus Linux and SONiC devices.

Listing 3: Juniper Junos Minimum Configuration for Off-box Agent

```

system {
  login {
    user aosadmin {
      uid 2000;
      class super-user;
      authentication {
        encrypted-password "xxxxxx";
      }
    }
  }
  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>;
      }
    }
  }
}

```

See *Juniper Device Agent* for more information.

Listing 4: Arista EOS Off-box Agent Minimum Configuration

```

!
service routing protocols model multi-agent
!
aaa authorization exec default local
!
username admin privilege 15 role network-admin secret <admin-password>
!
vrf definition management
  rd 100:100
!
interface Management1
  vrf forwarding management
  ip address <address>/<cidr>
!

```

(continues on next page)

(continued from previous page)

```
ip route vrf management 0.0.0.0/0 <management-default-gateway>
!
management api http-commands
  protocol http
  no shutdown
!
  vrf management
    no shutdown
!
```

Listing 5: Cisco NX-OS Off-box Agent Minimum Configuration

```
!
feature nxapi
feature bash-shell
feature scp-server
feature evmed
copp profile strict
nxapi http port 80
!
username admin password <admin-password> role network-admin
!
vrf context management
  ip route 0.0.0.0/0 <management-default-gateway>
!
nxapi http port 80
!
interface mgmt0
  ip address <address>/<cidr>
!
```

5.3.2.3 Uploading Packages

Before creating an agent, upload any *packages* that are to be included in the agent.

Upgrading Packages

Upload a package with an incremented version, and include it in the system agent or agent profile.

5.3.2.4 Retain Pre-existing Configuration

On Cisco NX-OS platforms, you can ensure that pre-existing configuration is retained when a device is brought under AOS management. For details, see [Device AAA support](#).

5.3.3 Creating Agent

5.3.3.1 On-box

If you're creating system agents for multiple devices that share configuration parameters, it is useful to create a *system agent profile*. This allows you to configure the parameters only once.

1. From the web interface, navigate to **Devices > System Agents > Agents** and click **Create Onbox Agent(s)**.

Creating Agent (Version 3.2.0)

Create System Agent(s)

Agent Parameters

Device Addresses (25 max) *

Comma-separated list of hostnames, individual IP addresses, and IP address ranges, e.g. '192.168.1.5-192.168.1.10,mydevice.local'

→

Agent Type *

☒ ONBOX ☐ OFFBOX

Operation Mode

☒ FULL CONTROL ☐ TELEMETRY ONLY

Username

☐ Set username?

Password

☐ Set password?

Agent Profile

Select...

Job to run after creation

☐ Check ☒ Install

Packages 0

Query: All

<

>

Page Size:

25

0

selected

Name

Version

No items

From Agent Profile

Selected agent profile doesn't have Packages

Create

Creating Agent (Version 3.1.1)

Create System Agent(s)

Agent Parameters

Device Addresses (25 max) *
Comma-separated list of hostnames, individual IP addresses, and IP address ranges, e.g. '192.168.1.5-192.168.1.10,mydevice.local'

Agent Type *
☒ ONBOX ☐ OFFBOX

Operation Mode
☒ FULL CONTROL ☐ TELEMETRY ONLY

Username
☐ Set username?

Password
☐ Set password?

Enabled?
☒

Advanced Options

Agent Profile
Select...

Installer Options

Install Requirements?
☐

Intent *
☒ Install
☐ Uninstall

Install Policy *
☐ Automatic
☒ Once

Packages 0

Query: All

Page Size:
25

0 selected

| Name | Version |
|----------|---------|
| No items | |

From Agent Profile
Selected agent profile doesn't have Packages

Create

- Specify *agent details*.
- 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**.

5.3.3.2 Off-box

If you're creating system agents for multiple devices that share configuration parameters, it is useful to create a *system agent profile*. This allows you to configure the parameters only once.

- From the web interface, navigate to **Devices > System Agents > Agents**, click the **OFFBOX** tab, and click **Create Offbox Agent(s)**.
- Specify *agent details*.

352

Chapter 5. Devices

- For **Platform**, as of version 3.3.0, entries are case-sensitive. Enter one of the following: `nxos`, `eos`, `junos`.
- Provide login credentials, OR select a previously created agent profile.
 - If an agent profile is selected, make sure it has the required packages associated to it.
 - If an agent profile is not selected, make sure required packages are uploaded, and check the ones needed.

Create System Agent(s)

Agent Parameters

Device Addresses (25 max) *

Comma-separated list of hostnames, individual IP addresses, and IP address ranges, e.g. '192.168.1.5-192.168.1.10,mydevice.local'



Agent Type *

☐ ONBOX ☒ OFFBOX

Operation Mode

☒ FULL CONTROL ☐ TELEMETRY ONLY

Platform *

Username *

Password *

Agent Profile

Packages 3

| | | |
|-------------------------------------|-------------------------------|---------|
| 1-3 of 3 | | |
| Query: All | | |
| Page Size: 25 | | |
| 3 selected | Name | Version |
| <input checked="" type="checkbox"/> | aos-deployment-helper-nxos | 0.1.0 |
| <input checked="" type="checkbox"/> | aosstdcollectors-builtin-nxos | 0.1.0 |
| <input checked="" type="checkbox"/> | aosstdcollectors-custom-nxos | 0.1.0 |

From Agent Profile

Selected agent profile doesn't have Packages

Open Options 2

| Key | Value | From Agent Profile |
|---------------|-------|--|
| proto | https | Selected agent profile doesn't have Open Options |
| port | 443 | |
| Add an option | | |

- 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**.

5.3.3.3 Agents on L2 Servers

You can install device agents on L2 servers running CentOS or Ubuntu. Telemetry collected from servers is limited compared to the data provided from network devices. Installing an agent on an L2 server follows the same steps as described above with a few specific requirements.

- **Operation Mode** for L2 servers must be **TELEMETRY ONLY** since they do not establish L3 peering with leafs they only provide telemetry data.
- Certain services (such as LLDP) require that additional packages be installed. Check the **Install Requirements** checkbox to install them.

After the agent is running on the device you can add the device to a blueprint and deploy it in the same way as any other device. See *Staging Device Profiles* for details.

Note: Only servers using **eth0** as the management interface are able to connect to the Apstra server. If you require a different naming schema you must install the agent manually. For more information, see *Server Agent* documentation.

5.3.4 Uninstalling Agent

1. From the web interface, navigate to **Devices > System Agents > Agents**.
2. Click the **Uninstall** button for the agent to uninstall.
3. Click **Confirm** to start the uninstall process and return to the list view.

5.3.5 Editing Agent

1. From the web interface, navigate to **Devices > System Agents > Agents**.
2. Click the **Edit** button for the agent to edit.
3. Make your changes.
4. Click **Update** to update the agent and return to the list view.

5.3.6 Deleting Agent

1. From the web interface, navigate to **Devices > System Agents > Agents**.
2. Click the **Delete** button for the agent to delete.
3. Click **Delete** to delete the agent and return to the list view.

5.3.7 Upgrading Device Operating System

5.3.7.1 Device Operating System Upgrade

Device Operating Systems (aka Network Operating Systems, NOS) can be upgraded for the following hardware devices directly from the AOS Web interface:

- Arista EOS (as of AOS 2.3)

- Cisco NX-OS (as of AOS 2.3)
- Cumulus Linux (as of AOS 3.1)
- Enterprise SONiC (as of AOS 3.3.0a)

Important: It is highly recommended this document is read in full detail before proceeding to use this feature.

Note:

- While Junos images can be uploaded to an AOS Server, Device Operating System upgrades are not yet supported for Junos devices.
 - This feature is not supported on virtualized switches.
-

Warning: It is the user's responsibility to ensure they select a compatible Device OS Image for the devices to be upgraded. If the user tries to use an incompatible image and the upgrade fails, a deployment lock will not be released. Even if the user recovers the device, it may be necessary for the user to remove the device assignment from the AOS Blueprint, decommission and normalize the device under **Devices / Managed Devices** then reassign the device to the AOS Blueprint to release the deployment lock and make the device active again. Please contact [Apstra Global Support](#) for assistance.

Warning: When upgrading Device OS, the device has to be under Devices / Managed Devices and the Admin State has to be set as NORMAL. **Admin State MAINT/DECOMM must not be used** for Device OS upgrade as it may put the device into unrecoverable state.

Warning: As part of the upgrade procedure, prior to the actual upgrade the pristine config is restored to the device. This is to ensure there are no configuration conflicts. However, if the upgrade for some reason fails, the device will still be running the Pristine config. Consequently, any subsequent successful upgrade means the device is still only running pristine. Reasons for upgrade failure can be:

- Insufficient disk space
- Incorrect MD5/SHA512 checksum
- Incorrect remote URL

To avoid this, do a full configuration push after the failed upgrade before proceeding to resolve the cause of the failure, and re-try the upgrade. This issue will be fixed in v4.0.1 and later versions.

Apstra will only test certain NOS upgrade paths. These are usually to a recommended Device NOS version in a version of AOS, from the previous recommended Device NOS version in the previous version of AOS, and between recommended NOS version on the same version of AOS.

Supported Device Operating System Upgrades

- **AOS 3.3.0a**
 - NXOS:
 - * From 7.0(3)I7(2), 7.0(3)I7(4) to 9.2(2)

- * From 7.0(3)I7(2), 7.0(3)I7(4) to 7.0(3)I7(7)
 - * From 7.0(3)I7(2), 7.0(3)I7(4), 7.0(3)I7(7) to 7.0(3)I7(8)
- EOS:
 - * From 4.20.11M to 4.21.5.1F or 4.21.9M or 4.22.3M
- Cumulus Linux:
 - * From 3.7.5, 3.7.11, 3.7.12 to 3.7.13
 - * From 3.7.5, 3.7.11 to 3.7.12
 - * From 3.7.5 to 3.7.11
- Enterprise SONiC:
 - * From SONiC 3.1.0-Enterprise_Base to SONiC 3.1.0a-Enterprise_Base
- **AOS 3.3.0**
 - NXOS:
 - * From 7.0(3)I7(2), 7.0(3)I7(4) to 9.2(2)
 - * From 7.0(3)I7(2), 7.0(3)I7(4) to 7.0(3)I7(7)
 - * From 7.0(3)I7(2), 7.0(3)I7(4), 7.0(3)I7(7) to 7.0(3)I7(8)
 - EOS:
 - * From 4.20.11M, 4.21.5.1F, 4.21.9M to 4.22.3M
 - Cumulus Linux:
 - * From 3.7.5, 3.7.11, 3.7.12 to 3.7.13
 - * From 3.7.5, 3.7.11 to 3.7.12
 - * From 3.7.5 to 3.7.11
- **AOS 3.2.4**
 - NXOS:
 - * From 9.2(2) to 9.3(3)
 - * From 7.0(3)I7(7), 7.0(3)I7(8) to 9.3(3)
 - EOS:
 - * From 4.20.11M, 4.21.5.1F, 4.22.3M to 4.23.4.2M
 - * From 4.20.11M, 4.21.5.1F to 4.22.3M
 - Cumulus Linux:
 - * From 3.7.5, 3.7.11, 3.7.12 to 3.7.13
 - * From 3.7.5, 3.7.11 to 3.7.12
 - * From 3.7.5 to 3.7.11
- **AOS 3.2**
 - NXOS:
 - * From 7.0(3)I7(7) to 9.2(2)
 - * From 7.0(3)I7(4) to 7.0(3)I7(7) or 9.2(2)

- EOS:
 - * From 4.20.11M to 4.21.5.1F or 4.22.3M
- Cumulus Linux:
 - * From 3.7.5 to 3.7.11
- **AOS 3.1**
 - NXOS:
 - * From 7.0(3)I7(4) to 9.2(2)
 - EOS:
 - * From 4.20.11M to 4.21.5.1F
 - Cumulus Linux:
 - * From 3.7.4 to 3.7.5
- **AOS 3.0**
 - NXOS:
 - * From 7.0(3)I7(2) to 7.0(3)I7(4)
 - EOS:
 - * From 4.18.4.1F to 4.20.11M

Additional upgrade paths may be available. Contact [Apstra Global Support](#) for more information about supported upgrade paths or to request support for a specific upgrade path.

Warning: If you have manually created/cloned Device Profiles in the blueprints, they will not be updated on AOS upgrade even though the upgraded AOS release supports new device NOS versions. You need to follow the steps in the “Upgrading Device Operating System with Manually Cloned/Created Device Profiles” section below.

Upgrading Device Operating System

After the necessary Device [OS images](#) have been uploaded or registered, you can upgrade the device. You are responsible for managing the state of any deployed devices being upgraded. See [Deploy Modes](#) for more information.

Important: NOS upgrade is not possible if agent’s AOS version is different from controller’s. As of AOS 3.3.0 release, if you try to upgrade NOS with different agent version, it does not show any warning but indicates “in progress” job state without doing anything.

Warning: If you are upgrading a **Cumulus Linux device**, there **must** be a DHCP server on the device Management Network that is configured with a static DHCP assignment for the Cumulus Linux device.

The Cumulus Linux device **must** get the same IP from the DHCP server as before the upgrade. AOS uses the Open Network Install Environment (ONIE) process to add a completely new Cumulus Linux to the device and return it to a factory default state. The eth0 management interface comes up configured for DHCP.

Agents

If the Agent for your device is not on the list and needs to be created see [Agents](#) for instructions.

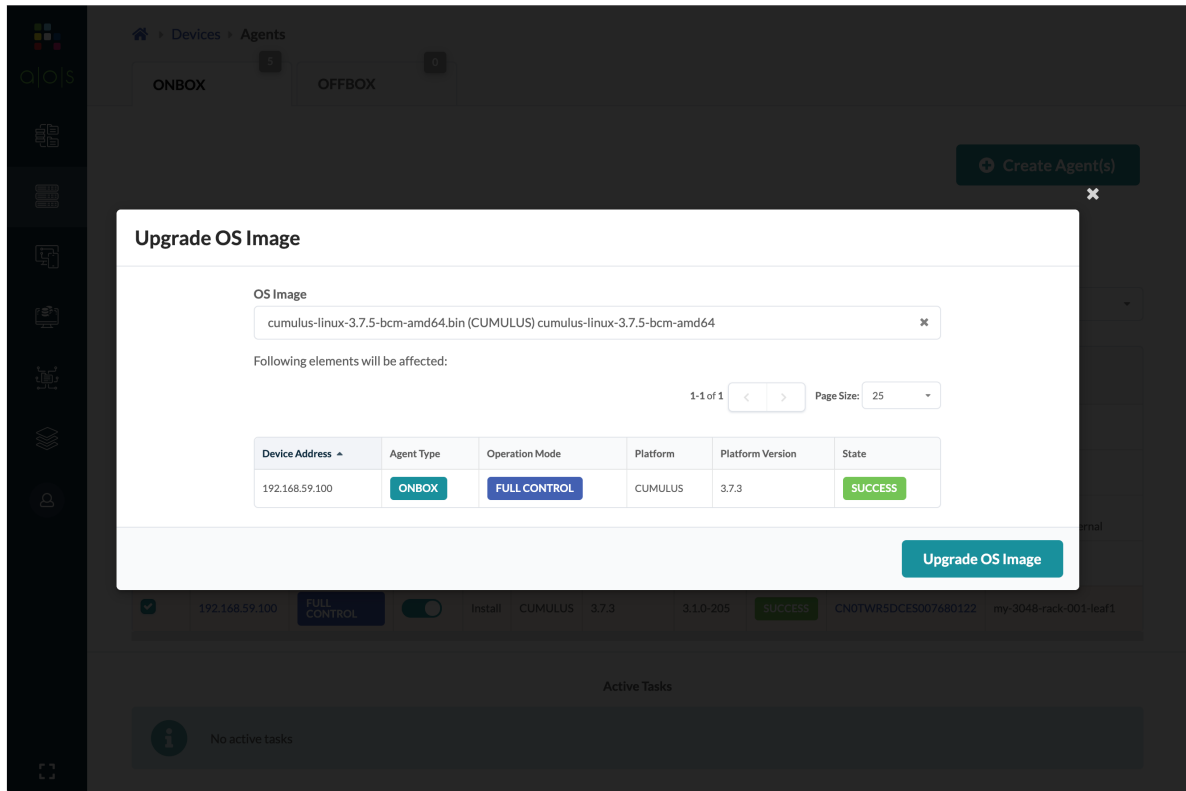
1. Navigate to **Devices / System Agents / Agents** to see the list of existing agents.
2. You can upgrade one or more devices at the same time. All selected devices must be of the same type and must be upgraded to the same image and version. You can use **Search criteria** to search for specific devices (such as all EOS devices). Select the device(s) to upgrade.
3. Click the **DOS Upgrade** button.

The screenshot shows the 'Agents' page in the Apstra interface. The breadcrumb navigation is 'Devices > Agents'. There are two tabs: 'ONBOX' (5 items) and 'OFFBOX' (0 items). A 'Create Agent(s)' button is in the top right. Below the tabs is a search bar with 'Query: All' and pagination controls showing '1-5 of 5' and 'Page Size: 25'. A table lists 5 devices. The 'Actions' column for each device contains several icons, with the 'DOS Upgrade' icon (a person with an arrow) highlighted by a green box in the first row.

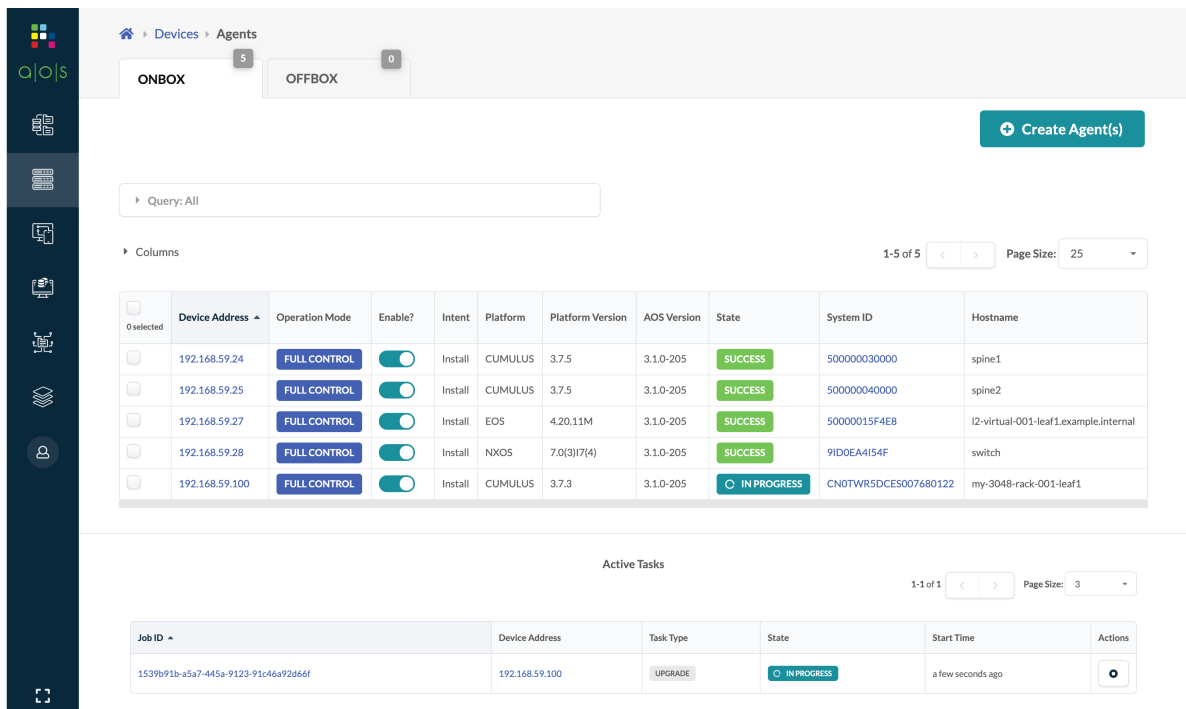
| Device Address | Operation Mode | Platform | Platform Version | AOS Version | Job State | System ID | Hostname | Device State | Actions |
|----------------|----------------|----------|------------------|-------------|-----------|--------------|----------------------|--------------|---|
| 172.20.191.16 | FULL CONTROL | EOS | 4.22.3M | 3.2.0-127 | SUCCESS | 5054000BF029 | I2-virtual-003-leaf1 | IS-ACTIVE | [Icons: Check, Download, Upload, DOS Upgrade , Eye, Refresh, Delete] |
| 172.20.191.15 | FULL CONTROL | CUMULUS | 3.7.9 | 3.2.0-127 | SUCCESS | 525400651bb3 | I2-virtual-001-leaf1 | IS-ACTIVE | [Icons: Check, Download, Upload, DOS Upgrade, Eye, Refresh, Delete] |
| 172.20.191.17 | FULL CONTROL | NXOS | 7.0(3)I7(7) | 3.2.0-127 | SUCCESS | 525400D591C4 | switch | OOS-READY | [Icons: Check, Download, Upload, DOS Upgrade, Eye, Refresh, Delete] |
| 172.20.191.13 | FULL CONTROL | CUMULUS | 3.7.9 | 3.2.0-127 | SUCCESS | 525400460c1e | spine1 | IS-ACTIVE | [Icons: Check, Download, Upload, DOS Upgrade, Eye, Refresh, Delete] |
| 172.20.191.14 | FULL CONTROL | EOS | 4.22.2F | 3.2.0-127 | SUCCESS | 5054009CDB33 | spine2 | IS-ACTIVE | [Icons: Check, Download, Upload, DOS Upgrade, Eye, Refresh, Delete] |

Active Jobs: 0

4. The **Upgrade OS Image** dialog lists the available Device OS Images that match the selected devices. Select the appropriate OS Image and click **Upgrade OS Image** to upload the new image to the device.



5. You can monitor the upgrade status from the **Active Jobs** section at the bottom of the **Agents** page .



Note: For Cumulus Linux:

- The local device username and password that was used to install the AOS Device System Agent is recre-

ated.

- The same Cumulus Linux license is reinstalled.
- The Cumulus Linux license is reinstalled and the AOS configuration and deployment state is restored.

6. If a checksum is provided with the DOS image, AOS checks the image checksum after the upload is complete. If the checksum is incorrect, or if any other failures occur, the job state changes to **FAIL** and the device does not reboot. 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.
7. If the checksum is correct and no other failures occur, the job state changes to **SUCCESS** and the device reboots.
8. When the device has rebooted with the new image and has reestablished its Agent connection with the AOS Controller, the upgrade is complete. The **Agents** page displays the new Device OS version.

The screenshot displays the 'Agents' page in the Apstra AOS interface. The page has a sidebar on the left with navigation icons. The main content area shows a table of agents. The table has columns for Device Address, Operation Mode, Enable?, Intent, Platform, Platform Version, AOS Version, State, System ID, and Hostname. There are 5 agents listed, all with a 'SUCCESS' state. Below the table, there is an 'Active Tasks' section which shows 'No active tasks'.

| Device Address | Operation Mode | Enable? | Intent | Platform | Platform Version | AOS Version | State | System ID | Hostname |
|----------------|----------------|---------|---------|----------|------------------|-------------|---------|----------------------|---------------------------------------|
| 192.168.59.24 | FULL CONTROL | ON | Install | CUMULUS | 3.7.5 | 3.1.0-205 | SUCCESS | 500000030000 | spine1 |
| 192.168.59.25 | FULL CONTROL | ON | Install | CUMULUS | 3.7.5 | 3.1.0-205 | SUCCESS | 500000040000 | spine2 |
| 192.168.59.27 | FULL CONTROL | ON | Install | EOS | 4.20.11M | 3.1.0-205 | SUCCESS | 50000015F4E8 | I2-virtual-001-leaf1.example.internal |
| 192.168.59.28 | FULL CONTROL | ON | Install | NXOS | 7.0(3I7/4) | 3.1.0-205 | SUCCESS | 9ID0EA4I54F | switch |
| 192.168.59.100 | FULL CONTROL | ON | Install | CUMULUS | 3.7.5 | 3.1.0-205 | SUCCESS | CN0TWR5DCES007680122 | cumulus |

Upgrading Device Operating System with Manually Cloned/Created Device Profiles

AOS built-in Device Profiles are managed by Apstra, and the changes in the built-in DP are automatically propagated to the imported Interface Maps in the Blueprints during the AOS upgrade. For example, AOS release 3.2.4 now supports Arista EOS 4.23 and the AOS 3.2.4 built-in DPs have the OS Version selector like “4.(18|20|21|22|23)”, although AOS 3.2.2 built-in DPs have “4.(18|20|21|22)”. During the AOS upgrade, the Interface Maps in the BPs which refer to the built-in DPs are also automatically updated so that the user can upgrade Arista devices to EOS version 4.23 after the AOS upgrade.

However, the manually created or cloned DPs are not managed by Apstra, and not updated during the AOS upgrade, e.g. as below “Clone DCS-7160-48YC6_abc”.

Query: Name == "DCS-7160-48YC6" 1-2 of 2 Page Size: 25

| Name | Manufacturer | Hardware Model | Modular? | OS Family | OS Version | Actions |
|--------------------------|--------------|----------------|----------|-----------|-------------------------|-------------------------|
| Arista DCS-7160-48YC6 | Arista | DCS-7160-48YC6 | no | EOS | 4\\(18 20 21 22 23)\\.* | [edit] [clone] [delete] |
| Clone DCS-7160-48YC6_abc | Arista | DCS-7160-48YC6 | no | EOS | 4\\(18 20 21 22 23)\\.* | [edit] [clone] [delete] |

If you have manually cloned/created DPs and want to upgrade the OS Version selector, you have to do the following steps.

1. Update the DP's OS Version selector via AOS UI.

Edit Device Profile

Device profiles need to accurately model various characteristics of a switch model. Make sure you update the profile to match the new switch model(s) you intend to use this profile for.

Updating the device profile ports may not be allowed because it is referenced by `Arista_DCS-7160TC_EOS_AOS-48x10_6x100-1` interface map.

Summary
Selector²
Capabilities
Ports

Manufacturer *

Arista

Model *

DCS-7160-48TC6

OS family *

EOS

Version *

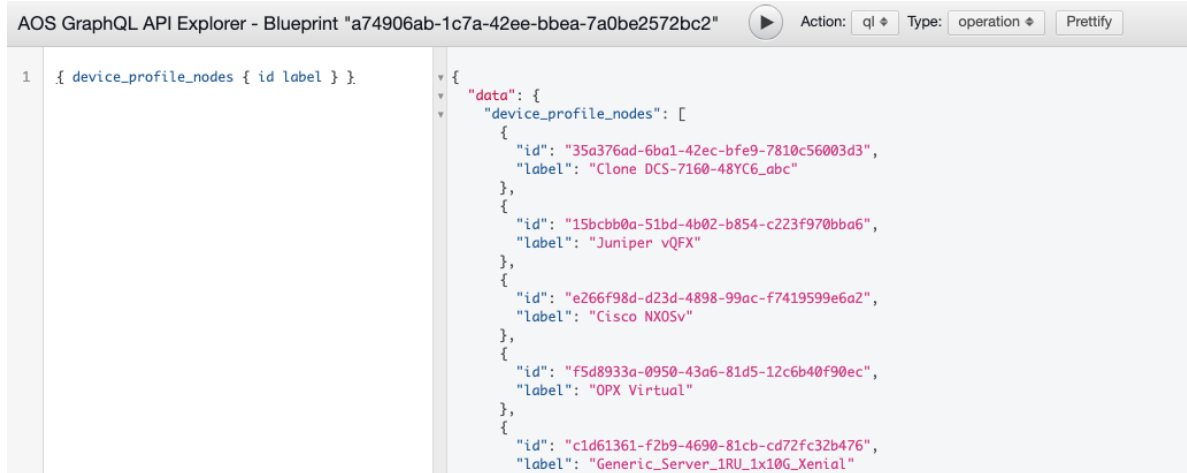
4\\(18|20|21|22|23)\\.*

4\\(18|20|21|22|23)\\.*

Update

2. Find the `device_profile_nodes` ID by using the AOS GraphQL API Explorer.

Query Variables " { device_profile_nodes { id label } } " can be used. In this example, find the "id" for the label "Clone DCS-7160-48YC6_abc" which is "35a376ad-6ba1-42ec-bfe9-7810c56003d3".



3. Update the Device Profile used in the BP via AOSCLI.

You can use your BP ID, also use the node ID from step2, and set the proper model ID (e.g. “DCS-7160-48YC6”), and then execute.

AOS-CLI command format:

```
blueprint set-node-property --blueprint <your blueprint ID> --node_type
device_profile --node <node ID from Step2> --property selector
--value-fn '{"os_version":"4\\.(18|20|21|22|23)\\.\\.*","model":"<your model>"
,"os": "EOS","manufacturer": "Arista"}'
```

Example:

```
aos> blueprint set-node-property --blueprint
a74906ab-1c7a-42ee-bbea-7a0be2572bc2 --node_type device_profile
--node 35a376ad-6ba1-42ec-bfe9-7810c56003d3 --property selector
--value-fn '{"os_version":"4\\.(18|20|21|22|23)\\.\\.*","model":"DCS-7160-48YC6",
"os": "EOS","manufacturer": "Arista"}'
```

4. Commit the changes via AOS UI.

The screenshot shows the AOS UI interface. The top navigation bar includes 'Blueprints', 'L2 Virtual EVPN', 'Uncommitted', and 'Full Nodes Diff'. The main content area shows a comparison between 'Staged' and 'Active' states. A warning message states: 'Long Operation: Calculating full diff between staging and active graphs is an expensive operation and might take a while'. Below this, there is a 'Calculate Full Diff' button. The bottom section displays a table with the following data:

| Node ID | Node Type | Action | Staged Value | Active Value |
|--------------------------------------|----------------|---------|---|---|
| 35a376ad-6ba1-42ec-bfe9-7810c56003d3 | device_profile | CHANGED | { "selector": { "os_version": "4\\.(18 20 21 22 23)\\.\\.*" } } | { "selector": { "os_version": "4\\.(18 20 21 22 23)\\.\\.*" } } |
| ba92af7a-a257-4b3f-932d-acc40a8a7e11 | metadata | CHANGED | see value | see value |

After the changes are committed, you should be able to upgrade the device operating system in the same manner as the devices associated with AOS built-in Device Profiles.

5.3.8 Vendor Specific Agent Information

5.3.8.1 Juniper Device Agent

This document describes how to manually install Juniper device agents.

Juniper ZTP

Note: For an option that's simpler and easier to support at scale, see [Apstra ZTP Documentation](#), which shows you how to automatically boot and install Apstra device agents and prerequisite switch configuration.

Disabling ZTP

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`.

Listing 6: Disabling ZTP

```
{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>
```

Initial Juniper Junos Configuration

Juniper Junos devices require the below minimal configuration before installing Apstra device system agents.

Listing 7: Juniper Junos Minimum Configuration for Off-box Agent

```
system {
  login {
    user aosadmin {
      uid 2000;
      class super-user;
```

(continues on next page)

(continued from previous page)

```

        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>;
            }
        }
    }
}
}

```

Configuring 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`.

Listing 8: Configuring super-user 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

```

(continues on next page)

(continued from previous page)

```
Exiting configuration mode

{master:0}
root>
```

Note: If you intend to use another authentication method for device access (e.g. RADIUS), you must use local password authentication first.

```
system authentication-order [ password radius ]
```

Configuring IP address and Management VRF

Device system agents use the Junos `mgmt_junos` management-instance VRF and the management interface (e.g. `em0`).

Listing 9: Configuring IP address and Management VRF

```
{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 another management interface (such as `vme.0`), configure the management IP address on it instead.

Configure SSH and NETCONF

Device system agents require Junos SSH and NETCONF access to be configured under `system services`.

Listing 10: Configure SSH and Netconf

```
{master:0}
root> edit
```

(continues on next page)

(continued from previous page)

```
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*.

5.3.8.2 Arista Device Agent

Initial Arista EOS Configuration

Note: Please see *Apstra ZTP Documentation* for an option for automatically booting and installing the AOS Device agent and prerequisite configuration on switches. This option is simpler and easier to support at scale than manually installing agents.

Disabling ZTP

If the switch is to be provisioned without ZTP (ZTP Disabled), we need to ensure that the ZTP process is disabled before proceeding. After logging into the switch for the first time, run the command *zerotouch disable*.

This will require a device reload.

Listing 11: Disabling ZTP

```
localhost login: admin
localhost> zerotouch disable
```

Configuring AAA and network-admin User

In order to install or manage the AOS System Agent from AOS, there must be a network-admin user configured on the device with a known password.

```
aaa authorization exec default local
username admin privilege 15 role network-admin secret <admin-password>
```

Configuring IP address and Management VRF

Apstra AOS Device Agent makes use of the ‘management’ VRF. Accordingly, any management interfaces must be moved from the default (none) VRF into the Management VRF.

Note: If the user is installing a on-box AOS Device Agent from AOS, the creation of the Management VRF can be skipped. The AOS Device Agent installer will automatically create the Management VRF if needed.

The device agent makes use of the *Management1* interface by default. On modular chassis such as the Arista 7504 or 7508, the management interface is Management0 - please check your platform if management interfaces appear as “Management1” or “Management1/1, Management1/2, and Management0”. Management0 is a shared mgmt interface between both supervisors

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 will immediately remove the IP address from the NIC and potentially lock 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>
```

Configuring DNS for EOS

AOS server discovery supports DNS-based discovery if the user is manually configuring the the AOS Device System Agent. By default, the `aos-config` file will look for `tbt://aos-server:29731` - accordingly, we can use a DNS nameserver to resolve `aos-server`.

Listing 12: DNS for EOS

```
ip name-server vrf management <dns-server-ip>
ip name-server vrf management <dns-server-ip>
```

Configure HTTP API for EOS

Note: If the user is installing a on-box AOS Device Agent from AOS, the creation of the configuration of HTTP API can be skipped. The AOS Device Agent installer will automatically configure the HTTP API if needed.

AOS makes use of the HTTP API and Unix sockets 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.

Additionally, the AOS agent connects using the unix-socket locally on the filesystem.

Listing 13: Configure HTTP API for AOS connectivity

```
management api http-commands
  protocol unix-socket
  no shutdown
  vrf management
  no shutdown
```

Configure multi-agent for EVPN

For Arista devices running EOS 4.22, the `service routing protocols model multi-agent` configuration command is required to run EVPN. The configuration also requires a device reboot to apply the configuration.

```
localhost(config)#service routing protocols model multi-agent
! Change will take effect only after switch reboot
localhost(config)#
```

Apstra recommends that this configuration is added to Arista device prior to the installation of the AOS Device System Agent. This will ensure the multi-agent configuration is added to the Device Pristine Configuration.

After configuring this, 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

Find the device under Devices, select it, and press the Decommission button in the UI. This will allow us to delete the device.

Delete the device from the Devices list Once the device is decommissioned and 'available' we can delete it.

Remove the AOS package from EOS

Uninstall the AOS Device agent package using the EOS CLI

Erasing the startup-configuration does not delete the installed EOS Extension files. We must explicitly remove the AOS device agent.

These steps must be done 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 the AOS Device agent using bash

Using the Bash CLI requires the administrator to edit `/mnt/flash/boot-extensions` to remove the reference to the extension, and deleting the extension from `/mnt/flash/.extensions/aos-device-agent.i386.rpm` - This filename will be unique depending on the version that AOS was installed

Listing 14: View the AOS Agent extension installation details

```
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
```

Listing 15: Remove the AOS device agents

```
[admin@localhost ~]$ vi /mnt/flash/boot-extensions
```

Remove remaining AOS data from system

AOS modifies and retains information on the filesystem from a few locations, these can be cleaned up by hand by running the below:

Warning: If AOS files are not removed (especially `/mnt/flash/.aos/` which includes checkpoint files), the next time AOS is installed, the configuration will be replaced by whatever configuration AOS rendered last – including quarantine configuration, which could shut down all interfaces.

Be very sure to remove `/mnt/flash/.aos/` when removing AOS data.

Listing 16: Remove AOS data from the filesystem

```
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 the config file

In order for the extension to be removed from bootup, we need to `wr mem` to ensure the extension no longer appears in `boot-extensions`, or else AOS may start the agent up again if the RPM is still installed in available extensions.

Restart the system

After AOS is uninstalled we need to reboot.

Note: Say ‘yes’ to configuration saving, this will ensure the extension is removed from boot extensions.

```
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 2017):

The system is going down for reboot NOW!
```

The configuration running on the switch at the time of removing the AOS agent will not be modified or changed in any way, it is completely safe to remove AOS without disrupting the network.

Arista Device Agent Manual Installation

This describes the process of **manual** Agent Installation on Arista EOS devices. For the recommended method using the UI, refer to *Device Agents*

Important: Only in rare exceptions is it needed to follow the manual process of Agent installation. In almost all cases agents should be installed by creating System Agents in the UI. Installing Agents manually is more bespoke, more effort and prone to user error.

In-depth understanding of the various device states, configuration stages and Agent operation is required when attempting manual Agent installation.

When in doubt, contact *Apstra Global Support*.

Download Agent Installer

The AOS Device agent is available over HTTPs from the AOS Server from the base URL https://aos-server/device_agent_images/aos_device_agent.run

Listing 17: Downloading the AOS Agent

```
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

Listing 18: Run the AOS Device agent installer

```
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 .
```

(continues on next page)

(continued from previous page)

```

++ pwd
+ script_dir=/tmp/selfgz726322812
++ date
+ echo 'Device Agent Installation : Wed' Oct 18 20:34:11 UTC 2017
Device Agent Installation : Wed Oct 18 20:34:11 UTC 2017
+ echo

+ UNKNOWN_PLATFORM=1
+ WRONG_PLATFORM=1
+ CANNOT_EXECUTE=126
+ '[' 0 -ne 0 ']'
+ arg_parse
+ start_aos=True
+ [[ 0 > 0 ]]
+ supported_platforms=(["centos"]="install_sysvinit_rpm" ["eos"]="install_on_arista"
↪["nxos"]="install_on_nxos" ["cumulus"]="install_sysvinit_deb" ["trusty"]="install_
↪sysvinit_deb" ["icos"]="install_sysvinit_rpm" ["snaproute"]="install_sysvinit_deb" [
↪"simulation"]="install_sysvinit_deb")
+ declare -A supported_platforms
++ /tmp/selfgz726322812/aos_get_platform
+ current_platform=eos
+ installer=install_on_arista
+ [[ -z install_on_arista ]]
+ [[ -x /etc/init.d/aos ]]
+ echo 'Stopping AOS'
Stopping AOS
+++ readlink /sbin/init
++ basename upstart
+ [[ systemd == upstart ]]
+ /etc/init.d/aos stop
+ install_on_arista
++ pwd
+ local pkg_dir=/tmp/selfgz726322812/arista
+ local to_be_installed=
+ local flash_dir_from_bash=/mnt/flash/aos-installer
+ local flash_dir_from_cli=flash:/aos-installer
+ cp aos_device_agent.img /mnt/flash/
+ mkdir -p /mnt/flash/aos-installer
++ ls /mnt/flash/.extensions/aos-device-agent-2.0.0-0.1.138.i386.rpm
+ existing_aos=/mnt/flash/.extensions/aos-device-agent-2.0.0-0.1.138.i386.rpm
+ for aos_rpm in '${existing_aos}'
++ basename /mnt/flash/.extensions/aos-device-agent-2.0.0-0.1.138.i386.rpm
+ ip netns exec default FastCli -p15 -c 'no extension aos-device-agent-2.0.0-0.1.138.
↪i386.rpm'
++ basename /mnt/flash/.extensions/aos-device-agent-2.0.0-0.1.138.i386.rpm
+ ip netns exec default FastCli -p15 -c 'delete extension:aos-device-agent-2.0.0-0.1.
↪138.i386.rpm'
+ pushd /tmp/selfgz726322812/arista
/tmp/selfgz726322812/arista /tmp/selfgz726322812
++ ls aos-device-agent-2.0.0-0.1.138.i386.rpm
+ aos_rpm=aos-device-agent-2.0.0-0.1.138.i386.rpm
+ cp aos-device-agent-2.0.0-0.1.138.i386.rpm /mnt/flash/aos-installer
+ ip netns exec default FastCli -p15 -c 'copy flash:/aos-installer/aos-device-agent-
↪2.0.0-0.1.138.i386.rpm extension:'
Copy completed successfully.
+ ip netns exec default FastCli -p15 -c 'extension aos-device-agent-2.0.0-0.1.138.
↪i386.rpm force'

```

(continues on next page)

(continued from previous page)

```
+ 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
```

AOS Device Agent Configuration File

The Arista device agent manages the running-configuration file. No other configuration files are modified throughout the device agent lifecycle. Configuration can be managed by editing the device agent configuration file directly. The Arista EOS device agent config file is located at `/mnt/flash/aos-conf`. See [AOS Device Agent Configuration file](#) for parameters. After updating the file, restart the AOS device agent.

Listing 19: Restart AOS Device Agent

```
localhost# bash sudo systemctl stop aos
localhost# bash sudo systemctl start aos
```

Arista Agent Troubleshooting

AOS Log files

AOS logs to a number of files in `/var/log/aos`.

Confirming installation of AOS Agent Check if the AOS device agent package is installed.

Listing 20: Show RPM Data

```
-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.idboiseeossdk
      Presence: available
      Status: installed
      Vendor:
```

(continues on next page)

(continued from previous page)

```

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

```

Check if the AOS Agent is running

Listing 21: The AOS services must be running

```

localhost#bash sudo service aos status
AOS is running

```

Listing 22: Check for presence of files in flash

```

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)

```

Listing 23: Check for AOS data in /var/log/aos

```

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

```

(continues on next page)

(continued from previous page)

```

-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

AOS 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.

Listing 24: Check for DNS failures

```

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)

```

(continued from previous page)

```
[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)
```

Listing running processes

AOS processes on an Arista switch run alongside other management components on the switch. List the services with the 'ps wax' command.

Listing 25: List processes

```
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/udevd -d |
| 1374 | ? | S | 0:01 | [kworker/0:2] |
| 1471 | ? | S< | 0:00 | /sbin/udevd -d |
| 1730 | ? | S | 0:00 | python /usr/bin/immortalize --daemonize --log=/var/log/ |

```
↪agents/ConnMgr --logpidsuffix --maxcredits=5 --cos
```

(continues on next page)

(continued from previous page)

```

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

```

(continues on next page)

(continued from previous page)

```

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

```

(continues on next page)

(continued from previous page)

```

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) -

```

(continues on next page)

(continued from previous page)

```

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=

```

(continues on next page)

(continued from previous page)

```

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 ?      S1      0:00 tacspawner --daemonize=/var/log/aos/aos.log --pidfile=/var/
↳run/aos.pid --name=000C29E808A1 --hostname=000
4602 ?      S      0:00 tacleafsysdb --agentName=000C29E808A1-LocalTasks-
↳000C29E808A1-0 --partition= --storage-mode=persistent --
4606 ?      S1      0:00 /usr/bin/python /usr/bin/aos_agent --class=aos.device.
↳common.DeviceKeeperAgent.DeviceKeeperAgent --name=D
4629 ?      S1      0:00 /usr/bin/python /usr/bin/aos_agent --class=aos.device.
↳common.ProxyDeploymentAgent.ProxyDeploymentAgent --
4670 ?      S1      0:00 /usr/bin/python /usr/bin/aos_agent --class=aos.device.
↳arista.AristaTelemetryAgent.AristaTelemetryAgent --
4684 ?      S1      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 ?      Ss1     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 ?      Ss1     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

```

(continues on next page)

(continued from previous page)

```
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 EOS Installation

When attempting to install the Arista EOS Device agent, you may receive a ‘connection refused’ error. This means that A) The SDK is not running, and B) the unix-socket is not listening, or C) you are attempting to run the AOS Installer in the management VRF.

Listing 26: Connection refused when running FastCli commands

```
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
```

To resolve this, Switch routing-contexts to ‘default’.

5.3.8.3 Cisco NX-OS Device Agent

This chapter describes the process of **manual** Agent Installation on Cisco NXOS devices. For the recommended method using the UI, refer to *Device Agents*

Important: Only in rare exceptions is it needed to follow the manual process of Agent installation. In almost all cases agents should be installed by creating System Agents in the UI. Installing Agents manually is more bespoke, more effort and prone to user error.

In-depth understanding of the various device states, configuration stages and Agent operation is required when attempting manual Agent installation.

When in doubt, contact *Apstra Global Support*.

Quick start

Manual installation of the Agent involves the following steps:

- Modifying the guestshell disk size, memory and cpu, as well as restarting the guestshell in order to take effect.
- Copying the device agent from the AOS Server and installing it.
- Modifying the aos config file.

Warning: The Cisco GuestShell is not partitioned to be unique with AOS. If there are other applications hosting on the guestshell, any changes in the guestshell could impact them.

Warning: Commands in the “Bootstrap” or “Pristine” configuration may interfere with configuration added by AOS during fabric deployment.

Adding NX-OS configuration “system jumbomtu” with a value lower than MTUs used by AOS will cause AOS MTU commands to fail.

Device configuration requirements

Configuration steps must happen in order on NX-OS - VRF, NXAPI, GuestShell, Create Management VRF. Apstra’s AOS Device agent requires the use of VRF of the name `management` to allow for agent-server communication. Ensure these lines appear in the running configuration.

```
!
no password strength-check
username admin password admin-password role network-admin
copp profile strict
!
vrf context management
  ip route 0.0.0.0/0 <Management Default Gateway>
!
interface mgmt0
  vrf member management
  ip address <Management CIDR Address>
!
```

Resize and Enable the Guestshell

Either the guestshell is running or not restarting or enabling the service is required after the the following step.

Resize the guestshell disk space, memory and cpu by executing the next commands:

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

If the guestshell is not enable, proceed to activate it by executing “guestshell enable”, otherwise, if it was already running please run “guestshell reboot” command in order to restart the shell.

Verify that the guestshell is activated again:

```
switch# show guestshell detail
```

Download Agent Installer

We can easily copy the installation agents over HTTPS from the AOS server. After downloading, please confirm the MD5sum of your downloaded copy matches what AOS stores.

Note: The Cisco device needs to connect to the AOS Server using HTTPS in order to retrieve the agent file, please make sure that this connectivity is OK before proceeding.

Apstra ships the AOS agent from the AOS Server. We can copy it to the `/volatile`, or `volatile:` filesystem location. AOS also ships with an `md5sum` file in the `/home/admin` folder on the AOS Server.

Replace the `aos_server_ip` variable and `aos_version` from the run file below, you can find this exact version from the AOS Server, Platform → About (i.e '3.2.2-12')

```
switch# guestshell run sudo chvrf management wget --no-check-certificate -o /volatile/
↪aos_download.log
-O /volatile/aos.run https://<aos_server_ip>/device_agent_images/aos_device_agent_
↪<aos_version>.run

guestshell run sudo chvrf management wget --no-check-certificate -o /volatile/aos_
↪download.log
-O /volatile/aos.run.md5 https://<aos_server_ip>/device_agent_images/aos_device_agent_
↪<aos_version>.run.md5
```

Validate that the file was downloaded correctly.

```
switch# show file volatile:aos.run md5
a28780880a8d674f6eb6a397509db101

switch# show file volatile:aos.run.md5
a28780880a8d674f6eb6a397509db101  aos_device_agent_<aos_version>.run
```

Install Cisco Device Agent

The AOS agent on Cisco is simply installed by running it as a shell script directly as root on the Cisco NXOS switch. This command must be done within the guest shell. After installing the agent and before starting the service, `aos.conf` file needs to be modified to connect to the server.

Note: It is recommended to save your current running-config to the startup-config 'copy running-config startup-config' to save your latest changes in case of any issue.

```
switch# guestshell run sudo chmod +x /volatile/aos.run
switch# guestshell run sudo /volatile/aos.run -- --no-start
<omitted output>
created 7855 files
created 1386 directories
created 602 symlinks
created 0 devices
created 0 fifos
+ [[ True == \T\r\u\e ]]
+ true
+ systemctl enable aos
```

Change the required parameters in the AOS configuration file before enabling the AOS service (see next steps).

AOS Device Agent Configuration File

Device agent configuration can be managed by editing the device agent configuration file directly. The Cisco NX-OS device agent config file is located at `/etc/aos/aos.conf`. See *AOS Device Agent Configuration file* for parameters. After updating the file, start the AOS device agent.

```
service aos start
```

Activating AOS Devices on the AOS Server

When the AOS Device agent communicates with AOS, it uses a 'device key' to identify itself. In the case of A Cisco NXOS switch, 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 the device

Once the Agent is up and running it will appear under Managed Devices, and can be Acknowledged and assigned to a Blueprint using the UI as normal.

Resetting the AOS agent

If you need to reset the AOS agent for some reason (changing blueprints, redeploying, restoring device from backup, etc) it is best to clear the AOS 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 ~]#
```

Uninstalling the AOS device agent

To uninstall the agent, first Undeploy and Unassign it from the blueprint as per standard procedures using the UI. It can also be deleted entirely from the Managed Devices page.

To remove the AOS package from NX-OS we can destroy the guestshell. This should only be done if no other applications are making use of 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 AOS EEM Scripts

The AOS 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

The AOS Agent runs under the NXOS guestshell to interact with the underlying bash and linux environments. This is an internal Linux Container (LXC) in which AOS operates. Under LXC, AOS 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 AOS 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 AOS 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
```

(continues on next page)

(continued from previous page)

Virtual Service List:

| Name | Status | Package Name |
|-------------|-----------|----------------|
| ----- | ----- | ----- |
| guestshell+ | Activated | guestshell.ova |

Confirm network reachability to AOS

Check ICMP Ping to the AOS Server by pinging within the guest shell. On NXOS, we have to use the ‘chvrf <vrf>’ command to run commands within the context of a VRF. In this case, ‘management’ VRF.

```
[guestshell@guestshell ~]$ chvrf management ping 172.20.65.3
PING 172.20.65.3 (172.20.65.3) 56(84) bytes of data.
64 bytes from 172.20.65.3: icmp_seq=1 ttl=64 time=0.239 ms
64 bytes from 172.20.65.3: icmp_seq=2 ttl=64 time=0.215 ms
```

Confirm agent installation

Check if the AOS device agent package is installed. In NXOS, the AOS 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 if the AOS 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
```

(continues on next page)

(continued from previous page)

```
[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/ssh_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   tacleafsadb --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, AOS 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 AOS data in /var/log/aos

AOS 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 root 0 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 root 0 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 root 0 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 root 0 Nov 15 00:10 C9K-0.115.1479168649.log
-rw-r--r-- 1 root root 0 Nov 15 00:10 C9K-0.err
-rw-r--r-- 1 root root 0 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 root 0 Nov 15 00:10 _C9K-00000000582a528a-0001744b-checkpoint-valid
-rw----- 1 root root 0 Nov 15 00:10 _C9K-00000000582a528a-0001744b-log
-rw-r--r-- 1 root root 0 Nov 15 00:10 _C9K-00000000582a528a-0001744b-log-valid
-rw-r--r-- 1 root root 0 Nov 15 00:10 aos.log
[root@guestshell aos]#
```

Determining AOS Agent version

The AOS 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

AOS 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
```

AOS Service takes long time to start on Cisco NXOS

The GuestShell feature on Cisco NXOS takes a few minutes to initialize the NXAPI within the LXC container. Apstra AOS does not have control over this to make it any faster. Apstra Engineering has added a wait-delay to the initialization of the AOS scripts to account for this delay. This wait is normal.

AOS stops and fails without any errors (MGMT VRF)

Please ensure that the guestshell is properly behind management VRF.

We should not be able to ping the AOS server when running 'ping' command by default:

Below - we expect a ping from global default routing table to AOS 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
```

(continues on next page)

(continued from previous page)

```
64 bytes from 172.20.156.3: icmp_seq=4 ttl=63 time=0.404 ms
^C
```

Verify MGMT VRF in NXOS 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
```

Contact Apstra Global Support

Apstra Global Support is available to assist with troubleshooting. Diagnostic information will most likely be needed to help resolve issues. Please see the [Apstra Global Support](#) page for details.

5.3.8.4 Cumulus Device Agent

This chapter describes the process of **manual** Agent Installation on Cumulus devices. For the recommended method using the UI, refer to [Device Agents](#)

Important: Only in rare exceptions is it needed to follow the manual process of Agent installation. In almost all cases agents should be installed by creating System Agents in the UI. Installing Agents manually is more bespoke, more effort and prone to user error.

In-depth understanding of the various device states, configuration stages and Agent operation is required when attempting manual Agent installation.

When in doubt, contact [Apstra Global Support](#).

Quick start

The below section is a quick-start guide on installing the Cumulus device agent. The contents of this document will describe this in detail.

1. Configure management IP and VRF

```
admin@cumulus:mgmt-vrf:~$ vi /etc/network/interfaces
```

```
auto mgmt
iface mgmt
    address 127.0.0.1/8
    vrf-table auto
    post-up sudo service aos start
```

(continues on next page)

(continued from previous page)

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

auto eth0
iface eth0 inet dhcp
    vrf mgmt
```

```
admin@cumulus:mgmt-vrf:~$ ifreload -a
```

2. Activate cumulus license

```
admin@cumulus:mgmt-vrf:~$ cl-license -i user@domain|cumulus-license-key
admin@cumulus:mgmt-vrf:~$ service switchd restart
```

3. Download AOS agent

```
admin@cumulus:mgmt-vrf:~$ sudo vrf task exec mgmt wget -nv \
--no-check-certificate \
https://aos-server/device_agent_images/aos_device_agent.run
```

4. Set IP of AOS server and enable configuration service

```
admin@cumulus:mgmt-vrf:~$ sudo vi /etc/aos/aos.conf
```

```
[controller]
metadb = tbt://aos-server:29731

[service]
enable_configuration_service = 1
```

5. Restart AOS agent

```
admin@cumulus:mgmt-vrf:~$ sudo service aos stop
admin@cumulus:mgmt-vrf:~$ sudo service aos start
```

6. Approve device in AOS

7. Assign device to blueprint

Cumulus Initial Configuration

The cumulus device agent requires some configuration prior to being used with the AOS software.

Hint: In the below installation instructions, Apstra has added a new local username **admin** with password **admin** as part of our provisioning process. By default, the Cumulus credentials are username **cumulus** and password **CumulusLinux!** - AOS does not depend on the username to be changed.

CumulusVX

If deploying CumulusVX, (Virtual appliance), ensure the virtual switch is given at least 1 vCPU and 2GB of ram. Cumulus VX 3.1.1 OVA template only provides for 512MB of ram. This will need to be increased.

Apstra has tested using the *virtio* and *e1000* network driver types on our hypervisors.

Management interface

Warning: The Cumulus device agent requires a management VRF to be set up before running the AOS device installer. The ‘bash’ shell must also be running under the context of the management VRF prior to installation to ensure the AOS agent can communicate to the AOS server

By default, cumulus does not have a management VRF activated.

```
root@cumulus:~# cat /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*.intf

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet dhcp
```

Add a new *auto mgmt* interface as a VRF and activate the eth0 interface for *vrf mgmt*.

```
root@cumulus:~# vi /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*.intf

# The loopback network interface
auto lo
iface lo inet loopback

auto mgmt
iface mgmt
    address 127.0.0.1/8
    vrf-table auto
    post-up sudo service aos start

auto eth0
iface eth0 inet dhcp
    vrf mgmt
```

After the VRF is activated, reload the network file with *ifreload -a*, and log back into the switch afterwards. This will ensure the Bash prompt is under the management VRF routing context. Pay particular note to the new bash prompt, *admin@cumulus:mgmt-vrf:~\$* which indicates bash is running under the mgmt-vrf context.

```
root@cumulus:/etc/network# ifreload -a
<reconnect with SSH>
Welcome to Cumulus VX (TM)
```

(continues on next page)

(continued from previous page)

```
Cumulus VX (TM) is a community supported virtual appliance designed for
experiencing, testing and prototyping Cumulus Networks' latest technology.
For any questions or technical support, visit our community site at:
http://community.cumulusnetworks.com
```

```
The registered trademark Linux (R) is used pursuant to a sublicense from LMI,
the exclusive licensee of Linus Torvalds, owner of the mark on a world-wide
basis.
```

```
Last login: Fri May 26 12:41:13 2017 from 192.168.25.1
```

```
admin@cumulus:mgmt-vrf:~$
```

Installing a cumulus license

On most platforms, a license needs to be installed for the switch to run properly. If a license is not installed, AOS may experience deployment failures because the command `ifreload -a` will fail because the `switchd` service isn't running.

The Apstra ZTP server can pre-provision a license file for cumulus. This license can also be installed manually.

If a license isn't installed, and this is not a cumulus VX platform, the command `cl-license` will fail.

```
admin@cumulus:mgmt-vrf:~$ sudo service switchd start
Job for switchd.service failed. See 'systemctl status switchd.service' and
↳ 'journalctl -xn' for details.

admin@cumulus:mgmt-vrf:~$ cl-license
1508122225.507863 2017-10-16 02:50:25 license.c:318 CRIT No license file.
No license installed!

admin@cumulus:mgmt-vrf:~$ sudo cat /var/log/syslog
2017-10-16T02:53:04.661850+00:00 cumulus systemd[1]: Failed to start Cumulus Linux_
↳ Switch Daemon.
2017-10-16T02:53:04.662375+00:00 cumulus systemd[1]: Dependency failed for Cumulus_
↳ Linux Port Watch Event Daemon.
2017-10-16T02:53:04.663217+00:00 cumulus systemd[1]: Dependency failed for Cumulus_
↳ Linux acltool.
2017-10-16T02:53:04.664004+00:00 cumulus systemd[1]: Unit switchd.service entered_
↳ failed state.
2017-10-16T02:53:04.687029+00:00 cumulus Failure: Not running cl-support for portwd.
↳ service, failure #2 status: Result=success ExecMainCode=2 ExecMainStatus=15
2017-10-16T02:53:04.730186+00:00 cumulus Failure: Not running cl-support for switchd.
↳ service, failure #2 status: Result=exit-code ExecMainCode=1 ExecMainStatus=2
```

Add the license key.

```
admin@cumulus:mgmt-vrf:~$ sudo cl-license -i
Paste license text here, then hit ctrl-d
user@company.com|example_license_text/here
License file installed.
Service 'switchd' is not running.
Run this command:
sudo systemctl restart switchd

Or reboot to enable functionality.
License file installed.
```


After the license is installed, restart the *switchd* service

```
admin@cumulus:mgmt-vrf:~$ sudo service switchd stop
admin@cumulus:mgmt-vrf:~$ sudo service switchd start
```

Download Agent Installer

The cumulus AOS device agent installation files are available from the AOS server, served from the URL https://aos-server/device_agent_images/aos_device_agent.run

For validating the downloaded file, a .md5 file is also available. Copy the .run file to the Cumulus switch, with the command `vrf task exec mgmt wget -nv --no-check-certificate https://aos-server/device_agent_images/aos_device_agent.run`

Note: This command assumes bash is running under the ‘mgmt’ vrf context. If this is not the case, omit the section `vrf task exec mgmt` and just download the file normally

Listing 27: Download AOS Agent

```
root@cumulus:mgmt-vrf:~# vrf task exec mgmt wget -nv --no-check-certificate https://
↪aos-server/device_agent_images/aos_device_agent.run
WARNING: The certificate of 'aos-server' is not trusted.
WARNING: The certificate of 'aos-server' hasn't got a known issuer.
The certificate's owner does not match hostname 'aos-server'
2017-10-15 23:32:55 URL:https://aos-server/device_agent_images/aos_device_agent.run_
↪[57245356/57245356] -> "aos_device_agent.run" [1]
```

Listing 28: Download AOS Agent

```
root@cumulus:mgmt-vrf:~#vrf task exec mgmt wget -nv --no-check-certificate https://
↪aos-server/device_agent_images/aos_device_agent.run.md5
WARNING: The certificate of 'aos-server' is not trusted.
WARNING: The certificate of 'aos-server' hasn't got a known issuer.
The certificate's owner does not match hostname 'aos-server'
2017-10-15 23:34:50 URL:https://aos-server/device_agent_images/aos_device_agent.run.
↪md5 [65/65] -> "aos_device_agent.run.md5" [1]

root@cumulus:mgmt-vrf:~# root@cumulus:mgmt-vrf:~# cat aos_device_agent.run.md5
70d58a0aaa5ed4519b87ced74003476a aos_device_agent_2.0.0-210.run
root@cumulus:mgmt-vrf:~# root@cumulus:mgmt-vrf:~# md5sum aos_device_agent.run
70d58a0aaa5ed4519b87ced74003476a aos_device_agent.run
```

Install Cumulus Device Agent

Installing the AOS device agent on cumulus is simply done by running the ‘.run’ file available from the AOS Server. Once the file is downloaded, run it as a shell command. Make sure that the AOS device agent is installed while bash is under the mgmt-vrf routing-context.

Run `sudo sh aos_device_agent.run`

```
admin@cumulus:mgmt-vrf:~$ sudo sh aos_device_agent.run
Verifying archive integrity... All good.
```

(continues on next page)

(continued from previous page)

```

Uncompressing AOS Device Agent installer 100%
+ set -o pipefail
+++ dirname ./agent_installer.sh
++ cd .
++ pwd
+ script_dir=/tmp/selfgz8796
++ date
+ echo 'Device Agent Installation : Mon' Oct 16 00:19:57 UTC 2017
Device Agent Installation : Mon Oct 16 00:19:57 UTC 2017
+ echo

+ UNKNOWN_PLATFORM=1
+ WRONG_PLATFORM=1
+ CANNOT_EXECUTE=126
+ '[' 0 -ne 0 ']'
+ arg_parse
+ start_aos=True
+ [[ 0 > 0 ]]
+ supported_platforms=(["centos"]="install_sysvinit_rpm" ["eos"]="install_on_arista" [
↪ "nxos"]="install_on_nxos" ["cumulus"]="install_sysvinit_deb" ["trusty"]="install_
↪ sysvinit_deb" ["icos"]="install_sysvinit_rpm" ["snaproute"]="install_sysvinit_deb" [
↪ "simulation"]="install_sysvinit_deb")
+ declare -A supported_platforms
++ /tmp/selfgz8796/aos_get_platform
+ current_platform=cumulus
+ installer=install_sysvinit_deb
+ [[ -z install_sysvinit_deb ]]
+ [[ -x /etc/init.d/aos ]]
+ install_sysvinit_deb
++ pwd
+ local pkg_dir=/tmp/selfgz8796/sysvinit_deb
+ dpkg -s aos-device-agent
+ dpkg --purge aos-device-agent
(Reading database ... 25364 files and directories currently installed.)
Removing aos-device-agent (2.0.0-210) ...
Purging configuration files for aos-device-agent (2.0.0-210) ...
+ dpkg -i /tmp/selfgz8796/sysvinit_deb/aos-device-agent-2.0.0-210.amd64.deb
Selecting previously unselected package aos-device-agent.
(Reading database ... 25364 files and directories currently installed.)
Preparing to unpack ../aos-device-agent-2.0.0-210.amd64.deb ...
Unpacking aos-device-agent (2.0.0-210) ...
Setting up aos-device-agent (2.0.0-210) ...
Processing triggers for systemd (215-17+deb8u4) ...
+ mkdir -p /opt/aos
+ cp aos_device_agent.img /opt/aos
+ post_install_common
+ /etc/init.d/aos config_gen
grep: /etc/aos/aos.conf: No such file or directory
+ [[ True == \T\r\u\e ]]
+++ readlink /sbin/init
++ basename /lib/systemd/systemd
+ [[ systemd == systemd ]]
+ systemctl start aos

```

AOS will create an aos.conf file if one does not exist at first startup of the agent.

AOS Device Agent Configuration File

Device agent configuration can be managed by editing the device agent configuration file directly. The Cumulus device agent config file is located at `/etc/aos/aos.conf`. See *AOS Device Agent Configuration file* for parameters. After updating the file, restart the AOS device agent.

Listing 29: Restart AOS Device Agent

```
service aos stop
service aos start
```

AOS Device Agent Management

Bootstrap configuration

The concept of *bootstrap* configuration relates to the `/etc/network/interfaces` configuration section as applicable to the management IP address. The bootstrap configuration is prepended to configuration jobs that AOS pushes. This helps ensure that AOS does not overwrite any network configuration that may prevent the agent from reaching the controller.

The bootstrap configuration is typically pushed by configuration by the user, or automated with software such as Apstra's Aeon ZTP server, or the AOS Device installer project.

Bootstrap configuration contain the minimum necessary network settings for AOS to connect to the AOS controller.

Listing 30: Example of `/etc/network/interfaces`

```
auto mgmt
iface mgmt
address 127.0.0.1/8
vrf-table auto
post-up sudo service aos start

# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

auto eth0
iface eth0 inet dhcp
    vrf mgmt
```

Cumulus Device Configuration Management

The Cumulus device agent manages the following files on the filesystem:

/etc/cumulus/ports.conf The `ports.conf` file specifies how port breakouts are consumed on the Cumulus platform

/etc/frr/frr.conf `frr.conf` contains all of the routing information for BGP on the device.

/etc/network/interfaces The network interfaces file handles all layer 2 and layer 3 configuration on the device. This includes CLAG, VLANs, VXLAN, IP Routing, etc.

/etc/hostname AOS manages the device hostname through `/etc/hostname`

/etc/default/isc-dhcp-relay AOS manages this file to specify interfaces that participate in DHCP Relay

Deploy the device

Once the Agent is up and running it will appear under Managed Devices, and can be Acknowledged and assigned to a Blueprint using the UI as normal.

Uninstall AOS Device Agent

Removing the AOS device agent is a three-step process: stop the agent, remove it from AOS and uninstall the agent.

The AOS Device Agent on Cumulus can be uninstalled with the below procedure. Since the AOS agent itself is mostly a compressed (squashfs) image, it can be uninstalled in a few simple steps.

When uninstalling the AOS configuration applied to the various files (frr.conf, /etc/network/interfaces, etc) is removed.

Stop the AOS Service

Before the AOS device agent is uninstalled, stop the AOS service on the agent. This will prevent the agent from immediately re-registering to the AOS server.

```
admin@cumulus:mgmt-vrf:~$ sudo service aos stop
admin@cumulus:mgmt-vrf:~$ sudo service aos status
aos.service - LSB: Start AOS device agents
   Loaded: loaded (/etc/init.d/aos)
   Active: inactive (dead) since Fri 2017-05-26 17:27:37 UTC; 5s ago
   Process: 32086 ExecStop=/etc/init.d/aos stop (code=exited, status=0/SUCCESS)
   Process: 31268 ExecStart=/etc/init.d/aos start (code=exited, status=0/SUCCESS)
```

Removing the device from AOS UI

Undeploy and Unassign the device from the blueprint as per standard procedures using the UI. It can also be deleted entirely from the Managed Devices page.

Important: If you uninstall the agent before removing it from the AOS UI, existing configuration can no longer be erased.

Remove the AOS package from Cumulus

The AOS package can be removed with `sudo dpkg -r aos-device-agent`

```
admin@cumulus:mgmt-vrf:~$ sudo dpkg -r aos-device-agent
(Reading database ... 25366 files and directories currently installed.)
Removing aos-device-agent (2.0.0-210) ...
Processing triggers for systemd (215-17+deb8u4) ...
```

Clean up any other lingering files left on the filesystem.

```
sudo rm -rf /opt/aos/
sudo rm -rf /etc/aos
sudo rm -rf /var/log/aos
sudo rm -rf /run/aos/*
```

(continues on next page)

(continued from previous page)

```

sudo rm -rf /mnt/persist/.aos
sudo rm -rf /run/aos
sudo rm -rf /run/lock/aos
sudo rm -rf /tmp/aos_show_tech
sudo rm -rf /usr/sbin/aos*

```

Check if any other AOS files remain on the filesystem.

```

admin@cumulus:mgmt-vrf:~$ sudo find / -iname '*aos*'
find: File system loop detected; `/.snapshots/1/snapshot' is part of the same file_
↪system loop as `/'.
/home/admin/aos_device_agent_1.2.0-137_cumulus.run
/var/lib/dpkg/info/aos-device-agent.postrm
/var/lib/dpkg/info/aos-device-agent.list

```

Optionally, remove the management VRF from /etc/network/interfaces.

Cumulus Agent Troubleshooting

Checking AOS status

Running `service aos status` will provide output of the AOS service status.

```

root@cumulus:mgmt-vrf:~# service aos status
aos.service - LSB: Start AOS device agents
  Loaded: loaded (/etc/init.d/aos)
  Active: active (running) since Fri 2017-05-26 17:42:39 UTC; 1min 59s ago
  Process: 32086 ExecStop=/etc/init.d/aos stop (code=exited, status=0/SUCCESS)
  Process: 32381 ExecStart=/etc/init.d/aos start (code=exited, status=0/SUCCESS)
  CGroup: /system.slice/aos.service
          └─32468 tacspawner --daemonize=/var/log/aos/aos.log --pidfile=/var/run/aos.
↪pid --name=000C29CBF3A8 --hostname=000C29CBF3A8 --domainSocket=aos_spawner_sock --
↪hostSysdbAddress=tbl://aos_localtasks_sock --jsonConfig=/etc/aos/cumulus/agent.json_
↪--eventLogSev=TaccSpawner/error,Mounter/error,Mountee/error,Nb...
          └─32470 tacleafsysdb --agentName=000C29CBF3A8-LocalTasks-000C29CBF3A8-0 --
↪partition= --storage-mode=persistent --eventLogDir=. --eventLogSev=TaccSpawner/
↪error,Mounter/error,Mountee/error,NboAttrLog/error
          └─32471 /usr/bin/python /usr/bin/aos_agent --class=aos.device.common.
↪ProxyDeploymentAgent.ProxyDeploymentAgent --name=DeploymentProxyAgent device_
↪type=Cumulus serial_number=@(SYSTEM_UNIQUE_ID)
          └─32474 /usr/bin/python /usr/bin/aos_agent --class=aos.device.common.
↪DeviceKeeperAgent.DeviceKeeperAgent --name=DeviceKeeperAgent serial_number=@(SYSTEM_
↪UNIQUE_ID)
          └─32484 /usr/bin/python /usr/bin/aos_agent --class=aos.device.common.
↪ProxyCountersAgent.ProxyCountersAgent --name=CounterProxyAgent device_type=Cumulus_
↪serial_number=@(SYSTEM_UNIQUE_ID)
          └─32511 /usr/bin/python /usr/bin/aos_agent --class=aos.device.cumulus.
↪CumulusTelemetryAgent.CumulusTelemetryAgent --name=DeviceTelemetryAgent serial_
↪number=@(SYSTEM_UNIQUE_ID)
          └─32748 sh -c curl -k -f -sS -H "Content-Type: application/json" -X POST -
↪d '{"version":"1.2.0-137","serial_number":"000C29CBF3A8","platform":"cumulus"}' _
↪https://aos-server/api/versions/device 2>&1
          └─32749 curl -k -f -sS -H Content-Type: application/json -X POST -d {
↪"version":"1.2.0-137","serial_number":"000C29CBF3A8","platform":"cumulus"} https://
↪aos-server/api/versions/device

```

Listing AOS running processes

We can attach to the AOS container with `service aos attach` command, then run normal linux commands within the container. Any changes within this container will be lost/destroyed after the container restarts again, it's a read-only instance.

```
root@cumulus:mgmt-vrf:/var/log/aos# service aos attach
aos@mclag-compute-1-leaf1:/# ps wax
```

| PID | TTY | STAT | TIME | COMMAND |
|-----|-----|------|------|-----------------|
| 1 | ? | Ss | 0:21 | /sbin/init |
| 2 | ? | S | 0:00 | [kthreadd] |
| 3 | ? | S | 0:04 | [ksoftirqd/0] |
| 5 | ? | S< | 0:00 | [kworker/0:0H] |
| 7 | ? | S | 1:28 | [rcu_sched] |
| 8 | ? | S | 0:00 | [rcu_bh] |
| 9 | ? | S | 0:00 | [migration/0] |
| 10 | ? | S | 0:00 | [watchdog/0] |
| 11 | ? | S | 0:00 | [watchdog/1] |
| 12 | ? | S | 0:00 | [migration/1] |
| 13 | ? | S | 0:04 | [ksoftirqd/1] |
| 15 | ? | S< | 0:00 | [kworker/1:0H] |
| 16 | ? | S | 0:00 | [watchdog/2] |
| 17 | ? | S | 0:00 | [migration/2] |
| 18 | ? | S | 0:08 | [ksoftirqd/2] |
| 20 | ? | S< | 0:00 | [kworker/2:0H] |
| 21 | ? | S | 0:00 | [watchdog/3] |
| 22 | ? | S | 0:00 | [migration/3] |
| 23 | ? | S | 0:07 | [ksoftirqd/3] |
| 25 | ? | S< | 0:00 | [kworker/3:0H] |
| 26 | ? | S< | 0:00 | [khelper] |
| 27 | ? | S | 0:00 | [kdevtmpfs] |
| 28 | ? | S< | 0:00 | [netns] |
| 29 | ? | S< | 0:00 | [perf] |
| 30 | ? | S | 0:00 | [khungtaskd] |
| 31 | ? | S< | 0:00 | [writeback] |
| 33 | ? | SN | 0:00 | [ksmd] |
| 34 | ? | SN | 0:00 | [khugepaged] |
| 35 | ? | S< | 0:00 | [crypto] |
| 36 | ? | S< | 0:00 | [kintegrityd] |
| 37 | ? | S< | 0:00 | [bioset] |
| 38 | ? | S< | 0:00 | [kblockd] |
| 39 | ? | S< | 0:00 | [ata_sff] |
| 40 | ? | S< | 0:00 | [edac-poller] |
| 42 | ? | S< | 0:00 | [rpciod] |
| 43 | ? | S | 0:00 | [kswapd0] |
| 44 | ? | S | 0:00 | [fsnotify_mark] |
| 45 | ? | S< | 0:00 | [nfsiod] |
| 54 | ? | S< | 0:00 | [kthrotld] |
| 57 | ? | S | 0:00 | [scsi_eh_0] |
| 58 | ? | S< | 0:00 | [scsi_tmf_0] |
| 59 | ? | S | 0:00 | [scsi_eh_1] |
| 60 | ? | S< | 0:00 | [scsi_tmf_1] |
| 61 | ? | S | 0:00 | [scsi_eh_2] |
| 62 | ? | S< | 0:00 | [scsi_tmf_2] |
| 63 | ? | S | 0:00 | [scsi_eh_3] |
| 64 | ? | S< | 0:00 | [scsi_tmf_3] |
| 69 | ? | S | 0:00 | [scsi_eh_4] |

(continues on next page)

(continued from previous page)

```

70 ?      S<      0:00 [scsi_tmf_4]
71 ?      S        0:00 [scsi_eh_5]
72 ?      S<      0:00 [scsi_tmf_5]
76 ?      S<      0:00 [ipv6_addrconf]
77 ?      S<      0:00 [deferwq]
128 ?     S<      0:00 [bioset]
133 ?     S        0:00 [scsi_eh_6]
134 ?     S<      0:00 [scsi_tmf_6]
135 ?     S        0:03 [usb-storage]
136 ?     S        0:00 [scsi_eh_7]
137 ?     S<      0:00 [scsi_tmf_7]
138 ?     S        0:01 [usb-storage]
146 ?     S<      0:00 [kworker/0:1H]
147 ?     S<      0:00 [kworker/1:1H]
148 ?     S<      0:00 [kworker/3:1H]
149 ?     S<      0:00 [kworker/2:1H]
166 ?     S<      0:00 [btrfs-worker]
168 ?     S<      0:00 [btrfs-worker-hi]
169 ?     S<      0:00 [btrfs-delalloc]
170 ?     S<      0:00 [btrfs-flush_del]
171 ?     S<      0:00 [btrfs-cache]
172 ?     S<      0:00 [btrfs-submit]
173 ?     S<      0:00 [btrfs-fixup]
174 ?     S<      0:00 [btrfs-endio]
175 ?     S<      0:00 [btrfs-endio-met]
176 ?     S<      0:00 [btrfs-endio-met]
177 ?     S<      0:00 [btrfs-endio-rai]
178 ?     S<      0:00 [btrfs-endio-rep]
179 ?     S<      0:00 [btrfs-rmw]
180 ?     S<      0:00 [btrfs-endio-wri]
181 ?     S<      0:00 [btrfs-freespace]
182 ?     S<      0:00 [btrfs-delayed-m]
183 ?     S<      0:00 [btrfs-readahead]
184 ?     S<      0:00 [btrfs-qgroup-re]
185 ?     S<      0:00 [btrfs-extent-re]
186 ?     S        0:00 [btrfs-cleaner]
187 ?     S        0:06 [btrfs-transacti]
262 ?     S        0:00 [kauditd]
266 ?     Ss       0:05 /lib/systemd/systemd-journald
269 ?     Ss       0:00 /lib/systemd/systemd-udev
337 ?     S<      0:00 [kvm-irqfd-clean]
542 ?     S<s1     0:00 /sbin/auditd -n
694 ?     SNs      0:00 /usr/sbin/cron -f -L 38
697 ?     Ss       0:00 /lib/systemd/systemd-logind
702 ?     Ss       0:02 /usr/bin/dbus-daemon --system --address=systemd: --nofork -
→-nopidfile --systemd-activation
708 ?     Ss       0:00 /usr/sbin/mcelog --daemon
710 ?     Ss       0:04 /usr/sbin/irqbalance --pid=/var/run/irqbalance.pid
715 ?     Ss       0:00 /usr/sbin/acpid
719 ?     Ss1      0:01 /usr/sbin/rsyslogd -n
733 ?     Ss       0:00 /usr/sbin/wd_keepalive
740 ?     Ss       9:17 /usr/bin/python /usr/sbin/smond
741 ?     Ss       0:24 /usr/bin/python /usr/sbin/ledmgrd
743 ?     S        0:00 /usr/sbin/dnsmasq -x /var/run/dnsmasq/dnsmasq.pid -u
→dnsmasq -7 /etc/dnsmasq.d,.dpkg-dist,.dpkg-old,.dpkg
744 ?     Ss       0:43 /usr/bin/python /usr/sbin/pwmd
745 ?     S<s      0:05 /sbin/mstpd -d -v2

```

(continues on next page)

(continued from previous page)

```

1053 ?      Ss      0:00 /usr/sbin/uidd --socket-activation
1196 ?      Ssl     1:34 /usr/bin/python /usr/bin/arp_refresh
1200 ?      Ss      0:00 /usr/bin/python /usr/lib/python2.7/dist-packages/clcmd_
↪server.py > /dev/null 2>&1
1240 ?      Ss      0:02 /usr/sbin/ntpd -n -u ntp:ntp -g
1242 ?      Ss      0:00 /usr/sbin/ptmd -l INFO
1255 ?      Ss      0:00 /usr/sbin/sshd -D
1978 ?      Ss      0:00 /sbin/dhclient -pf /run/dhclient.eth0.pid -lf /var/lib/
↪dhcp/dhclient.eth0.leases eth0
2310 ?      S       0:00 [kworker/2:2]
4633 ?      Ss      0:00 sshd: admin [priv]
4661 ?      S       0:00 sshd: admin@pts/2
4662 ?      Ss      0:00 -bash
4699 ?      S       0:00 sudo su -
4704 ?      S       0:00 su -
4726 ?      S       0:00 -su
5853 ?      S       0:00 [kworker/3:2]
8515 ?      S       0:00 [kworker/u8:1]
9264 ?      S<      0:00 [kworker/u9:6]
11195 ?     S       0:00 [kworker/2:1]
13477 ?     S<      0:00 [kworker/u9:0]
13480 ?     S<      0:00 [kworker/u9:1]
13481 ?     S<      0:00 [kworker/u9:2]
13482 ?     S<      0:00 [kworker/u9:3]
13810 ?     S       0:00 /bin/bash /etc/init.d/aos attach
13824 ?     S       0:00 /lib/systemd/systemd-udev
13832 ?     S       0:00 /bin/bash
13835 ?     R+      0:00 ps wax
14541 ?     S       0:00 sudo su -
14547 ?     S       0:00 su -
14575 ?     S       0:00 -su
14645 ?     S       0:00 [kworker/u8:2]
14675 ?     S+      0:00 tail -f DeploymentProxyAgent.err
15057 ?     S<      0:00 [kloopd0]
15097 ?     S<      0:00 [kworker/u9:7]
15099 ?     Sl      0:00 tacspawner --daemonize=/var/log/aos/aos.log --pidfile=/var/
↪run/aos.pid --name=571254X1448071 --hostname=5
15103 ?     S       0:00 tacleafsysdb --agentName=571254X1448071-LocalTasks-
↪571254X1448071-0 --partition= --storage-mode=persisten
15107 ?     Sl      0:02 /usr/bin/python /usr/bin/aos_agent --class=aos.device.
↪common.DeviceKeeperAgent.DeviceKeeperAgent --name=D
15117 ?     Sl      0:03 /usr/bin/python /usr/bin/aos_agent --class=aos.device.
↪common.ProxyCountersAgent.ProxyCountersAgent --name
15138 ?     Sl      0:06 /usr/bin/python /usr/bin/aos_agent --class=aos.device.
↪cumulus.CumulusTelemetryAgent.CumulusTelemetryAgent
15139 ?     Sl      0:02 /usr/bin/python /usr/bin/aos_agent --class=aos.device.
↪common.ProxyDeploymentAgent.ProxyDeploymentAgent --
15373 ttySl  Ss      0:00 /bin/login --
15403 ttySl  S       0:00 -bash
15440 ttySl  S       0:00 sudo su
15445 ttySl  S       0:00 su
15467 ttySl  S+      0:00 bash
15756 ?     Ss      0:00 dhclient -lf /var/lib/dhcp/dhclient.eth0.leases eth0
16287 ?     S<sl    5:55 /usr/sbin/switchd
16583 ?     Ds      0:29 /usr/bin/python /usr/sbin/portwd
16586 ?     S       0:00 [kworker/1:0]
16763 ?     S<s     0:00 /usr/lib/quagga/zebra -s 90000000 --daemon -A 127.0.0.1

```

(continues on next page)

(continued from previous page)

```

16770 ?      S<s    0:00 /usr/lib/quagga/bgpd --daemon -A 127.0.0.1
16777 ?      S<s    0:00 /usr/lib/quagga/watchquagga -adz -r /usr/sbin/
↪servicebBquaggabBrestartbB%s -s /usr/sbin/servicebBquaggabB
16787 ?      Ss     0:00 sshd: admin [priv]
16815 ?      S      0:00 sshd: admin@pts/1
16816 ?      Ss     0:00 -bash
16853 ?      S      0:00 sudo su -
16858 ?      S      0:00 su -
16880 ?      S      0:00 -su
17070 ?      S+     0:00 tail -f DeploymentProxyAgent.err
18757 ?      Ss     0:00 lldpd: monitor .
18760 ?      S      0:01 lldpd: connected to dutmgmtsw2-eth0.dc1.apstra.com
21990 ?      S      0:00 [kworker/2:0]
22942 ?      S      0:00 [kworker/3:0]
24846 ?      S      0:00 [kworker/0:0]
27138 ?      S      0:00 [kworker/3:1]
27389 ?      S<     0:00 [bond41]
27584 ?      Ssl    0:06 /usr/bin/python /usr/sbin/clagd --daemon 10.0.0.63 bond41.
↪2999 44:38:39:ff:00:01 --priority 32768 --backu
27665 ?      S      0:00 /sbin/bridge monitor fdb
27959 ?      S      0:00 [kworker/1:1]
28342 ?      S      0:00 [kworker/u8:0]
31882 ?      Ss     0:00 sshd: admin [priv]
31912 ?      S      0:00 sshd: admin@pts/0
31913 ?      Ss     0:00 -bash
32063 ?      S      0:00 [kworker/0:2]

```

Displaying and reading AOS log files

AOS log files are stored in the `/var/log/aos` folder. There is typically one log file for each AOS agent that runs. The AOS device agent will spawn off a series of other device agents for various purposes - telemetry, configuration rendering, counters, and agent health. The `.err` files are plain-text and can be read with any text editors

```

root@cumulus:mgmt-vrf:/var/log/aos# ls -lah /var/log/aos
total 876K
drwxr-xr-x 1 root root 4.9K Oct 16 01:42 .
drwxr-xr-x 1 root root 360 Oct 16 01:44 ..
-rw----- 1 root root 644 Oct 16 01:42 _571254X1448071-0000000059e40e8c-000ea27f-
↪checkpoint
-rw-r--r-- 1 root root 0 Oct 16 01:42 _571254X1448071-0000000059e40e8c-000ea27f-
↪checkpoint-valid
-rw----- 1 root root 0 Oct 16 01:42 _571254X1448071-0000000059e40e8c-000ea27f-log
-rw-r--r-- 1 root root 0 Oct 16 01:42 _571254X1448071-0000000059e40e8c-000ea27f-
↪log-valid
-rw-r--r-- 1 root root 0 Oct 16 01:40 571254X1448071-0.14640.1508118008.log
-rw-r--r-- 1 root root 0 Oct 16 01:42 571254X1448071-0.15103.1508118156.log
-rw-r--r-- 1 root root 0 Oct 16 01:40 571254X1448071-0.err
-rw-r--r-- 1 root root 0 Oct 16 01:40 571254X1448071-0.out
-rw----- 1 root root 61K Oct 16 01:40 571254X1448071-LocalTasks-571254X1448071-0_
↪2017-10-16--01-40-08_14640-2017-10-16--01-40-08.tel
-rw----- 1 root root 105K Oct 16 02:16 571254X1448071-LocalTasks-571254X1448071-0_
↪2017-10-16--01-42-36_15103-2017-10-16--01-42-36.tel
-rw-r--r-- 1 root root 0 Oct 16 01:42 aos.log
-rw-r--r-- 1 root root 0 Oct 16 01:40 CounterProxyAgent.14642.1508118012.log
-rw-r--r-- 1 root root 0 Oct 16 01:40 CounterProxyAgent.14665.1508118023.log

```

(continues on next page)

(continued from previous page)

```

-rw-r--r-- 1 root root 0 Oct 16 01:42 CounterProxyAgent.15105.1508118159.log
-rw-r--r-- 1 root root 0 Oct 16 01:42 CounterProxyAgent.15117.1508118171.log
-rw----- 1 root root 33K Oct 16 01:40 CounterProxyAgent571254X1448071_2017-10-16--
↪01-40-14_14642-2017-10-16--01-40-14.tel
-rw----- 1 root root 42K Oct 16 01:40 CounterProxyAgent571254X1448071_2017-10-16--
↪01-40-24_14665-2017-10-16--01-40-24.tel
-rw----- 1 root root 33K Oct 16 01:42 CounterProxyAgent571254X1448071_2017-10-16--
↪01-42-41_15105-2017-10-16--01-42-41.tel
-rw----- 1 root root 42K Oct 16 01:42 CounterProxyAgent571254X1448071_2017-10-16--
↪01-42-52_15117-2017-10-16--01-42-52.tel
-rw-r--r-- 1 root root 3.0K Oct 16 01:42 CounterProxyAgent.err
-rw-r--r-- 1 root root 0 Oct 16 01:40 CounterProxyAgent.out
-rw-r--r-- 1 root root 0 Oct 16 01:40 DeploymentProxyAgent.14641.1508118018.log
-rw-r--r-- 1 root root 0 Oct 16 01:42 DeploymentProxyAgent.15104.1508118166.log
-rw-r--r-- 1 root root 0 Oct 16 01:42 DeploymentProxyAgent.15139.1508118176.log
-rw----- 1 root root 39K Oct 16 01:40 DeploymentProxyAgent571254X1448071_2017-10-
↪16--01-40-19_14641-2017-10-16--01-40-19.tel
-rw----- 1 root root 33K Oct 16 01:42 DeploymentProxyAgent571254X1448071_2017-10-
↪16--01-42-47_15104-2017-10-16--01-42-47.tel
-rw----- 1 root root 39K Oct 16 01:43 DeploymentProxyAgent571254X1448071_2017-10-
↪16--01-42-58_15139-2017-10-16--01-42-58.tel
-rw-r--r-- 1 root root 31K Oct 16 02:03 DeploymentProxyAgent.err
-rw-r--r-- 1 root root 0 Oct 16 01:40 DeploymentProxyAgent.out
-rw-r--r-- 1 root root 0 Oct 16 01:40 DeviceKeeperAgent.14644.1508118014.log
-rw-r--r-- 1 root root 0 Oct 16 01:42 DeviceKeeperAgent.15107.1508118166.log
-rw----- 1 root root 38K Oct 16 01:40 DeviceKeeperAgent571254X1448071_2017-10-16--
↪01-40-15_14644-2017-10-16--01-40-15.tel
-rw----- 1 root root 38K Oct 16 01:43 DeviceKeeperAgent571254X1448071_2017-10-16--
↪01-42-48_15107-2017-10-16--01-42-48.tel
-rw-r--r-- 1 root root 1.3K Oct 16 01:42 DeviceKeeperAgent.err
-rw-r--r-- 1 root root 0 Oct 16 01:40 DeviceKeeperAgent.out
-rw-r--r-- 1 root root 0 Oct 16 01:40 DeviceTelemetryAgent.14643.1508118012.log
-rw-r--r-- 1 root root 0 Oct 16 01:40 DeviceTelemetryAgent.14674.1508118021.log
-rw-r--r-- 1 root root 0 Oct 16 01:42 DeviceTelemetryAgent.15106.1508118165.log
-rw-r--r-- 1 root root 0 Oct 16 01:42 DeviceTelemetryAgent.15138.1508118177.log
-rw----- 1 root root 33K Oct 16 01:40 DeviceTelemetryAgent571254X1448071_2017-10-
↪16--01-40-15_14643-2017-10-16--01-40-15.tel
-rw----- 1 root root 56K Oct 16 01:40 DeviceTelemetryAgent571254X1448071_2017-10-
↪16--01-40-23_14674-2017-10-16--01-40-23.tel
-rw----- 1 root root 33K Oct 16 01:42 DeviceTelemetryAgent571254X1448071_2017-10-
↪16--01-42-47_15106-2017-10-16--01-42-47.tel
-rw----- 1 root root 56K Oct 16 01:43 DeviceTelemetryAgent571254X1448071_2017-10-
↪16--01-42-59_15138-2017-10-16--01-42-59.tel
-rw-r--r-- 1 root root 8.8K Oct 16 01:49 DeviceTelemetryAgent.err
-rw-r--r-- 1 root root 0 Oct 16 01:40 DeviceTelemetryAgent.out
-rw----- 1 root root 50K Oct 16 01:40 Spawner-571254X1448071_2017-10-16--01-40-08_
↪14635-2017-10-16--01-40-08.tel
-rw----- 1 root root 51K Oct 16 01:43 Spawner-571254X1448071_2017-10-16--01-42-36_
↪15096-2017-10-16--01-42-36.tel

```

DeviceTelemetryAgent.err This file contains all diagnostic output as it relates to device telemetry.

DeviceKeeperAgent.err This file tracks the health of the AOS Agent itself and how it mounts AOS Graph Datastore to the AOS Server.

DeploymentProxyAgent.err This file logs all configuration options managed by AOS, and describes whether the configuration job is a complete apply or partial apply. All configuration changes by AOS are seen here.

CounterProxyAgent.err This file captures any custom counter collectors deployed on the switch

aos.log Unused

XXXXXXXXXXXX-0.err Unused

The *.tel* files are *tacc event logs* output, used for internal support at Apstra.

Configuration not being pushed

The device agent may be configured in 'telemetry-only' mode. Check */etc/aos/aos.conf* for `enable_configuration_service = 0`.

You may also observe these log lines in */var/log/aos/DeviceProxyAgent.err*, stating *Configuration service disabled. Not setting mount timer*.

Correcting this will then show *handle device deployment config* in the log file.

```
2017-10-16 01:42:58,571 15139:INFO:aos.device.common.ProxyDeploymentSm:Device init_
↳sanity check completed at 1508118178.571466
2017-10-16 01:42:58,571 15139:INFO:aos.device.common.ProxyDeploymentSm:Device_
↳rebooted: False
2017-10-16 01:42:58,572 15139:INFO:aos.device.common.ProxyDeploymentSm:Device undergo_
↳ztp: True
2017-10-16 01:42:58,630 15139:INFO:aos.device.common.ProxyDeploymentSm:handle device_
↳deployment config: 571254X1448071 ddc: 1 dds: 1
2017-10-16 01:42:58,630 15139:INFO:aos.device.common.ProxyDeploymentSm:deploy ddc: 1_
↳dds: 1
2017-10-16 01:42:58,635 15139:INFO:aos.device.common.ProxyDeploymentSm:Config up-to-
↳date after restart, not re-applying
```

Cant ping AOS or use other network tools

Attempting to ping the AOS Server on the CLI while the device is configured in a VRF may give confusing error messages. Make sure to use `ping -I mgmt` on an ICMP echo to source it from the management VRF properly.

For other commands, use `vrf task exec mgmt <command>`

```
admin@cumulus:mgmt-vrf:~$ ping aos-server
sudo: unable to resolve host mclag-compute-1-leaf1
connect: Network is unreachable

admin@cumulus:mgmt-vrf:~$ ping -I mgmt aos-server
sudo: unable to resolve host mclag-compute-1-leaf1
ping: Warning: source address might be selected on device other than mgmt.
PING aos-server (172.20.85.3) from 172.20.85.6 mgmt: 56(84) bytes of data.
64 bytes from aos-server (172.20.85.3): icmp_seq=1 ttl=64 time=0.245 ms
```

Avg state is CRITICAL

When the Cumulus VM runs out of ram, we get kernel panics and the system continues to restart by itself. The cumulus default OVA VM ships with 512MB by default.

The root cause here is there was not enough ram to run the VM.

```
Broadcast message from root@cumulus (somewhere) (Mon Jan 15 20:44:51 2018):  
  
Avg state is CRITICAL. Last 10 values: [100.0, 100.0, 100.0, 100.0, 100.0, 100.  
0, 100.0, 100.0, 100.0, 100.0]. System will shutdown in 26 secs
```

Contact Apstra Global Support

Apstra Global Support is available to assist with troubleshooting. Diagnostic information will most likely be needed to help resolve issues. Please see the [Apstra Global Support](#) page for details.

5.3.8.5 SONiC Device Agent

This describes the process of manual AOS agent installation on supported SONiC devices. For the recommended method using the web interface, refer to [Device Agents](#).

Important: Only in rare exceptions is it needed to follow the manual process of agent installation. In almost all cases agents should be installed by creating system agents in the web interface. Installing agents manually is more bespoke, more effort, and prone to user error.

An in-depth understanding of the various device states, configuration stages, and agent operations is required when attempting manual agent installation.

When in doubt, contact [Apstra Global Support](#).

SONiC Support

Starting in AOS version 3.3.0a, the only Software for Open Networking in the Cloud (SONiC) distribution is SONiC Enterprise.

The SONiC device agent currently 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.

Warning: The user should not edit the `config_db.json` or `frr.conf` files manually at any time, before or after AOS Device System Agent installation. The AOS Agent will overwrite any existing configuration in these files.

Manual SONiC Device Management IP Configuration

SONiC automatically creates a management VRF for the “eth0” management interface. By default, “eth0” will get a DHCP address from the management network. In most cases, no management configuration should be needed.

However, if the user needs to manually configure a SONiC device management IP address, the user **must** configure this 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(config-if-eth0)# ip address 192.168.59.7/24 gwaddr 192.168.59.1
sonic(config-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#

```

The Managment VRF can be checked 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 addrgenmode 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

```

(continues on next page)

(continued from previous page)

```

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:~$

```

Manual AOS Agent Installation

The section below is a guide on manually installing the SONiC AOS device agent.

1. Download the AOS 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 AOS 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 ]]

```

(continues on next page)

(continued from previous page)

```

+ supported_platforms=("centos"="install_centos" ["eos"]="install_on_arista" [
↪ "nxos"]="install_on_nxos" ["cumulus"]="install_sysvinit_deb" ["opx"]="install_
↪ systemd_deb opx" ["trusty"]="install_sysvinit_deb" ["xenial"]="install_sysvinit_
↪ deb" ["icos"]="install_sysvinit_rpm" ["snaproute"]="install_sysvinit_deb" [
↪ "simulation"]="install_sysvinit_deb" ["sonic"]="install_systemd_deb sonic" [
↪ "bionic"]="install_sysvinit_deb")
+ declare -A supported_platforms
++ /tmp/selfgz334323135/aos_get_platform
+ current_platform=sonic
+ installer='install_systemd_deb sonic'
+ [[ -z install_systemd_deb sonic ]]
+++ readlink /sbin/init
++ basename /lib/systemd/systemd
+ [[ systemd == systemd ]]
+ systemd_available=true
+ [[ -x /etc/init.d/aos ]]
+ echo 'Stopping AOS'
Stopping AOS
+ true
+ systemctl stop aos
+ install_systemd_deb sonic
++ pwd
+ local pkg_dir=/tmp/selfgz334323135/sonic
+ install_deb /tmp/selfgz334323135/sonic
+ local pkg_dir=/tmp/selfgz334323135/sonic
+ dpkg -s aos-device-agent
+ dpkg --purge aos-device-agent
(Reading database ... 34189 files and directories currently installed.)
Removing aos-device-agent (3.3.0a-93) ...
Purging configuration files for aos-device-agent (3.3.0a-93) ...
Processing triggers for systemd (232-25+deb9u12) ...
+ dpkg -i /tmp/selfgz334323135/sonic/aos-device-agent-3.3.0a-93.amd64.deb
Selecting previously unselected package aos-device-agent.
(Reading database ... 34180 files and directories currently installed.)
Preparing to unpack .../aos-device-agent-3.3.0a-93.amd64.deb ...
Unpacking aos-device-agent (3.3.0a-93) ...
Setting up aos-device-agent (3.3.0a-93) ...
Synchronizing state of aos.service with SysV service script with /lib/systemd/
↪ systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable aos
/var/lib/dpkg/info/aos-device-agent.postinst: line 7: /usr/sbin/aosconfig: No_
↪ such file or directory
Processing triggers for systemd (232-25+deb9u12) ...
+ mkdir -p /opt/aos
+ cp aos_device_agent.img /opt/aos
+ post_install_common
+ /etc/init.d/aos config_gen
+ [[ False == \T\r\u\e ]]
+ true
+ systemctl enable aos
Synchronizing state of aos.service with SysV service script with /lib/systemd/
↪ systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable aos
admin@sonic:~$

```

3. Set the IP of the AOS server and enable configuration service by editing `/etc/aos/aos.conf` with the `sudo vi /etc/aos/aos.conf` command.

- For the following, replace “aos-server” with the IP address or valid FQDN of your AOS 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 AOS.

```
[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 AOS agent (usually “admin”).

```
[credential]
username = admin
```

4. Start the AOS 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
```

(continues on next page)

(continued from previous page)

```

└─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. The SONiC device will be listed in **Device / Managed Devices** in the AOS web interface and can be acknowledged and assigned to a blueprint.

The screenshot shows the 'Managed Devices' page in the AOS web interface. The page has a sidebar with navigation icons and a main content area. At the top right, there is a 'Stock Device' button. Below it, there is a search bar with the text 'Query: All' and pagination controls showing '1-1 of 1' and 'Page Size: 25'. The main table lists the following device:

| Device Key | Device Profile | Operation Mode | Management IP | AOS Version | Hostname | Location | OS | Acknowledged? | State | Blueprint | Comms |
|--------------|-----------------------|----------------|---------------|------------------|----------|----------|----------------------|---------------|-----------------|--------------|-------|
| 525400E1E77A | VS_SONIC_BUZZNIK_PLUS | FULL CONTROL | 172.20.74.6 | AOS_3.3.0a_OB.93 | sonic | | SONIC 3.1.0a-Generic | ⊖ | OOS-QUARANTINED | Not assigned | ⊕ |

Manual AOS Agent Uninstallation

The section below is a guide on manually uninstalled the SONiC AOS device agent.

1. Stop the AOS agent with the `sudo service aos stop` command.

```

admin@sonic:~$ sudo service aos stop
admin@sonic:~$

```

2. Uninstall the AOS 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 AOS 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:~$
```

5.3.8.6 Server Agent

We recommend using the [web interface](#) for tasks related to [device agents](#). However, if you are using a server interface naming schema different to `eth0` or if you would like to manually install the device agent for any other reason, this document guides you through that process.

Knowledge of the various device states and configuration stages is vital for managing AOS day-to-day operations. See [AOS Device Configuration Lifecycle](#) and [AOS Device Agent Configuration File](#) for details.

In a first instance, you will need to optionally install the desired extra telemetry packages in order for AOS to show related information.

After this step, you can go to the next section **L3 Server Agent** or the last section **Manual Agents installation on Servers** if you are just installing a L2 server agent. This last section will be required in both cases.

Warning: User is responsible for the verification and installation steps for the manual agent installation procedure.

CentOS extra Packages

Install the following packages in CentOS if you would like to collect the related telemetry information.

```
yum install -y epel-release
yum install -y lldpd
yum install -y python-ipaddr
yum install -y libselinux-python
yum install -y iproute
```

Ubuntu extra Packages

Install the following packages in Ubuntu if you would like to collect the related telemetry information.

```
apt update
apt install -y lldpd
apt install -y python-ipaddr
apt install -y libc-ares2
apt install -y iproute2
```

L3 Server Agent

If you are installing a L3 Server agent you will need to go through the FRR installation procedure described on this section. Otherwise you can go to the next section **Manual Agents installation on Servers** directly.

CentOS

1. Delete and clean any previous Routing related software from the server

```
yum erase -y quagga
yum erase -y frr
yum erase -y frr-pythontools
```

2. Delete existing frr user/groups

```
userdel frr
groupdel frrvty
```

3. Create FRR group

```
groupadd frr
groupadd frrvty
```

4. Ensure the FRR user configuration

```
useradd --shell /sbin/nologin --create-home --home-dir /var/run/frr/ --
↪comment 'FRRouting suite' -g frr -G frrvty frr
getent passwd frr
```

5. Make sure that you get the proper FRR image and Install FRR

```
yum install -y https://github.com/FRRouting/frr/releases/download/frr-6.0.
↪2/frr-6.0.2-01.el6.x86_64.rpm
yum install -y https://github.com/FRRouting/frr/releases/download/frr-6.0.
↪2/frr-pythontools-6.0.2-01.el6.x86_64.rpm
```

Ubuntu

1. Delete and clean any previous Routing related software from the server

```
service quagga stop
apt remove -y quagga
service frr stop
apt remove -y frr
apt remove -y frr-pythontools
```

2. Delete existing FRR user/groups

```
userdel frr
groupdel frrvty
```

3. Verify your current version by running `lsb_release -rs` and make sure you get the FRR packages according to this version. Install FRR

This example shows a FRR download for version 16.04:

```
wget -O frr_6.0.2-0.ubuntu16.04.1_amd64.deb -t 3 -T 3 https://github.com/
↪FRRouting/frr/releases/download/frr-6.0.2/frr_6.0.2-0.ubuntu16.04.1_
↪amd64.deb
dpkg -i --force-confnew ./frr_6.0.2-0.ubuntu16.04.1_amd64.deb
wget -O frr-pythontools_6.0.2-0.ubuntu16.04.1_all.deb -t 3 -T 3 https://
↪github.com/FRRouting/frr/releases/download/frr-6.0.2/frr-pythontools_6.
↪0.2-0.ubuntu16.04.1_all.deb
dpkg -i --force-confnew ./frr-pythontools_6.0.2-0.ubuntu16.04.1_all.deb
chown frr /etc/frr/vtysh.conf
chgrp frr /etc/frr/vtysh.conf
```

4. Configure LLDP

```
mkdir -p /etc/lldpd.d
grep 'configure lldp portidsubtype ifname' /etc/lldpd.d/port_info.conf
echo 'configure lldp portidsubtype ifname' > /etc/lldpd.d/port_info.conf
service lldpd restart
```

Manual Agents installation on Servers

The following procedure applies for L2 and L3 Servers. This last will need some extra steps that are described in the previous section **L3 Server Agent**. Please ensure that they are accomplished before proceeding further.

To install the agent manually please follow the next steps:

1. Ensure that the server can reach the AOS Server IP.
2. Delete any previous aos.run file if any.

```
rm -f /tmp/aos.run
```

3. Download the agent from the AOS Server, replace the `aos_version` by the running AOS version and the exact build (i.e 3.2.1-298), you can verify this information from the **AOS Dashboard** → **Platform** → **About** and the `aos_server_ip` variable by the actual AOS Server IP.

```
curl -o /tmp/aos.run -k -O https://<aos_server_ip>/device_agent_images/
aos_device_agent_<aos_version>.run
```

4. Run the installer.

```
/bin/bash /tmp/aos.run -- --no-start
```

5. Modify the `aos.conf` file, you will need to update the desired interface among other parameters, please refer to this document for details [AOS Device Agent Configuration File](#). In this `aos.conf` file it is important to enable the configuration service if you are configuring a L3 Server only. If it is a L2 Server the default setting (disabled) is correct.

```
enable_configuration_service = 1
```

6. Start the service.

```
sudo aos service start
```

If everything went as expected, the server will show up in the **Managed Devices** tab in the AOS Dashboard. Then acknowledge and assign to the desired blueprint following the standard procedure.

The following capture shows an example of L2 Server agent with **Telemetry only**. L3 Server would display **Full Control**.

0 selected

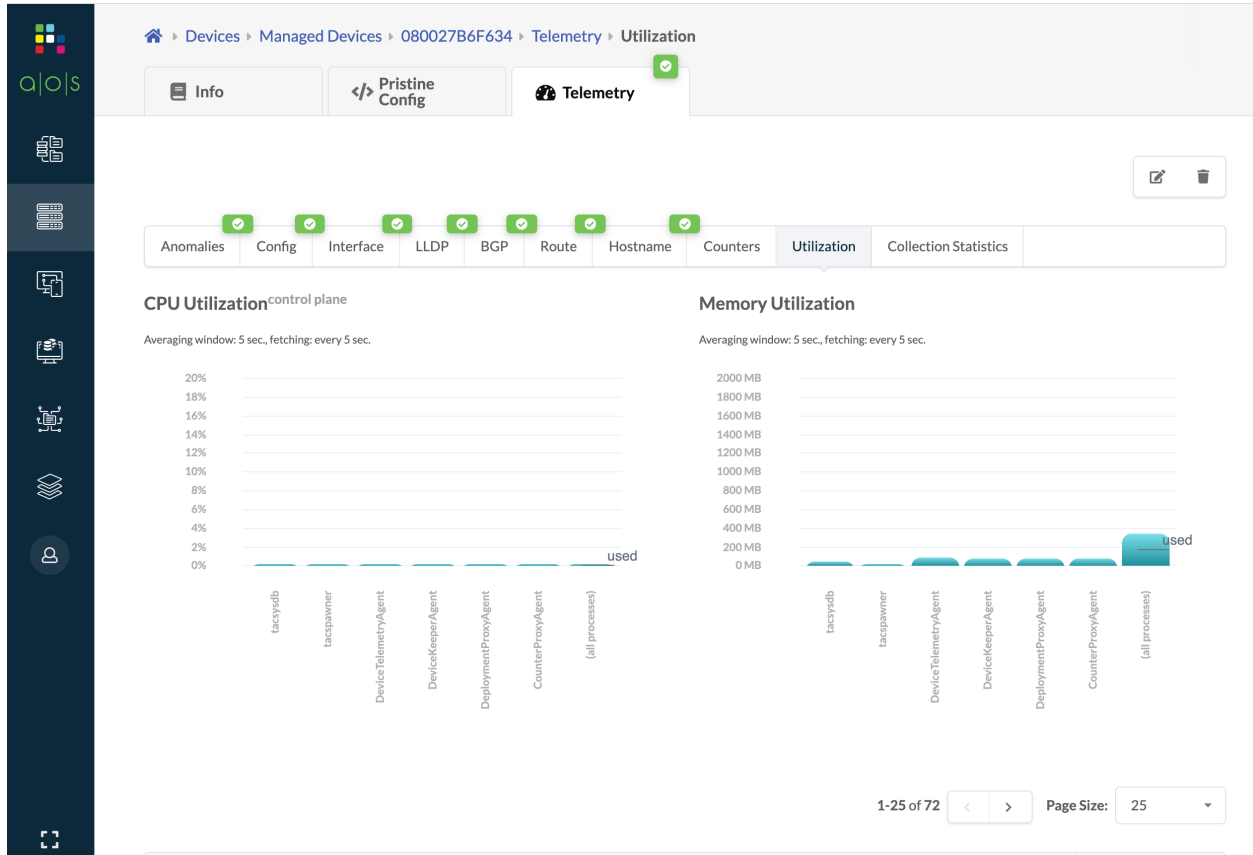
Query: All

1-1 of 1

Page Size: 25

| Device Key | Device Profile | Operation Mode | Management IP | AOS Version | Hostname | Location | OS | Acknowledged? | State | Blueprint | Comms |
|--------------|----------------|----------------|---------------|-----------------|-----------------------|----------|-----------------|---------------|-----------------|--------------|-------|
| 080027B6F634 | | TELEMETRY ONLY | 192.168.40.52 | AOS_3.2.2_OB.12 | localhost.localdomain | | CentOS 7.8.2003 | | OOS-QUARANTINED | Not assigned | |

Click on the device key and then go to the Telemetry tab to see the collected information.



5.4 Agent Profiles

Agent profiles enable the logical link between device credentials, a key-value store to be used in the device configuration, and a selection of user uploaded packages. With Agent profiles, you can configure parameters for a certain class of devices that exist in the network and edit their device agent settings as a group.

The screenshot displays the Apstra web interface for the Agent Profiles section. The interface includes a sidebar with navigation options like Blueprints, Devices, Managed Devices, Telemetry, System Agents, Agents, Agent Profiles, Packages, OS Images, and ZTP Logs. The main content area shows the Agent Profiles tab selected, displaying a table of agent profiles. The table has columns for Name, Platform, Has Username?, Has Password?, Packages Count, and Open Options Count. The first row shows 'example_profile' with Platform 'NXOS', Has Username? 'no', Has Password? 'no', Packages Count '1', and Open Options Count '0'. The interface also includes a search bar, a 'Create Agent Profile' button, and a table of agent profiles.

To access agent profiles - from the AOS web interface, navigate to **Devices / System Agents / Agent Profiles**

☆
🏠
> Devices > Agent Profiles > example_profile

✎
📄
🗑️

Expanded View
Compact View

Profile Parameters

| | |
|---------------|-----------------|
| Name | example_profile |
| Platform | NXOS |
| Has Username? | no |
| Has Password? | no |

Packages
1

| | |
|----------|--|
| Packages | aosstdcollectors-custom-nxosn3k==0.1.0.post462 |
|----------|--|

To see details - click an agent profile. Agent profiles include the following details:

Name Identifies the profile..

Platform Device OS family..

Username / Password Admin/root username and password on the device..

Open Options Key-value list for custom variables that need to be passed to agents on the device..

Packages Admin-provided software packages stored on the AOS server that can be applied to each device agent that is created using the profile..

Cloning Agent Profile

Instead of entering all details for a new agent profile, you can clone an existing one, give it a unique name and customize it.

5.4.1 Creating Agent Profile

Before creating an agent profile, upload any *packages* that are to be included in the agent profile.

1. From the list view (Devices / System Agents / Agent Profiles) click **Create Agent Profile**.
2. Enter a name, platform (EOS, CUMULUS, NXOS, SONIC, JUNOS), and optional username and password.
3. Add open options (optional).
4. Select package(s) (optional).
5. Click **Create** to create the agent profile and return to the list view.

5.4.2 Editing Agent Profile

1. Either from the list view (Devices / System Agents / Agent Profiles) or the details view, click the **Edit** button for the profile to edit.
2. Make your changes.
3. Click **Update** to update the profile and return to the list view.

5.4.3 Deleting Agent Profile

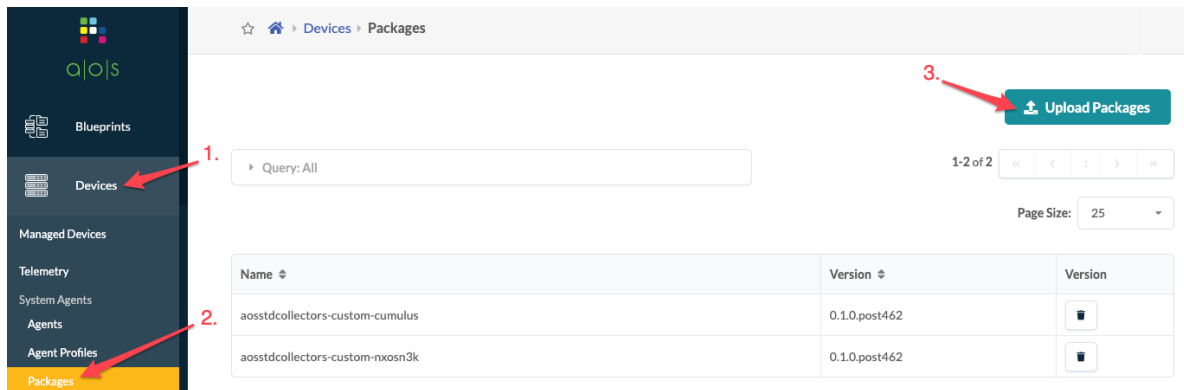
1. Either from the list view (Devices / System Agents / Agent Profiles) or the details view, click the **Delete** button for the profile to delete.
2. Click **Delete** to delete the profile and return to the list view.

5.5 Packages

You can extend Apstra capabilities to add 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 *agents* and *agent profiles*. 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.

5.5.1 Uploading Packages

1. Download required package(s) from the portal at <https://support.juniper.net/support/downloads/?p=apstra-fabric-conductor>.
2. From the web interface, navigate to **Devices > System Agents > Packages**.



3. Click **Upload Packages**, then, for each package to upload, either click **Choose File** and navigate to the downloaded file, or drag and drop the file into the dialog window.
4. Click **Upload**, then close the dialog to return to the list view.

5.6 OS Images

In preparation for *upgrading device operating systems* (DOS), OS images (obtained from device vendors) must be registered with AOS.

5.6.1 Registering Device OS Image

The screenshot shows the Apstra web interface. On the left sidebar, 'Devices' is highlighted with a red arrow and '1.' next to it. Below it, 'OS Images' is also highlighted with a red arrow and '2.' next to it. At the top right, there is a 'Register OS Image' button with a red arrow and '3.' next to it. The main content area shows a table of OS images. Above the table is a search bar with the text 'Query: All' and a red arrow pointing to it with the text 'Click to search'. To the right of the search bar are pagination controls showing '1-10 of 10' and 'Page Size: 25'. The table has columns: Name, Platform, Type, Description, Checksum, and Actions. The first row shows 'cl-3.7.12-bcm-amd64.bin' as the Name, 'CUMULUS' as the Platform, 'uploaded' as the Type, and 'cl-3.7.12-bcm-amd64.bin' as the Description. The second row shows 'cumulus-linux-3.7.12-bcm-amd64.bin' as the Name, 'CUMULUS' as the Platform, 'external' as the Type, and 'cumulus-linux-3.7.12-bcm-amd64.bin' as the Description. The third row shows 'EOS-4.20.11M.swi' as the Name, 'EOS' as the Platform, 'external' as the Type, and 'EOS-4.20.11M.swi' as the Description. The fourth row shows 'EOS-4.21.5.1F.swi' as the Name, 'EOS' as the Platform, 'external' as the Type, and 'EOS-4.21.5.1F.swi' as the Description. The fifth row shows 'EOS-4.22.3M.swi' as the Name, 'EOS' as the Platform, 'uploaded' as the Type, and 'EOS-4.22.3M.swi' as the Description. The Actions column for each row contains 'Edit' and 'Delete' icons. A red arrow points to the 'Platform' header with the text 'Click to sort'.

| Name | Platform | Type | Description | Checksum | Actions |
|---|----------|----------|------------------------------------|----------|---|
| cl-3.7.12-bcm-amd64.bin ^{URL} | CUMULUS | uploaded | cl-3.7.12-bcm-amd64.bin | | Edit Delete |
| cumulus-linux-3.7.12-bcm-amd64.bin ^{URL} | CUMULUS | external | cumulus-linux-3.7.12-bcm-amd64.bin | | Edit Delete |
| EOS-4.20.11M.swi ^{URL} | EOS | external | EOS-4.20.11M.swi | | Edit Delete |
| EOS-4.21.5.1F.swi ^{URL} | EOS | external | EOS-4.21.5.1F.swi | | Edit Delete |
| EOS-4.22.3M.swi ^{URL} | EOS | uploaded | EOS-4.22.3M.swi | | Edit Delete |

1. From the AOS web interface, navigate to **Devices / System Agents / OS Images**, then click **Register OS Image** (top-right).
2. Select the platform from the drop-down list (EOS, CUMULUS, NXOS, JUNOS), then enter a description.
3. Proceed by either *uploading an image* directly to the AOS server or by *providing a URL* download link pointing to an image file on another HTTP server.

5.6.1.1 Method One: Upload Image

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)

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**.
2. *Add a checksum* (optional).
3. Click **Upload** to upload and register the image with AOS.

HTTP URLs Only

Only HTTP URLs are supported. HTTPS, FTP, SFTP, SCP; others are not.

5.6.1.2 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 Device OS Image instead of uploading it.

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

1. Select **Provide Image URL**.
2. Enter the Image URL that is pointing to the image on another server.
3. *Add a checksum* (optional).
4. Click **Register** to register the URL in AOS.

5.6.1.3 Adding Checksum

You have the option of adding a checksum before uploading or registering an OS image. After the image is uploaded to the device being upgraded, AOS verifies the checksum. If the checksum is incorrect, AOS stops the upgrade before rebooting the device. The type of checksum is determined by the platform:

- Arista EOS - SHA512 (128 characters)
- Cisco NX-OS - SHA512 (128 characters)
- Cumulus Linux - MD5 (32 characters)
- Enterprise SONiC - MD5 (32 characters)

If the OS Image Vendor provides a checksum file, we recommend that you download it and copy the checksum 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
dbfd28d3597777a6ee5946b52277205fc714e11ab992574b7ef1156ffcd6e379979979f8c009f665fc21212e4d38d1794a412
↪ EOS-4.20.11M.swi
$
```

```
$ md5sum cumulus-linux-3.7.5-bcm-amd64.bin
MD5 (cumulus-linux-3.7.5-bcm-amd64.bin) = 8ffe069651cf4ffe8cfbe1162491761a
$
```

5.6.2 Editing OS Image

Important: AAA authentication must be disabled before proceeding with the new Image upgrade, otherwise it will lead to failure.

1. From the list view (Devices / System Agents / OS Images) click the **Edit** button for the OS Image to edit.
2. Make your changes.
3. Click **Update** to update the OS image.

5.6.3 Deleting OS Image

1. From the list view (Devices / System Agents / OS Images) click the **Delete** button for the OS Image to delete.
2. Click **Delete** to delete the OS image.

5.7 Apstra ZTP

5.7.1 Overview

Apstra ZTP is a Zero-Touch-Provisioning server for data center infrastructure systems. It enables Apstra data center devices to be bootstrapped without consideration of differences in underlying NOS mechanisms. Apstra ZTP replaces the community-supported Aeon-ZTPS software previously used by Apstra for ZTP implementation.

Zero-Touch Provisioning (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 the device agent the device communicates with and is managed by Apstra). 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 device OS
- Installing license (Cumulus only)
- On-box or Off-box Device System Agent installation

Important: ZTP uses default, hard-coded credentials for when there is a problem during the ZTP process. This prevents being locked out from the device. These credentials are:

- root / admin
- aosadmin / aosadmin

Apstra ZTP is a reference implementation. Apstra currently has two supported versions of Apstra ZTP, 1.0.0, and 2.0.0.

Apstra ZTP 1.0.0 supports versions of version 3.1.1 through version 3.3.0 and devices running Cisco NX-OS, Arista EOS, Cumulus Linux, and Juniper Junos.

Apstra ZTP 2.0.0 released with version 3.3.0a currently supports devices running Enterprise SONiC, Cisco NX-OS, Arista EOS, Cumulus Linux, and Juniper Junos.

The user has the option of using an Apstra-provided VM image (.ova, .tar.gz, .qcow2.gz, .vmdx.gz) or the user can build their 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 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 preparing it for Apstra management. This includes verifying that the device is running an Apstra-supported OS.

3. Agent Installation Phase

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

5.7.2 Apstra ZTP 2.0.0

Apstra ZTP 2.0.0 was introduced with version 3.3.0a and supports ZTP for Arista EOS, Cisco NX-OS, Cumulus Linux, Juniper Junos, and Enterprise SONiC devices.

5.7.2.1 Apstra ZTP 2.0.0 VM Server Resource Requirements

Apstra ZTP 2.0.0 runs as an Ubuntu 18.04LTS server running a DHCP, HTTP, and TFTP server and includes Apstra provided ZTP scripts that must be customized for the user's 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 |

5.7.2.2 Apstra ZTP 2.0.0 Network requirements

| Source | Destination | Ports | Role |
|--------------------------------|---|------------------|-----------------------------------|
| Device agents | DHCP Server (renewals) & Broadcast (requests) | udp/67 -> udp/68 | DHCP Client |
| Device agents | Apstra ZTP | any -> tcp/80 | Bootstrap and API scripts |
| Arista and Cisco Device agents | Apstra ZTP | any -> udp/69 | TFTP for POAP and ZTP |
| Apstra ZTP | Apstra Server | 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 Apstra. Refer to Apstra [Network Security Requirements](#) for more information.

5.7.2.3 Download and Deploy Apstra ZTP VM on Standalone Apstra ZTP VM

1. Download the appropriate Apstra 2.0.0 ZTP image from <https://support.juniper.net/support/downloads/?p=apstra-fabric-conductor>.
 - Linux KVM QCOW2 image - apstra-ztp-2.0.0-<build-version>.qcow2.gz (example: apstra-ztp-2.0.0-53.qcow2.gz)
 - VMware OVA image - apstra-ztp-2.0.0-<build-version>.ova (example: apstra-ztp-2.0.0-53.ova)
 - Microsoft Hyper-V - apstra-ztp-2.0.0-<build-version>.vhd.gz (example: apstra-ztp-2.0.0-53.vhd.gz)
2. Validate the downloaded file against the SHA512/MD5 checksums provided.
3. Deploy the VM with the appropriate resources.
4. By default, TFTP, NGINX (HTTP) and DHCP Server Docker containers will be enabled and run by default.

```

admin@apstra-ztp:~$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED
↪ STATUS          PORTS              NAMES
↪
3f6c53304501       apstra/tftp        "sh /init.sh"          4 hours ago
↪ Up 12 minutes    0.0.0.0:69->69/udp    tftp
↪
ba53ebd62b04       apstra/nginx       "sh /init.sh"          4 hours ago
↪ Up 12 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
0fbda7d068e1       apstra/status      "sh /init.sh"          4 hours ago
↪ Up 12 minutes    8080/tcp            status
↪
59a61c037096       apstra_ztp_dhcpd   "sh /init.sh"          4 hours ago
↪ Up 12 minutes
↪ dhcpd
caf901aac317       mysql:8            "docker-entrypoint.s..." 4 hours ago
↪ Up 12 minutes    3306/tcp, 33060/tcp    db
↪
admin@apstra-ztp:~$

```

5. If the user does not want to use the Apstra ZTP DHCP Server, the user can 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:~$
```

5.7.2.4 Configuring Static Management IP Address

By default, the Apstra ZTP Server will attempt to assign an IP address for its eth0 interface via DHCP. If using the Apstra ZTP Server as a DHCP server, the user must set a static management IP address.

1. Log into the Apstra server as user admin. To configure a static management IP address, edit the `/etc/netplan/01-netcfg.yaml` file.

```
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]
```

2. Apply the IP address change with the `sudo netplan apply` command, or reboot the Apstra server with the `sudo reboot` command.

For more information on using netplan, see <https://netplan.io/examples>.

5.7.2.5 Configure DHCP Server

An ISC DHCP server is provided on the Apstra ZTP VM. The user may decide to use another DHCP for the device management network. This guide describes the process to edit the supplied DHCP server. The user is responsible for configuring the same options if using a different DHCP server. For example, if using Cumulus Linux, the user must ensure their server contains the following so devices will download the Apstra ZTP `ztp.py` file.

```
option cumulus-provision-url "tftp://192.168.59.4/ztp.py";
```

On the Apstra ZTP VM, the DHCP configuration files are 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`. The user 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`.

```
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 (`fixed-address`) for device with hardware ethernet MAC. Use the Switch MAC address (`hardware ethernet`) of the management interface used for DHCP negotiations.

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 using ZTP with Cumulus Linux, the user must edit the following:

```
class "cumulus" {
    match if (substring(option host-name, 0, 7) = "cumulus");
    option cumulus-provision-url "tftp://192.168.59.4/ztp.py";
}
```

- `cumulus-provision-url`: TFTP URL with IP address of ZTP server

5. If using ZTP with SONiC, the user 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

- After modifying any DHCP configuration change, restart the Apstra ZTP DHCP process with the `sudo docker restart dhcpd` command.

```
admin@apstra-ztp:~$ docker restart dhcpd
dhcpd
admin@apstra-ztp:~$
```

5.7.2.6 Configure ZTP User on Apstra Server

The user may use any configured web interface user that has API write access (such as admin), but it is recommended that the user create a designated user (e.g. “ztp”) that is assigned the predefined role `device_ztp`. The `device_ztp` role allows users to make API calls to the Apstra server to request device system agent installation.

5.7.2.7 Configure Apstra Server IP for ZTP

The Apstra server IP and the Apstra ZTP username must be configured in the `/containers_data/status/app/aos.conf` file on the Apstra ZTP 2.0.0 Server.

```
admin@apstra-ztp:~$ sudo nano /containers_data/status/app/aos.conf
```

```
{
  "ip": "192.168.0.3",
  "user": "ztp",
  "password": "ztp-user-password"
}
```

- `ip`: The IP Address of the Apstra server

- **user:** The username of the ZTP or admin user
- **password:** The user's password

5.7.2.8 Edit ZTP Configuration File

Apstra ZTP VM includes a TFTP and nginx HTTP server. These servers do not require configuration. Both servers serve files out of the `/containers_data/tftp` directory.

```
admin@apstra-ztp:~$ sudo ls -l /containers_data/tftp/
total 220
-rw----- 1 root root 2448 Oct 21 00:35 config_verifier.py
-rw----- 1 root root 393 Oct 21 00:35 container_init.sh
-rw----- 1 root root 170 Oct 21 00:35 cumulus_custom.sh
-rw----- 1 root root 55 Oct 21 00:35 cumulus_license_file
-rw----- 1 root root 192 Oct 21 00:35 Dockerfile
-rw----- 1 root root 107 Oct 21 00:35 eos_custom.sh
-rw----- 1 root root 4721 Oct 21 00:35 junos_apstra_ztp_bootstrap.sh
-rw----- 1 root root 1799 Oct 21 00:35 junos_custom.sh
-rw----- 1 root root 86 Oct 21 00:35 nxos_custom.sh
-rwx----- 1 root root 205 Oct 21 00:35 poap-md5sum
-rw----- 1 root root 1843 Oct 21 00:35 rsyslog.conf
-rw----- 1 root root 1910 Oct 21 00:35 ztp.json
-rw----- 1 root root 84855 Oct 21 00:36 ztp.py
-rw----- 1 root root 84812 Oct 21 00:36 ztp.py.md5
admin@apstra-ztp:~$
```

The `ztp.json` file contains all the configuration for the Apstra ZTP script `ztp.py`.

1. Edit the `ztp.json` file with `vi` or `nano`.

```
admin@apstra-ztp:~$ sudo nano /containers_data/tftp/ztp.json
```

2. The `ztp.json` file is organized by the following:

- **defaults:** Values are used for all devices unless more specific keys are defined.

```
"defaults": {
  "device-root-password": "root-password-123",
  "device-user": "admin",
  "device-user-password": "admin-password-123",
  "system-agent-params": {
    "agent_type": "onbox",
    "install_requirements": false
  }
}
```

- **platform:** Values are used for all devices for a network platform (“cumulus”, “nxos”, “eos”, “junos”, “sonic”) unless more specific keys are defined.

```
"cumulus": {
  "cumulus-versions": ["3.7.12", "3.7.11"],
  "cumulus-image": "http://192.168.59.4/cumulus-linux-3.7.12-bcm-amd64.bin",
  "license": "cumulus_license_file",
  "custom-config": "cumulus_custom.sh",
}
```

- **model:** Values are used for all devices for a specific device model (e.g. “N9K-C93180YC-FX”).

```
"N9K-C93180YC-FXC3396": {
  "custom-config": "93180_cumulus_custom.sh",
}
```

- **serial number:** Values are used for a device matching a specific device serial number (e.g. “525400B3C311”).

```
"525400B3C311": {
  "cumulus-versions": [ "3.7.13" ],
  "cumulus-image": "http://192.168.59.4/cumulus-linux-3.7.13-bcm-amd64.bin"
}
```

More specific data will take precedence over other data. For example, data for a specific serial number will take precedence over any other data, then model, then platform, then finally default data.

3. The `ztp.json` file uses the following keys:

- **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": [ "7.0(3)I7(4)", "9.2(2)" ]
```

- **nxos-image:** This is filename of the NX-OS image to be loaded if the running version does not match a version in the `nxos-versions` list.

By default, the image name will be loaded from the ZTP server via TFTP from the ZTP server's / `container_data/tftp/` directory. 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.9.2.2.bin"
```

The above example will use HTTP from the ZTP server to transfer the Cisco NX-OS image.

- **cumulus-versions:** Valid versions for Cumulus Linux devices. If a device is not running a version in this list, ZTP upgrades the device with the `cumulus-image` image.

```
"cumulus-versions": [ "3.7.12", "3.7.11" ]
```

- **cumulus-image:** This is the filename of the Cumulus Linux ONIE BIN image to be loaded if the running version does not match a version in the `cumulus-versions` list.

By default, the image name will be loaded from the ZTP server via TFTP from the ZTP server's / `container_data/tftp/` directory. To use any HTTP server for image transfer, enter a valid HTTP URL with IP address. For example:

```
"cumulus-image": "http://192.168.59.4/cumulus-linux-3.7.12-bcm-amd64.bin"
```

The above example will use HTTP from the ZTP server to transfer the Cumulus Linux 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": [ "4.20.11M", "4.21.5.1F" ]
```

- **eos-image:** This is the filename of the Arista EOS SWI image to be loaded if the running version does not match a version in the `eos-versions` list.

By default, the image name will be loaded from the ZTP server via TFTP from the ZTP server's / container_data/tftp/ directory. 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-4.21.5.1F.swi"
```

The above example will use HTTP from the Apstra server to transfer the Arista EOS image.

- **junos-versions:** Valid versions for Juniper Junos devices. If a device is not running a version in this list, ZTP upgrades the device with the **junos-image** image.

```
"junos-versions": [ "18.4R3-S4.2" ]
```

- **junos-image:** This is the filename of the Juniper Junos TGZ image to be loaded if the running version does not match a version in the **junos-versions** list.

By default, the image name will be loaded from the ZTP server via TFTP from the ZTP server's / container_data/tftp/ directory. 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-5-18.4R3-S4.2-signed.tgz"
↪
```

The above example will use HTTP from the Apstra server to transfer the Juniper Junos image.

- **sonic-versions:** Valid versions for SONiC devices. If a device is not running a version in this list, ZTP upgrades the device with the **sonic-image** image.

```
"sonic-versions": [ "SONiC-OS-3.1.0a-Enterprise_Base" ]
```

- **sonic-image:** This is filename of the SONiC ONIE BIN image to be loaded if the running version does not match a version in the **sonic-versions** list.

By default, the image name will be loaded from the ZTP server via TFTP from the ZTP server's / container_data/tftp/ directory. 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"
```

The above example will use HTTP from the Apstra server to transfer the SONiC image.

- **device-root-password:** The ZTP process will set the device root password to this value. For Arista EOS and Cisco NX-OS devices, the **device-root-password** will be used to set the password for the system admin password.

```
"device-root-password": "root-admin-password"
```

- **device-user / device-user-password:** This is the username and password Apstra will use for the device system agent. Also, if necessary, the ZTP process will create a user on the device with this username and password.

```
"device-user": "aosadmin",
"device-user-password": "aosadmin-password"
```

- **license:** The filename of the Cumulus Linux license file in the TFTP directory or a URL pointing to the file on a HTTP server.

```
"license": "cumulus_license_file"
```

- `custom-config`: The filename of the custom configuration shell script in the TFTP directory or a URL pointing to the file on a HTTP server. This shell script will be run during ZTP allowing the user to add custom configuration to the device. See *Platform Specific Information* section below for more information.

```
"custom-config": "cumulus_custom.sh"
```

- `system-agent-params`: The information that Apstra uses to create a new user and device system agent on the device.

- `agent_type`: The 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`: (On-box agents only) Set to `install` to install the on-box agent on the device.

```
"job_on_create": "install"
```

- `platform`: (Required for off-box agents only) Set to the device platform (“eos”, “nxos”, “junos”). Must be in all lowercase.

```
"platform": "junos"
```

- `open_options`: (Off-box agents only) Set to enable HTTPS between off-box agent to device API interface. Connection will default to HTTP if `open_options` is not defined.

```
"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"
]
```

Refer to the Swagger Platform REST API Documentation for `/api/system-agents` for all available `system-agent-params` options.

5.7.3 Apstra ZTP 1.0.0

5.7.3.1 Apstra ZTP 1.0.0 VM Server Resource Requirements

Apstra ZTP 1.0.0 runs as an Ubuntu 18.04LTS Server running a DHCP, HTTP, and TFTP server and includes Apstra provided ZTP scripts that must be customized for the user’s environment. Below table 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 |

5.7.3.2 Apstra ZTP 1.0.0 Network requirements

| Source | Destination | Ports | Role |
|--------------------------------|---|------------------|-----------------------------------|
| Device agents | DHCP Server (renewals) & Broadcast (requests) | udp/67 -> udp/68 | DHCP Client |
| Device agents | Apstra ZTP | any -> tcp/80 | Bootstrap and API scripts |
| Arista and Cisco Device agents | Apstra ZTP | any -> udp/69 | TFTP for POAP and ZTP |
| Apstra ZTP | Apstra Server | 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 Apstra. Refer to Apstra [Network Security Requirements](#) for more information.

5.7.3.3 Download and Deploy the Apstra ZTP VM

Download the [Apstra ZTP VM image](#) Linux KVM/QEMU QCOW2 and VMware OVA images are available:

- Linux KVM QCOW2 image `apstra-ztp-<version>.qcow2.gz` (e.g. `apstra-ztp-1.0.0-48.qcow2.gz`)
- VMware OVA image `apstra-ztp-<version>.ova` (e.g. `apstra-ztp-1.0.0-48.ova`)

Validate the downloaded file against the SHA512/MD5 checksums provided.

Deploy the VM with the appropriate resources outlined in [Apstra ZTP 1.0.0 VM Server Resource Requirements](#).

5.7.3.4 Deploy Apstra ZTP on the Apstra Server VM

The user may install Apstra ZTP directly on the Apstra server, or on another Ubuntu Linux 18.04.3 LTS server with Docker installed. This involves the following steps:

1. Download the `apstra-ztp-<version>.tar.gz` (e.g. `apstra-ztp-1.0.0-48.tar.gz`) from <https://support.juniper.net/support/downloads/?p=apstra-fabric-conductor> and copy it to the Apstra server.

Note: If using the Apstra ZTP DHCP server on the Apstra server, We recommend changing the Apstra server network interface to a static IP address. See [Configuring Static Management IP Address](#) for more information.

2. On the AOS server VM Linux CLI, as root, use the tar command to extract the files to the root (/) directory:

```
admin@aos-server:~$ sudo tar zxvf apstra-ztp-1.0.0-48.tar.gz -C /
etc/apstra_ztp/
etc/apstra_ztp/version
etc/apstra_ztp/docker-compose-nohttp.yml
```

(continues on next page)

(continued from previous page)

```

etc/apstra_ztp/docker-compose.yml
containers_data/
containers_data/tftp/
containers_data/tftp/eos_custom.sh
containers_data/tftp/ztp.py
containers_data/tftp/ztp.json
containers_data/tftp/junos_custom.sh
containers_data/tftp/cumulus_custom.sh
containers_data/tftp/poap-md5sum
containers_data/tftp/cumulus_license_file
containers_data/tftp/nxos_custom.sh
containers_data/tftp/junos_apstra_ztp_bootstrap.sh
containers_data/Dockerfile.tftp
containers_data/dhcp/
containers_data/dhcp/dhcpd.conf
containers_data/dhcp/dhcpd.leases
containers_data/init

```

3. On the AOS server VM Linux CLI, as root, run the `docker-compose` command to install the Apstra ZTP dhcpd and tftp Docker containers.

Note: This operation requires Internet access to download the required packages from the Docker repository. If the Apstra server VM does not have access, the user will need to manually upload and install the dhcpd and tftp Docker containers images before running the `docker-compose` command. Contact [Juniper JTAC Apstra Support](#) for assistance.

```

admin@aos-server:~$ sudo docker-compose -f /etc/apstra_ztp/docker-compose-nohttp.
↪.yml up --detach
WARNING: The CONTAINER_DATA_BASE_DIR variable is not set. Defaulting to a blank_
↪string.
Creating network "apstra_ztp_default" with the default driver
Building tftp
Step 1/7 : FROM alpine:3.4
3.4: Pulling from library/alpine
cle54eec4b57: Pull complete
Digest: sha256:b733d4a32c4da6a00a84df2ca32791bb03df95400243648d8c539e7b4cce329c
Status: Downloaded newer image for alpine:3.4
--> b7c5ffe56db7
Step 2/7 : RUN apk update && apk add --no-cache tftp-hpa rsyslog
--> Running in 5261c8b831f6
fetch http://dl-cdn.alpinelinux.org/alpine/v3.4/main/x86_64/APKINDEX.tar.gz
fetch http://dl-cdn.alpinelinux.org/alpine/v3.4/community/x86_64/APKINDEX.tar.gz
v3.4.6-316-g63ea6d0 [http://dl-cdn.alpinelinux.org/alpine/v3.4/main]
v3.4.6-160-g14ad2a3 [http://dl-cdn.alpinelinux.org/alpine/v3.4/community]
OK: 5973 distinct packages available
fetch http://dl-cdn.alpinelinux.org/alpine/v3.4/main/x86_64/APKINDEX.tar.gz
fetch http://dl-cdn.alpinelinux.org/alpine/v3.4/community/x86_64/APKINDEX.tar.gz
(1/9) Installing libestr (0.1.10-r0)
(2/9) Installing libfastjson (0.99.2-r0)
(3/9) Installing libpgp-error (1.23-r0)
(4/9) Installing libgcrypt (1.7.9-r0)
(5/9) Installing liblogging (1.0.5-r1)
(6/9) Installing libnet (1.1.6-r2)
(7/9) Installing libuuid (2.28-r3)

```

(continues on next page)

(continued from previous page)

```

(8/9) Installing rsyslog (8.18.0-r0)
(9/9) Installing tftp-hpa (5.2-r2)
Executing busybox-1.24.2-rl4.trigger
OK: 8 MiB in 20 packages
Removing intermediate container 5261c8b831f6
---> 07754ef3a795
Step 3/7 : COPY init /
---> c964d72e19aa
Step 4/7 : VOLUME /var/tftpboot
---> Running in 231e4640f053
Removing intermediate container 231e4640f053
---> 3951abf2f5f6
Step 5/7 : EXPOSE 69/udp
---> Running in 2eaaa53dcc59
Removing intermediate container 2eaaa53dcc59
---> edf66a3456ae
Step 6/7 : ENTRYPOINT ["sh"]
---> Running in c0c705a3f73d
Removing intermediate container c0c705a3f73d
---> f929bb9c9b93
Step 7/7 : CMD ["/init"]
---> Running in cla3bed09729
Removing intermediate container cla3bed09729
---> 301c97f4656a

Successfully built 301c97f4656a
Successfully tagged apstra/tftp:latest
WARNING: Image for service tftp was built because it did not already exist. To
↳ rebuild this image you must use `docker-compose build` or `docker-compose up --
↳ build`.
Pulling dhcpd (networkboot/dhcpd)...
latest: Pulling from networkboot/dhcpd
898c46f3b1a1: Pull complete
63366dfa0a50: Pull complete
041d4cd74a92: Pull complete
6e1bee0f8701: Pull complete
114483241095: Pull complete
ef446bdcb1f0: Pull complete
Digest: sha256:fdc7ff6f265249a104f32f1d7aed0aedaf2f2fc62ea10eebf596e2af3b670477
Status: Downloaded newer image for networkboot/dhcpd:latest
Creating apstra_ztp_dhcpd_1 ... done
Creating apstra_ztp_tftp_1 ... done
admin@aos-server:~$

```

4. The user can verify the running Docker Containers with the `docker ps` command.

```

admin@aos-server:~$ docker ps
CONTAINER ID   IMAGE                                COMMAND                                  CREATED
↳ STATUS      PORTS                               NAMES
1319db66b7b6   networkboot/dhcpd                  "/entrypoint.sh eth0"                  About a minute
↳ ago        Up About a minute                  apstra_ztp_dhcpd_1
2f6e5f5ce4f3   apstra/tftp                         "sh /init"                             About a minute
↳ ago        Up About a minute    0.0.0.0:69->69/udp    apstra_ztp_tftp_1
452f4c93c5f5   aos:3.1.1-179                      "/usr/bin/aos_launch..."             About an
↳ hour ago    Up About an hour                  aos_metadb_1
40f226709b4a   aos:3.1.1-179                      "/usr/bin/aos_launch..."             About an
↳ hour ago    Up About an hour                  aos_auth_1

```

(continues on next page)

(continued from previous page)

```

021a50908640      aos:3.1.1-179      "/usr/bin/aos_launch..."  About an
↪hour ago      Up About an hour      aos_sysdb_1
b41de404ec14      aos:3.1.1-179      "/usr/bin/aos_launch..."  About an
↪hour ago      Up About an hour      aos_controller_1
ed2e77565552      nginx:1.14.2-upload  "nginx -g 'daemon of..."  About an
↪hour ago      Up About an hour      aos_nginx_1
admin@aos-server:~$

```

5.7.3.5 Using the Apstra Server VM as an HTTP image server for ZTP

By default, the ZTP process uses TFTP to transfer ZTP upgrade images to devices. The TFTP process can be slow for large images and networks with higher latency.

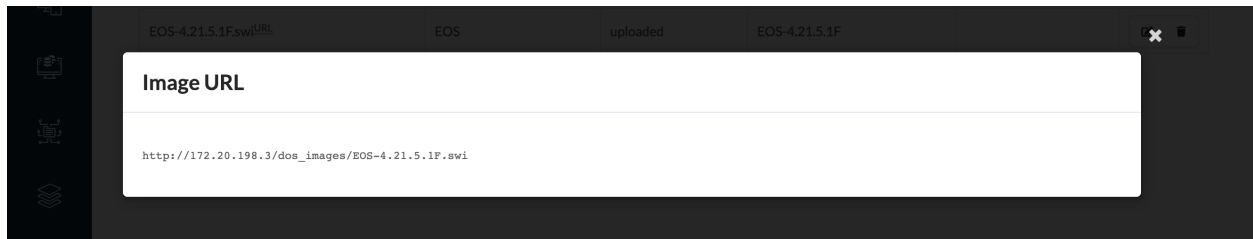
Beginning with Apstra ZTP version 1.0.0-33, the user can use HTTP to transfer ZTP upgrade images to devices.

If the user is using the Apstra ZTP server VM, a Nginx HTTP server has been added beginning with Apstra ZTP version 1.0.0-33. The HTTP server serves the same directory as the TFTP server (/containers_data/tftp).

No matter if the user is using Apstra ZTP on an Apstra ZTP VM or an Apstra server VM, the user may use the Nginx HTTP server on the Apstra ZTP server VM for ZTP images.

The user must upload “OS images” to the Apstra Server via the web interface. Information for this can be found at [Device Operating System Upgrade](#).

The HTTP URL for files will be available under **Devices / OS Images**.



By default, the Apstra server VM redirects HTTP connections to an HTTPS connection. If the Apstra server is not configured with a signed SSL certificate, the user must reconfigure the Apstra server Nginx process to not redirect for the dos_images directory.

Listing 31: /etc/aos/nginx.conf.d/nginx.conf

```

server {
    listen      80;
    server_name _;
    location /api { return 307 https://$host$request_uri; }
    location / { return 301 https://$host$request_uri; }
    # for this URI we allow plain HTTP access
    location /dos_images { root    /usr/share/nginx/html; }
}

```

After reconfiguring this file, signal Nginx to reload the configuration file.

```

root@aos-server:/etc/aos/nginx.conf.d# docker kill -s HUP aos_nginx_1
aos_nginx_1
root@aos-server:/etc/aos/nginx.conf.d#

```

5.7.3.6 Configure the Apstra ZTP Server VM

After the Apstra ZTP server completes its first boot-up, we are ready to log in and set management IP address information as well as the Apstra server IP address.

Note: Apstra ZTP expects a DHCP IP address on first boot. The user will need to connect to the Apstra ZTP server VM via a console to configure a static IP address.

Log in through the hypervisor console to configure the management IP address the first time.

Note: Initial username and password are **admin** and **admin**. The admin password should be changed with the `passwd` command.

Listing 32: Logging in to Apstra ZTP

```
Ubuntu 18.04.3 LTS apstra-ztp ttyl
apstra-ztp login: admin
Password:

Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-55-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

admin@apstra-ztp:~$
```

5.7.3.7 Management IP address

Note: If using the Apstra ZTP server as a DHCP server, the `eth0` interface must be reconfigured with a static IP address.

Configure the IP address by modifying the `/etc/netplan/01-netcfg.yaml` file with the `sudo vi /etc/netplan/01-netcfg.yaml` command (or another text editor).

Listing 33: `/etc/netplan/01-netcfg.yaml`

```
# 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
```

(continues on next page)

(continued from previous page)

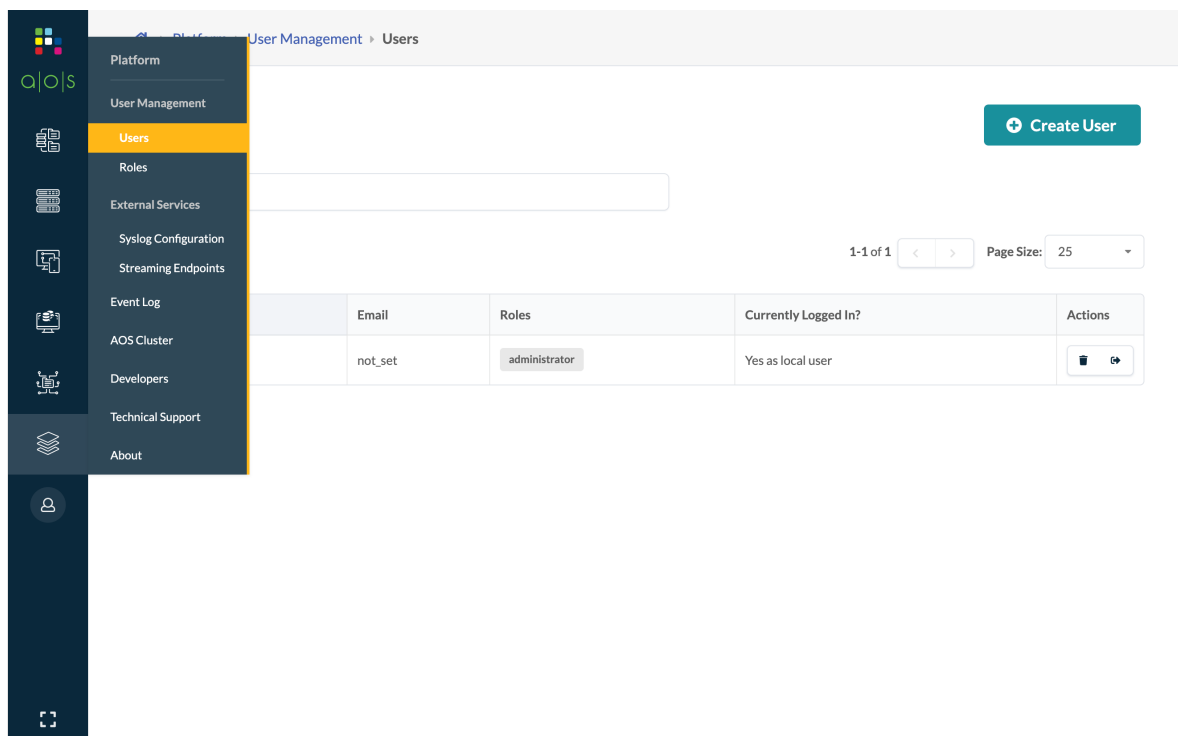
```
addresses: [192.168.59.4/24]
gateway4: 192.168.59.1
```

Apply the IP address change with the `sudo netplan apply` command or reboot the Apstra ZTP server with the `sudo reboot` command.

5.7.3.8 Configure Apstra ZTP 1.0.0 User on Apstra Server

Apstra recommends creating a “ztp” user with a Role-Based Access Control (RBAC) role of “device_ztp”. This role only allows this user to make an API call to the Apstra server to request device system agent installation.

1. From the AOS Web interface, navigate to **Platform / User Management / Users**.



2. Click **Create User** and fill in the **Username**, **Password** select the **Role** of **device_ztp**.

3. Click **Create** to create the profile.

For more information about RBAC see the [Providers](#) section.

Use of this role is optional. Users may use the `admin` Apstra user or any other configured user that has API write access.

5.7.3.9 Configuring the DHCP Server

An ISC DHCP server is provided on the Apstra ZTP VM. The user may decide to use another DHCP for the device management network. This guide describes the process to edit the supplied DHCP server. The user is responsible for configuring the same options if using a different DHCP server. For example, if using Cumulus Linux, the user must ensure their server contains the following so devices will download the Apstra ZTP `ztp.py` file.

```
option cumulus-provision-url "tftp://192.168.59.4/ztp.py";
```

On the Apstra ZTP VM, the DHCP configuration files are in the `/containers_data/dhcp` directory:

```
admin@apstra-ztp:/containers_data/dhcp$ ls -l
total 12
-rw-r--r-- 1 root          root          1526 Aug 27 19:36 dhcpd.conf
-rw-r--r-- 1 root          root           274 Aug 28 00:37 dhcpd.leases
-rw-r--r-- 1 systemd-resolve systemd-journal 274 Aug 27 19:36 dhcpd.leases~
admin@apstra-ztp:/containers_data/dhcp$
```

Note: All configuration files are owned by `root`. The user must use `sudo` to run commands as `root`.

1. Edit the `dhcpd.conf` file with `vi` or `nano`.

```
admin@apstra-ztp:/containers_data/dhcp$ sudo nano 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 (`fixed-address`) for device with `hardware ethernet` MAC. Use the Switch MAC address of the management interface used for DHCP negotiations.

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;
```

Warning: Incorrect DNS settings may result in timeouts as the server attempts to resolve IP addresses. When in doubt, remove the DNS section.

1. If using ZTP with Cumulus Linux, the user must edit the following:

```
class "cumulus" {
    match if (substring(option host-name, 0, 7) = "cumulus");
    option cumulus-provision-url "tftp://192.168.59.4/ztp.py";
}
```

- `cumulus-provision-url`: TFTP URL with IP address of ZTP server

After modifying any DHCP configuration change, restart the Apstra ZTP DHCP process with the `sudo docker restart apstra_ztp_dhcpd_1` command.

```
admin@apstra-ztp:~$ sudo docker restart apstra_ztp_dhcpd_1
apstra_ztp_dhcpd_1
admin@apstra-ztp:~$
```

5.7.3.10 Editing the ZTP Configuration File

A TFTP server is provided on the Apstra ZTP VM. The TFTP server requires no configuration. The TFTP server serves files out of the `/containers_data/tftp` directory.

```
admin@apstra-ztp:/containers_data/tftp$ ls -l
total 56
-rw-r--r-- 1 root root 170 Aug 27 19:36 cumulus_custom.sh
-rw-r--r-- 1 root root 55 Aug 27 19:36 cumulus_license_file
-rw-r--r-- 1 root root 82 Aug 27 19:36 eos_custom.sh
-rw-r--r-- 1 root root 86 Aug 27 19:36 nxos_custom.sh
-rw-r--r-- 1 root root 117 Aug 27 19:36 poap-md5sum
-rw-r--r-- 1 root root 2103 Aug 27 19:36 ztp.json
-rw-r--r-- 1 root root 35100 Aug 27 19:36 ztp.py
admin@apstra-ztp:/containers_data/tftp$
```

Note: All configuration files are owned by `root`. The user must use `sudo` to run commands as `root`.

The `ztp.json` file contains all the configuration for the Apstra ZTP script `ztp.py`.

1. Edit the `ztp.json` file with `vi` or `nano`.

```
admin@apstra-ztp:/containers_data/tftp$ sudo nano 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.
- **Platform:** Values are used for all devices for a network platform (“cumulus”, “nxos”, “eos”) unless more specific keys are defined.
- **Model:** Values are used for all devices for a specific device model (e.g. “C3396”).
- **Serial Number:** Values are used for a device matching a specific device serial number.

```
{
  "defaults": {
    "_comment# if nxos-versions is empty, we will just assume all_
    versions are accepted": 0,
    "nxos-versions": ["9.2(2)", "7.0(3)I7(4)"],
    "nxos-image": "aos_nxos_image.bin",
    .....
  },

  "cumulus": {
    "device-root-password": "admin123",
    "custom-config": "cumulus_custom.sh"
  },

  "nxos": {
    "custom-config": "nxos_custom.sh"
  },

  "C3396": {
    "Device-root-password": "c3396-custom-pass"
  }
}
```

(continues on next page)

(continued from previous page)

```

    "some-serial-number": {
        "nxos-image": "device-specific-nxos-image"
    }
}

```

3. The `ztp.json` file uses the following keys:

- `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-image`: This is filename of the NX-OS image to be loaded if the running version does not match a version in the `nxos-versions` list.

By default, the image name will be loaded from the ZTP server via TFTP from the ZTP server's / `container_data/tftp/` directory. Beginning in Apstra ZTP version 1.0.0-33, the user can use HTTP to transfer ZTP upgrade images to devices. To use any HTTP server for image transfer, enter a valid HTTP URL with IP address for `nxos-image`. For example:

```

"nxos-versions": [ "7.0(3)I7(4)", "9.2(2)" ],
"nxos-image": "nxos.9.2.2.bin",

```

The above example will use TFTP from the ZTP server to transfer the Cisco NX-OS image.

- `cumulus-versions`: Valid versions for Cumulus Linux devices. If a device is not running a version in this list, ZTP upgrades the device with the `cumulus-image` image.
- `cumulus-image`: This is filename of the Cumulus Linux ONIE image to be loaded if the running version does not match a version in the `cumulus-versions` list.

By default, the image name is loaded from the ZTP server via TFTP from the ZTP server's / `container_data/tftp/` directory. Beginning in Apstra ZTP version 1.0.0-33, the user can use HTTP to transfer ZTP upgrade images to devices. To use any HTTP server for image transfer, enter a valid HTTP URL with IP address for `cumulus-image`. For example:

```

"cumulus-versions": ["3.7.11", "3.7.12"],
"cumulus-image": "http://192.168.59.4/cumulus-linux-3.7.12-bcm-amd64.bin",

```

The above example will use HTTP from the ZTP server to transfer the Cumulus Linux 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-image`: This is the filename of the Arista EOS SWI image to be loaded if the running version does not match a version in the `eos-versions` list.

By default, the image name is loaded from the ZTP server via TFTP from the ZTP server's / `container_data/tftp/` directory. Beginning in Apstra ZTP version 1.0.0-33, the user can use HTTP to transfer ZTP upgrade images to devices. To use any HTTP server for image transfer, enter a valid HTTP URL with IP address for `eos-image`. For example:

```

"eos-versions": ["4.21.5.1F", "4.22.3M"],
"eos-image": "http://192.168.59.4/dos_images/EOS-4.22.3M.swi",

```

The above example will use HTTP from the Apstra server to transfer the Arista EOS image.

- `aos-server`: The IP Address of the Apstra server.
- `aos-user`: The ZTP user that has been configured on the Apstra server (such as "ztp"), see [Configure Apstra ZTP 1.0.0 User on Apstra Server](#).

- **aos-password:** The password configured for the ZTP user on the Apstra Server.
- **device-root-password:** The ZTP process will set the device root password to this value
- **license:** The filename of the Cumulus Linux license file in the TFTP directory.
- **custom-config:** The filename of the custom configuration file in the TFTP directory. See NOS-specific section below for more information.
- **system-agent-params:** The information that Apstra uses to create a new user and device system agent on the device.
 - **username:** The username of an account on the device that is used to create the device system agent. This user will be newly created on the device.
 - **password:** The password for the new account on the device that is used to create the device system agent.
 - **agent_type:** The agent type, onbox or offbox.
 - **installer_options:** (only for version 3.1) Options for the device system agent.
 - * **enable:** Set `false` to let the user manually enable the agent. Set `true` to automatically enable the agent.
 - * **intent:** Set to `install` to install the device system agent.
 - * **install_policy:** Set to `once`.
 - * **install_requirements:** Set to `false`. Not needed for Cumulus Linux, Cisco NX-OS or Arista EOS.
 - **job_on_create:** (only for version 3.2) Set to `install` to install the on-box agent. Do not use this option for off-box agents.

Refer to the Swagger Platform REST API documentation for `/api/system-agents` for all available `system-agent-params` options.

Note: To skip agent install, only specify `username` and `password` under `system-agent-params`.

5.7.4 Platform Specific Information

5.7.4.1 Cumulus Linux

Note: Apstra ZTP has limited support for virtual Cumulus VX (CVX) devices.

- ZTP Cumulus Linux upgrades are not supported on CVX devices. Cumulus Linux versions for CVX device must match `cumulus-versions` set in `ztp.json` file.
 - ZTP Logging to an Apstra controller does not work for CVX devices due to an unsupported device serial number (MAC address). This will be addressed in a future version of Apstra.
-

Ensure that sufficient disk space is available on the switch. As part of the ZTP process a new OS image will be copied to the switch. To make sure this step does not fail it is helpful to ensure beforehand there is sufficient disk space on the device.


```
cumulus@cumulus:~$ df -h /dev/sda4
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda4        5.8
```

If ZTP is installing Cumulus Linux image, the image (e.g. `cumulus-linux-3.7.12-bcm-amd64.bin`) must be copied to the `/containers_data/tftp` directory. The file permissions should be modified to allow for download via tftp or http, for example:

```
admin@aos-server:/containers_data/tftp$ sudo chmod a+r cl-3.7.12-mlx-amd64.bin
[sudo] password for admin:
admin@aos-server:/containers_data/tftp$
```

Important: If the user is using ONIE to install Cumulus Linux on a device, the image must be copied to the `/containers_data/tftp` directory and renamed `onie-installer` or another ONIE download name (e.g. `onie-installer-x86_64-dell_s3000_c2338-r0`). When rebooting in ONIE, the device will look for this file on the TFTP server. ZTP will fail if the file is not found.

Example Cumulus Linux Apstra ZTP `ztp.json`

Listing 34: 3.2.0 On-box Agent / Apstra ZTP 1.0.0

```
{
  "cumulus": {
    "cumulus-versions": ["3.7.11"],
    "cumulus-image": "http://192.168.59.4/cumulus-linux-3.7.11-bcm-amd64.bin",
    "aos-server": "192.168.59.3",
    "aos-user": "ztp",
    "aos-password": "ztp-password",
    "device-root-password": "root-password",
    "license": "cumulus_license_file",
    "custom-config": "cumulus_custom.sh",
    "system-agent-params": {
      "username": "admin",
      "password": "admin-password",
      "agent_type": "onbox",
      "job_on_create": "install"
    }
  }
}
```

Listing 35: 3.3.0a/3.3.0c On-box Agent / Apstra ZTP 2.0.0

```
{
  "cumulus": {
    "cumulus-versions": ["3.7.12"],
    "cumulus-image": "http://192.168.59.4/cumulus-linux-3.7.12-bcm-amd64.bin",
    "device-root-password": "root-password",
    "license": "cumulus_license_file",
    "custom-config": "cumulus_custom.sh",
    "device-user": "admin",
    "device-user-password": "admin-password",
    "system-agent-params": {
```

(continues on next page)

(continued from previous page)

```

    "agent_type": "onbox",
    "job_on_create": "install"
  }
}
}

```

Cumulus Linux Custom Config File

When configuring custom-config for Cumulus Linux devices, refer to the example `cumulus_custom.sh` which is a bash executable file executed during the ZTP process. It can set the SSH login banner or other system configuration to be set prior to device system agent installation.

```

#!/bin/bash

sed -i s/"#Banner.*"/"Banner \/\etc\/issue.net"/ /etc/ssh/sshd_config

cat >& /etc/issue.net << EOF
Provisioned by AOS
Date: $(date)
EOF

service ssh restart

```

Restarting Cumulus Linux ZTP

Warning: If an agent is already installed on the device, the user must remove the agent either via the web interface device agent installer or manually via the device CLI before restarting the device ZTP process.

```

admin@cumulus:mgmt-vrf:~$ sudo dpkg -r aos-device-agent
(Reading database ... 25366 files and directories currently installed.)
Removing aos-device-agent (2.0.0-210) ...
Processing triggers for systemd (215-17+deb8u4) ...

```

See the documentation how to [Remove the AOS package from Cumulus](#) for more information.

To restart the ONIE install and Cumulus Linux ZTP process:

```

cumulus@cumulus:mgmt-vrf:~$ sudo curl -o image.bin tftp://<tftpserver-ip>/cumulus-
↪linux-3.7.5-bcm-amd64.bin
cumulus@cumulus:mgmt-vrf:~$ sudo onie-install -a -f -i image.bin
cumulus@cumulus:mgmt-vrf:~$ sudo reboot

```

To just do the Cumulus Linux ZTP process:

```

cumulus@cumulus:mgmt-vrf:~$ sudo echo "cumulus" >& /etc/hostname
cumulus@cumulus:mgmt-vrf:~$ sudo net del vrf mgmt && net commit
cumulus@cumulus:mgmt-vrf:~$ sudo ztp -R && sudo ztp -s
cumulus@cumulus:mgmt-vrf:~$ sudo reboot

```

Troubleshooting Cumulus Linux ZTP

When in ZTP mode, the switch downloads the `ztp.py`, `ztp.json`, OS image and license files to the `/mnt/persist` directory. For diagnostics, take note of the `/mnt/persist/aosztp.log` file.

Additional useful messages can be found in `/var/log/syslog` (search for ‘ztp’)

5.7.4.2 Cisco NX-OS

Ensure that sufficient disk space is available on the switch. As part of the ZTP process a new OS image will be copied to the switch. To make sure this step does not fail it is helpful to ensure beforehand there is sufficient disk space on the device.

```
switch1# dir bootflash: | include free|total
1296171008 bytes free
3537219584 bytes total
```

If ZTP is installing Cisco NX-OS image, the image (e.g. `nxos.7.0.3.I7.7.bin`) must be copied to the `/containers_data/tftp` directory ensuring correct file permissions.

Example Cisco NX-OS Apstra ZTP `ztp.json`

Listing 36: 3.2.0 On-box Agent / Apstra ZTP 1.0.0

```
{
  "nxos": {
    "nxos-versions": ["9.2(2)"],
    "nxos-image": "http://192.168.0.6/nxos.9.2.2.bin",
    "aos-server": "192.168.0.3",
    "aos-user": "ztp",
    "aos-password": "ztp-password",
    "device-root-password": "admin-password",
    "custom-config": "nxos_custom.sh",
    "system-agent-params": {
      "username": "admin",
      "password": "admin-password",
      "agent_type": "onbox",
      "job_on_create": "install"
    }
  }
}
```

Listing 37: 3.2.0 Off-box Agent / Apstra ZTP 1.0.0

```
{
  "nxos": {
    "nxos-versions": ["9.2(2)"],
    "nxos-image": "http://192.168.0.6/nxos.9.2.2.bin",
    "aos-server": "192.168.0.3",
    "aos-user": "ztp",
    "aos-password": "ztp-password",
    "device-root-password": "admin-password",
    "custom-config": "nxos_custom.sh",
    "system-agent-params": {
```

(continues on next page)

(continued from previous page)

```

    "username": "admin",
    "password": "admin-password",
    "agent_type": "offbox",
    "operation_mode": "full_control",
    "platform": "nxos",
    "open_options": {
        "proto": "https",
        "port": "443"
    },
    "packages": [
        "aos-deployment-helper-nxos",
        "aosstdcollectors-builtin-nxos",
        "aosstdcollectors-custom-nxos"
    ]
}
}
}

```

This configuration will enable secure off-box agent HTTPS (port 443) between the off-box agent on the server and the device API.

Cisco NX-OS Custom Config File

When configuring custom-config for Cisco NX-OS devices, refer to the example `nxos_custom.sh`, which is a bash executable file executed during the ZTP process. It can execute NX-OS configuration commands to set the SSH login banner or other system configuration to be set prior to device system agent installation.

Note: The user 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 off-box agents, the user must have the custom-config file enable the following NX-OS configuration commands.

```

feature nxapi
feature bash-shell

```

(continues on next page)

(continued from previous page)

```
feature scp-server
feature evmed
copp profile strict
nxapi http port 80
```

The following `nxos_custom.sh` can be used to add these along with a banner.

```
#!/bin/sh

/isan/bin/vsh -c "conf ; feature nxapi ; nxapi http port 80 ; 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.
#####
~"
```

Restarting Cisco NX-OS ZTP

Warning: If an agent is already installed on the device, the user must remove the agent either via the web interface device agent installer or manually via the device CLI before restarting the device ZTP process.

```
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 the documentation about [Uninstalling the AOS device agent](#) for more information.

To restart Cisco NX-OS ZTP process:

```
switch# write erase
switch# reload
```

5.7.4.3 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 an Apstra controller will not work for vEOS devices due to lack of a device serial number. This will be addressed in a future version of Apstra.

Ensure that sufficient disk space is available on the switch. As part of the ZTP process a new OS image will be copied to the switch. To make sure this step does not fail it is helpful to ensure beforehand there is sufficient disk space on the device.

```
switch1#dir flash:
Directory of flash:/

<...>

3957878784 bytes total (3074723840 bytes free)
```

If ZTP is installing Arista EOS image, the image (e.g. EOS-4.22.3M.swi) must be copied to the /containers_data/tftp directory.

Example Arista EOS Apstra ZTP ztp.json

Listing 38: 3.2.0 On-box Agent / Apstra ZTP 1.0.0

```
{
  "eos": {
    "eos-versions": ["4.22.3M"],
    "eos-image": "http://192.168.59.4/EOS-4.22.3M.swi",
    "aos-server": "192.168.59.3",
    "aos-user": "ztp",
    "aos-password": "ztp-password",
    "device-root-password": "admin-password",
    "custom-config": "eos_custom.sh",
    "system-agent-params": {
      "username": "admin",
      "password": "admin-password",
      "agent_type": "onbox",
      "job_on_create": "install"
    }
  }
}
```

Listing 39: 3.2.0 Off-box Agent / Apstra ZTP 1.0.0

```
{
  "eos": {
    "eos-versions": ["4.21.9M"],
    "eos-image": "http://192.168.59.4/EOS-4.21.9M.swi",
    "aos-server": "192.168.59.3",
    "aos-user": "ztp",
    "aos-password": "ztp-password",
    "device-root-password": "admin-password",
    "custom-config": "eos_custom.sh",
    "system-agent-params": {
      "username": "admin",
      "password": "admin-password",
      "agent_type": "offbox",
      "operation_mode": "full_control",
      "platform": "eos",
      "open_options": {
```

(continues on next page)

(continued from previous page)

```

        "proto": "https",
        "port": "443"
    }
}
}
}

```

Listing 40: 3.3.0a/3.3.0c On-box Agent / Apstra ZTP 2.0.0

```

{
  "eos": {
    "eos-versions": [ "4.22.3M" ],
    "eos-image": "http://192.168.59.3/EOS-4.22.3M.swi",
    "custom-config": "eos_custom.sh",
    "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` which is a bash executable file executed during the ZTP process. It can execute EOS configuration commands to set the SSH login banner or other system configuration to be set prior to device system agent installation.

```

#!/bin/sh

FastCli -p 15 -c $'conf t\n service routing protocols model multi-agent\n hardware_
↪tcam\n system profile vxlan-routing\n banner login\n
#####
UNAUTHORIZED ACCESS TO THIS DEVICE IS PROHIBITED
#####\n EOF\n'

```

Note: During the ZTP process, the EOS banner login is set to text saying “The device is in Zero Touch Provisioning mode ...”. By default, this will be copied to the permanent configuration by the ZTP script.

To prevent this, the user **must** configure the custom-config pointing to a script (e.g. `eos_custom.sh`), which will configure a different banner login or configure no banner login.

There must be a space after any `\n`.

Note: For users 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 (e.g. `system profile vxlan-routing`). This ensures that this configuration is applied on reboot and added to the device pristine configuration.

Restarting Arista EOS ZTP

Warning: If an Agent is already installed on the device, the user must remove the Agent extension either via the UI Device Agent Installer or manually via the device CLI before restarting the device ZTP process.

```
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#
```

See [Arista Device Agent](#) for more information.

To restart Arista EOS ZTP process:

```
localhost# delete flash:zerotouch-config
localhost# write erase
localhost# reload
```

5.7.4.4 Juniper Junos

Note: Apstra ZTP 1.0.0-48 used with version 3.3.0 has support for Juniper Junos devices. ZTP is supported for devices which start the ZTP process with Junos version 14.1, 15.1, and 17.3 only.

Apstra manages Juniper Junos devices using off-box agents. Apstra ZTP manages the bootstrap and the lifecycle of Juniper Junos devices. Apstra ZTP handles off-box agent creation, local user creation and can set other system configuration using a custom script.

Ensure that sufficient disk space is available on the switch. As part of the ZTP process a new OS image will be copied to the switch. To make sure this step does not fail it is helpful to ensure beforehand there is sufficient disk space on the device.

```
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 Apstra ZTP ztp.json

Listing 41: 3.3.0 Off-box Agent / Apstra ZTP 1.0.0

```
{
  "junos": {
```

(continues on next page)

(continued from previous page)

```

"junos-version": [ "18.4R2-S4.10" ],
"junos-image": "http://192.168.59.4/jinstall-host-qfx-5-18.4R2-S4.10-signed.tgz",
"aos-password": "admin",
"aos-user": "aos-admin-password",
"aos-server": "192.168.59.3",
"device-root-password": "root-password"
"system-agent-params": {
  "username": "admin",
  "password": "admin-password",
  "platform": "junos",
  "agent_type": "offbox",
  "operation_mode": "full_control"
}
}
}

```

Listing 42: 3.3.0a/3.3.0c Off-box Agent / Apstra ZTP 2.0.0

```

{
  "junos": {
    "junos-versions": [ "18.4R2-S4.10" ],
    "junos-image": "http://192.168.59.4/jinstall-host-qfx-5-18.4R2-S4.10-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"
    }
  }
}

```

Juniper Junos Bootstrap File

Apstra ZTP uses a python script `ztp.py` to provision the device during ZTP. On Junos, additional configurations are required to allow a python script to run on the device. The script `junos_apstra_ztp_bootstrap.sh` is used to bootstrap Apstra ZTP on Junos. It downloads the ZTP script and runs it.

Juniper Junos Custom Config File

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

```

#!/bin/sh

SOURCE_IP=$(cli -c "show conf interfaces em0.0" | grep address | sed 's/.*address_
↪ \([0-9.]*\).*\/1/')

# Syslog
SYSLOG_SERVER="192.168.59.4"
SYSLOG_PORT="514"

```

(continues on next page)

(continued from previous page)

```
# 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"
```

Warning: If the user sets external AAA authentication (e.g. authentication-order), the user will need to ensure that the device system agent device-user and device-user-password are replicated in the AAA system otherwise the user will get an authentication error for the device system agent.

Restarting Juniper Junos ZTP

To erase (zeroize) the device and restart Juniper Junos ZTP process:

```
root@leaf3> request system zeroize
```

5.7.4.5 Enterprise SONiC

Note: Apstra ZTP 2.0.0 used with 3.3.0a/3.3.0c has support for SONiC Enterprise Distribution devices. There is no support for any SONiC devices with earlier versions of Apstra ZTP and Apstra.

Apstra manages Enterprise SONiC devices only using on-box agents. Apstra ZTP manages the bootstrap and the lifecycle of Enterprise SONiC devices. Apstra ZTP handles on-box agent creation, local user creation and can set other system configuration using a custom script.

Ensure that sufficient disk space is available on the switch.

Important: If the user is using ONIE to install Enterprise SONiC on a device, the image must be copied to the /containers_data/tftp directory and renamed onie-installer or another ONIE download name (e.g. onie-installer-x86_64-dell_z9100_c2538-r0). When rebooting in ONIE, the device will look for this file on the HTTP then TFTP server. ZTP will fail if the file is not found. Once ONIE SONiC installation successfully completes, the SONiC device will automatically start ZTP.

Example Enterprise SONiC Apstra ZTP ztp.json

Listing 43: 3.3.0a On-box Agent / Apstra ZTP 2.0.0

```
{
  "sonic": {
    "sonic-versions": [ "SONiC-OS-3.1.0a-Enterprise_Base" ],
    "sonic-image": "http://192.168.59.4/sonic-3.1.0a-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"
    }
  }
}
```

Listing 44: 3.3.0a On-box Agent / Apstra ZTP 2.0.0

```
{
  "sonic": {
    "sonic-versions": [ "SONiC-OS-3.1.0a-Enterprise_Base" ],
    "sonic-image": "http://192.168.59.4/sonic-3.1.0a-bcm.bin",
    "device-root-password": "root-password",
    "device-user": "aosadmin",
    "device-user-password": "aosadmin-password",
    "custom-config": "sonic_custom.sh",
    "system-agent-params": {
      "agent_type": "onbox",
      "job_on_create": "install"
    }
  }
}
```

Note: If the user uses another device-user besides admin (e.g. aosadmin) Apstra ZTP will create this new user, but will 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` which is a bash executable file executed during the ZTP process. It can set system configuration e.g. 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
```

Restarting 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:~$
```

5.7.5 Monitoring DHCP, TFTP and HTTP Logs

From the Apstra ZTP VM, the user may monitor logs from the DHCP Server (if used), the TFTP server and the HTTP server with the `docker logs` commands for the `apstra_ztp_dhcpd_1`, `apstra_ztp_tftp_1`, and `apstra_ztp_http_1` Docker containers.

```
admin@apstra-ztp:~$ docker logs --tail 10 apstra_ztp_dhcpd_1
DHCPACK on 192.168.59.30 to 50:00:00:0a:00:00 (cumulus) via eth0
DHCPREQUEST for 192.168.59.24 from 50:00:00:03:00:00 (spine1) via eth0
DHCPACK on 192.168.59.24 to 50:00:00:03:00:00 (spine1) via eth0
DHCPREQUEST for 192.168.59.26 from 00:50:00:00:06:00 (ubuntu-xenial) via eth0
DHCPACK on 192.168.59.26 to 00:50:00:00:06:00 (ubuntu-xenial) via eth0
DHCPDISCOVER from 50:00:00:0b:00:00 via eth0
DHCPPOFFER on 192.168.59.31 to 50:00:00:0b:00:00 via eth0
DHCPREQUEST for 192.168.59.31 (192.168.59.4) from 50:00:00:0b:00:00 via eth0
DHCPACK on 192.168.59.31 to 50:00:00:0b:00:00 via eth0
DHCPRELEASE of 192.168.59.31 from 50:00:00:0b:00:00 via eth0 (found)
admin@apstra-ztp:~$
```

The following log messages indicate successful download of the various files. If none or some of these are seen this is an indication the ZTP process did not run correctly. The user will need to review their ZTP configuration against what has been detailed in this document:

```
root@apstra-ztp:/containers_data/tftp# docker logs --tail 10 apstra_ztp_tftp_1
2019-10-07T15:14:14.695422+00:00 e4f048f72774 in.tftpd[4193]: RRQ from 192.168.59.10_
↪ filename ztp.py
```

(continues on next page)

(continued from previous page)

```

2019-10-07T15:14:14.955079+00:00 e4f048f72774 in.tftpd[4194]: RRQ from 192.168.59.10
↳filename ztp.json
2019-10-07T15:14:15.037284+00:00 e4f048f72774 in.tftpd[4195]: RRQ from 192.168.59.10
↳filename ztp.py
2019-10-07T15:14:15.703449+00:00 e4f048f72774 in.tftpd[4197]: RRQ from 192.168.59.10
↳filename cumulus_license_file
2019-10-07T15:14:36.020385+00:00 e4f048f72774 in.tftpd[4218]: RRQ from 192.168.59.10
↳filename cumulus_custom.sh

```

```

admin@apstra-ztp:~# docker logs --tail 10 apstra_ztp_http_1
2019/12/16 18:36:43 [error] 6#6: *1 directory index of "/usr/share/nginx/html/" is
↳forbidden, client: 10.1.252.70, server: localhost, request: "GET / HTTP/1.1", host:
↳"192.168.59.6"
10.1.252.70 - - [16/Dec/2019:18:36:43 +0000] "GET / HTTP/1.1" 403 555 "-" "Mozilla/5.
↳0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/
↳79.0.3945.79 Safari/537.36" "-"
2019/12/16 18:36:43 [error] 6#6: *1 open() "/usr/share/nginx/html/favicon.ico" failed
↳(2: No such file or directory), client: 10.1.252.70, server: localhost, request:
↳"GET /favicon.ico HTTP/1.1", host: "192.168.59.6", referer: "http://192.168.59.6/"
10.1.252.70 - - [16/Dec/2019:18:36:43 +0000] "GET /favicon.ico HTTP/1.1" 404 555
↳"http://192.168.59.6/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/
↳537.36 (KHTML, like Gecko) Chrome/79.0.3945.79 Safari/537.36" "-"
10.1.252.70 - - [16/Dec/2019:18:51:46 +0000] "GET /EOS-4.21.5.1F.swi HTTP/1.1" 200
↳749757299 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36
↳(KHTML, like Gecko) Chrome/79.0.3945.79 Safari/537.36" "-"
192.168.59.101 - - [16/Dec/2019:19:16:08 +0000] "GET /EOS-4.21.5.1F.swi HTTP/1.1" 200
↳749757299 "-" "lftp/4.6.4" "-"
admin@apstra-ztp:~#

```

See the `docker logs --help` command for more information.

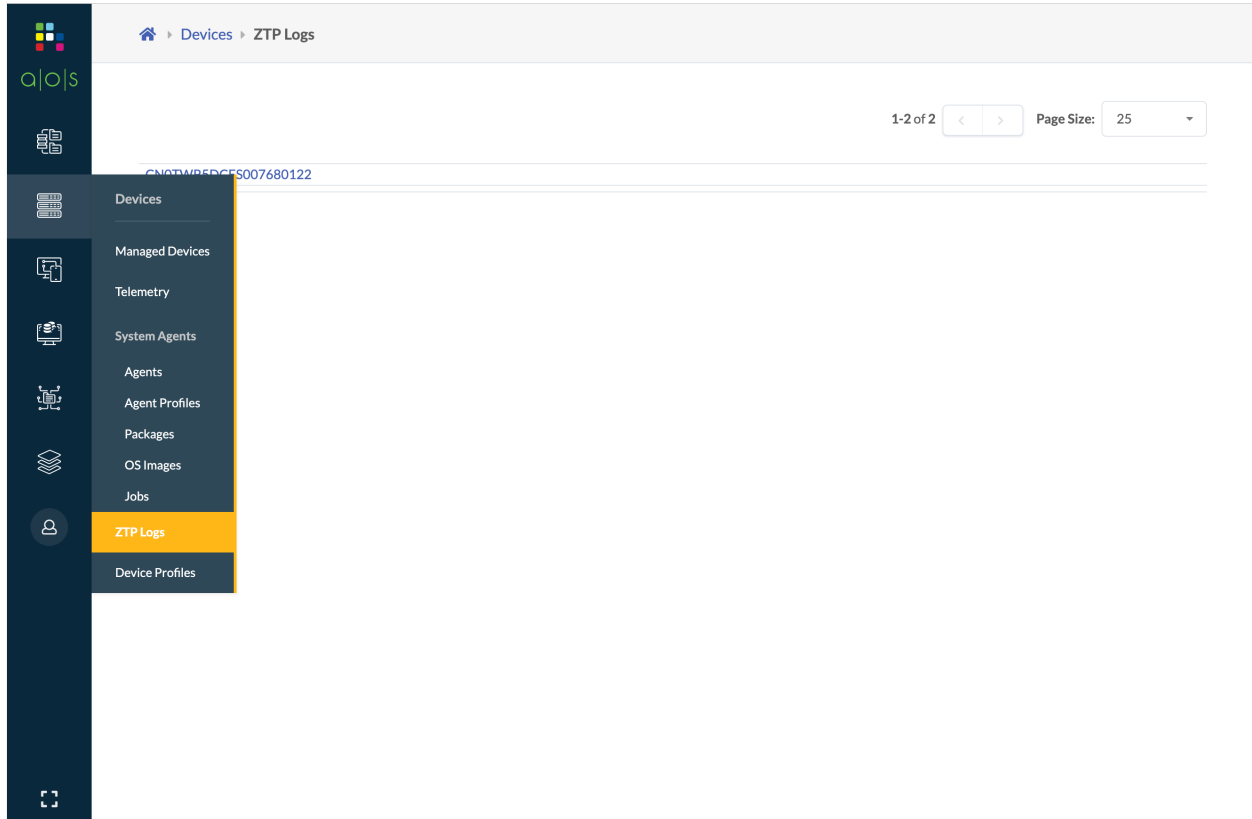
5.7.5.1 ZTP Logs

Note: There is currently no support for ZTP logs for virtual Arista EOS (vEOS) and virtual Cumulus VX (CVX) devices.

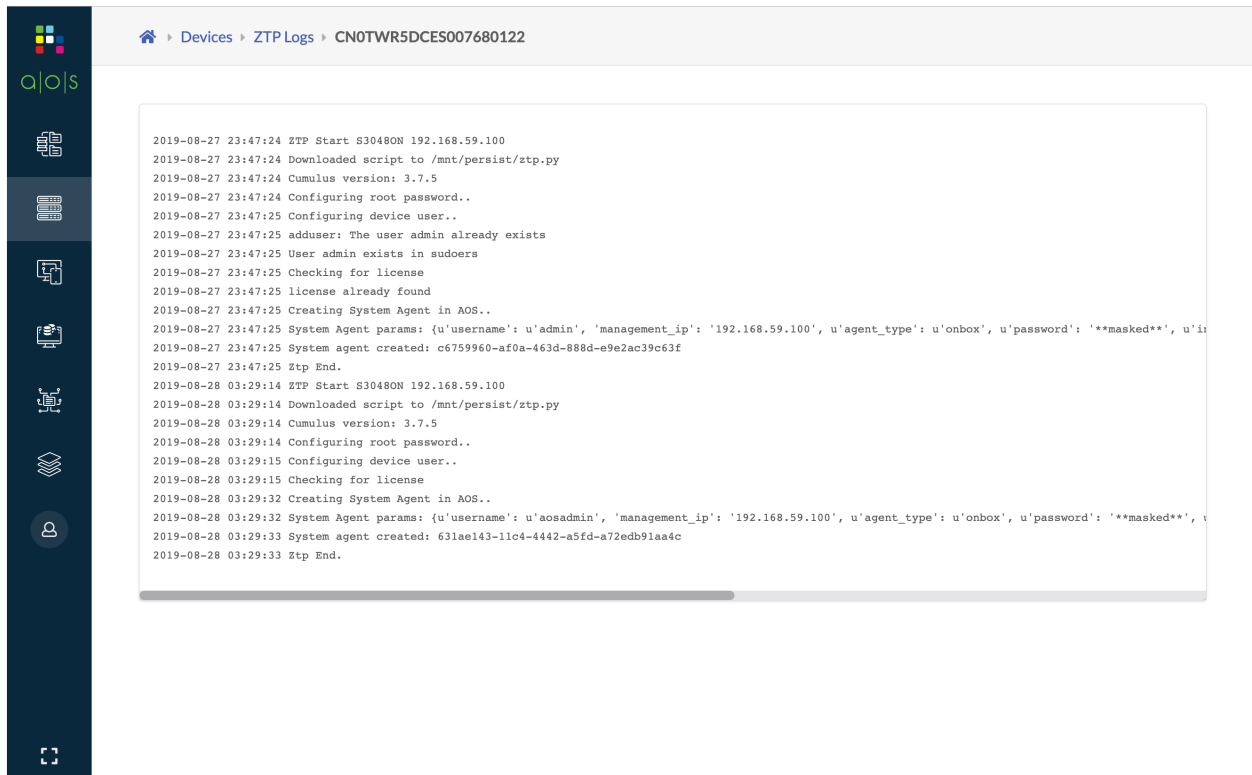
The ZTP script, once executed, will send logs to the Apstra server via API.

Monitoring Apstra ZTP 1.0.0

Apstra ZTP 1.0.0 users can monitor the ZTP process from the web interface by going to **Devices / ZTP Logs**.



Click on the serial number of the device to view the logs.



The user can monitor the device system agent installation. The user can go to **Devices / System Agent / Agents** to

view “In Process” and “Success” agents.

Devices > Agents

ONBOX 3 OFFBOX 0

Create Agent(s)

Query: All

Columns 1-3 of 3 Page Size: 25

| 0 selected | Device Address | Operation Mode | Enable? | Intent | Platform | Platform Version | AOS Version | State | System ID | Hostname |
|--------------------------|----------------|----------------|-------------------------------------|---------|----------|------------------|-------------|-------------|---------------------|----------|
| <input type="checkbox"/> | 192.168.59.22 | FULL CONTROL | <input checked="" type="checkbox"/> | Install | | | unknown | IN PROGRESS | | |
| <input type="checkbox"/> | 192.168.59.23 | FULL CONTROL | <input checked="" type="checkbox"/> | Install | EOS | 4.20.11M | absent | IN PROGRESS | 50000F6AD37 | |
| <input type="checkbox"/> | 192.168.59.100 | FULL CONTROL | <input checked="" type="checkbox"/> | Install | CUMULUS | 3.7.5 | 3.1.0-205 | SUCCESS | CN0TW5DCES007680122 | cumulus |

Active Tasks 1-2 of 2 Page Size: 3

| Job ID | Device Address | Task Type | State | Start Time | Actions |
|--------------------------------------|----------------|-----------|-------------|-------------------|---------|
| 7d852fb1-65c4-4ea1-a2e3-876b03c7085d | 192.168.59.23 | INSTALL | IN PROGRESS | a few seconds ago | |
| 7d852fb1-65c4-4ea1-a2e3-876b03c7085d | 192.168.59.22 | INSTALL | IN PROGRESS | a few seconds ago | |

Monitoring Apstra ZTP 2.0.0

Starting with Apstra ZTP 2.0.0, log files for all processes can be found in the `/containers_data/logs` directory. Apstra ZTP 2.0.0 logs are not currently available from the web interface.

```
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"}'
```

(continues on next page)

(continued from previous page)

```
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#
```

5.8 Device Profiles

5.8.1 Cumulus Device Profile

5.8.1.1 Background

The Device Profiles (DP) are a way of allowing a new device to be recognized and usable by AOS. They essentially capture much of the device specific semantics which are required not only to be discovered by AOS but also run network configs that work well for the datapath once inside the blueprint.

The Logical Device(LD) provides an abstracted view of a network device to be used for a given intent. The user is able to make a logical representation of a device including all information about how it can be used, including roles.

Interface Maps(IM) maps the physical ports as available in a DP to a Logical Device to generate and run port configs on the device.

It is clear that the sanity of the DP is crucial to ensure successful config deploys and effective network traffic passage.

The user can create, edit, delete, list DPs during the design phase of AOS. This DP is then used in the creation of Interface Maps (IM), which get directly used inside the AOS config rendering engine when the Blueprints are deployed.

This document tries to record all the necessary knowledge required to create (and edit) a semantically correct Cumulus DP, so that not only does it pass the validations in place in AOS 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.

5.8.1.2 Problem Statement

A user who creates the DP needs to have the device specification (usually released by the vendor). It does not stop there. The user also needs to learn a few things about how to translate these specifications into AOS DP data model so as to create the valid and config-friendly JSON. This is because the DP is a vendor semantics aware data structure.

When there is a new device to be added to AOS, which is not already part of production, one needs to go through the laborious process of first learning the specifications and then tailoring it to the AOS semantics, which has been found to be non-trivial without a guide which explains the purpose, meaning and semantic requirements behind each and every DP data model field.

5.8.1.3 Solution

This document will exhaustively iterate over each and every field in the DP and record the way one should use it when creating a new DP.

The high level data model is the same for all DPs, i.e. there are the same keys for every vendor's DP, just the way we get the values might differ, or might be loaded with a vendor constraint.

The document will try and enlist the following:

- The schema of the DP and the nested elements inside the DP.
- The meaning of each key value pair in the schema.
- The vendor specific recipe the values are populated.
- List any constraints, corner cases which need to be considered, especially for port configurations for certain (group of) models.
- Any lessons learnt along the way creating those DPs already in production useful in creating future ones.

5.8.1.4 User Interface

The AOS UI which helps the user to create the DP, while it is intelligent to warn the user against some semantic validations, is not wholly capable of ensuring deep vendor specific constraints and requirements (See Inter port constraints section). If the user can take care of having the exact vendor specification, then the AOS UI will certainly help the user create a semantically valid DP which can be inserted into AOS database successfully without any errors.

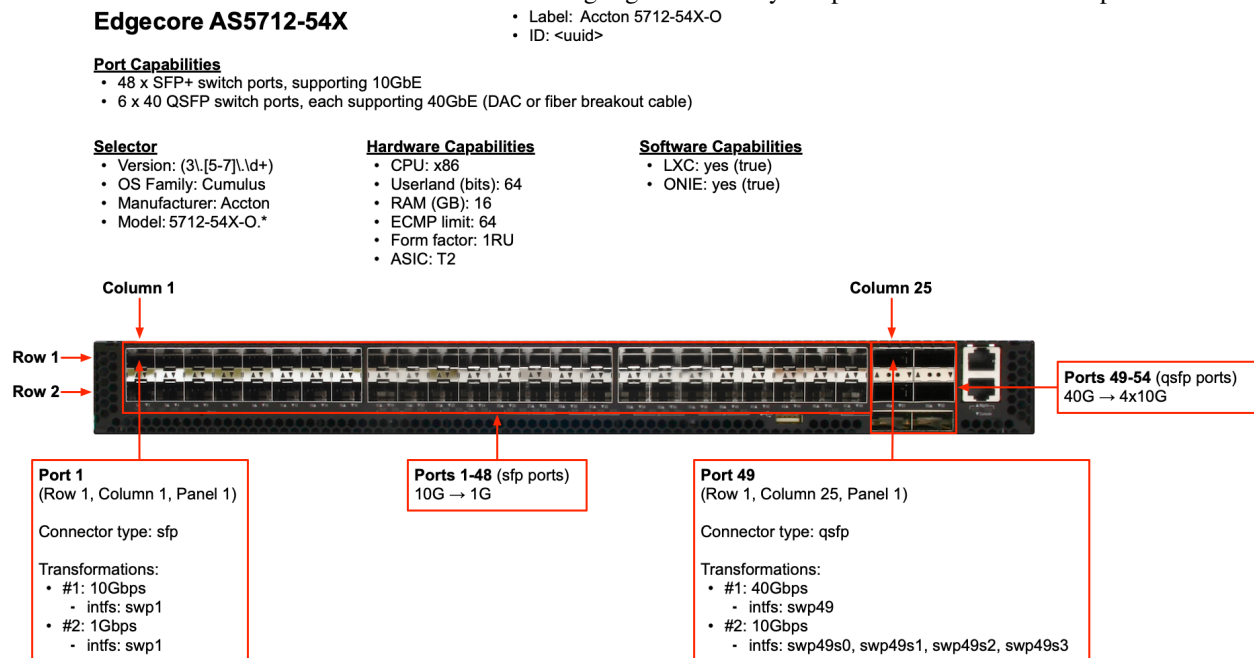
This document will enlist what it takes to get the vendor specifications and make them part of AOS DP data model. Once the user has these ready, AOS UI can be used to populate the fields and create the DP successfully.

Alternatively, the user can write his own Python code which takes into account the vendor specifications, normalize it as per AOS DP data model and generate the JSON to then import this into AOS UI.

5.8.1.5 DP Data Model

This section discusses the general DP data model in order to set the ground for Cumulus specific details. Some of the Optional fields might not apply to Cumulus and we will discuss those in more detail in subsequent sections.

Here is the front panel of the device Edgecore AS5712-54X as seen on the vendor's [data specification](#). Some of the elements of the device are marked to highlight how they map in the AOS device profile model.



| Key | Value data type | Value required(R) or optional(O) | Meaning | Purpose/Examples |
|-----------------------|-----------------|----------------------------------|---|---|
| id | string | R | The id of the device profile REST entity, needs to be unique | The DP is a rest entity like some of the other AOS elements like logical device and IM etc. Each need a unique identifier which is also used as an index into the database when looking up DPs. |
| failure_domain_id | int | R | When a device has multiple failure domains, this represents the one that is currently active. | |
| hardware_capabilities | dict | R | Captures those specifications of the device which are purely hardware | AOS needs a way to capture some of the HW capabilities which identify the device or highlight some of its unique strengths like ram, cpu, ecmp limit etc |
| userland | int | R | The type of application binary/kernel that the device supports. Usually this value is 32 or 64. | |
| copp_strict | list | O | when set to 'True, generate strict copp 'profile config for NXOS | This field is used in the nxos jinja files to generate the config. Usually for most models, this value is True, except for models like Cisco 3172 and Cisco nxosv |
| breakout_capable | list | O | When set to true for a module, indicates that ports of that module are breakout capable | |
| ram | int | R | How much memory does the device have | The unit is GB. |
| asic | string | R | Type of chipset on the device | Example: "T2", "T3", "Arad (3)", "Alta" |
| as_seq_num_supported | list | O | | |
| form_factor | string | R | How much rack space/units does the device need | Examples: '1RU', '2RU', '7RU', '13RU' |

Continued on next page

Table 4 – continued from previous page

| Key | Value data type | Value required(R) or optional(O) | Meaning | Purpose/Examples |
|-----------------------|-----------------|----------------------------------|---|--|
| ecmp_limit | int | R | Maximum number of next hops for ECMP routes | This field changes the BGP configuration on the device (ecmp max-paths) |
| cpu | string | R | What is the cpu architecture of the device. Note: AOS can only support “x86” if users intend to use onbox device agents because of TACC limitations. If the users intend to use offbox agents, then there is no such restriction. | Example: ‘x86’ |
| label | string | R | A human readable name for the DP | |
| software_capabilities | dict | R | A device might have some unique software capabilities. AOS requires these fields to have been set correctly for proper config rendering and deployment. | |
| onie | bool | R | Does the device OS support onie | |
| lxc_support | bool | R | Does the device support LXC containers | |
| slot_count | int | R | How many slots does the device have | If the device is a modular device, it will have a non-zero slot_count. |
| selector | dict | R | The selector information is used by the AOS to match the DP to the actual device. Each of the four fields in the selector need to exactly match the ones as derived from the device. | As part of device bring up, the device agent does an exact match on each of these fields to successfully assign the DP to the device. Unless this happens, the device cannot be made part of AOS |

Continued on next page

Table 4 – continued from previous page

| Key | Value data type | Value required(R) or optional(O) | Meaning | Purpose/Examples |
|--------------|-----------------|----------------------------------|---|--|
| os_version | string | R | Regex for the OS version on the device. AOS will use this regex from the DP to match and assign a system device to a DP | |
| model | string | R | Regex for the Model information as seen on syseeprom that AOS will use to match and assign a system device to a DP | |
| os | string | R | Regex for OS name that AOS will use to match and assign a system device to a DP | |
| manufacturer | string | R | Regex of vendor/manufacturer that AOS will use to match and assign a system device to a DP | |
| ports | list | R | Number of ports that the device specifies. | Port is a physical entity on the device as seen from the switch's front panel |
| panel_id | int | R | Being a per port field, This field is used by UI to display the rows and columns. For devices where the ports are divided between multiple panels, user will need to rightly map ports to the correct panel | panel_id is informational only at this point. The AOS UI uses this information to geometrically place the ports as close to how they appear on the device's physical front panel. The panel_id is not used in the config rendering engine inside AOS and is purely for cosmetic mapping between the physical appearance of the ports on the device to that on AOS UI port display. |

Continued on next page

Table 4 – continued from previous page

| Key | Value data type | Value required(R) or optional(O) | Meaning | Purpose/Examples |
|-----------------|-----------------|----------------------------------|--|------------------|
| slot_id | int | R | Being a per port field, this represents the module/slot this port belongs to. For modular devices, there will be multiple slots. | |
| connector_type | string | R | Being a per port field, this records the connector type for the physical port | |
| row_id | int | R | This is used to represent the port layout. User can see the specification and figure out how many rows the layout has and the row ID for this port. | |
| column_id | int | R | This is used to represent the port layout. User can see the specification and figure out how many columns the layout has and the column ID for this port. | |
| transformations | list | R | Transformations capture the breakout capabilities of the port. If a port is 40G->4x10G, there will be 2 transformations. There is at least one transformation per port, because every port is capable of at least one speed. | |

Continued on next page

Table 4 – continued from previous page

| Key | Value data type | Value required(R) or optional(O) | Meaning | Purpose/Examples |
|------------|-----------------|----------------------------------|--|---|
| is_default | bool | R | Being a per transformation field, this represents if the speed this transformation represents is the default speed of the port. This information can be read from the spec. For example for a qsfp28 port, the default speed is 100G, so the transformation with 100G interfaces sets the is_default to True | |
| interfaces | list | R | This is a per transformation field. For the 4x10G transformation, there will be 4 active interfaces, each capable of 10G. Similarly for the 40G transformation, there will be one active interface with speed 40G. | Interfaces are the effective port state when put in a break-out mode. |
| state | string | R | Per interface field. Represents if it is active or inactive interface. This field is useful when building IMs using this DP, especially for models with inter-port constraints | On certain vendors like EOS, the OS expects inactive and active interfaces in the config. For example, in a 40G->4x10G, transformation #1 will have 1 active 40G interface and 3 inactive interfaces. Transformation #2 will have 4 active 10G interfaces. Example values: "active", "inactive" |

Continued on next page

Table 4 – continued from previous page

| Key | Value data type | Value required(R) or optional(O) | Meaning | Purpose/Examples |
|-------------------|-----------------|----------------------------------|---|------------------|
| setting | string | Optional | Per interface field. This captures the vendor specific command to apply an interface speed andand to bring the port into selected breakout mode. | |
| speed | dict | Optional | Per interface field. This represents the speed this interface is to run once configured on the device. | |
| unit | string | R | Per speed field, represents the speed unit | |
| value | int | R | Per speed field, represents the speed value | |
| name | string | R | Per interface field, this is the name of the interface as the vendor/operating system requires the name when the interface is configuredin the selected breakout transformationon the device. This is a semantic heavy field. | |
| interface_id | int | R | A unique identifier which is used to ID the interface in the database. | |
| transformation_id | int | R | A unique identifier which is used to ID the transformation in the database. | |
| port_id | int | R | A unique identifier which is used to ID the port in the database. | |

Getting selector information from the device

Entering the correct information in all the 4 fields of the selector is critical for the device to get matched to the device profile.

In this section we will describe how AOS gets each of this information in detail. In the following table, mostly column 4 is enough for us to get all the 4 fields. Column 5 discusses a little more in detail how AOS gets these fields and processes in Python. Reading Column 5 helps the user gain certainty, nevertheless it is not necessary if one logs into the box and is able to read the file, type the commands and manually human-read this simple information.

Reference: Please read *this link* <<https://docs.cumulusnetworks.com/cumulus-linux/Monitoring-and-Troubleshooting/Monitoring-System-Hardware/>> ‘_ to understand the Cumulus TLV codes for the sys eeprom fields.

General Example output of a typical Cumulus device syseeprom information.

```
admin@leaf-1-1:mgmt-vrf:~$ decode-syseeprom
TlvInfo Header:
  Id String:      TlvInfo
  Version:        1
  Total Length: 168
TLV Name          Code Len Value
-----
Manufacture Date   0x25  19 10/23/2015 23:11:35
Diag Version       0x2E   7 2.0.1.4
Label Revision     0x27   4 R01I
Platform Name      0x28  27 x86_64-accton_as5712_54x-r0
ONIE Version       0x29  13 2014.08.00.05
Manufacturer       0x2B   6 Accton
Manufacture Country 0x2C   2 TW
Base MAC Address   0x24   6 CC:37:AB:62:F3:4B
Serial Number      0x23  14 571254X1541008
Part Number        0x22  13 FP1ZZ5654001A
Product Name       0x21  15 5712-54X-O-AC-F
MAC Addresses      0x2A   2 74
Vendor Name        0x2D   8 Edgecore
CRC-32             0xFE   4 0x95E4178F
(checksum valid)
admin@leaf-1-1:mgmt-vrf:~$
```

Now, let's try and apply these to an example in AOS, let's pick Accton 5712-54X.

| Selec- tor field | Value | Ex- am- ple | What files on the device have this information | [Extra information] Python processing to get the exact fields after we run the command/read the file |
|-----------------------------|--|-------------------|---|--|
| model | 0x21 | 5712-54X-O-AC-F | de- code_syseeprom_command = “/usr/cumulus/bin/decode- syseeprom” | Not much, read the syseeprom, use the 0x21 value in TLV |
| man- u- fac- turer | If 0x2D in syseeprom, 0x2D else 0x2B | Edgecore | de- code_syseeprom_command = “/usr/cumulus/bin/decode- syseeprom” | Not much, read the syseeprom, use the (0x2D or 0x2B) value in TLV |
| ver- sion | cat /etc/lsb- release, pick the DIS- TRIB_RELEASE | 3.7.3 | lsb_release_file = “/host_etc/lsb- release” | cumulus_info = read the lsb-release file match = re.search(r'%s=(.+)' % re.escape('NAME'), cumu- lus_info) value = match.group(1) if value[0] == value[-1] and value[0] in “””: value = value[1:-1] return value.split(“””)[0] |
| OS fam- ily | cat /etc/os- release, pick the first part of the NAME | Cu- mu- lus | os_release_file = “/host_etc/os- release” | Read the file cumulus_info = <file contents> match = re.search(r'%s=(S+)' % re.escape('DISTRIB_RELEASE'), cumulus_info) return match.group(1) |

Model and manufacturer/vendor

```
admin@leaf-1-1:mgmt-vrf:~$ decode-syseeprom
TlvInfo Header:
  Id String:   TlvInfo
  Version:    1
  Total Length: 168
TLV Name      Code Len Value
-----
Manufacture Date    0x25  19 10/23/2015 23:11:35
Diag Version        0x2E   7 2.0.1.4
Label Revision       0x27   4 R01I
Platform Name       0x28  27 x86_64-accton_as5712_54x-r0
ONIE Version         0x29  13 2014.08.00.05
Manufacturer         0x2B   6 Accton
Manufacture Country  0x2C   2 TW
Base MAC Address     0x24   6 CC:37:AB:62:F3:4B
Serial Number        0x23  14 571254X1541008
Part Number          0x22  13 FP1ZZ5654001A
Product Name         0x21  15 5712-54X-O-AC-F
MAC Addresses        0x2A   2 74
Vendor Name          0x2D   8 Edgecore
CRC-32               0xFE   4 0x95E4178F
(checksum valid)
admin@leaf-1-1:mgmt-vrf:~$
```

OS Version

```
admin@leaf-1-1:mgmt-vrf:~$ cat /etc/lsb-release
DISTRIB_ID="Cumulus Linux"
DISTRIB_RELEASE=3.7.3
DISTRIB_DESCRIPTION="Cumulus Linux 3.7.3"
admin@leaf-1-1:mgmt-vrf:~$
```

OS Family

```
admin@leaf-1-1:mgmt-vrf:~$ cat /etc/os-release
NAME="Cumulus Linux"
VERSION_ID=3.7.3
VERSION="Cumulus Linux 3.7.3"
PRETTY_NAME="Cumulus Linux"
ID=cumulus-linux
ID_LIKE=debian
CPE_NAME=cpe:/o:cumulusnetworks:cumulus_linux:3.7.3
HOME_URL="http://www.cumulusnetworks.com/"
SUPPORT_URL="http://support.cumulusnetworks.com/"
admin@leaf-1-1:mgmt-vrf:~$
```

Using the above gotten information to Accton 5712 54X's selector as displayed in UI:

 ▸ [Devices](#) ▸ [Device Profiles](#) ▸ **Accton 5712-54X-O**

Selector[?]

| | |
|---------------------|---------------|
| Manufacturer | Accton |
| Model | 5712-54X-O.* |
| OS family | Cumulus |
| Version | (3\.[5-7]\d+) |

Some common default values as used in AOS

What makes the AOS SDK generators fast tools when it comes to generating the JSONs is using some of the knowledge specific to a vendor and re-using them in Python code to generate more than one DP.

If you look at the list of key value pairs that make the DP, there are a lot! However, on careful observation, we have found some commonalities across models belonging to the same vendor. We call these defaults. If one can confirm if it is safe to use this default value for a particular key, then it saves some effort in the overall process.

Note: While every field in the DP should be populated per device carefully consulting the specification, one can also

use these defaults as a reference when populating the fields.

| Field | Default value | Meaning |
|------------|---|---|
| ports | -NA- (list) | Ports for a device are one of those things which mostly vary by device. There is no point in having a default for ports, more often than not it won't be used. |
| label | -NA- (string) | Every label needs to be unique, so a default does not apply |
| slot_count | 0 (int) | Affects config. For non-modular devices, leave this 0. For modular device, this is the number of linecards excludes the supervisor units. |
| id | No default. When user is constructing json, make sure to provide a unique string as id for the DP. This is especially applicable when user builds the json and imports the payload into AOS using the REST POST/PUT API. However, if the user uses the UI to populate the fields, the UI prompts for the label but assigns an id behind the scenes. Nevertheless, this id field is required for a valid DP. | |
| os_version | On[5-7].d+)" (string) | Usually the default reflects the versions as supported by AOS for the release. As of AOS 3.1.1 we support 3.5-3.7. AOS maintains an official feature matrix by device OS here- http://docs.dc1.apstra.com/feature_matrix.html The os_version value in the DP is more permissive than what is officially certified, to facilitate experiments of (not officially supported) OS versions for given hardware model. |

5.8.1.6 Capabilities

The hardware and software capabilities are relatively straight forward, if the user has access to the device specification, it is easy to fetch these fields.

The software capabilities for Cumulus, lxc and onie is mostly set to true.

Here are some commonly found values on Cumulus devices (this data is based on the devices AOS already supports):

| | | |
|-------------|---------------------------------------|--|
| userland | 64 (int) | This does not affect config as of AOS 3.1.1 |
| form_factor | '100' (string) | This does not affect config as of AOS 3.1.1 |
| ecmp | 64 (int) | This does not affect config as of AOS 3.1.1 |
| asic | 'T2' (string) | AOS looks at this for certain scenarios like Cumulus release note RN 766. The RN states that on the Broadcom Trident II+, Trident 3, and Maverick platforms, in an external VXLAN routing environment, the switch does not rewrite MAC addresses and TTL, so packets are dropped by the next hop. To configure this, the asic field is looked at in conjunction with the os version field to determine if RN766 applies to the DP/device version and if yes, to handle it accordingly. |
| cpu | 'x86' (string) | This does not affect config as of AOS 3.1.1 |
| ram | 16 (int) (Note, the unit is in GB) | This does not affect config as of AOS 3.1.1 |
| onie | True (bool) | This does not affect config as of AOS 3.1.1 |
| lxc | True (bool) | This does not affect config as of AOS 3.1.1 |

Note: AOS does not require the user to configure any extra configurables for “supported features” for Cumulus device profiles as of AOS 3.1.1. Thus the UI says “no supported features”.

5.8.1.7 Port specific semantics

Every interface per transformation per port per device profile, requires a setting.

This port setting is used by the configuration renderer(jinja files) to correctly configure the interfaces on the device. The DP allows the user to configure these for the device. Thus, it is very important these setting fields are populated correctly in the DP. In this section, we will look at what is the schema AOS enforces for Cumulus port settings.

Example port setting:

```
{
  "interface": {
    "speed": "10000",
    "lane_map": "13"
  }
}
```

Where is this field found: If you have 32 ports, each port capable of 40G and 4x10G, you will have:

- 2 transformations per port
- 1 active interface in 1st transformation; setting can be found in each of these interfaces
- 4 active interfaces in the 2nd transformation; setting can be found in each of these interfaces

Interface Name semantics on Cumulus

The rules for naming an interface, as of AOS 3.1.1, are as follows:

- Arguments: port_id(required), lane_id(optional, applies only in case of a broken out port)

Example of an interface name on Cumulus, **swp49s0**. First get the lane_suffix:

Example: **s0**

If lane_id: Lane_suffix = "s%d" % lane_id

Else: Lane_suffix = ""

Then build the name, using port id and lane_suffix Name = "'swp%d%s' % (port_id, lane_suffix)"

Examples: Interfaces for 4x10G transformation on port id 49: - **swp49s0** - **swp49s1** - **swp49s2** - **swp49s3** Interfaces for 40G transformation on port id 49: - **swp49**

Port setting schema on Cumulus

| | | | |
|----------------------|--------|--|--|
| Cumulus port setting | dict | Required/Optional field | The high level dictionary |
| interface | dict | R | The interface this setting is applicable to |
| command | string | R. This is the speed to command mapping as required by cumulus. For example, if interface speed is 1G, then command is 'link-speed 1000' command = { 1: 'link-speed 1000', 10: '', 20: '', 25: '', 40: '', 50: '', 100: '', } | The command if applicable when user needs to apply the speed on this interface |
| speed | string | R () | The speed to be applied to this interface |

Editing interface setting per interface on UI

The setting contains two important pieces of information, the command and the speed.

The "command" goes underneath the interface paragraph in /etc/network/interfaces, and "speed" is the value that goes to ports.conf.

- I know my port breakout capability
- I am ready to create ports in AOS UI
- I am in the DP->edit->ports->port
- How do I populate the Setting?

Summary

Selector[?]

Capabilities

Ports

Panel #1 Select port(s) to fill in the details

Port breakoutAutonegotiation

357911131517192123252729313335373941434547495153

24681012141618202224262830323436384042444648505254

Edit selected port #1

Connector type[?]

sfp28

Transformations

| Default | Speed | Active | Name template [?] | Setting [?] | |
|----------------------------------|---------|-------------------------------------|----------------------------|-----------------------------|-----------------------------------|
| <input checked="" type="radio"/> | 25 Gbps | <input checked="" type="checkbox"/> | swp1 | [*interface": [*command": " | <div><div></div><div></div></div> |
| <input type="radio"/> | 10 Gbps | <input type="checkbox"/> | swp1 | | <div><div></div><div></div></div> |

Add new transformation

+

 Add Panel

Update

| Interfaces applicable in this transformation | This interface speed | to_count | No breakout? | Speed to be populated in interface setting | command | Settings as found per interface on UI |
|--|----------------------|----------|--|--|-----------------|---|
| 40G | 40G | 1 | Yes (Note the setting remains same even for a port which is capable of 40G breakout. For example On the Accton 6712, we have all ports capable of 40G->4x10G. Whereas on the Dell S6010, we have ports 13-16 and 29-32 which are 40G non breakout capable ports. Irrespective of whether it is breakout capable or non breakout capable, the setting for 40G interface remains same) | '40G' | '' | { "interface": { "command": "", "speed": "40G" } } |
| 1x40G->4x10G | 10G | 4 | No | '4x10G' | '' | { "interface": { "command": "", "speed": "4x10G" } } |
| 1G | 1G | 1 | Yes | '1G' | link-speed 1000 | { "interface": { "command": "link-speed 1000", "speed": "10G" } } |
| 1G auto neg | 1G autoneg | 1 | Yes | '' | '' | { "interface": { "command": "link-speed 1000", "speed": "" } } |
| 10G | 10G | 1 | Yes | '10G' | '' | { "interface": { "command": "", "speed": "10G" } } |
| 1x10G->1x1G | 1G | 1 | No | '10G' | link-speed 1000 | { "interface": { "command": "link- |

Auto-negotiation for 10GBaseT ports

On the 10GBaseT, Cumulus supports auto negotiation for some devices.

To incorporate this into the DP, on that port which supports the 10G->1x1G breakout, for the 1G transformation, have the interface setting as follows:

```
{
  "interface": {
    "command": "link-speed 1000",
    "speed": ""
  }
}
```

Note the “speed” inside the “interface” setting is set to “”. This allows the interface to be configured in auto-neg mode and device will auto-negotiate the interface speed.

In the IM, pick the auto-neg transformation for this port.

How to make the interface as auto neg in the DP:

1. Go to the 10GBaseT port
2. Create a 1G transformation as follows:
 - state: active
 - speed: max speed possible for interface speed
 - setting->interface->speed: empty
 - name: as per interface name rules

For example, on the Dell S4148T-ON, the ports from 1-24 and 31-54 are 10GBaseT ports. Thus, Transformation #2, configures the speed in the interface setting as “”.

Panel #1

INTERFACES CAPACITY

72 x 10 Gbps 48 x 1 Gbps 4 x 100 Gbps 8 x 50 Gbps 6 x 40 Gbps 16 x 25 Gbps

PORTS Click on port to toggle the details

PORT DETAILS

| |
|----------------|
| ID |
| Connector type |

Transformations

| |
|-----------------------|
| #1 (10 Gbps, default) |
| #2 (1 Gbps) |

swp31

| | |
|----------|---|
| Name | swp31 |
| Active? | yes |
| Speed | 1 Gbps |
| Setting? | { "interface": { "command": "link-speed 1000", "speed": "" } } |

5.8.1.8 Inter port constraints

To capture all the essence of ports of a particular device, not only does the user need to read the device specification, but also the OS specific port rules. Fortunately, Cumulus packages these rules in the `ports.conf` in the `dpkg` bundle.

When we are writing device profiles for new Cumulus devices, the procedure ideally to be followed is:

1. To generate a list of ports(with correct transformations and interfaces) in the DP, a good way is to have access to the `ports.conf` and work our way backwards to then generate the ports list in the DP such that the Cumulus OS likes it. To know the detailed port constraints, read the `ports.conf` of that particular model in the `dpkg`.
2. run the following command on any Cumulus box(even VX) to get access to `ports.conf`: `dpkg -L cumulus-platform | grep ports.conf`

Pay attention to the comments sections which lists the port breakout constraints usually. It looks something like this:

`ports.conf` –

This file controls port aggregation and subdivision. For example, QSFP+ ports are typically configurable as either one 40G interface or four 10G/1000/100 interfaces. This file sets the number of interfaces per port while `/etc/network/interfaces` and `ethtool` configure the link speed for each interface.

You must restart `switchd` for changes to take effect.

The DELL S6010 has:

32 QSFP ports numbered 1-32 These ports are configurable as 40G, split into 4x10G ports or disabled.

The X pipeline covers QSFP ports 1 through 16 and the Y pipeline covers QSFP ports 17 through 32.

The Trident2+ chip can only handle 52 logical ports per pipeline.

This means 13 is the maximum number of 40G ports you can ungang per pipeline, with the remaining three 40G ports set to “disabled”. The 13 40G ports become 52 unganged 10G ports, which totals 52 logical ports for that pipeline.

QSFP+ ports <port label 1-32> = [4x10G|40G|disabled]

3. Use the comments when writing a new Cumulus DP.

Dell Z9100-ON

Port Constraint:

The Dell Z9100 has:

32 QSFP28 ports numbered 1-32 These ports are configurable as 100G, 50G, 40G, 2x50G, 4x25G, 4x10G or disabled.

Two SFP+ ports. These ports are configurable as 10G or disabled. The system can only handle 128 logical ports. This means that if all 32 QSFP28 ports are broken out into 4x25G or 4x10G mode, the two 10G ports (33 and 34) must be set to “disabled”.

To honor this, in the DP, the ports 33-34 should provide a transformation with the interface with disabled setting.

```
{
  "interface": {
    "command": "",
    "speed": "disabled"
  }
}
```

IM constraint:

If all the 32 QSFP28 ports are broken out into 4x25G or 4x10G, then the two 33-34 need to be disabled. So explicitly pick the disabled transformation for 33-34 in this case.

Dell 4128F

Port Constraint:

The Dell S4128 has:

28 SFP+ ports numbered 1-24 and 27-30 These ports are configurable as 10G, or groups of four adjacent ports can be configured as 40G.

2 QSFP28 ports numbered 25-26 These ports are configurable as 100G, 50G, 40G, or split into 2x50G, 4x25G, or 4x10G ports.

Note: SFP+ ports 13, 14, 23, and 24 are not in a group of four adjacent ports and cannot be configured as part of a ganged 40G.

AOS does not support the configuration of ganged ports.

For the 1G broken out interface, the speed inside the interface setting should be “1G” and not “10G” (which usually is for other cumulus devices)

To honor this, in the DP, the ports 33-34 should provide a transformation with the interface with disabled setting.

```
{
  "interface": {
    "command": "",
    "speed": "disabled"
  }
}
```

IM constraint:

If all the 32 QSFP28 ports are broken out into 4x25G or 4x10G, then the two 33-34 need to be disabled. So explicitly pick the disabled transformation for 33-34 in this case.

Dell 4148T

Port Constraint:

The Dell S4148T has:

48 10GT ports numbered 1-24 and 31-54 These ports are not configurable.

2 QSFP+ ports numbered 27-28 These ports are configurable as 40G or split into 4x10G ports.

4 QSFP28 ports numbered 25-26 and 29-30 These ports are configurable as 100G, 50G, 40G, or split into 2x50G, 4x25G, or 4x10G ports.

Note: The two QSFP+ ports are DISABLED by default, and the four QSFP28 ports are configured for 100G by default. In order to enable the two QSFP+ ports for 40G or 4x10G, all four QSFP28 ports must also be configured for either 40G or 4x10G.

In the DP, to honor this constraint we need a transformation with “disabled” setting as follows for the ports 27-28

```
{
  "interface": {
    "command": "",
    "speed": "disabled"
  }
}
```

IM constraint:

In order to enable the two QSFP+ ports(27-28) for 40G or 4x10G, all four QSFP28 ports must also be configured for either 40G or 4x10G.

Dell 4148F

Port Constraint:

The Dell S4148 has:

48 SFP+ ports numbered 1-24 and 31-54 These ports are configurable as 10G, or groups of four adjacent ports can be configured as 40G.

2 QSFP+ ports numbered 27-28 These ports are configurable as 40G or split into 4x10G ports.

4 QSFP28 ports numbered 25-26 and 29-30 These ports are configurable as 100G, 50G, 40G, or split into 2x50G, 4x25G, or 4x10G ports.

Note: The two QSFP+ ports are DISABLED by default, and the four QSFP28 ports are configured for 100G by default. In order to enable the two QSFP+ ports for 40G or 4x10G, all four QSFP28 ports must also be configured for either 40G or 4x10G.

The 2 QSFP+ and 4 QSFP28 have same rules as Dell 4148T. The 48 SFP+ ports have same 1G setting constraints as Dell 4128F, i.e.

```
{
  "interface": {
    "command": "link-speed 1000",
    "speed": "10G"
  }
}
```

Edgecore 6812_32x_O

Port Constraint:

This file controls port aggregation and subdivision. For example, QSFP+ ports are typically configurable as either one 40G interface or four 10G/1000/100 interfaces. This file sets the number of interfaces per port while /etc/network/interfaces and ethtool configure the link speed for each interface.

You must restart switchd for changes to take effect.

Accton AS6812_32X has:

32 QSFP+ ports numbered 1-32 These ports are configurable as 40G, split into 4x10G ports or disabled.

The X pipeline covers QSFP ports 1-16 and the Y pipeline covers QSFP ports 17-32.

The Trident2+ chip can only handle 52 logical ports per pipeline.

This means 13 is the maximum number of 40G ports you can ungang per pipeline, with the remaining three 40G ports set to “disabled”. The 13 40G ports become 52 ungang 10G ports, which totals 52 logical ports for that pipeline.

To honor this constraint, in the DP, all the 40g->4x10G ports need to have a transformation with an interface with disabled setting as follows:

```
{
  "interface": {
    "command": "",
    "speed": "disabled"
  }
}
```

IM Constraint:

When picking 10G interfaces, make sure there are only a maximum of 52 10G interfaces (coming from the 13 ports) for this IM. Also for the rest of the 3 leftover 40G ports, explicitly select the “disabled” transformation.

This way, the port config will be generated correctly as per port constraints.

Mellanox 2700

Port Constraint:

Odd numbered ports can do full breakout up to 4x10G and 4x25G.

Even numbered ports can only have upto 2x50G interfaces and should be disabled if immediately preceding odd-numbered port is broken out into 4 interfaces. (Source: <https://docs.mellanox.com/display/sn2000pub/Cable+Installation>)

SN2700 and SN2740 Splitting Options

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

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

All QSFP28 ports can be split to 2 QSFP28 ports.



How does this translate in the DP?

The odd numbered ports, like port 1, have following transformations:


| | |
|------------------------|-----------------------------|
| #1 (100 Gbps, default) | swp1 |
| #2 (50 Gbps) | swp1s0 swp1s1 |
| #3 (50 Gbps) | swp1 |
| #4 (40 Gbps) | swp1 |
| #5 (25 Gbps) | swp1s0 swp1s1 swp1s2 swp1s3 |
| #6 (10 Gbps) | swp1s0 swp1s1 swp1s2 swp1s3 |
| #7 (1 Gbps) | swp1s0 swp1s1 swp1s2 swp1s3 |

Panel #1

INTERFACES CAPACITY

32 x 100 Gbps 64 x 50 Gbps 32 x 40 Gbps 64 x 25 Gbps 64 x 10 Gbps 64 x 1 Gbps

PORTS *Click on port to toggle the details*



PORT DETAILS

| | |
|----------------|--------|
| ID | 1 |
| Connector type | qsfp28 |

Transformations

| | |
|------------------------------------|-----------------------------|
| Port #1 Tr. #1 (100 Gbps, default) | swp1 |
| Port #1 Tr. #2 (50 Gbps) | swp1s0 swp1s1 |
| Port #1 Tr. #3 (50 Gbps) | swp1 |
| Port #1 Tr. #4 (40 Gbps) | swp1 |
| Port #1 Tr. #5 (25 Gbps) | swp1s0 swp1s1 swp1s2 swp1s3 |
| Port #1 Tr. #6 (10 Gbps) | swp1s0 swp1s1 swp1s2 swp1s3 |
| Port #1 Tr. #7 (1 Gbps) | swp1s0 swp1s1 swp1s2 swp1s3 |

| | |
|----------|---|
| Name | swp1s3 |
| Active? | yes |
| Speed | 25 Gbps |
| Setting? | <pre>{ "interface": { "command": "", "speed": "4x25G" } }</pre> |

Whereas the even numbered ports, like port 2, have following transformations.

| | |
|------------------------|---------------|
| #1 (100 Gbps, default) | swp2 |
| #2 (50 Gbps) | swp2s0 swp2s1 |
| #3 (50 Gbps) | swp2 |
| #4 (40 Gbps) | swp2 |
| #5 (100 Gbps) | swp2 |

Note that even ports do not have the 10G and 25G transformations. Additionally, they have a “disabled” interface in transformation 5 with the following setting:


```
{
  "interface": {
    "command": "",
    "speed": "disabled"
  }
}
```

Panel #1

INTERFACES CAPACITY

32 x 100 Gbps 64 x 50 Gbps 32 x 40 Gbps 64 x 25 Gbps 64 x 10 Gbps 64 x 1 Gbps

PORTS Click on port to toggle the details



PORT DETAILS

| | |
|------------------------------------|---|
| ID | 2 |
| Connector type | |
| Transformations | |
| Port #2 Tr. #1 (100 Gbps, default) | |
| Port #2 Tr. #2 (50 Gbps) | |
| Port #2 Tr. #3 (50 Gbps) | |
| Port #2 Tr. #4 (40 Gbps) | |
| Port #2 Tr. #5 (100 Gbps) | |

| | |
|----------|--|
| Name | swp2 |
| Active? | no |
| Speed | 100 Gbps |
| Setting? | <pre>{ "interface": { "command": "", "speed": "disabled" } }</pre> |

swp2

IM constraints:

If you want to include an odd numbered port which is broken out into 4 interfaces, then make sure you also pick the disabled transformation of the following even numbered port.

For example in the Interface Map Mellanox_MSN2700_Cumulus__AOS-48x10_8x100-1

Because we are using the ports: [1,3,5,7,9,11,13,15,17,19,21,23] to generate the 4x10G interfaces, we also pick the 5th transformation(disabled) for ports [2,4,6,8,10,12,14,16,18,20,22,24]. Thus on the UI, for the Interface Map, you see 12 unused interfaces corresponding to these disabled interfaces.

Only if these are included in the Interface Map, will AOS generate the right ports.conf honoring the above Cumulus Mellanox 2700 constraint. Refer to the use case [Handling Inter Port Constraints - Disabled Ports](#) for more details on Interface Map creation.

5.8.1.9 Debugging and Recovery

Device mismatch

Usually this is seen at the very beginning of device's lifecycle. If a device profile is not being picked by the device, then it is advisable to check the 4 fields as entered in the selector section of the DP.

Deploy errors

One of the reasons deploy errors are seen are because of incorrect port configurations. This could be either the ports were configured with incorrect speeds, or the OS specific port constraints were not handled in the DP or in the IM.

A possible flow for root cause would be:

1. Check the DP for obvious port capabilities errors. Is the port really capable of the speeds the DP has configured. The `/var/log/switchd` logs on the device are a good resource to parse for ERROR messages.
2. Check if the DP has configured autoneg or disabled interfaces correctly. Autoneg and disabled can both be expressed in the interface setting field.
3. The port constraints can be read from the following files: `admin@leaf1:mgmt-vrf:~$ dpkg -L cumulus-platform | grep ports.conf`

5.8.1.10 Appendix

Mapping the DP to LD in the IM

The mapping specifies how exactly the user wants to map the physical ports to the logical interfaces.

The 5 fields on the mapping are: [DP portID, DP transformationID, DP InterfaceID, LD panelID, LD portID].

These five fields allow the user to map the DP exactly to the LD. The first three fields specify which port, transformation and interface in the DP is the interface in the IM referencing:

1. (port id, transformation id) - all active interfaces in the given transformation are used; active interfaces are mapped in the order they appear in the transformation.
2. (port id, transformation id, interface id) - If the user specifies the third field in the mapping, the particular interface in the transformation is used in mapping.

5.8.1.11 Example for DP JSON for Edgecore / Accton AS5712-54X

Edgecore AS5712-54X

Port Capabilities

- 48 x SFP+ switch ports, supporting 10GbE
- 6 x 40 QSFP switch ports, each supporting 40GbE (DAC or fiber breakout cable)

Selector

- Version: (3\.[5-7]\.\d+)
- OS Family: Cumulus
- Manufacturer: Accton
- Model: 5712-54X-O.*

• Label: Accton 5712-54X-O

• ID: <uuid>

Hardware Capabilities

- CPU: x86
- Userland (bits): 64
- RAM (GB): 16
- ECMP limit: 64
- Form factor: 1RU
- ASIC: T2

Software Capabilities

- LXC: yes (true)
- ONIE: yes (true)

```

{
  "hardware_capabilities": {
    "userland": 64,
    "ram": 16,
  }

```

(continues on next page)

(continued from previous page)

```

    "asic": "T2",
    "form_factor": "1RU",
    "ecmp_limit": 64,
    "cpu": "x86"
  },
  "created_at": "2019-08-30T05:53:38.369782Z",
  "last_modified_at": "2019-08-30T05:53:38.369782Z",
  "selector": {
    "os_version": "(3\\. [5-7]\\.\\.\\d+)",
    "model": "5712-54X-O.*",
    "os": "Cumulus",
    "manufacturer": "Accton"
  },
  "software_capabilities": {
    "onie": true,
    "lxc_support": true
  },
  "id": "Accton_5712-54X-O_Cumulus",
  "slot_count": 0,
  "label": "Accton 5712-54X-O",
  "ports": [
    {
      "panel_id": 1,
      "slot_id": 0,
      "connector_type": "sfp",
      "row_id": 1,
      "transformations": [
        {
          "is_default": false,
          "interfaces": [
            {
              "state": "active",
              "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
              "speed": {
                "unit": "G",
                "value": 10
              },
              "name": "swp1",
              "interface_id": 1
            }
          ],
          "transformation_id": 1
        },
        {
          "is_default": true,
          "interfaces": [
            {
              "state": "active",
              "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}",
              "speed": {
                "unit": "G",
                "value": 1
              },
              "name": "swp1",
              "interface_id": 1
            }
          ]
        }
      ]
    }
  ]
}

```

(continues on next page)

(continued from previous page)

```

        ],
        "transformation_id": 2
    }
],
"port_id": 1,
"failure_domain_id": 1,
"column_id": 1
},
{
    "panel_id": 1,
    "slot_id": 0,
    "connector_type": "sfp",
    "row_id": 2,
    "transformations": [
        {
            "is_default": false,
            "interfaces": [
                {
                    "state": "active",
                    "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
                    "speed": {
                        "unit": "G",
                        "value": 10
                    },
                    "name": "swp2",
                    "interface_id": 1
                }
            ],
            "transformation_id": 1
        },
        {
            "is_default": true,
            "interfaces": [
                {
                    "state": "active",
                    "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}",
                    "speed": {
                        "unit": "G",
                        "value": 1
                    },
                    "name": "swp2",
                    "interface_id": 1
                }
            ],
            "transformation_id": 2
        }
    ],
    "port_id": 2,
    "failure_domain_id": 1,
    "column_id": 1
},
{
    "panel_id": 1,
    "slot_id": 0,
    "connector_type": "sfp",
    "row_id": 1,

```

(continues on next page)

(continued from previous page)

```

"transformations": [
  {
    "is_default": false,
    "interfaces": [
      {
        "state": "active",
        "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
        "speed": {
          "unit": "G",
          "value": 10
        },
        "name": "swp3",
        "interface_id": 1
      }
    ],
    "transformation_id": 1
  },
  {
    "is_default": true,
    "interfaces": [
      {
        "state": "active",
        "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}",
        "speed": {
          "unit": "G",
          "value": 1
        },
        "name": "swp3",
        "interface_id": 1
      }
    ],
    "transformation_id": 2
  }
],
"port_id": 3,
"failure_domain_id": 1,
"column_id": 2
},
{
  "panel_id": 1,
  "slot_id": 0,
  "connector_type": "sfp",
  "row_id": 2,
  "transformations": [
    {
      "is_default": false,
      "interfaces": [
        {
          "state": "active",
          "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
          "speed": {
            "unit": "G",
            "value": 10
          },
          "name": "swp4",
          "interface_id": 1
        }
      ]
    }
  ]
}

```

(continues on next page)

(continued from previous page)

```

    }
  ],
  "transformation_id": 1
},
{
  "is_default": true,
  "interfaces": [
    {
      "state": "active",
      "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}",
      "speed": {
        "unit": "G",
        "value": 1
      },
      "name": "swp4",
      "interface_id": 1
    }
  ],
  "transformation_id": 2
}
],
"port_id": 4,
"failure_domain_id": 1,
"column_id": 2
},
{
  "panel_id": 1,
  "slot_id": 0,
  "connector_type": "sfp",
  "row_id": 1,
  "transformations": [
    {
      "is_default": false,
      "interfaces": [
        {
          "state": "active",
          "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
          "speed": {
            "unit": "G",
            "value": 10
          },
          "name": "swp5",
          "interface_id": 1
        }
      ],
      "transformation_id": 1
    }
  ],
  "is_default": true,
  "interfaces": [
    {
      "state": "active",
      "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}",
      "speed": {
        "unit": "G",

```

(continues on next page)

(continued from previous page)

```

        "value": 1
      },
      "name": "swp5",
      "interface_id": 1
    }
  ],
  "transformation_id": 2
},
{
  "port_id": 5,
  "failure_domain_id": 1,
  "column_id": 3
},
{
  "panel_id": 1,
  "slot_id": 0,
  "connector_type": "sfp",
  "row_id": 2,
  "transformations": [
    {
      "is_default": false,
      "interfaces": [
        {
          "state": "active",
          "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
          "speed": {
            "unit": "G",
            "value": 10
          },
          "name": "swp6",
          "interface_id": 1
        }
      ],
      "transformation_id": 1
    },
    {
      "is_default": true,
      "interfaces": [
        {
          "state": "active",
          "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}",
          "speed": {
            "unit": "G",
            "value": 1
          },
          "name": "swp6",
          "interface_id": 1
        }
      ],
      "transformation_id": 2
    }
  ],
  "port_id": 6,
  "failure_domain_id": 1,
  "column_id": 3
},

```

(continues on next page)

(continued from previous page)

```

{
  "panel_id": 1,
  "slot_id": 0,
  "connector_type": "sfp",
  "row_id": 1,
  "transformations": [
    {
      "is_default": false,
      "interfaces": [
        {
          "state": "active",
          "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
          "speed": {
            "unit": "G",
            "value": 10
          },
          "name": "swp7",
          "interface_id": 1
        }
      ],
      "transformation_id": 1
    },
    {
      "is_default": true,
      "interfaces": [
        {
          "state": "active",
          "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}",
          "speed": {
            "unit": "G",
            "value": 1
          },
          "name": "swp7",
          "interface_id": 1
        }
      ],
      "transformation_id": 2
    }
  ],
  "port_id": 7,
  "failure_domain_id": 1,
  "column_id": 4
},
{
  "panel_id": 1,
  "slot_id": 0,
  "connector_type": "sfp",
  "row_id": 2,
  "transformations": [
    {
      "is_default": false,
      "interfaces": [
        {
          "state": "active",
          "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
          "speed": {

```

(continues on next page)

(continued from previous page)

```

        "unit": "G",
        "value": 10
    },
    "name": "swp8",
    "interface_id": 1
}
],
"transformation_id": 1
},
{
    "is_default": true,
    "interfaces": [
        {
            "state": "active",
            "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}",
            "speed": {
                "unit": "G",
                "value": 1
            },
            "name": "swp8",
            "interface_id": 1
        }
    ],
    "transformation_id": 2
}
],
"port_id": 8,
"failure_domain_id": 1,
"column_id": 4
},
{
    "panel_id": 1,
    "slot_id": 0,
    "connector_type": "sfp",
    "row_id": 1,
    "transformations": [
        {
            "is_default": false,
            "interfaces": [
                {
                    "state": "active",
                    "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
                    "speed": {
                        "unit": "G",
                        "value": 10
                    },
                    "name": "swp9",
                    "interface_id": 1
                }
            ],
            "transformation_id": 1
        },
        {
            "is_default": true,
            "interfaces": [
                {

```

(continues on next page)

(continued from previous page)

```

        "state": "active",
        "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}\",
        "speed": {
            "unit": "G",
            "value": 1
        },
        "name": "swp9",
        "interface_id": 1
    }
],
    "transformation_id": 2
}
],
"port_id": 9,
"failure_domain_id": 1,
"column_id": 5
},
{
    "panel_id": 1,
    "slot_id": 0,
    "connector_type": "sfp",
    "row_id": 2,
    "transformations": [
        {
            "is_default": false,
            "interfaces": [
                {
                    "state": "active",
                    "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}\",
                    "speed": {
                        "unit": "G",
                        "value": 10
                    },
                    "name": "swp10",
                    "interface_id": 1
                }
            ],
            "transformation_id": 1
        },
        {
            "is_default": true,
            "interfaces": [
                {
                    "state": "active",
                    "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}\",
                    "speed": {
                        "unit": "G",
                        "value": 1
                    },
                    "name": "swp10",
                    "interface_id": 1
                }
            ],
            "transformation_id": 2
        }
    ]
}

```

(continues on next page)

(continued from previous page)

```

    ],
    "port_id": 10,
    "failure_domain_id": 1,
    "column_id": 5
  },
  {
    "panel_id": 1,
    "slot_id": 0,
    "connector_type": "sfp",
    "row_id": 1,
    "transformations": [
      {
        "is_default": false,
        "interfaces": [
          {
            "state": "active",
            "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
            "speed": {
              "unit": "G",
              "value": 10
            },
            "name": "swp11",
            "interface_id": 1
          }
        ],
        "transformation_id": 1
      },
      {
        "is_default": true,
        "interfaces": [
          {
            "state": "active",
            "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}",
            "speed": {
              "unit": "G",
              "value": 1
            },
            "name": "swp11",
            "interface_id": 1
          }
        ],
        "transformation_id": 2
      }
    ],
    "port_id": 11,
    "failure_domain_id": 1,
    "column_id": 6
  },
  {
    "panel_id": 1,
    "slot_id": 0,
    "connector_type": "sfp",
    "row_id": 2,
    "transformations": [
      {
        "is_default": false,

```

(continues on next page)

(continued from previous page)

```

    "interfaces": [
      {
        "state": "active",
        "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
        "speed": {
          "unit": "G",
          "value": 10
        },
        "name": "swp12",
        "interface_id": 1
      }
    ],
    "transformation_id": 1
  },
  {
    "is_default": true,
    "interfaces": [
      {
        "state": "active",
        "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}",
        "speed": {
          "unit": "G",
          "value": 1
        },
        "name": "swp12",
        "interface_id": 1
      }
    ],
    "transformation_id": 2
  }
],
"port_id": 12,
"failure_domain_id": 1,
"column_id": 6
},
{
  "panel_id": 1,
  "slot_id": 0,
  "connector_type": "sfp",
  "row_id": 1,
  "transformations": [
    {
      "is_default": false,
      "interfaces": [
        {
          "state": "active",
          "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
          "speed": {
            "unit": "G",
            "value": 10
          },
          "name": "swp13",
          "interface_id": 1
        }
      ]
    },
    {
      "transformation_id": 1
    }
  ]
}

```

(continues on next page)

(continued from previous page)

```

    },
    {
      "is_default": true,
      "interfaces": [
        {
          "state": "active",
          "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}",
          "speed": {
            "unit": "G",
            "value": 1
          },
          "name": "swp13",
          "interface_id": 1
        }
      ],
      "transformation_id": 2
    }
  ],
  "port_id": 13,
  "failure_domain_id": 1,
  "column_id": 7
},
{
  "panel_id": 1,
  "slot_id": 0,
  "connector_type": "sfp",
  "row_id": 2,
  "transformations": [
    {
      "is_default": false,
      "interfaces": [
        {
          "state": "active",
          "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
          "speed": {
            "unit": "G",
            "value": 10
          },
          "name": "swp14",
          "interface_id": 1
        }
      ],
      "transformation_id": 1
    }
  ],
},
{
  "is_default": true,
  "interfaces": [
    {
      "state": "active",
      "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}",
      "speed": {
        "unit": "G",
        "value": 1
      },
      "name": "swp14",

```

(continues on next page)

(continued from previous page)

```

        "interface_id": 1
    }
],
    "transformation_id": 2
}
],
    "port_id": 14,
    "failure_domain_id": 1,
    "column_id": 7
},
{
    "panel_id": 1,
    "slot_id": 0,
    "connector_type": "sfp",
    "row_id": 1,
    "transformations": [
        {
            "is_default": false,
            "interfaces": [
                {
                    "state": "active",
                    "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
                    "speed": {
                        "unit": "G",
                        "value": 10
                    },
                    "name": "swp15",
                    "interface_id": 1
                }
            ],
            "transformation_id": 1
        },
        {
            "is_default": true,
            "interfaces": [
                {
                    "state": "active",
                    "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}",
                    "speed": {
                        "unit": "G",
                        "value": 1
                    },
                    "name": "swp15",
                    "interface_id": 1
                }
            ],
            "transformation_id": 2
        }
    ],
    "port_id": 15,
    "failure_domain_id": 1,
    "column_id": 8
},
{
    "panel_id": 1,
    "slot_id": 0,

```

(continues on next page)

(continued from previous page)

```

"connector_type": "sfp",
"row_id": 2,
"transformations": [
  {
    "is_default": false,
    "interfaces": [
      {
        "state": "active",
        "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
        "speed": {
          "unit": "G",
          "value": 10
        },
        "name": "swp16",
        "interface_id": 1
      }
    ],
    "transformation_id": 1
  },
  {
    "is_default": true,
    "interfaces": [
      {
        "state": "active",
        "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}",
        "speed": {
          "unit": "G",
          "value": 1
        },
        "name": "swp16",
        "interface_id": 1
      }
    ],
    "transformation_id": 2
  }
],
"port_id": 16,
"failure_domain_id": 1,
"column_id": 8
},
{
  "panel_id": 1,
  "slot_id": 0,
  "connector_type": "sfp",
  "row_id": 1,
  "transformations": [
    {
      "is_default": false,
      "interfaces": [
        {
          "state": "active",
          "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
          "speed": {
            "unit": "G",
            "value": 10
          },
          "name": "swp16",
          "interface_id": 1
        }
      ],
      "transformation_id": 1
    }
  ]
}

```

(continues on next page)

(continued from previous page)

```

        "name": "swp17",
        "interface_id": 1
    },
    ],
    "transformation_id": 1
},
{
    "is_default": true,
    "interfaces": [
        {
            "state": "active",
            "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}",
            "speed": {
                "unit": "G",
                "value": 1
            },
            "name": "swp17",
            "interface_id": 1
        }
    ],
    "transformation_id": 2
}
],
"port_id": 17,
"failure_domain_id": 1,
"column_id": 9
},
{
    "panel_id": 1,
    "slot_id": 0,
    "connector_type": "sfp",
    "row_id": 2,
    "transformations": [
        {
            "is_default": false,
            "interfaces": [
                {
                    "state": "active",
                    "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
                    "speed": {
                        "unit": "G",
                        "value": 10
                    },
                    "name": "swp18",
                    "interface_id": 1
                }
            ],
            "transformation_id": 1
        },
        {
            "is_default": true,
            "interfaces": [
                {
                    "state": "active",
                    "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}",

```

(continues on next page)

(continued from previous page)

```

        "speed": {
            "unit": "G",
            "value": 1
        },
        "name": "swp18",
        "interface_id": 1
    }
],
    "transformation_id": 2
}
],
"port_id": 18,
"failure_domain_id": 1,
"column_id": 9
},
{
    "panel_id": 1,
    "slot_id": 0,
    "connector_type": "sfp",
    "row_id": 1,
    "transformations": [
        {
            "is_default": false,
            "interfaces": [
                {
                    "state": "active",
                    "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
                    "speed": {
                        "unit": "G",
                        "value": 10
                    },
                    "name": "swp19",
                    "interface_id": 1
                }
            ],
            "transformation_id": 1
        },
        {
            "is_default": true,
            "interfaces": [
                {
                    "state": "active",
                    "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}",
                    "speed": {
                        "unit": "G",
                        "value": 1
                    },
                    "name": "swp19",
                    "interface_id": 1
                }
            ],
            "transformation_id": 2
        }
    ],
    "port_id": 19,
    "failure_domain_id": 1,

```

(continues on next page)

(continued from previous page)

```

    "column_id": 10
  },
  {
    "panel_id": 1,
    "slot_id": 0,
    "connector_type": "sfp",
    "row_id": 2,
    "transformations": [
      {
        "is_default": false,
        "interfaces": [
          {
            "state": "active",
            "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
            "speed": {
              "unit": "G",
              "value": 10
            },
            "name": "swp20",
            "interface_id": 1
          }
        ],
        "transformation_id": 1
      },
      {
        "is_default": true,
        "interfaces": [
          {
            "state": "active",
            "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}",
            "speed": {
              "unit": "G",
              "value": 1
            },
            "name": "swp20",
            "interface_id": 1
          }
        ],
        "transformation_id": 2
      }
    ],
    "port_id": 20,
    "failure_domain_id": 1,
    "column_id": 10
  },
  {
    "panel_id": 1,
    "slot_id": 0,
    "connector_type": "sfp",
    "row_id": 1,
    "transformations": [
      {
        "is_default": false,
        "interfaces": [
          {
            "state": "active",

```

(continues on next page)

(continued from previous page)

```

        "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
        "speed": {
            "unit": "G",
            "value": 10
        },
        "name": "swp21",
        "interface_id": 1
    },
    ],
    "transformation_id": 1
},
{
    "is_default": true,
    "interfaces": [
        {
            "state": "active",
            "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}",
            "speed": {
                "unit": "G",
                "value": 1
            },
            "name": "swp21",
            "interface_id": 1
        }
    ],
    "transformation_id": 2
}
],
"port_id": 21,
"failure_domain_id": 1,
"column_id": 11
},
{
    "panel_id": 1,
    "slot_id": 0,
    "connector_type": "sfp",
    "row_id": 2,
    "transformations": [
        {
            "is_default": false,
            "interfaces": [
                {
                    "state": "active",
                    "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
                    "speed": {
                        "unit": "G",
                        "value": 10
                    },
                    "name": "swp22",
                    "interface_id": 1
                }
            ],
            "transformation_id": 1
        }
    ],
    "is_default": true,

```

(continues on next page)

(continued from previous page)

```

    "interfaces": [
      {
        "state": "active",
        "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}",
        "speed": {
          "unit": "G",
          "value": 1
        },
        "name": "swp22",
        "interface_id": 1
      }
    ],
    "transformation_id": 2
  }
],
"port_id": 22,
"failure_domain_id": 1,
"column_id": 11
},
{
  "panel_id": 1,
  "slot_id": 0,
  "connector_type": "sfp",
  "row_id": 1,
  "transformations": [
    {
      "is_default": false,
      "interfaces": [
        {
          "state": "active",
          "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
          "speed": {
            "unit": "G",
            "value": 10
          },
          "name": "swp23",
          "interface_id": 1
        }
      ],
      "transformation_id": 1
    }
  ],
  "is_default": true,
  "interfaces": [
    {
      "state": "active",
      "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}",
      "speed": {
        "unit": "G",
        "value": 1
      },
      "name": "swp23",
      "interface_id": 1
    }
  ]
},

```

(continues on next page)

(continued from previous page)

```

        "transformation_id": 2
    }
],
"port_id": 23,
"failure_domain_id": 1,
"column_id": 12
},
{
    "panel_id": 1,
    "slot_id": 0,
    "connector_type": "sfp",
    "row_id": 2,
    "transformations": [
        {
            "is_default": false,
            "interfaces": [
                {
                    "state": "active",
                    "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
                    "speed": {
                        "unit": "G",
                        "value": 10
                    },
                    "name": "swp24",
                    "interface_id": 1
                }
            ],
            "transformation_id": 1
        },
        {
            "is_default": true,
            "interfaces": [
                {
                    "state": "active",
                    "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}",
                    "speed": {
                        "unit": "G",
                        "value": 1
                    },
                    "name": "swp24",
                    "interface_id": 1
                }
            ],
            "transformation_id": 2
        }
    ],
    "port_id": 24,
    "failure_domain_id": 1,
    "column_id": 12
},
{
    "panel_id": 1,
    "slot_id": 0,
    "connector_type": "sfp",
    "row_id": 1,
    "transformations": [

```

(continues on next page)

(continued from previous page)

```

    {
      "is_default": false,
      "interfaces": [
        {
          "state": "active",
          "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
          "speed": {
            "unit": "G",
            "value": 10
          },
          "name": "swp25",
          "interface_id": 1
        }
      ],
      "transformation_id": 1
    },
    {
      "is_default": true,
      "interfaces": [
        {
          "state": "active",
          "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}",
          "speed": {
            "unit": "G",
            "value": 1
          },
          "name": "swp25",
          "interface_id": 1
        }
      ],
      "transformation_id": 2
    }
  ],
  "port_id": 25,
  "failure_domain_id": 1,
  "column_id": 13
},
{
  "panel_id": 1,
  "slot_id": 0,
  "connector_type": "sfp",
  "row_id": 2,
  "transformations": [
    {
      "is_default": false,
      "interfaces": [
        {
          "state": "active",
          "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
          "speed": {
            "unit": "G",
            "value": 10
          },
          "name": "swp26",
          "interface_id": 1
        }
      ]
    }
  ]
}

```

(continues on next page)

(continued from previous page)

```

    ],
    "transformation_id": 1
  },
  {
    "is_default": true,
    "interfaces": [
      {
        "state": "active",
        "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}",
        "speed": {
          "unit": "G",
          "value": 1
        },
        "name": "swp26",
        "interface_id": 1
      }
    ],
    "transformation_id": 2
  }
],
"port_id": 26,
"failure_domain_id": 1,
"column_id": 13
},
{
  "panel_id": 1,
  "slot_id": 0,
  "connector_type": "sfp",
  "row_id": 1,
  "transformations": [
    {
      "is_default": false,
      "interfaces": [
        {
          "state": "active",
          "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
          "speed": {
            "unit": "G",
            "value": 10
          },
          "name": "swp27",
          "interface_id": 1
        }
      ],
      "transformation_id": 1
    },
    {
      "is_default": true,
      "interfaces": [
        {
          "state": "active",
          "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}",
          "speed": {
            "unit": "G",
            "value": 1
          }
        }
      ]
    }
  ]
}

```

(continues on next page)

(continued from previous page)

```

        },
        "name": "swp27",
        "interface_id": 1
    }
],
"transformation_id": 2
}
],
"port_id": 27,
"failure_domain_id": 1,
"column_id": 14
},
{
    "panel_id": 1,
    "slot_id": 0,
    "connector_type": "sfp",
    "row_id": 2,
    "transformations": [
        {
            "is_default": false,
            "interfaces": [
                {
                    "state": "active",
                    "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
                    "speed": {
                        "unit": "G",
                        "value": 10
                    },
                    "name": "swp28",
                    "interface_id": 1
                }
            ],
            "transformation_id": 1
        },
        {
            "is_default": true,
            "interfaces": [
                {
                    "state": "active",
                    "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}",
                    "speed": {
                        "unit": "G",
                        "value": 1
                    },
                    "name": "swp28",
                    "interface_id": 1
                }
            ],
            "transformation_id": 2
        }
    ],
    "port_id": 28,
    "failure_domain_id": 1,
    "column_id": 14
},
{

```

(continues on next page)

(continued from previous page)

```

"panel_id": 1,
"slot_id": 0,
"connector_type": "sfp",
"row_id": 1,
"transformations": [
  {
    "is_default": false,
    "interfaces": [
      {
        "state": "active",
        "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
        "speed": {
          "unit": "G",
          "value": 10
        },
        "name": "swp29",
        "interface_id": 1
      }
    ],
    "transformation_id": 1
  },
  {
    "is_default": true,
    "interfaces": [
      {
        "state": "active",
        "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}",
        "speed": {
          "unit": "G",
          "value": 1
        },
        "name": "swp29",
        "interface_id": 1
      }
    ],
    "transformation_id": 2
  }
],
"port_id": 29,
"failure_domain_id": 1,
"column_id": 15
},
{
  "panel_id": 1,
  "slot_id": 0,
  "connector_type": "sfp",
  "row_id": 2,
  "transformations": [
    {
      "is_default": false,
      "interfaces": [
        {
          "state": "active",
          "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
          "speed": {
            "unit": "G",

```

(continues on next page)

(continued from previous page)

```

        "value": 10
    },
    "name": "swp30",
    "interface_id": 1
}
],
"transformation_id": 1
},
{
    "is_default": true,
    "interfaces": [
        {
            "state": "active",
            "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}",
            "speed": {
                "unit": "G",
                "value": 1
            },
            "name": "swp30",
            "interface_id": 1
        }
    ],
    "transformation_id": 2
}
],
"port_id": 30,
"failure_domain_id": 1,
"column_id": 15
},
{
    "panel_id": 1,
    "slot_id": 0,
    "connector_type": "sfp",
    "row_id": 1,
    "transformations": [
        {
            "is_default": false,
            "interfaces": [
                {
                    "state": "active",
                    "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
                    "speed": {
                        "unit": "G",
                        "value": 10
                    },
                    "name": "swp31",
                    "interface_id": 1
                }
            ],
            "transformation_id": 1
        },
        {
            "is_default": true,
            "interfaces": [
                {
                    "state": "active",

```

(continues on next page)

(continued from previous page)

```

        "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}\",
        "speed": {
            "unit": "G",
            "value": 1
        },
        "name": "swp31",
        "interface_id": 1
    }
],
    "transformation_id": 2
}
],
"port_id": 31,
"failure_domain_id": 1,
"column_id": 16
},
{
    "panel_id": 1,
    "slot_id": 0,
    "connector_type": "sfp",
    "row_id": 2,
    "transformations": [
        {
            "is_default": false,
            "interfaces": [
                {
                    "state": "active",
                    "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}\",
                    "speed": {
                        "unit": "G",
                        "value": 10
                    },
                    "name": "swp32",
                    "interface_id": 1
                }
            ],
            "transformation_id": 1
        },
        {
            "is_default": true,
            "interfaces": [
                {
                    "state": "active",
                    "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}\",
                    "speed": {
                        "unit": "G",
                        "value": 1
                    },
                    "name": "swp32",
                    "interface_id": 1
                }
            ],
            "transformation_id": 2
        }
    ]
},

```

(continues on next page)

(continued from previous page)

```

    "port_id": 32,
    "failure_domain_id": 1,
    "column_id": 16
  },
  {
    "panel_id": 1,
    "slot_id": 0,
    "connector_type": "sfp",
    "row_id": 1,
    "transformations": [
      {
        "is_default": false,
        "interfaces": [
          {
            "state": "active",
            "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
            "speed": {
              "unit": "G",
              "value": 10
            },
            "name": "swp33",
            "interface_id": 1
          }
        ],
        "transformation_id": 1
      },
      {
        "is_default": true,
        "interfaces": [
          {
            "state": "active",
            "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}",
            "speed": {
              "unit": "G",
              "value": 1
            },
            "name": "swp33",
            "interface_id": 1
          }
        ],
        "transformation_id": 2
      }
    ],
    "port_id": 33,
    "failure_domain_id": 1,
    "column_id": 17
  },
  {
    "panel_id": 1,
    "slot_id": 0,
    "connector_type": "sfp",
    "row_id": 2,
    "transformations": [
      {
        "is_default": false,
        "interfaces": [

```

(continues on next page)

(continued from previous page)

```

        {
            "state": "active",
            "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
            "speed": {
                "unit": "G",
                "value": 10
            },
            "name": "swp34",
            "interface_id": 1
        }
    ],
    "transformation_id": 1
},
{
    "is_default": true,
    "interfaces": [
        {
            "state": "active",
            "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}",
            "speed": {
                "unit": "G",
                "value": 1
            },
            "name": "swp34",
            "interface_id": 1
        }
    ],
    "transformation_id": 2
}
],
"port_id": 34,
"failure_domain_id": 1,
"column_id": 17
},
{
    "panel_id": 1,
    "slot_id": 0,
    "connector_type": "sfp",
    "row_id": 1,
    "transformations": [
        {
            "is_default": false,
            "interfaces": [
                {
                    "state": "active",
                    "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
                    "speed": {
                        "unit": "G",
                        "value": 10
                    },
                    "name": "swp35",
                    "interface_id": 1
                }
            ],
            "transformation_id": 1
        }
    ],
},

```

(continues on next page)

(continued from previous page)

```

        {
            "is_default": true,
            "interfaces": [
                {
                    "state": "active",
                    "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}\",
                    "speed": {
                        "unit": "G",
                        "value": 1
                    },
                    "name": "swp35",
                    "interface_id": 1
                }
            ],
            "transformation_id": 2
        }
    ],
    "port_id": 35,
    "failure_domain_id": 1,
    "column_id": 18
},
{
    "panel_id": 1,
    "slot_id": 0,
    "connector_type": "sfp",
    "row_id": 2,
    "transformations": [
        {
            "is_default": false,
            "interfaces": [
                {
                    "state": "active",
                    "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}\",
                    "speed": {
                        "unit": "G",
                        "value": 10
                    },
                    "name": "swp36",
                    "interface_id": 1
                }
            ],
            "transformation_id": 1
        }
    ],
    {
        "is_default": true,
        "interfaces": [
            {
                "state": "active",
                "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}\",
                "speed": {
                    "unit": "G",
                    "value": 1
                },
                "name": "swp36",
                "interface_id": 1
            }
        ]
    }
}

```

(continues on next page)

(continued from previous page)

```

    }
    ],
    "transformation_id": 2
  }
],
"port_id": 36,
"failure_domain_id": 1,
"column_id": 18
},
{
  "panel_id": 1,
  "slot_id": 0,
  "connector_type": "sfp",
  "row_id": 1,
  "transformations": [
    {
      "is_default": false,
      "interfaces": [
        {
          "state": "active",
          "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
          "speed": {
            "unit": "G",
            "value": 10
          },
          "name": "swp37",
          "interface_id": 1
        }
      ],
      "transformation_id": 1
    },
    {
      "is_default": true,
      "interfaces": [
        {
          "state": "active",
          "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}",
          "speed": {
            "unit": "G",
            "value": 1
          },
          "name": "swp37",
          "interface_id": 1
        }
      ],
      "transformation_id": 2
    }
  ],
  "port_id": 37,
  "failure_domain_id": 1,
  "column_id": 19
},
{
  "panel_id": 1,
  "slot_id": 0,
  "connector_type": "sfp",

```

(continues on next page)

(continued from previous page)

```

"row_id": 2,
"transformations": [
  {
    "is_default": false,
    "interfaces": [
      {
        "state": "active",
        "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
        "speed": {
          "unit": "G",
          "value": 10
        },
        "name": "swp38",
        "interface_id": 1
      }
    ],
    "transformation_id": 1
  },
  {
    "is_default": true,
    "interfaces": [
      {
        "state": "active",
        "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}",
        "speed": {
          "unit": "G",
          "value": 1
        },
        "name": "swp38",
        "interface_id": 1
      }
    ],
    "transformation_id": 2
  }
],
"port_id": 38,
"failure_domain_id": 1,
"column_id": 19
},
{
  "panel_id": 1,
  "slot_id": 0,
  "connector_type": "sfp",
  "row_id": 1,
  "transformations": [
    {
      "is_default": false,
      "interfaces": [
        {
          "state": "active",
          "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
          "speed": {
            "unit": "G",
            "value": 10
          },
          "name": "swp39",

```

(continues on next page)

(continued from previous page)

```

        "interface_id": 1
    }
],
    "transformation_id": 1
},
{
    "is_default": true,
    "interfaces": [
        {
            "state": "active",
            "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}",
            "speed": {
                "unit": "G",
                "value": 1
            },
            "name": "swp39",
            "interface_id": 1
        }
    ],
    "transformation_id": 2
}
],
    "port_id": 39,
    "failure_domain_id": 1,
    "column_id": 20
},
{
    "panel_id": 1,
    "slot_id": 0,
    "connector_type": "sfp",
    "row_id": 2,
    "transformations": [
        {
            "is_default": false,
            "interfaces": [
                {
                    "state": "active",
                    "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
                    "speed": {
                        "unit": "G",
                        "value": 10
                    },
                    "name": "swp40",
                    "interface_id": 1
                }
            ],
            "transformation_id": 1
        }
    ],
    "is_default": true,
    "interfaces": [
        {
            "state": "active",
            "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}",
            "speed": {

```

(continues on next page)

(continued from previous page)

```

        "unit": "G",
        "value": 1
    },
    "name": "swp40",
    "interface_id": 1
}
],
"transformation_id": 2
}
],
"port_id": 40,
"failure_domain_id": 1,
"column_id": 20
},
{
    "panel_id": 1,
    "slot_id": 0,
    "connector_type": "sfp",
    "row_id": 1,
    "transformations": [
        {
            "is_default": false,
            "interfaces": [
                {
                    "state": "active",
                    "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
                    "speed": {
                        "unit": "G",
                        "value": 10
                    },
                    "name": "swp41",
                    "interface_id": 1
                }
            ],
            "transformation_id": 1
        },
        {
            "is_default": true,
            "interfaces": [
                {
                    "state": "active",
                    "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}",
                    "speed": {
                        "unit": "G",
                        "value": 1
                    },
                    "name": "swp41",
                    "interface_id": 1
                }
            ],
            "transformation_id": 2
        }
    ],
    "port_id": 41,
    "failure_domain_id": 1,
    "column_id": 21
}

```

(continues on next page)

(continued from previous page)

```

    },
    {
      "panel_id": 1,
      "slot_id": 0,
      "connector_type": "sfp",
      "row_id": 2,
      "transformations": [
        {
          "is_default": false,
          "interfaces": [
            {
              "state": "active",
              "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
              "speed": {
                "unit": "G",
                "value": 10
              },
              "name": "swp42",
              "interface_id": 1
            }
          ],
          "transformation_id": 1
        },
        {
          "is_default": true,
          "interfaces": [
            {
              "state": "active",
              "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}",
              "speed": {
                "unit": "G",
                "value": 1
              },
              "name": "swp42",
              "interface_id": 1
            }
          ],
          "transformation_id": 2
        }
      ],
      "port_id": 42,
      "failure_domain_id": 1,
      "column_id": 21
    },
    {
      "panel_id": 1,
      "slot_id": 0,
      "connector_type": "sfp",
      "row_id": 1,
      "transformations": [
        {
          "is_default": false,
          "interfaces": [
            {
              "state": "active",
              "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",

```

(continues on next page)

(continued from previous page)

```

        "speed": {
            "unit": "G",
            "value": 10
        },
        "name": "swp43",
        "interface_id": 1
    }
],
"transformation_id": 1
},
{
    "is_default": true,
    "interfaces": [
        {
            "state": "active",
            "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}",
            "speed": {
                "unit": "G",
                "value": 1
            },
            "name": "swp43",
            "interface_id": 1
        }
    ],
    "transformation_id": 2
}
],
"port_id": 43,
"failure_domain_id": 1,
"column_id": 22
},
{
    "panel_id": 1,
    "slot_id": 0,
    "connector_type": "sfp",
    "row_id": 2,
    "transformations": [
        {
            "is_default": false,
            "interfaces": [
                {
                    "state": "active",
                    "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
                    "speed": {
                        "unit": "G",
                        "value": 10
                    },
                    "name": "swp44",
                    "interface_id": 1
                }
            ],
            "transformation_id": 1
        }
    ],
    "is_default": true,
    "interfaces": [

```

(continues on next page)

(continued from previous page)

```

        {
            "state": "active",
            "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}\",
            "speed": {
                "unit": "G",
                "value": 1
            },
            "name": "swp44",
            "interface_id": 1
        }
    ],
    "transformation_id": 2
}
],
"port_id": 44,
"failure_domain_id": 1,
"column_id": 22
},
{
    "panel_id": 1,
    "slot_id": 0,
    "connector_type": "sfp",
    "row_id": 1,
    "transformations": [
        {
            "is_default": false,
            "interfaces": [
                {
                    "state": "active",
                    "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}\",
                    "speed": {
                        "unit": "G",
                        "value": 10
                    },
                    "name": "swp45",
                    "interface_id": 1
                }
            ],
            "transformation_id": 1
        },
        {
            "is_default": true,
            "interfaces": [
                {
                    "state": "active",
                    "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}\",
                    "speed": {
                        "unit": "G",
                        "value": 1
                    },
                    "name": "swp45",
                    "interface_id": 1
                }
            ],
            "transformation_id": 2
        }
    ]
}

```

(continues on next page)

(continued from previous page)

```

    }
  ],
  "port_id": 45,
  "failure_domain_id": 1,
  "column_id": 23
},
{
  "panel_id": 1,
  "slot_id": 0,
  "connector_type": "sfp",
  "row_id": 2,
  "transformations": [
    {
      "is_default": false,
      "interfaces": [
        {
          "state": "active",
          "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
          "speed": {
            "unit": "G",
            "value": 10
          },
          "name": "swp46",
          "interface_id": 1
        }
      ],
      "transformation_id": 1
    },
    {
      "is_default": true,
      "interfaces": [
        {
          "state": "active",
          "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}",
          "speed": {
            "unit": "G",
            "value": 1
          },
          "name": "swp46",
          "interface_id": 1
        }
      ],
      "transformation_id": 2
    }
  ],
  "port_id": 46,
  "failure_domain_id": 1,
  "column_id": 23
},
{
  "panel_id": 1,
  "slot_id": 0,
  "connector_type": "sfp",
  "row_id": 1,
  "transformations": [
    {

```

(continues on next page)

(continued from previous page)

```

    "is_default": false,
    "interfaces": [
      {
        "state": "active",
        "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
        "speed": {
          "unit": "G",
          "value": 10
        },
        "name": "swp47",
        "interface_id": 1
      }
    ],
    "transformation_id": 1
  },
  {
    "is_default": true,
    "interfaces": [
      {
        "state": "active",
        "setting": "{\"interface\": {\"command\": \"link-speed 1000\", \"speed\
↪\": \"10G\"}}",
        "speed": {
          "unit": "G",
          "value": 1
        },
        "name": "swp47",
        "interface_id": 1
      }
    ],
    "transformation_id": 2
  }
],
"port_id": 47,
"failure_domain_id": 1,
"column_id": 24
},
{
  "panel_id": 1,
  "slot_id": 0,
  "connector_type": "sfp",
  "row_id": 2,
  "transformations": [
    {
      "is_default": false,
      "interfaces": [
        {
          "state": "active",
          "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"10G\"}}",
          "speed": {
            "unit": "G",
            "value": 10
          },
          "name": "swp48",
          "interface_id": 1
        }
      ]
    }
  ]
},

```

(continues on next page)

(continued from previous page)

```

        "transformation_id": 1
    },
    {
        "is_default": true,
        "interfaces": [
            {
                "state": "active",
                "setting": "{\\"interface\\": {\\"command\\": \\"link-speed 1000\\", \\"speed\\": \\"10G\\"}}",
                "speed": {
                    "unit": "G",
                    "value": 1
                },
                "name": "swp48",
                "interface_id": 1
            }
        ],
        "transformation_id": 2
    }
],
"port_id": 48,
"failure_domain_id": 1,
"column_id": 24
},
{
    "panel_id": 1,
    "slot_id": 0,
    "connector_type": "qsfp",
    "row_id": 1,
    "transformations": [
        {
            "is_default": true,
            "interfaces": [
                {
                    "state": "active",
                    "setting": "{\\"interface\\": {\\"command\\": \\"\\", \\"speed\\": \\"40G\\"}}",
                    "speed": {
                        "unit": "G",
                        "value": 40
                    },
                    "name": "swp49",
                    "interface_id": 1
                }
            ],
            "transformation_id": 1
        }
    ],
    "is_default": false,
    "interfaces": [
        {
            "state": "active",
            "setting": "{\\"interface\\": {\\"command\\": \\"\\", \\"speed\\": \\"4x10G\\"}}",
            "speed": {
                "unit": "G",
                "value": 10
            },
            "name": "swp49s0",

```

(continues on next page)

(continued from previous page)

```

        "interface_id": 1
    },
    {
        "state": "active",
        "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"4x10G\"}}",
        "speed": {
            "unit": "G",
            "value": 10
        },
        "name": "swp49s1",
        "interface_id": 2
    },
    {
        "state": "active",
        "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"4x10G\"}}",
        "speed": {
            "unit": "G",
            "value": 10
        },
        "name": "swp49s2",
        "interface_id": 3
    },
    {
        "state": "active",
        "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"4x10G\"}}",
        "speed": {
            "unit": "G",
            "value": 10
        },
        "name": "swp49s3",
        "interface_id": 4
    }
],
    "transformation_id": 2
}
],
    "port_id": 49,
    "failure_domain_id": 1,
    "column_id": 25
},
{
    "panel_id": 1,
    "slot_id": 0,
    "connector_type": "qsfp",
    "row_id": 2,
    "transformations": [
        {
            "is_default": true,
            "interfaces": [
                {
                    "state": "active",
                    "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"40G\"}}",
                    "speed": {
                        "unit": "G",
                        "value": 40
                    },
                    "name": "swp50",

```

(continues on next page)

(continued from previous page)

```

        "interface_id": 1
    }
],
"transformation_id": 1
},
{
    "is_default": false,
    "interfaces": [
        {
            "state": "active",
            "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"4x10G\"}}",
            "speed": {
                "unit": "G",
                "value": 10
            },
            "name": "swp50s0",
            "interface_id": 1
        },
        {
            "state": "active",
            "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"4x10G\"}}",
            "speed": {
                "unit": "G",
                "value": 10
            },
            "name": "swp50s1",
            "interface_id": 2
        },
        {
            "state": "active",
            "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"4x10G\"}}",
            "speed": {
                "unit": "G",
                "value": 10
            },
            "name": "swp50s2",
            "interface_id": 3
        },
        {
            "state": "active",
            "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"4x10G\"}}",
            "speed": {
                "unit": "G",
                "value": 10
            },
            "name": "swp50s3",
            "interface_id": 4
        }
    ],
    "transformation_id": 2
}
],
"port_id": 50,
"failure_domain_id": 1,
"column_id": 25
},
{

```

(continues on next page)

(continued from previous page)

```

"panel_id": 1,
"slot_id": 0,
"connector_type": "qsfp",
"row_id": 1,
"transformations": [
  {
    "is_default": true,
    "interfaces": [
      {
        "state": "active",
        "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"40G\"}}",
        "speed": {
          "unit": "G",
          "value": 40
        },
        "name": "swp51",
        "interface_id": 1
      }
    ],
    "transformation_id": 1
  },
  {
    "is_default": false,
    "interfaces": [
      {
        "state": "active",
        "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"4x10G\"}}",
        "speed": {
          "unit": "G",
          "value": 10
        },
        "name": "swp51s0",
        "interface_id": 1
      },
      {
        "state": "active",
        "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"4x10G\"}}",
        "speed": {
          "unit": "G",
          "value": 10
        },
        "name": "swp51s1",
        "interface_id": 2
      },
      {
        "state": "active",
        "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"4x10G\"}}",
        "speed": {
          "unit": "G",
          "value": 10
        },
        "name": "swp51s2",
        "interface_id": 3
      },
      {
        "state": "active",
        "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"4x10G\"}}",

```

(continues on next page)

(continued from previous page)

```

        "speed": {
            "unit": "G",
            "value": 10
        },
        "name": "swp51s3",
        "interface_id": 4
    }
],
    "transformation_id": 2
},
    "port_id": 51,
    "failure_domain_id": 1,
    "column_id": 26
},
{
    "panel_id": 1,
    "slot_id": 0,
    "connector_type": "qsfp",
    "row_id": 2,
    "transformations": [
        {
            "is_default": true,
            "interfaces": [
                {
                    "state": "active",
                    "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"40G\"}}",
                    "speed": {
                        "unit": "G",
                        "value": 40
                    },
                    "name": "swp52",
                    "interface_id": 1
                }
            ],
            "transformation_id": 1
        },
        {
            "is_default": false,
            "interfaces": [
                {
                    "state": "active",
                    "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"4x10G\"}}",
                    "speed": {
                        "unit": "G",
                        "value": 10
                    },
                    "name": "swp52s0",
                    "interface_id": 1
                },
                {
                    "state": "active",
                    "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"4x10G\"}}",
                    "speed": {
                        "unit": "G",
                        "value": 10
                    }
                }
            ]
        }
    ]
}

```

(continues on next page)

(continued from previous page)

```

        "name": "swp52s1",
        "interface_id": 2
    },
    {
        "state": "active",
        "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"4x10G\"}}",
        "speed": {
            "unit": "G",
            "value": 10
        },
        "name": "swp52s2",
        "interface_id": 3
    },
    {
        "state": "active",
        "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"4x10G\"}}",
        "speed": {
            "unit": "G",
            "value": 10
        },
        "name": "swp52s3",
        "interface_id": 4
    }
],
"transformation_id": 2
}
],
"port_id": 52,
"failure_domain_id": 1,
"column_id": 26
},
{
    "panel_id": 1,
    "slot_id": 0,
    "connector_type": "qsfp",
    "row_id": 1,
    "transformations": [
        {
            "is_default": true,
            "interfaces": [
                {
                    "state": "active",
                    "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"40G\"}}",
                    "speed": {
                        "unit": "G",
                        "value": 40
                    },
                    "name": "swp53",
                    "interface_id": 1
                }
            ],
            "transformation_id": 1
        },
        {
            "is_default": false,
            "interfaces": [

```

(continues on next page)

(continued from previous page)

```

        "state": "active",
        "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"4x10G\"}}",
        "speed": {
            "unit": "G",
            "value": 10
        },
        "name": "swp53s0",
        "interface_id": 1
    },
    {
        "state": "active",
        "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"4x10G\"}}",
        "speed": {
            "unit": "G",
            "value": 10
        },
        "name": "swp53s1",
        "interface_id": 2
    },
    {
        "state": "active",
        "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"4x10G\"}}",
        "speed": {
            "unit": "G",
            "value": 10
        },
        "name": "swp53s2",
        "interface_id": 3
    },
    {
        "state": "active",
        "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"4x10G\"}}",
        "speed": {
            "unit": "G",
            "value": 10
        },
        "name": "swp53s3",
        "interface_id": 4
    }
],
"transformation_id": 2
}
],
"port_id": 53,
"failure_domain_id": 1,
"column_id": 27
},
{
    "panel_id": 1,
    "slot_id": 0,
    "connector_type": "qsfp",
    "row_id": 2,
    "transformations": [
        {
            "is_default": true,
            "interfaces": [
                {

```

(continues on next page)

(continued from previous page)

```

        "state": "active",
        "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"40G\"}}",
        "speed": {
            "unit": "G",
            "value": 40
        },
        "name": "swp54",
        "interface_id": 1
    },
    ],
    "transformation_id": 1
},
{
    "is_default": false,
    "interfaces": [
        {
            "state": "active",
            "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"4x10G\"}}",
            "speed": {
                "unit": "G",
                "value": 10
            },
            "name": "swp54s0",
            "interface_id": 1
        },
        {
            "state": "active",
            "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"4x10G\"}}",
            "speed": {
                "unit": "G",
                "value": 10
            },
            "name": "swp54s1",
            "interface_id": 2
        },
        {
            "state": "active",
            "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"4x10G\"}}",
            "speed": {
                "unit": "G",
                "value": 10
            },
            "name": "swp54s2",
            "interface_id": 3
        },
        {
            "state": "active",
            "setting": "{\"interface\": {\"command\": \"\", \"speed\": \"4x10G\"}}",
            "speed": {
                "unit": "G",
                "value": 10
            },
            "name": "swp54s3",
            "interface_id": 4
        }
    ],
    "transformation_id": 2
}

```

(continues on next page)

(continued from previous page)

```
    }
  ],
  "port_id": 54,
  "failure_domain_id": 1,
  "column_id": 27
}
]
```

5.8.1.12 References

[Reading syseeprom on Cumulus device] <https://docs.cumulusnetworks.com/cumulus-linux/Monitoring-and-Troubleshooting/Monitoring-System-Hardware/>

[Edgecore AS5712-54X-0] <https://www.edge-core.com/productsInfo.php?cls=1&cls2=8&cls3=44&id=15>

5.8.2 SONiC Device Profile

5.8.2.1 Background

The Device Profiles (DP) are a way of getting a new device to be recognized and usable inside AOS. They essentially capture much of the device specific semantics which are required not only to be discovered by AOS but also run network configs that work well for the datapath once inside the blueprint.

The Logical Device(LD) provides an abstracted view of a network device to be used for a given intent. The user is able to make a logical representation of a device including all information about how it can be used, including roles.

Interface Maps(IM) maps the physical ports as available in a DP to a Logical Device to generate and run port configs on the device.

It is clear that the sanity of the DP is crucial to ensure successful config deploys and effective network traffic passage.

The DP is a REST entity, thus allowing the user to create, edit, delete, list during the design phase of AOS. This DP is then used in the creation of Interface Maps (IM), which get directly used inside the AOS config rendering engine when the Blueprints are deployed.

This document tries to record all the necessary knowledge required to create (and edit) a semantically correct Sonic DP, so that not only does it pass the validations in place in AOS 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.

5.8.2.2 Problem Statement

A user who creates the DP needs to have the device specification (usually released by the vendor). It does not stop there. The user also needs to learn a few things about how to translate these specifications into AOS DP data model so as to create the valid and config-friendly JSON. This is because the DP is a vendor semantics aware data structure.

When there is a new device to be added to AOS, which is not already part of production, one needs to go through the laborious process of first learning the specifications and then tailoring it to the AOS semantics, which has been found to be non-trivial without a guide which explains the purpose, meaning and semantic requirements behind each and every DP data model field.

5.8.2.3 Solution

This document will exhaustively iterate over each and every field in the DP and record the way one should use it when creating a new DP.

The high level data model is the same for all DPs, i.e. there are the same keys for every vendor's DP, just the way we get the values might differ, or might be loaded with a vendor constraint. The document will try and enlist the following:

- The schema of the DP and the nested elements inside the DP.
- The meaning of each key value pair in the schema.
- The vendor specific recipe the values are populated.
- List any constraints, corner cases which need to be considered, especially for port configurations for certain (group of) models.
- Any lessons learnt along the way creating those DPs already in production useful in creating future ones.

5.8.2.4 User Interface

The AOS UI which helps the user to create the DP, while it is intelligent to warn the user against some semantic validations, is not wholly capable of ensuring deep vendor specific constraints and requirements. If the user can take care of having the exact vendor specification, then the AOS UI will certainly help the user create a semantically valid DP which can be inserted into AOS database successfully without any errors.

This document will enlist what it takes to get the vendor specifications and make them part of AOS DP data model. Once the user has these ready, AOS UI can be used to populate the fields and create the DP successfully.

Alternatively, the user can write his own Python code which takes into account the vendor specifications, normalize it as per AOS DP data model and generate the json to then import this into AOS UI.

5.8.2.5 Selector information

Entering the correct information in all the 4 fields of the selector 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 |

5.8.2.6 Capabilities

The hardware and software capabilities are relatively straight forward, if the user has access to the device specification, it is easy to fetch these fields. The LXC and onie fields are set to True by default

Here are some commonly found values in sonic devices(this data is based on the devices AOS already supports)

| Selector Field | Value | Command to get the information on device |
|----------------|------------------------------------|---|
| userland | 64 (int) | This does not affect config as of AOS 3.1.1 |
| form_factor | '1RU' (string) | This does not affect config as of AOS 3.1.1 |
| ecmp_limit | 64 (int) | This does not affect config as of AOS 3.1.1 |
| asic | 'T2' (string) | This does not affect config as of AOS 3.1.1 |
| cpu | 'x86' (string) | This does not affect config as of AOS 3.1.1 |
| ram | 16 (int) (Note, the unit is in GB) | This does not affect config as of AOS 3.1.1 |
| onie | True (bool) | This does not affect config as of AOS 3.1.1 |
| lxc | True (bool) | This does not affect config as of AOS 3.1.1 |

Note: On supported_features in Sonic DPs, AOS does not require the user to configure any extra configurables for “supported features” for Sonic device profiles as of AOS 3.1.1. Thus the UI says “no supported features”.

5.8.2.7 Port Specific Semantics

Interface naming conventions

Sonic follows the naming conventions as 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 author a SONiC device profile, a user must read through the device specific port_config.ini (for example, sonic-buildimage/device/mellanox/x86_64-mlnx_msn2100-r0/ACS-MSN2100/port_config.ini) file and follow the instructions in the above link to come up with the right interface names.

The interface names are explicitly mentioned in the port_config.ini and whatever is given there is what will be used by SONiC and AOS will need to have DP with matching interface names which will be used to generate the PORT configs in the configuration file (config_db.json) . For this document purposes, port_config.ini and config_db.json should have the same interface naming standard. The user should use those interface names in their DP along with the lane numbers provided in the port_cfg.ini file. Once a device profile has been generated based on the aforementioned steps, AOS will use that along with the LD to generate the Interface Map (IM). AOS 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 AOS 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. AOS will try to validate the generated port configs with any of the available options for that platform. AOS currently does not use the Dynamic Port breakout feature which is on-going in the SONiC project.

5.8.2.8 Debugging and recovery

Device mismatch

Usually this is seen at the very beginning of device’s lifecycle. If a device profile is not being picked by the device, then it is advisable to check the 4 fields as entered in the selector section of the DP.

Deploy errors

One of the reasons deploy errors are seen are because of incorrect port configurations. This could be either the ports were configured with incorrect speeds, or the OS specific port constraints were not handled in the DP or in the IM.

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
- The naming constraints can be read from the SoNIC official documentation. For example if the user wants to generate the interface names for Accton 5712 54X running SONIC, the port_config.ini is the authority. https://github.com/Azure/sonic-buildimage/blob/master/device/accton/x86_64-accton_as5712_54x-r0/Accton-AS5712-54X/port_config.ini Sometimes the device might have inter-port constraints. For sonic, it is kind of all laid out in the port_config.ini file. There may be multiple port_config.ini files for a specific platform and a specific manufacturer with each port_config.ini file residing in their on HWSKU folders in the sonic image (like the one I had pointed above). The ability to try out different port speeds on (outside of what is listed in the port_config.ini) will need knowledge of the chipset and also the physical switch manufacturer to see what can be achieved. This information may not be available in any white papers unless we ask the vendors.

5.8.2.9 Example of 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 |

(continues on next page)

(continued from previous page)

| | | | |
|-------------|-----------------|-----------------|----|
| Ethernet96 | 117,118,119,120 | hundredGigE1/25 | 25 |
| Ethernet100 | 113,114,115,116 | hundredGigE1/26 | 26 |
| Ethernet104 | 125,126,127,128 | hundredGigE1/27 | 27 |
| Ethernet108 | 121,122,123,124 | hundredGigE1/28 | 28 |
| Ethernet112 | 5,6,7,8 | hundredGigE1/29 | 29 |
| Ethernet116 | 1,2,3,4 | hundredGigE1/30 | 30 |
| Ethernet120 | 13,14,15,16 | hundredGigE1/31 | 31 |
| Ethernet124 | 9,10,11,12 | hundredGigE1/32 | 32 |

Translate port_config to a port-to-lane_map data structure using parse.py script:

```

Parse.py
=====
#!/usr/bin/python
# Copyright (c) 2017 Apstrktr, Inc. All rights reserved.
# Apstrktr, Inc. Confidential and Proprietary.
#
# This source code is licensed under End User License Agreement found in the
# LICENSE file at http://apstra.com/eula
#
# pylint: disable=line-too-long

import sys
from pprint import pprint

# Run the program as ./parse.py <path_to_sonic_platform_port_config.ini>
# ex: ./parse.py sonic-buildimage/device/mellanox/x86_64-mlnx_msn2100-r0/ACS-MSN2100/
#    port_config.ini
def get_lanemap(buf):
    if not buf:
        return None
    d = {}
    interface_indices = []
    for line in buf.split('\n'):
        if line.startswith('#'):
            continue
        words = line.split(' ')
        words = [word for word in words if len(word)]
        if not len(words):
            continue
        intf = words[0][8:]
        lane = words[1].split(',')
        interface_indices.append(intf)
        if len(lane) > 1:
            one = 'Ethernet' + str(intf)
            two = 'Ethernet' + str(int(intf)+1)
            three = 'Ethernet' + str(int(intf)+2)
            four = 'Ethernet' + str(int(intf)+3)
            d.update({one:lane[0]})
            d.update({two:lane[1]})
            d.update({three:lane[2]})
            d.update({four:lane[3]})
        else:
            d.update({words[0]:words[1]})
    return {'interface_names' : interface_indices, 'lane_mapping' : d}

def parse_portconfig(f):

```

(continues on next page)

(continued from previous page)

```

    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',

```

(continues on next page)

(continued from previous page)

```
        '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',
```

(continues on next page)

(continued from previous page)

```
'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',
```

(continues on next page)

(continued from previous page)

```

'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 AOS:

```

{
  "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",

```

(continues on next page)

(continued from previous page)

```

        "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
        }
    ],
    "is_default": true,
    "transformation_id": 1
},

```

(continues on next page)

(continued from previous page)

```

    {
      "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"
          }
        ]
      }
    ]
  }
]

```

(continues on next page)

(continued from previous page)

```

    }
    ],
    "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,

```

(continues on next page)

(continued from previous page)

```

    "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": {

```

(continues on next page)

(continued from previous page)

```

        "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": [
                {

```

(continues on next page)

(continued from previous page)

```

        "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,

```

(continues on next page)

(continued from previous page)

```

        "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,

```

(continues on next page)

(continued from previous page)

```

    "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
      }
    ]
  }
]

```

(continues on next page)

(continued from previous page)

```

        "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\"}}",

```

(continues on next page)

(continued from previous page)

```

        "speed": {
            "unit": "G",
            "value": 40
        },
        "state": "active"
    }
],
"is_default": false,
"transformation_id": 2
}
]
},
{
    "column_id": 7,
    "connector_type": "qsfp28",
    "failure_domain_id": 1,
    "panel_id": 1,
    "port_id": 12,
    "row_id": 1,
    "slot_id": 0,
    "transformations": [
        {
            "interfaces": [
                {
                    "interface_id": 1,
                    "name": "Ethernet48",
                    "setting": "{\"interface\": {\"speed\": \"100000\", \"lane_map\": \"81,
↪82,83,84\"}}",
                    "speed": {
                        "unit": "G",
                        "value": 100
                    },
                    "state": "active"
                }
            ],
            "is_default": true,
            "transformation_id": 1
        },
        {
            "interfaces": [
                {
                    "interface_id": 1,
                    "name": "Ethernet48",
                    "setting": "{\"interface\": {\"speed\": \"40000\", \"lane_map\": \"81,
↪82,83,84\"}}",
                    "speed": {
                        "unit": "G",
                        "value": 40
                    },
                    "state": "active"
                }
            ],
            "is_default": false,
            "transformation_id": 2
        }
    ]
}
],
},

```

(continues on next page)

(continued from previous page)

```

{
  "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": [
        {

```

(continues on next page)

(continued from previous page)

```

        "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,

```

(continues on next page)

(continued from previous page)

```

        "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
                    }
                }
            ]
        }
    ]
}

```

(continues on next page)

(continued from previous page)

```

        },
        "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",

```

(continues on next page)

(continued from previous page)

```

    "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\"}}",

```

(continues on next page)

(continued from previous page)

```

        "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": [

```

(continues on next page)

(continued from previous page)

```

        {
            "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"
                }
            ]
        }
    ],

```

(continues on next page)

(continued from previous page)

```

        "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,

```

(continues on next page)

(continued from previous page)

```

"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
          }
        }
      ]
    }
  ]
}

```

(continues on next page)

(continued from previous page)

```

        },
        "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",

```

(continues on next page)

(continued from previous page)

```

        "setting": "{\"interface\": {\"speed\": \"40000\", \"lane_map\": \"113,
↪114,115,116\"}}",
        "speed": {
            "unit": "G",
            "value": 40
        },
        "state": "active"
    }
],
    "is_default": false,
    "transformation_id": 2
}
],
{
    "column_id": 14,
    "connector_type": "qsfp28",
    "failure_domain_id": 1,
    "panel_id": 1,
    "port_id": 26,
    "row_id": 1,
    "slot_id": 0,
    "transformations": [
        {
            "interfaces": [
                {
                    "interface_id": 1,
                    "name": "Ethernet104",
                    "setting": "{\"interface\": {\"speed\": \"100000\", \"lane_map\": \"125,
↪126,127,128\"}}",
                    "speed": {
                        "unit": "G",
                        "value": 100
                    },
                    "state": "active"
                }
            ],
            "is_default": true,
            "transformation_id": 1
        },
        {
            "interfaces": [
                {
                    "interface_id": 1,
                    "name": "Ethernet104",
                    "setting": "{\"interface\": {\"speed\": \"40000\", \"lane_map\": \"125,
↪126,127,128\"}}",
                    "speed": {
                        "unit": "G",
                        "value": 40
                    },
                    "state": "active"
                }
            ],
            "is_default": false,
            "transformation_id": 2
        }
    ]
}

```

(continues on next page)

(continued from previous page)

```

    ]
  },
  {
    "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": [
      {

```

(continues on next page)

(continued from previous page)

```

    "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"
        }
      ]
    }
  ]
}

```

(continues on next page)

(continued from previous page)

```

    ],
    "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": {

```

(continues on next page)

(continued from previous page)

```

        "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
        }
    ]
}
],
}
],

```

(continues on next page)

(continued from previous page)

```

"selector": {
  "manufacturer": "Dell|DELL",
  "model": "Z9100-ON",
  "os": "SONiC",
  "os_version": ".*"
},
"slot_count": 0,
"software_capabilities": {
  "lxc_support": false,
  "onie": true
}
}

```

5.8.3 Juniper Device Profile

5.8.3.1 Overview

New in version 3.3.0: AOS now supports Juniper QFX10002, QFX5110 and QFX5100 series Junos devices.

Apstra AOS provides Device Profiles for Juniper QFX10002, QFX5110 and QFX5100 series Junos devices.

This will document any constraints for these AOS Device Profiles.

5.8.3.2 Juniper QFX10002

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

☆ 🏠 > Devices > Device Profiles > Juniper_QFX10002-36Q

| |
|-------------|
| 2RU |
| ASIC® Q5(3) |

Ports

Panel #1

INTERFACES CAPACITY

36 x 40 Gbps 144 x 10 Gbps 12 x 100 Gbps

PORTS Click on port to toggle the details Port breakout Autonegotiation

PORT DETAILS

| | |
|----------------|--------|
| ID | 2 |
| Connector type | qsfp28 |

Transformations

| | |
|-----------------------------------|---|
| Port #2 Tr. #1 (40 Gbps, default) | et-0/0/1 |
| Port #2 Tr. #2 (100 Gbps) | et-0/0/1 |
| Port #2 Tr. #3 (10 Gbps) | xe-0/0/1.0 xe-0/0/1.1 xe-0/0/1.2 xe-0/0/1.3 |

If these ports are used as 100G, then the adjacent QSFP 40G ports cannot be used. The AOS Device Profile cannot automatically disable the adjacent QSFP 40G ports. The user must create an AOS Interface Map with these ports unused and disabled.

For the QFX10002 when creating the AOS Interface Map, after selecting the 100G ports, AOS will display an option to the user. “Do you want to select the disabled interfaces for unused device profile ports?”

Create Interface Map

Name: Juniper_QFX10002-36Q__AOS-12x100-1

Logical device: AOS-12x100-1

Device profile: Juniper_QFX10002-36Q

Map interfaces

| Logical device port groups | | Mapped/required number of interfaces | Device profile interfaces |
|----------------------------|--------------|--------------------------------------|---------------------------|
| Speed | Connected To | | |
| 100 Gbps | Leaf | 12 / 12 | Select interfaces |

Transformation #2

Interface #1 (12 ports)

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

Interface map preview

Create Another?

For 100G ports on the QFX10002, the user must click the **OK** option for this so the unused QSFP ports are disabled and cannot be used.

5.8.4 Device Profiles Overview

5.8.4.1 Hardware Capabilities

As of version 3.1.0, AOS supports configuring device-specific hardware features. Device Profiles are where you express how you want to configure the capabilities of the device. AOS supports three such capabilities that affect configuration rendering. Some feature capabilities have different behaviors across NOS versions and thus, capabilities can be expressed per NOS version. By default, the version matches all supported versions.

Edit Device Profile

Device profiles need to accurately model various characteristics of a switch model. Make sure you update the profile to match the new switch model(s) you intend to use this profile for.

Updating the device profile ports may not be allowed because it is referenced by `Cisco_9372PX_NXOS_AOS-48x10_6x40-1` interface map.

Summary

Selector?

Capabilities

Ports

Hardware Capabilities

CPU *

x86

Userland (bits) *

64

RAM (GB) *

16

ECMP limit *

64

Form factor *

1RU

ASIC

T2

Supported Features

COPP?

Enabled? *




Version *

.



[Home](#) > [Devices](#) > [Device Profiles](#) > Cisco C9372PX

| | |
|---------------|---|
| Modular? | no |
| Slot count | 0 |
| Ports preview |  |

Selectors[?]

| | |
|--------------|------------------------------------|
| Manufacturer | Cisco |
| Model | C9372PX |
| OS family | NXOS |
| Version | 7L.D(3V)(4-7)L(w(a-z)V)/9.L.2(lw+) |

Capabilities

Hardware Capabilities

| | |
|-----------------|-----|
| CPU | x86 |
| Userland (bits) | 64 |
| RAM (GB) | 16 |
| ECMP limit | 64 |
| Form factor | 1RU |
| ASIC | T2 |

Supported Features

| | |
|---------------------------------------|----------------------------|
| COPP [?] | yes (Version: *) |
| Breakout [?] | no (Version: :, Module: 1) |
| Sequence Numbers Support [?] | yes (Version: *) |

Software Capabilities

| | |
|------|----|
| LXC | no |
| ONIE | no |

Ports

When CoPP is enabled for a specific version, AOS renders strict copp profile config on the NX-OS devices resulting in the following configuration rendering:

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)#
```

5.8.4.3 Autonomous Systems Sequencing Support

Autonomous Systems Sequencing Support applies to the autonomous system (AS) path. When enabled, AOS sequences into the entry list to resequence and render the right AS config. For platforms, such as Cisco3172, that do not support sequence numbers, disabling this feature ensures that AOS removes the AS sequence numbers from the device model dictionary to avoid addition and negation in the event AOS resequences something. In such scenarios, there is no requirement to render anything on these platforms, because we cannot even sequence the entry.

If **Sequence Numbers Support** is enabled, the device supports sequence numbers and AOS can use the sequence number to 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$
```

Note that the numbers 5 and 15 are sequence numbers applicable to devices that support AS sequencing.

5.8.4.4 Interface Breakout

Fundamentally, this field is telling whether ports on a specific module in a device can be broken down to lower speed split ports or not.

In AOS 3.1.0, the field is used only by NX-OS Device Profiles. However, any vendor OS which wants to express breakout capability of a port can use this field.

Enabling breakout for a specific module indicates that the ports on that module in a device can be broken down to lower speed split ports. If the value for a particular module (let's say, here, 1) is set to True, then AOS renders the following config:

```
no interface breakout module 1
!
```

Because the negation command is always applicable per module, this particular capability in the DP also has to be configured per module. There are a few advantages in making the capability per module:

- In modular systems, not all linecards have breakout capable ports.
- In non-modular systems, the breakout capable ports may not always be in module 1.

Assumptions

AOS assumes we first un-breakout all ports that are breakout-capable, and then apply the proper breakout commands according to intent. This is based on another assumption that the global negation command “no interface breakout module <module_number>” always can be successfully applied to a module with breakout capable ports. 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.

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. AOS reverts any such breakout config that might have been done during the POAP stage to ensure that the ports are put back to default speed by applying the negation command.

To do this, AOS runs the following global negation command, which is idempotent when applied on ports that are not broken out:

```
no interface breakout module <module_number>
```

Interface Maps

When you create *Interface Maps*, you'll be assigning Device Profiles (and Logical Devices) to them. Make sure Device Profiles exist for all physical devices you'll be using.

Device Profiles (DP) specify the details of hardware devices that AOS supports. As additional hardware models are qualified, they are added to the *list of supported devices*.

5.8.5 Listing Device Profiles

From the AOS Web interface, navigate to **Devices / Device Profiles** to see the list of Device Profiles. AOS ships with many pre-defined profiles. To sort the list, click the header of the category you'd like to sort. Click the header again to reverse the sort order.

Click to sort by

Click for details

| Label | Manufacturer | Hardware Model | Modular? | OS Family | OS Version | Actions |
|--------------------------|----------------|-------------------|----------|-----------|---------------------|-------------------|
| Accton 5712-54X-O | Accton | 5712-54X-O* | no | Cumulus | (3\.[5-7])\d+ | Edit Clone Delete |
| Accton 6712-32X-O | Accton | 6712-32X-O* | no | Cumulus | (3\.[5-7])\d+ | Edit Clone Delete |
| Accton-AS5712-54X_SONIC | Esacore Accton | 5712-54X-O* | no | SONIC | * | Edit Clone Delete |
| Arista DCS-7050CX3-32S | Arista | DCS-7050CX3-32S | no | EOS | 4\.(18 20 21 22)\.* | Edit Clone Delete |
| Arista DCS-7050QX-32 | Arista | DCS-7050QX-32 | no | EOS | 4\.(18 20 21 22)\.* | Edit Clone Delete |
| Arista DCS-7050QX-32S | Arista | DCS-7050QX-32S | no | EOS | 4\.(18 20 21 22)\.* | Edit Clone Delete |
| Arista DCS-7050SX2-72Q | Arista | DCS-7050SX2-72Q | no | EOS | 4\.(18 20 21 22)\.* | Edit Clone Delete |
| Arista DCS-7050SX3-48YC8 | Arista | DCS-7050SX3-48YC8 | no | EOS | 4\.(18 20 21 22)\.* | Edit Clone Delete |
| Arista DCS-7050SX-64 | Arista | DCS-7050SX-64 | no | EOS | 4\.(18 20 21 22)\.* | Edit Clone Delete |
| Arista DCS-7050SX-128 | Arista | DCS-7050SX-128 | no | EOS | 4\.(18 20 21 22)\.* | Edit Clone Delete |
| Arista DCS-7050T-36 | Arista | DCS-7050T-36 | no | EOS | 4\.(18 20 21 22)\.* | Edit Clone Delete |
| Arista DCS-7050TX-48 | Arista | DCS-7050TX-48 | no | EOS | 4\.(18 20 21 22)\.* | Edit Clone Delete |
| Arista DCS-7050TX-64 | Arista | DCS-7050TX-64 | no | EOS | 4\.(18 20 21 22)\.* | Edit Clone Delete |

5.8.6 Viewing a Device Profile

Device Profile Details

AOS does not necessarily use all this information for modeling. It's made available to other AOS API orchestration tools for collection and use.

From the list view (**Devices / Device Profiles**) click a Device Profile name to view its details. Device Profile parameters include:

Summary Section

Label Name of the Device Profile

Slot count Number of slots or modules on the device. Modular switches have multiple slots.

Start from ID

Selector Section Device-specific information to match the hardware device to the Device Profile

Manufacturer Name of the manufacturer

Model Determines whether a Device Profile can be applied to specific hardware. Select from the drop-down list or use a Regular Expression (regex) to define one that's not on the list.

OS family Defines how configuration is generated, how telemetry commands are rendered, and how configuration is deployed on a device. Select a currently supported OS family (CentOS, Cumulus, EOS, NX-OS, SONiC, Ubuntu GNU/Linux) from the drop-down list or add an OS family for any other Operating System.

Version Determines whether a Device Profile can be applied to specific hardware. Select from the drop-down list or define your own regex.

Capabilities Section The hardware and software capabilities defined in this section can be leveraged in other parts of AOS to adapt the generated configuration, or to prevent an incompatible situation.

Hardware Capabilities With the exception of ECMP, Hardware Capabilities modify configuration rendering or deployment.

CPU (cpu:string) Architecture of the CPU. Example: x86

Userland (bits) (userland:integer) Type of userland (application binary/kernel) that is supported. Example: 64

RAM (GB) (ram:integer) Amount of memory that the device has. Example: 16

ECMP limit (ecmp_limit:integer) Maximum number of Equal Cost Multi Path routes. This field changes BGP configuration on the device (ecmp max-paths). Example: 64

Form factor (form_factor:string) Amount of rack space the device uses. Examples: 1RU, 2RU, 6RU, 7RU, 11RU, 13RU

ASIC (asic:string) Type of Chipset (ASIC) on the switch. Examples: “ , ``T2, T2(3), T2(6), Arad(3), Alta, TH, Spectrum, XPliant XP80, ASE2, Jericho

Supported Features (as of 3.1.0, Cisco only)

COPP When Control Plane Policing is enabled, strict CoPP profile config is rendered for the specified NX-OS version. Multiple versions can be specified.

Enabled for all versions by default (except Cisco 3172PQ NXOS, which is disabled by default).

Breakout Enable breakout to indicate that ports on the specified module can be broken out to lower speed split ports. Each module is specified individually.

Disabled for the following devices with modules incapable of breaking out ports: Cisco 3172PQ NXOS, Cisco 9372TX NXOS, Cisco C9372PX NXOS, Cisco C9396PX NXOS, Cisco NXOSv

You can express breakout capability of a port for any vendor OS.

Sequence Numbers Support (for Autonomous System (AS) path) Enable for the ability to sequence into the entry list to resequence and render the right AS config.

Enabled for all Cisco Device Profiles by default (except Cisco 3172PQ NXOS, which does not support sequence numbers).

Other supported features known to AOS 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

Software Capabilities

LXC (lxc_support: boolean) Select if the device supports LXC containers.

ONIE (onie: boolean) Select if the device supports ONIE.

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.

Port Index (port_id: integer) Indicates a unique port in the collection of ports in the Device Profile.

Row Index (row_id: integer) Represents the top-to-bottom dimensions of the port panel. Shows where the port is placed in the device's panel. For instance, in a panel with two rows and many columns the row index is either 1 or 2.

Column Index (column_id: integer) Represents the left-to-right dimensions of the port panel. Shows where the port is placed in the device's panel. For instance, in a panel with thirty-two ports and two rows, the column index is in the range of 1 through 16.

Panel Index (panel_id: integer) Indicates the panel that the port belongs to given the physical layout of ports in the device specification

Slot ID (slot_id: integer) Represents the module that the port belongs to. A modular switch has more than one slot. In fixed function network function devices, Slot ID is usually 0.

Failure Domain (failure_domain_id: integer) Indicates if multiple panels are relying on the same hardware components. Used when creating the cabling plan to ensure that two uplinks are not attached to the same failure domain.

Connector Type (connector_type: string) Port transceiver type. Speed capabilities of the port are directly related to the connector type, given that certain connector types can run in certain speeds. For instance, sfp, sfp28, qsfp, qsfp28.

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

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. Thus, AOS 2.3.1 uses a vendor specific schema to govern the user input of the interface setting and catch any potential validation errors at time of create and edit of device profiles.

Based on the OS information you enter in the device profile's selector field, the Web interface displays the applicable fields that you need to enter for the settings. The fields vary with the vendor OS (as found in examples below). The `setting` is validated based on the vendor specific schema (as listed below) when a Device Profile is created or edited.

The actual and expected vendor specific schema for the settings is as follows:

```
eos_port_setting = Dict({
    'interface': Dict({
        'speed': Enum([
            '', '1000full', '10000full', '25gfull', '40gfull',
            '50gfull', '100gfull',
        ])),
    'global': Dict({
        'port_group': Integer(),
```

(continues on next page)

(continued from previous page)

```

        '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())
    })
})

opx_port_setting = Dict({
    'interface': Dict({
        "command": String(),
        "speed": String(),
        "mode": String(),
        "default_speed": String()
    })
})

sonic_port_setting = Dict({
    'interface': Dict({
        "command": Optional(String()),
        "speed": String(),
        "lane_map": Optional(String())
    })
})

cumulus_port_setting = Dict({
    'interface': Dict({
        'speed': String(),
        'command': String()
    })
})

```

5.8.6.1 Vendor Specific Schema for Port Settings

As of version 2.3.1 AOS supports vendor specific schema for port settings in Device Profiles and Interface Maps. Prior to AOS 2.3.1, the setting field in the Device Profile and Interface Map was a string whose contents were not bound by any schema. The dictionary inside the string would be built differently for different vendors. The lack of schema for these vendor-based settings pose multiple problems on how you would configure features like auto-negotiation on various vendor switches.

AOS now allows you to create a well-defined schema for the port settings that provide you with the flexibility to choose the keys and values for the port settings in expectation of effective configuration rendering. You have better control over selecting certain keys and values in the design phase to expect a corresponding config in the deploy phase.

For example, to turn auto negotiation on, the vendor-based schema serves as a bound on what you can select in-order to enable auto neg all the way on the device interfaces.

Introduction of vendor-based schema for port settings does not only provide you a flexible way to configure the ports with features, but also allows AOS backend to elegantly consume the information in the DP and IM to calculate the rendering of these interfaces thus making the Jinja templates simpler.

5.8.7 Creating a Device Profile

Important: Device Profiles contain extensive *hardware model details*. When creating a Device Profile, ensure that the profile accurately describes all hardware characteristics. When in doubt, contact [Apstra Support](#) for assistance.

1. From the list view (**Devices / Device Profiles**), click **Create Device Profile** to see the dialog for creating a Device Profile.
2. If you've created JSON payload you can import it by clicking on **Import Device Profile** and selecting the file. Otherwise, continue to the next step.
3. Enter a Device Profile name.
4. Configure the Device Profile to match the characteristics of the physical device. See above for *parameter descriptions*
5. Click **Create** to create the Device Profile.

5.8.8 Cloning a Device Profile

Instead of creating a Device Profile from scratch that is similar to one that already exists, you can clone that existing one and change the few details that are different.

1. Either from the list view (**Devices / Device Profiles**), or the details view, click the **Clone** button corresponding to the Device Profile to clone. You'll see the dialog for cloning a Device Profile.
2. Enter a unique name for the new Device Profile, and make your changes.
3. Click **Create** (bottom-right) to create the Device Profile.

Warning: AOS built-in Device Profiles are managed and updated by Apstra, and the changes are automatically propagated to the imported Interface Maps in the Blueprints on AOS upgrade. However, manually created or cloned Device Profiles are not managed or updated by Apstra.

5.8.9 Editing a Device Profile

Important: Changes in built-in Device Profiles made by users are not reflected on AOS upgrade. Hence, cloning the built-in Device Profiles instead of editing is recommended.

Warning: Manually changing a Device Profile can lead to a mismatch between the profile's stated capabilities and the device's actual capabilities, potentially leading to unexpected results.

If a Device Profile is referenced by an *Interface Map*, you may not be able to change it if it would adversely affect that Interface Map.

1. Either from the list view (**Devices / Device Profiles**) or the details view, click the **Edit** button corresponding to the Device Profile to edit. You'll see the dialog for editing a Device Profile.
2. Make your changes.
3. Click **Update** (bottom-right) to update the Device Profile.

5.8.9.1 Use Case: Enabling Auto-negotiation

This example shows how to set a port on an **Arista DCS-7050TX-72Q** device to auto-negotiate its speed.

1. From the AOS Web interface, navigate to **Devices / Device Profiles** to see the list of existing Device Profiles.
2. Click the **Edit** button corresponding to **Arista DCS-7050TX-72Q** to see the dialog for editing that Device Profile.
3. Click **Ports** (left) to see a graphical representation of the ports.
4. Select one or more ports in the panel to see port details.
5. In the **Transformations** section, click the **Edit** button corresponding to the port to be edited.
6. Populate the port setting field with the string below. An empty string in the *speed* field signifies that the port is set to auto-negotiate. AOS models the maximum speed for the port to the speed in the fabric.

Listing 45: Port setting

```
{ "interface": { "speed": "" }, "global": { "select": "", "port_group": -1 } }
```

You can edit the fields, such as *command*, inside the port settings instead. AOS validates the command contents before the update is applied.

Arista Devices

Enabling auto-negotiation on an Arista device results in an automatically updated port setting in AOS, because the schema for enabling auto-negotiation on an Arista device is well-defined, as shown below.

Listing 46: Port setting

```
{ "interface":
  { "speed": "<String> example ",
    "global": "<Dict> example" {
      "select": <String> example "",
      "port_group": <Integer> example -1
    }
  }
```

(continues on next page)

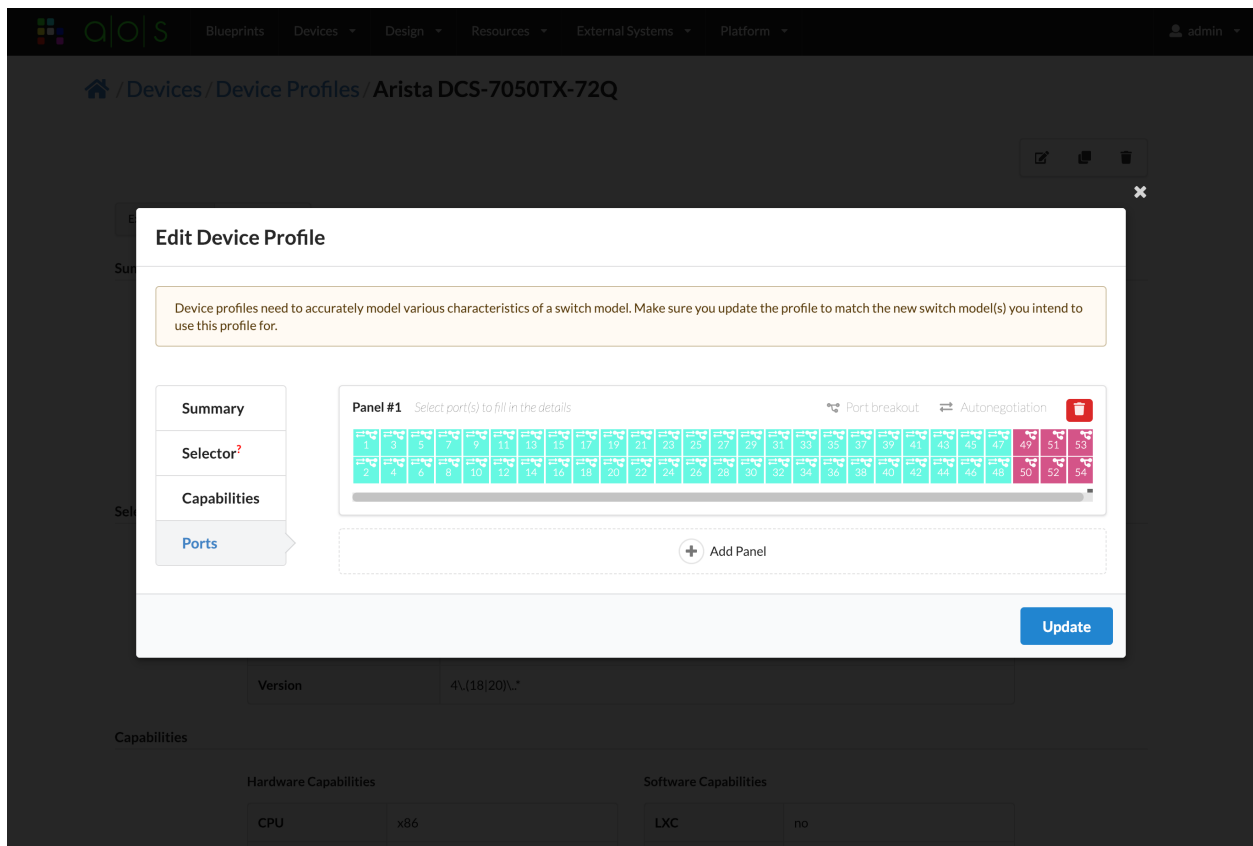


Fig. 1: Device Profile - Port View

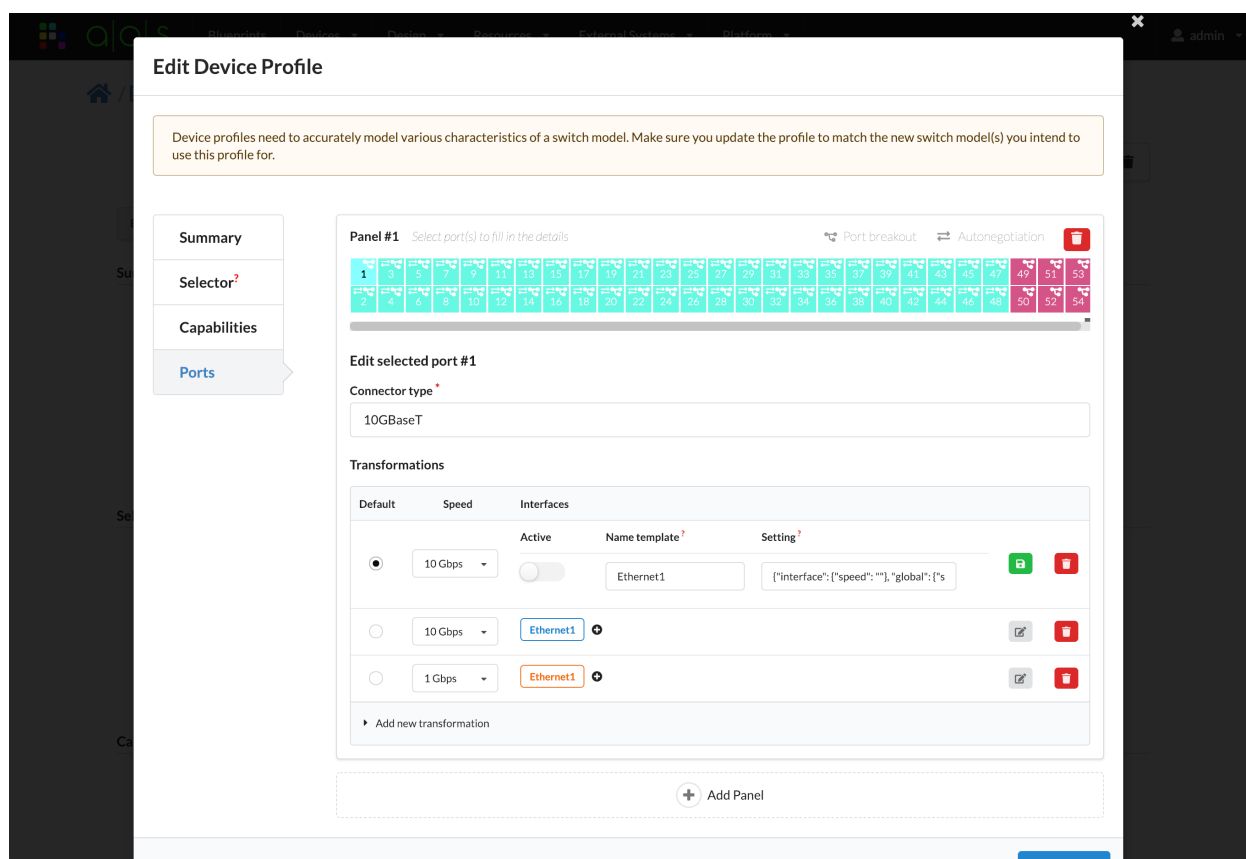


Fig. 2: Device Profile - Port Details View

(continued from previous page)

```
"command": <to turn on auto neg on the port>
}
```

7. Click **Update** to save your changes.

5.8.10 Deleting a Device Profile

If a Device Profile is referenced by an *Interface Map*, you cannot delete it.

1. Either from the list view (**Devices / Device Profiles**) or the details view, click the **Delete** button corresponding to the Device Profile to delete. You'll see the dialog for deleting a Device Profile.
2. Click **Delete Device Profile** to delete the Device Profile.

5.8.11 Device Profiles and REST API

You can also work with Device Profiles via REST API. From the AOS Web interface, navigate to **Platform / Developers** to access REST API documentation.

6.1 Logical Devices

6.1.1 Logical Device Overview

Networks can be designed in AOS before considering hardware vendors by using logical devices (abstractions of physical devices) to specify common device form factors such as number, speeds and roles of each port in one or more rack units (panels). Some of the capabilities of logical devices include:

- Specifying speed and roles for specific ports (For example, the 48th port is always an L3 router edge, 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, when using a 1 GbE SFP in a 10 GbE port AOS correctly configures the underlying hardware.)
- Solving for automatic cable map generation that takes into account failure domains on modular systems (For example, a line card).

When hardware devices have been selected, logical devices can be mapped to them with interface maps. Logical devices are also embedded into rack types and rack-based templates.

1. Click for details

2. Click to sort

3. Create Logical Device

4. Click to search

| Name | Capabilities | Panels Count | Ports Count | Ports Summary | Actions |
|------------|--------------|--------------|-------------|--|-------------------|
| AOS-1x1-1 | 1 x 1 Gbps | 1 | 1 | AOS-1x1-1 1 x 1 Gbps Leaf • Access | Edit Clone Delete |
| AOS-1x10-1 | 1 x 10 Gbps | 1 | 1 | AOS-1x10-1 1 x 10 Gbps Leaf • Access | Edit Clone Delete |

To access logical devices - from the AOS web interface, navigate to **Design /Logical Devices**. AOS ships with many predefined logical devices.

Name

AOS-96x10-8x40-2

PANEL #1

TOTAL

PORT GROUPS

Connected to ▾

96 ports

96 x 10 Gbps
L2 Server • L3 Server

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 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 |

PANEL #2

TOTAL

PORT GROUPS

Connected to ▾

8 ports

8 x 40 Gbps
Superspine • Spine •
External Router

| | | | |
|---|---|---|---|
| 1 | 3 | 5 | 7 |
| 2 | 4 | 6 | 8 |

To see details - click a logical device name. Logical devices include the following details:

Name 64 characters or fewer to identify the logical device

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

Roles

Superspine Port is configured to face a superspine device. Applies to 5-stage Clos datacenter fabric only.

Spine Port is configured to face a spine device.

Leaf Port is configured to face a leaf device.

Access (Not supported in AOS 3.3.0) Port is configured to face an access device.

L2 Server Port is configured to face an L2 server and trunk towards it, with SVI for VLANs, LAG/MLAG for single and dual-attached servers.

L3 Server Port is configured to face an L3 server with routing on the host (BGP).

External Router Port is configured to face a router that is not managed by AOS.

Peer Port is configured as a peer link between two leaf devices.

Unused AOS does not attempt to render configuration on a port with the unused role (ex: a dead port)

Important: **Access Switches** have limited support in AOS version 3.3.0. Please contact Apstra Support to learn more about this feature and the limitations.

Cloning Logical Device

Instead of entering all details for a new logical device, you can clone an existing one, give it a new name and customize it.

6.1.2 Creating Logical Device

1. From the list view (Design / Logical Devices), click **Create Logical Device** and enter a name.
2. 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.
3. Select the ports for the port group by dragging to select contiguous ports, or by clicking individual ports. Clicking a port again deselects it.
4. Select port speed, and applicable role(s) for the selected ports.
5. Click **Create Port Group** (bottom-middle) to create the port group.
6. 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.
7. To add a panel, click **Add Panel** (bottom-middle) and repeat the steps as for the first panel.
8. Click **Create** (bottom-right) to create the logical device and return to the list view.

6.1.2.1 Creating Logical Device - Example

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 list view (Design / Logical Devices) click **Create Logical Device**.

Create Logical Device

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

Name *

AOS-96x10-8x40-2 ← Enter name

PANEL #1

TOTAL

24 ports
0 assigned • 24 available

PORT GROUPS

No port groups created

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

Drag to change port configuration ←

Create port group

Number of ports *

24

Speed *

10 Gbps

Connected To *

☐ Superspine
☐ Spine
☐ Leaf
☐ Access
☐ L2 Server
☐ L3 Server
☐ External Router
☐ Peer
☐ Unused

Create Port Group

- A descriptive name is helpful when referring to the logical device later. For our example we entered **AOS-96x10-8x40-2**, which represents the following characteristics:
 - AOS - the operating system using this Logical Device
 - 96x10 - one panel with 96 - 10 GbE ports
 - 8x40 - one panel with 8 - 40 GbE ports
 - 2 - number of panels (rack units)
- 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 **L2 Server** and **L3 Server** for roles (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 *

AOS-96x10-8x40-2

PANEL #1

TOTAL

96 ports
0 assigned • 96 available

PORT GROUPS

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

☒ L2 Server

☒ L3 Server

☐ External Router

☐ Peer

☐ Unused

Select roles

Create Port Group

+ Add Panel

- Click **Create Port Group** (bottom-middle), then click **Add Panel** (bottom-middle).
- Drag the bottom-right corner of the port layout to change the configuration to 2x4. Leave the number or ports (8) as is, change the speed to **40 Gbps**, and connect them to **Superspine**, **Spine**, and **External Router**.
- Click **Create Port Group**, then click **Create** (bottom-right). The finalized logical device can be seen in the logical device overview *above*.

6.1.3 Editing Logical Device

If a logical device is linked to an *interface map*, it cannot be changed.

To prevent potentially unintended changes to existing rack types or templates, changes to logical devices do not affect them. If the intent is for a rack type or template to use a modified logical device, then the rack type must be *re-imported into the template*.

- Either from the list view (Design / Logical Device) or the details view, click the **Edit** button for the logical device to edit.
- 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 it and return to the list view.

6.1.4 Deleting Logical Device

If a logical device is linked to an *interface map*, it cannot be deleted.

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

6.2 Interface Maps

6.2.1 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.

1. Design

2. Interface Maps

3. Create Interface Map

Query: All Click to search 1-25 of 144

Table View Card View Page Size: 25

| Name | Device Profile | Logical Device | Actions |
|---|------------------------------|------------------|-------------------|
| Accton-A55712-54X_SONIC_AOS-72x10-2 | Accton-A55712-54X_SONIC | AOS-72x10-2 | Edit Clone Delete |
| Accton-A55712-54X_SONIC_BRCM_AOS-48x10_6x40-1 | Accton-A55712-54X_SONIC_BRCM | AOS-48x10+6x40-1 | Edit Clone Delete |

To access interface maps - from the AOS web interface, navigate to **Design / Interface Maps**. AOS ships with many predefined interface maps.

6.2.2 Creating Interface Map

1. From the list view (Design / Interface Maps), click **Create Interface Map**, then enter a 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.
2. Select a logical device from the drop-down list. If you don't see a logical device that fits your requirements, you can *create* one.
3. Select a device profile from the drop-down list. If you don't see a device profile that fits your requirements, you can *create* one.
4. Map the logical device to the device profile. See example below for details.
5. Click **Create** to create the interface map and return to the list view.

6.2.2.1 Creating Interface Map - Example: 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 an **AOS-96x10-8x40-2** logical device.

1. From the list view (Design / Interface Maps), click **Create Interface Map**, and leave the name blank for AOS to name it for you.

AOS 96x10 8x40

AOS-96x10-8x40-2 is not one of the predefined logical devices that ships with AOS, so if you have not created it you will not find it in the drop-down list. If you'd like to follow along with this example, you can create the *logical device* before continuing.

2. From the **Logical Device** drop-down list, select **AOS-96x10-8x40-2**. This logical device has 96-10 GbE ports for servers and 8-40 GbE ports for uplinks to spine switches or external routers.
3. From the **Device Profile** drop-down list, select **Arista DCS-7050QX-32**. This device has 24-40 GbE QSFP+ ports that are transformable (4x10 GbE or 1x40 GbE) and 8-40 GbE QSFP+ ports that are not transformable. As soon as both the logical device and device profile are selected, the interface map name is automatically populated.
4. Under **Device profile interfaces** (middle-right), click **Select Interfaces** for the 10 GbE logical ports to see the port layout.

Create Interface Map

Name *

Arista DCS-7050QX-32__AOS-96x10-8x40-2

Logical device *

AOS-96x10-8x40-2

Device profile *

Arista DCS-7050QX-32

Map interfaces

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

Interface map preview Click on interface to toggle the details

- Drag to select the first 24 ports. As the ports are selected the white numbers turn gray. When all interfaces are selected the red circle turns green.
- Under **Device profile interfaces** (middle-right), click **Select Interfaces** for the 40 GbE ports to see 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

- 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

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 | 8 / 8 | ▶ Select interfaces |

Interface map preview Click on interface to toggle the details

☐ Create Another? **Create**

- Click **Create** to create the interface map. The finalized interface map can be seen in the interface map overview *above*.

6.2.2.2 Use Case: Inter Port Constraints - Disabled Ports

Inter Port Constraint Overview

Inter port constraints for Cumulus devices are handled in both the *device profile* and the interface map. For AOS 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 5th transformation for ports 2,4,6,8,10,12,14,16,18,20,22, and 24 have been disabled.

| | |
|----------------|---|
| Name | Mellanox_MS2700_Cumulus_AOS-48x10_8x100-1 |
| Logical device | AOS-48x10+8x100-1 ↗ |
| Device profile | Mellanox MSN2700 ↗ |

[illegible]

Disabling Unused Ports

When creating an interface map that requires disabling ports for inter port constraints, AOS will flag you with the prompt **Do you want to select the disabled interfaces for unused device profile ports?** Click **OK** and AOS will automatically set the corresponding ports to disabled.

Create Interface Map

[illegible]

6.2.3 Editing Interface Map

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

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

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

6.2.4 Deleting Interface Map

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

6.3 Rack Types

6.3.1 Rack Type Overview

Rack Types in Blueprints

Racks in running blueprints can be added, deleted, also can be edited (as of version 3.2). Editing Racks can be done either by changing the Rack Types manually, or by modifying the Blueprints directly, e.g. Changing Link Speeds, Adding Servers, Adding & Removing Links, etc. which is so-called “Flexible Fabric Expansion (FFE)”. Rack Types in Blueprints have timestamps. FFE operations will modify the embedded Rack Types then update the timestamps. Embedded Rack Types can be exported into the global catalog. FFE provides complete flexibility in fabric design and the day-2 operations.

Rack types define the type and number of leafs and servers to be used in a rack build. They use *logical devices* (abstractions of physical devices that specify port roles and speeds) so you can design your racks before selecting a specific vendor’s device. Rack types are then used when creating *templates*.

Numerous predefined rack types are included in the global catalog based on common rack designs. You can create your own, clone, edit, and delete existing ones as described in the sections below.

Rack types include the following details:

Summary

Name 17 characters or fewer, and (optional) description

Connectivity Type L2 - Connects to servers via VLAN or VXLAN.

L3 - IP addresses assigned to switch ports, IP address and BGP routing enabled via server agent for servers

IP Version (for L3 connectivity type) IPv4, IPv6, or IPv4-IPv6

Leafs

Leaf Name 64 characters or fewer

Leaf Logical Device Used as ToR leaf switch network device(s)

Links per spine (and speed) Number of leaf-spine links and their speed

Redundancy Protocol

None For single-homed server connections

MLAG For dual-homed server connections. Both switches use the same logical device

Peer Links (and speed) Number of links between MLAG devices, and their speed

Peer Link Port Channel ID

L3 peer links (and speed) Applies when connectivity type is L3. Used mainly for BGP peering between border MLAG leafs in non-default security zones. Mainly used for routed L3 traffic to solve EVPN blackhole issues or if upstream routers go down. L3 peer-links act as backup paths for the north-south traffic. Other than border leaf it can be used on any other ToR leafs as well for avoiding blackholing traffic for a VRF.

L3 Peer Link Port Channel ID

ESI - Junos ONLY (New in version 3.3.0) Ethernet Segment ID assigned to the bundled links. Specifying device platforms other than Juniper Junos (such as Cisco, Cumulus, Arista) will result in blueprint build errors. See [Juniper EVPN Support](#) for information about Juniper ESI support and [ESI MAC MSB blueprint settings](#) for more information about ESI.

Important: When designing racks, make sure the intended platform supports the chosen redundancy protocol. For example, SONiC does not support L3 MLAG peers, and ESI is only supported on Junos.

External Facing When enabled, connectivity is configured from the leaf to the external router. In an MLAG pair, external connectivity is applied to both leafs. For example, 2 x external links with MLAG enabled create four external links.

Access Switches - Junos ONLY (New in version 3.3.0) Limited support. See [Adding Access Layer Switches](#) for more information.

Servers

Name 64 characters or fewer

Server Count Number of servers in the set

Port Channel ID Min and Max Port channel IDs define aggregated Ethernet interfaces that connect each leaf device to the server. default: 1-4096. You can leave the default or (as of version 3.3.0) you can specify it.

Logical Device The server network device

Link

Name 64 characters or fewer

Switch Leaf

LAG Mode (for L2 connectivity type) LACP (Active)

LACP (Passive)

Static LAG (no LACP)

No LAG

Physical link count per leaf (and link speed) Number of links from each server to each leaf and their speed. If using dual leaf switches, this number should be half of the total links attached to the server.

From the web interface, navigate to **Design > Rack Types**.

1. Design

2. Rack Types

3. Create Rack Type

Click to search

Click to sort

Click for details

| Name | Connectivity Type | IP Version | Leaf Count | External Connectivity? | Server Count | Actions |
|-------------------|-------------------|------------|----------------------------|------------------------|--------------|-------------------|
| L2 Access 4x | L2 | IPv4 | 1 single leaf | no | 4 | Edit Clone Delete |
| L2 ESI 2x Links | L2 | IPv4 | 1 ESI group | no | 1 | Edit Clone Delete |
| L2 HPC | L2 | IPv4 | 1 single leaf | no | 16 | Edit Clone Delete |
| L2 Leaf External | L2 | IPv4 | 1 single leaf | yes | 48 | Edit Clone Delete |
| L2 MLAG 1x access | L2 | IPv4 | 1 MLAG pair | no | 2 | Edit Clone Delete |
| L2 MLAG 2acs+1lef | L2 | IPv4 | 1 MLAG pair, 1 single leaf | no | 4 | Edit Clone Delete |
| L2 MLAG 2x access | L2 | IPv4 | 1 MLAG pair | no | 2 | Edit Clone Delete |

Cloning Logical Device

Instead of entering all details for a new logical device, you can clone an existing one, give it a new name and customize it.

6.3.2 Creating Rack Type

1. From the web interface, navigate to **Design > Rack Type** and click **Create Rack Type**.
2. Enter a name, (optional) description, then select a connectivity type (and IP version if L3 connectivity).
3. Configure the panel as required for your design.
 - See rack type overview above for details and the example below for a specific use case.
 - To add a **Logical link**, click **Add New Link**.
 - To add an additional Server Group, click **Add New Server Group** and enter/select details.
 - To clone or delete a Logical Link or Server Group within a Rack Type, click the **Clone** icon or **Delete** icon (top-right of section).

6.3.2.1 Creating Rack Type - Example

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 L2 servers.

1. From the list view (Design > Rack Types) click **Create Rack Type**, enter a name (**MyFirstRackType** in this example), then select **L2** connectivity type.
2. In the **Leafs** section, enter a name (**MyLeaf1** 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 shows the first leaf.

Create Rack Type ✕

Summary

Name *
MyFirstRackType

Description

Connectivity Type *
☒ L2 ☐ L3

Configuration

Leafs Access Switches Servers

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

☐ External facing?

+ Add new leaf

Preview

Topology Logical Devices

MyLeaf1_1

- Click **Add New Leaf** and enter a name for the second leaf (**MyLeaf2** in this example), select **AOS-48x10+6x100-1** from the **Leaf Logical Device** drop-down list, then change the **Links per spine** to **2**. Notice the **Topology** preview on the right side now shows both leaves.
- Click **Servers**, click **Add new server group** and enter a name (**MyServerGroup1** for this example), change the **Server 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.

Create Rack Type ✕

Configuration

Leafs Access Switches Servers

Server

Name *
MyServerGroup1

Server count *
20

Port Channel ID Min Max
0 0

Logical Device *
AOS-2x10-1

+ Add link

+ Add new server group

Preview

Topology Logical Devices

MyLeaf1_1 MyLeaf2_1

| | | | |
|-------------------|-------------------|-------------------|-------------------|
| MyServerGroup1_1 | MyServerGroup1_2 | MyServerGroup1_3 | MyServerGroup1_4 |
| MyServerGroup1_5 | MyServerGroup1_6 | MyServerGroup1_7 | MyServerGroup1_8 |
| MyServerGroup1_9 | MyServerGroup1_10 | MyServerGroup1_11 | MyServerGroup1_12 |
| MyServerGroup1_13 | MyServerGroup1_14 | MyServerGroup1_15 | MyServerGroup1_16 |
| MyServerGroup1_17 | MyServerGroup1_18 | MyServerGroup1_19 | MyServerGroup1_20 |

- Click **Add 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**.

6. Click **Add new server group**, and enter a name (**MyServerGroup2** for this example), change the **Server Count** to **20**, then select **AOS-2x10-1** from the **Logical Device** drop-down list.
7. Click **Add 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**.
8. 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

Leaves

Access Switches

Servers

Server

Name *

MyServerGroup1

Server count *

20

Port Channel ID Min

0

Max

0

Logical Device *

AOS-2x10-1

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

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

MyServerGroup1

20 servers

2 x 10 Gbps

MyLogicalLink1 single-homed at MyLeaf1

LAG Mode: LACP (Active)

2 x 10 Gbps

MyLogicalLink2 single-homed at MyLeaf2

LAG Mode: LACP (Active)

AOS-2x10-1

AOS-2x10-1

9. Click **Create** to create the rack type in the global catalog and return to the list view. The details view is shown below:

6.3. Rack Types

595

Summary

| | |
|-------------------|-----------------|
| Display Name | MyFirstRackType |
| Connectivity Type | L2 |
| IP Version | IPv4 |


Leafs

⌵ ⌶

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



Servers


⌵ ⌶

MyServerGroup1 20 servers

2 x 10 Gbps

MyLogicalLink1 single-homed at MyLeaf1
LAG Mode: LACP (Active)

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




MyServerGroup2 20 servers

2 x 10 Gbps

MyLogicalLink2 single-homed at MyLeaf2
LAG Mode: LACP (Active)

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



6.3.3 Editing Rack Type

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

6.3.3.1 Editing Rack types in Templates

Changes to rack types in the global catalog do not affect rack types that have already been embedded into templates. If the intent is for a template to use a modified rack type, then after editing the rack type it must be *updated in the template*.

6.3.3.2 Editing Rack types in Blueprints

Rack types that are being used in a running blueprint, can be edited (as of version 3.2.0). See [Editing Rack](#) in the staged blueprint section for more information.

6.3.4 Deleting Rack Type

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

6.4 Templates

6.4.1 Template Overview

Templates combine powerful logic to build networks using policy intent and AOS components such as rack types and logical devices. AOS includes two types of templates: rack-based and pod-based.

Rack-based templates import one or more rack types that define how servers connect to top-of-rack switches (or a pair of ToR switches); logical devices define the spines.

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

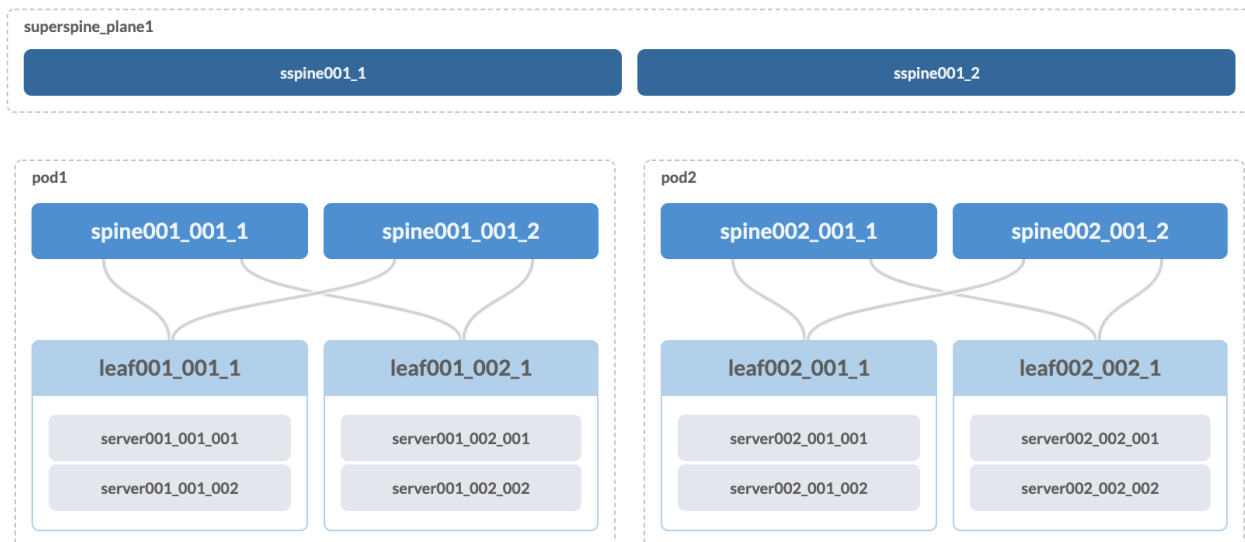


Fig. 1: Single plane, dual superspine

After a template is created, it can be used to create as many different blueprints as needed to build specific networks.

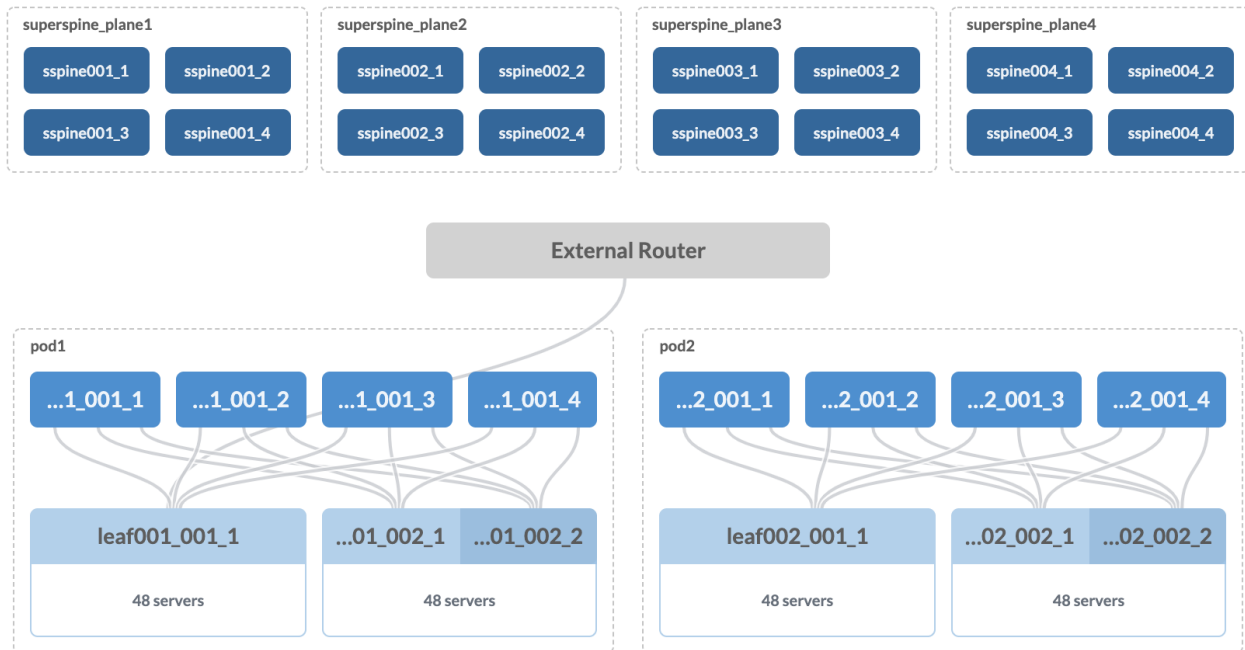


Fig. 2: 4 x plane, 4 x superspine

Query: All Click to search

1-21 of 21 « « 1 » »

Page Size: 25

| Name | Type | Overlay Control Protocol | Actions |
|-----------------|------------|--------------------------|-------------------|
| L2 Pod | RACK BASED | Static VXLAN | Edit Clone Delete |
| L2 Pod External | RACK BASED | Static VXLAN | Edit Clone Delete |
| L2 Pod Mlag | RACK BASED | Static VXLAN | Edit Clone Delete |
| L2 Pod Single | RACK BASED | Static VXLAN | Edit Clone Delete |

To go to templates - from the AOS web interface, navigate to **Design / Templates**. AOS ships with numerous predefined templates. To see more information, click a template name. Templates include the following details:

Common Parameters Name - 64 characters or fewer

Type - RACK BASED or POD BASED

Policies

ASN Allocation Scheme (spine) (rack-based only) Unique - applies to 3-stage designs. A different ASN is assigned to each spine.

Single - applies to 5-stage designs. One ASN is assigned to all spines within a pod, and another ASN is assigned to all superspines.

Routing Policy (import) (rack-based only) Default Only - accepts 0.0.0.0/0 BGP route

ALL - accepts all routes - It sends an **Internet full table** (700k routes), which may cause a crash or other

undesirable behavior for the AOS fabric network devices that are attached to external routers. Verify that your network devices can accept the appropriate number of routes.

Overlay Control Protocol - defines the inter-rack virtual network overlay protocol used in the AOS fabric. This cannot be changed after a blueprint is deployed.

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. VXLAN/EVPN capabilities for inter-rack virtual networks are dependent on the make and model of the network devices being used. See [Virtual Networks](#) for more information.

Spine to Leaf links Type (or for 5-stage, Spine to Superspine Links) IPv4 - uses addresses from IPv4 resource pools.

IPv6 - uses addresses from IPv6 resource pools. IPv6 is not supported when MP-EBGP EVPN overlay control protocol is specified.

IPv4-IPv6 - dual-stack

Structure

For Rack-based Templates: Rack Types - type of rack and number of each selected rack type. Specifying an ESI-based rack type in a rack-based templates without EVPN is invalid.

Spine Logical Device and Count - type and number of spine logical devices

External Links Count and Speed - number of spine links and speed to any external routers

Superspine Link Count and Speed - number and speed of links to any superspines

For Pod-based Templates: Pods - type of rack-based template and number of each selected template

Superspine Logical Device - type of logical device

Plane - number of planes and number of superspines per plane

External Links Count and Speed - number of links and speed to any external routers

Cloning Template

Instead of entering all details for a new template, you can clone an existing one, give it a new name and customize it.

6.4.2 Creating Rack Based Template

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

1. If your design requires *rack types* and/or *logical devices* that are not in the global catalog, create them before proceeding.
2. From the list view (Design / Templates), click **Create Template**, enter a name (64 characters or fewer) and select **RACK BASED**.
3. Select applicable policies.
4. Select a *rack type* from the drop-down list and select how many of that type to include in the template. Notice that as you enter information, the topology preview on the right changes accordingly.

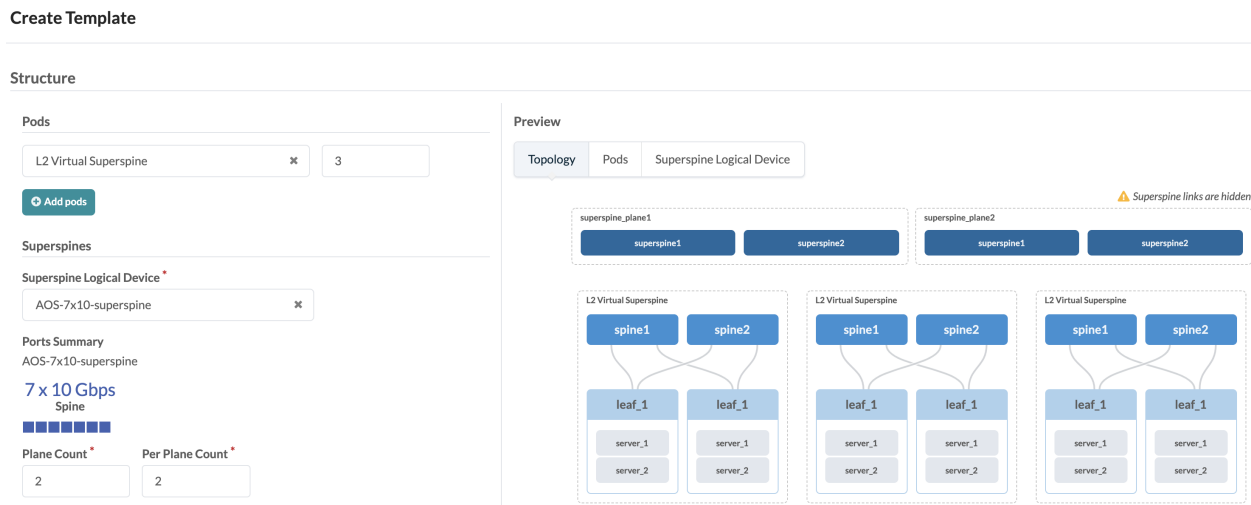
- To add another rack, click **Add racks**.
5. Select the **Spine Logical Device** from the drop-down list, then select the number of them to include in the template. Make sure to select one that provides a sufficient number of spine ports for your design. For 5-stage designs, make sure to select a logical device that includes the **Superspine** role.
 6. For **Spine External Connectivity**, enter the number and connection speed of any external links.
 7. For 5-stage designs, enter the number and connection speed of links for **Superspine Connectivity**.
 8. Click **Create** to create the template. It can now be used to [create a blueprint](#).

6.4.3 Creating Pod Based Template

A pod-based template consists of multiple rack-based templates; it is essentially a “template of templates” used to build [5-stage Clos networks](#).

1. If your design requires [templates](#), [rack types](#) and/or [logical devices](#) that are not in the global catalog, create them before proceeding.
2. From the list view (Design / Templates), click **Create Template**, enter a name (64 characters or fewer) and select **POD BASED**.
3. Select applicable policies.
4. 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.
5. Select a **Superspine Logical Device** from the drop-down list.
6. Select the number of planes and the number of superspines per plane.
7. Enter the number and connection speed of any external links.
8. Click **Create** to create the template. It can now be used to [create a blueprint](#).

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



6.4.4 Editing Template

Changes to templates in the global catalog do not affect templates that have already been used to create blueprints, thereby preventing potentially unintended changes to those blueprints.

1. Either from the list view (Design / Templates) or the details view, click the **Edit** button (top-right) for the template to update.
2. Make your changes.
 - If you've modified a rack type that's used in a rack type template and you want it be updated in the template, you need to delete the rack type in the template (click **X** to the right of the template), then immediately select the same rack type from the drop-down list before saving the changes.
1. Click **Update** (bottom-right) to update the template.

6.4.4.1 Updating Rack Type in Rack Type Template

Changes to rack types in the global catalog do not affect templates that have already imported them, 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. Either from the list view (Design / Templates) or the details view, click the **Edit** button (top-right) for the template to update.
2. Click the **X** to the right of the rack type to remove it. Don't save the template yet.
3. Select the same rack type from the drop-down list.
4. Click **Update** (bottom-right) to update the template with the modified rack type.

6.4.5 Deleting Template

Do not delete a template if it is referenced by a blueprint.

1. Either from the list view (Design / Templates) or the details view, click the **Delete** button for the template to delete.
2. Click **Delete** to delete the template.

6.5 Configlets

6.5.1 Configlet Overview

What about multi-line banners?

Using configlets to configure multi-line banners (such as banner motd) on Cisco NX-OS and Arista EOS is problematic because of an extra non-ASCII character that cannot be entered.

We recommend that, before you install the AOS agent on the device, you configure multi-line banners with ZTP (Arista Zero Touch Provisioning) or Cisco POAP (Power-on Auto Provisioning). The banner configuration will become part of the device's pristine configuration, and it will persist throughout the AOS configuration.

Another option is to *manually* configure multi-line banners on the device. This method causes a *configuration deviation* anomaly that you can clear by accepting the new configuration as the *Golden Configuration*. For more information, see [Configuration Deviation](#).

Configlets are configuration templates that are applied to network devices. They augment the AOS reference design with non-native device configuration. Circumstances when configlets might be employed include syslog, SNMP access policy, TACACS / RADIUS, management ACLs, control plane policing, NTP and username / passwords. Configlets are powerful, but, if used improperly, they do pose risks to deployment stability and feature interactions with the AOS reference design.

Configlets must *augment* existing configuration, and not attempt to replace it. While it *is* technically possible to modify AOS-rendered configuration, we strongly recommend that you refrain from modifying any configuration that affects routing or connectivity. If configlets are used to change interface configuration, the *intended* AOS interface configuration can be inadvertently overwritten. For example, if you add a configlet to create a network span port, you must make sure that the configlet is properly applied to an **Unused** port, or it might override one that is already in use.

Configlets are created and added to the global catalog, then they are imported into the catalogs of existing blueprints and assigned to one or more devices. They can be assigned to spines and/or leafs, and to specific devices. Devices must be deployed for configlets to be rendered. If you want to change a configlet that has been imported into a blueprint catalog, you must import it again into the blueprint after it's been updated in the global catalog.

Additional characteristics of configlets include:

- Configlets can contain syntax for different vendor NOS types.
- Configlets can use key-values stored in a *property set* to parameterize configuration.
- Passwords and other secret keys are not encrypted in configlets.
- You can use the same configlets across the entire enterprise, but creating and applying regionally-specific *property sets* is recommended instead.
- Avoid using shortened versions of commands. AOS may validate the exact commands, then return them as-is in the rendered configuration after pushing changes.

Warning: Using configlets to add non-native configuration is not always appropriate, or possible, depending on the application. Use care with configlets to prevent deployment issues.

6.5.1.1 Rendering Order

| Configuration Rendering order Cisco NXOS | | Configuration Rendering order Arista EOS | | Configuration Rendering order Cumulus Linux | | |
|---|--|--|--|--|--|-------------------------|
| system_top negation_template_text | User-defined configlets (Before reference design) | system_top negation_template_text | User-defined configlets (Before reference design) | Hostname | | |
| system_top template_text | | system_top template_text | | | | |
| Feature Enablement - bgp, lldp, bash, nxapi, vpc, etc - Enablement of vxlan and evpn features | AOS Reference Design (No configlets here) | System configuration - Spanning-tree - Hostname - Breakout | AOS Reference Design (No configlets here) | AOS Reference Design (No configlets here) | | |
| System level config - Spanning-tree - Hostname - Breakout modules | | Security Zones (VRFs) | | | | |
| Security Zones (VRFs) | | Interfaces - L2, L3, Port-channels - DHCP Relay - Additional breakout modules | | | | |
| Vlans - L2 vian and SVI | | Routing - Loopback - BGP - Static routes - L2/L3 VNI to RD mapping - EVPN | | | | |
| Mlag - Cisco VPC, VPC related vlans and SVIs | | VLANs - L2 vian and SVI - DHCP Relay | | | | |
| Interfaces - Port-channels, L2, L3 interfaces - DHCP Relay - Additional breakout modules | | MLAG - Arista MLAG, related vlans, and svi - Port-channels, L2, L3 interfaces | | | | |
| Routing - BGP - Route-maps, prefix-lists, etc - Static routes - EVPN - L2/L3 VNI to RD mapping | | VXLAN - VXLAN/L2 & L3 VNI mappings | | | | |
| Vxlan - VNIs and VTEPs - VLAN to VNI mapping - EVPN | | Feature negation | | | | |
| Feature negation | | Interface configlet negation_template_text | | | | User-Defined configlets |
| Interface configlet negation_template_text | | SVI Configlet negation negation_template_text | | | | |
| SVI Configlet negation negation_template_text | | System configlet negation negation_template_text | | | | |
| System configlet negation negation_template_text | System configlet template_text | | | | | |
| System configlet template_text | Interface configlet template_text | | | | | |
| Interface configlet template_text | SVI configlet template_text | | | | | |
| SVI configlet template_text | | | | | | |

| | |
|-------------------------|--|
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |
| User-Defined configlets | |

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

6.5.1.2 Configlet Details

Configlets include the following details:

Config Style (NOS) Cumulus Linux (applicable sections: SYSTEM, INTERFACE, FILE, OSPF, FRR)

Cisco NX-OS (applicable sections: SYSTEM, INTERFACE, SYSTEM TOP, OSPF)

Arista EOS (applicable sections: SYSTEM, INTERFACE, SYSTEM TOP, OSPF)

Juniper Junos (applicable section: SYSTEM)

SONiC (applicable sections: SYSTEM, FILE, OSPF, FRR)

Section

System Top (NXOS, EOS) Configuration is rendered *before* the AOS reference design is applied. System top is ideal for ensuring that AOS can overwrite a setting to implement the programmed intent. If you “turn off” a needed feature, AOS will reenale it when the reference design is applied.

System (All NOSs)

Configuration is rendered *after* the AOS reference design is applied and *after* any file-type configlets are applied. System-type configlets allow you to run commands on a Linux device as the root user. This location offers absolute insertion of the configlet. These changes could **potentially break the functionality of the reference design**.

It's also used in conjunction with file-type configlets to restart processes or perform administrative tasks after file-type operations have completed. System-type configlets can nest other configuration.

On Cumulus, the `net commit` command is required.

System type configlets require negation equivalents, so when a configlet is unassigned from the node, the configuration is removed from the device. For example, if the template text is `username example privilege 15 secret 0 MyPassword`, the negation template text might be `no username example`.

Note: For NX-OS and EOS, AOS ensures that the appropriate **configure terminal** context is applied. This does not need to be part of the configlet.

Warning: Template text and negation template text are not validated by AOS. They are issued directly to the devices. If you improperly configure a configlet, AOS does not raise warnings or restrictions. To ensure that a configlet performs exactly as intended, test configlet templates and negation templates on a separate dedicated device.

Warning: Use system-type configlets with great caution. Contents are written to the file as user root. Improper use can take down a network.

Interface (Cumulus, NXOS, EOS) Configuration is rendered *after* any system-type configlets are applied. Interface-type configlets apply config within the context of an interface. Only physical interfaces are supported. The actual interfaces that the configlet is to be applied to is not specified within the configlet itself. The configlet scope is specified when *importing the configlet into a blueprint*.

For Cumulus on AOS version 3.2, `net clu` (NCLU) syntax are no longer used. The config line must match `/etc/network/interfaces` syntax exactly.

For Cumulus on AOS version 3.1 and earlier, do not use `net commit`. (In interface context this is handled by AOS.) Also, omit `net add` and specific interface designation (e.g. `net add interface swp1`) from both the template text and negation template text, as AOS adds it automatically.

File (Cumulus, SONiC) File-type configlets add configuration to Cumulus devices that cannot be applied with Cumulus NCLU. The text file is added to the system and the contents of the targeted file are replaced with the template text *after* the AOS reference design is applied and *before* any system-type or interface-type configlets are applied. See [example](#) below.

File-type configlets enable you to replace text in an existing file located under `/etc` *only* (because of the Docker container host mount of AOS). File-type configlets do not support negation.

Important:

- The entire contents of the file must be present within the configlet.
- **All** contents of the file are overwritten.

- There is no versioning or storing of the original file contents.
- A file that has been overwritten cannot be restored to its original contents.

Warning: Use file-type configlets with great caution. Contents are written to the file as user root. Improper use can take down a network.

File-type configlets should *never* be used on configuration files of critical processes (e.g. `/etc/frr/frr.conf` or `/etc/network/interfaces/`).

Tip: If you need to write to a file outside of `/etc` (`/usr` for example) build the file-type configlet, then use a system-type configlet to move the file afterward.

OSPF (Cumulus, NX-OS, EOS, SONiC) OSPF configlets are designed for controlling **SPF Timers** and **LSA Pacing and throttling timers**. They apply config within the context of the OSPF process. Configuration is rendered *after* any system-type configlets are applied. Configuration is specified in **section_condition** to define the scope (i.e. which VRF it is applied to). See [Staging Configlets](#). Under **section_condition**, VRF name must be checked. If no VRFs are checked, the OSPF configlet applies to all VRFs.

On Cumulus, OSPF-type configlets are added directly into the OSPF section. Negation is not supported.

On NX-OS and EOS, the configuration is rendered under **router ospf**.

FRR (Cumulus as of AOS version 3.2.0, SONiC as of AOS version 3.3.0a) FRR configlets gives the ability to append FRR configuration to the configuration file (`/etc/frr/frr.conf`) generated by AOS. The configlet content is added at the *bottom* of the file, so the user-defined configlet becomes part of FRR “intent” and the configlet configuration is *incrementally included* in `frr-reload`.

Important: AOS cannot validate template text. It is the user’s responsibility to ensure that the FRR configuration in the configlet is valid. Errors in the FRR configlet are likely to cause deployment errors, unintended configuration, and device impact.

Warning: Use configlet types **File** and **System** with great caution. Contents are written to the file as user root. Improper use can take down a network.

Template Text CLI commands to add custom configuration to a device. Contents are not validated by AOS.

Negation Template Text (as applicable) CLI commands to disable functionality of the configlet. Contents are not validated by AOS.

Filename (Cumulus, SONiC as of AOS version 3.3.0a) For file-type configlets

From the AOS web interface, navigate to **Design / Configlets**.

1. Click to search

3. Create Configlet

Table View Card View

| Name | Generators | Actions |
|------------------------------------|------------------|-------------------|
| NXOS L2-Hardware TCAM Carving - HW | NXOS: SYSTEM TOP | Edit Clone Delete |
| Cumulus Hostname | Cumulus: FILE | Edit Clone Delete |
| configlet_test copy | EOS: SYSTEM | Edit Clone Delete |
| NXOS Hardware TCAM Carving - HW | NXOS: SYSTEM TOP | Edit Clone Delete |
| NXOS Software TCAM Carving | NXOS: SYSTEM TOP | Edit Clone Delete |
| configlet_test | EOS: SYSTEM | Edit Clone Delete |
| NXOS HTTP API | NXOS: SYSTEM | Edit Clone Delete |
| Cumulus LLDP Hostname override | Cumulus: SYSTEM | Edit Clone Delete |

Click for details

Cloning Configlets

Instead of entering all details for a new configlet, you can clone an existing one, give it a new name and customize it.

6.5.2 Creating Configlet

1. From the AOS web interface, navigate to **Design / Configlets**, then click **Create Configlet**.
2. Enter a configlet name (64 characters or fewer).
3. Select the *config style* (Cumulus, NXOS, EOS, Junos, SONiC). If you are creating a configlet on Cumulus, please see additional information in the **Configlets on Cumulus** section below.
4. Select the *section* where the configlet is to be rendered (SYSTEM TOP, SYSTEM, INTERFACE, FILE, OSPF, FRR). The available choices depend on the selected config style.
5. In the **Template Text** field, enter the CLI commands for the custom configuration.

Important: Using a raw text editor (OSX TextEdit, Windows Notepad++) is critical. The inclusion/addition of hidden characters cause unforeseen issues when deploying configlets.

6. If **Negation Template Text** is required, enter the CLI commands to remove the configuration.
7. For file-type configlets, in the **Filename** field, enter the filename.
8. If you want to add another style, click **Add a style**, then add applicable details.
9. Click **Create** to create the configlet in the global catalog. It is now available for *importing into the catalog of a blueprint*.

Tip: Multiple styles are useful in a mixed vendor environment. Create one configlet with a single-purpose that contains a style for each vendor.

6.5.2.1 Configlets on Cumulus

Configlets on Cumulus Linux have specifics to be aware of.

For AOS Version 3.2

What about Time Voyager?

When using Time Voyager you roll back to a previous AOS commit. As such, it is not to be confused with an UNDO function, as things deleted on the last commit are re-applied when rolling back.

- On system-type configlets, `net commit` is required in both **Template Text** and **Negation Template Text**.
- On interface-type configlets, `net clu` (NCLU) syntax is no longer used. The config line must match `/etc/network/interfaces` syntax exactly.
- On file-type configlets, all file contents are overwritten. Removing the configlet will not restore its original content.
- On OSPF-type configlets, configuration will be directly added into the OSPF section.
- On FRR-type configlets, all content of the configlet is appended to the file `/etc/frr/frr.conf`.

For AOS Versions 3.1 and Earlier

- On system-type configlets, **net commit** is required in both **Template Text** and **Negation Template Text**.
- On interface-type configlets, **net commit** should *not* be used. AOS will ensure this is correctly applied.
- On interface-type configlets, the **net add** and specific interface designation (e.g. `net add interface swp1`) should be omitted from both the **Template Text** and **Negation Template Text**. This is added by AOS automatically.
- On file-type configlets, *all* contents of the file are overwritten. Removing the configlet will not restore its original contents.

See *Cumulus examples* below.

6.5.2.2 Configlets and Property Sets

Instead of hard-coding data into a configlet, you can refer to a *property set* (key-value pairs). The example below refers to a property set named NTP SERVER.

Generators *

Config Style *

☒ Cumulus ☐ NXOS ☐ EOS ☐ Junos ☐ SONiC

Section *

☒ SYSTEM ☐ INTERFACE ☐ FILE ☐ OSPF ☐ FRR

Template Text *

```
net add time ntp server {{NTP_SERVER}} iburst
net commit
```

Negation Template Text

```
net del time ntp server {{NTP_SERVER}}
net commit
```

6.5.2.3 Cumulus Linux Configlet Examples

Cumulus Configlet Example: NTP in Management VRF

NTP in Cumulus by default runs within the default VRF. For certain scenarios, NTP is required to run in the mgmt VRF. It requires stopping the services in the default VRF and starting services in the mgmt VRF. The following system-level configlet shows how NTP configuration can be applied on the mgmt VRF on Cumulus devices :

- Config Style - **Cumulus**
- Section - **SYSTEM**
- Template Text

```
systemctl stop ntp.service
systemctl disable ntp.service
systemctl daemon-reload
systemctl start ntp@mgmt.service
systemctl enable ntp@mgmt.service
```

- Negation Template Text

```
systemctl stop ntp@mgmt.service
systemctl disable ntp@mgmt.service
systemctl daemon-reload
systemctl start ntp.service
systemctl enable ntp.service
```

Generators *

Config Style *

☒ Cumulus ☐ NXOS ☐ EOS ☐ Junos ☐ SONiC

Section *

☒ SYSTEM ☐ INTERFACE ☐ FILE ☐ OSPF ☐ FRR

Template Text *

```
systemctl stop ntp.service
systemctl disable ntp.service
systemctl daemon-reload
systemctl start ntp@mgmt.service
systemctl enable ntp@mgmt.service
```

Negation Template Text

```
systemctl stop ntp@mgmt.service
systemctl disable ntp@mgmt.service
systemctl daemon-reload
systemctl start ntp.service
```

Cumulus Configlet Example: SNMP

The following system-level configlet shows how to apply SNMP server configuration on Cumulus Linux using NCLU. Note the `net commit` command:

- Config Style - **Cumulus**
- Section - **SYSTEM**
- Template Text

```
net add snmp-server listening-address 172.20.40.10
net add snmp-server readonly-community MyCommunity access
10.0.0.1
net commit
```

- Negation Template Text

```
net del snmp-server listening-address 172.20.40.10
net del snmp-server readonly-community MyCommunity access
10.0.0.1
net commit
```

Generators *

Config Style *

☒ Cumulus ☐ NXOS ☐ EOS ☐ Junos ☐ SONiC

Section *

☒ SYSTEM ☐ INTERFACE ☐ FILE ☐ OSPF ☐ FRR

Template Text *

```
net add snmp-server listening-address 172.20.40.10
net add snmp-server readonly-community MyCommunity access
10.0.0.1
net commit
```

Negation Template Text

```
net del snmp-server listening-address 172.20.40.10
net del snmp-server readonly-community MyCommunity access
10.0.0.1
net commit
```

Cumulus Configlet Example: OSPF

The following OSPF configlet shows how to modify SPF and LSA timers on Cumulus Linux. Note configuration will be directly added into the OSPF section.

- Config Style - **Cumulus**
- Section - **OSPF**
- Template Text

```
timers throttle spf {{ospf_spf_delay_msec}}
{{ospf_spf_hold_msec}} {{ospf_spf_max_msec}}
```


Generators *

Config Style *

☒ Cumulus ☐ NXOS ☐ EOS ☐ Junos ☐ SONiC

Section *

☐ SYSTEM ☐ INTERFACE ☐ FILE ☒ OSPF ☐ FRR

Template Text *

```
timers throttle spf {{ospf_spf_delay_msec}}  
{{ospf_spf_hold_msec}} {{ospf_spf_max_msec}}
```

Cumulus Configlet Example: FRR

The following FRR configlet shows how to add a static route to VRF SZ_ART_DR.

- Config Style - **Cumulus**
- Section - **FRR**
- Template Text

```
vrf SZ_ART_DR  
ip route 10.90.0.0/16 10.136.118.24
```

Generators ***Config Style** *

☒ Cumulus ☐ NXOS ☐ EOS ☐ Junos ☐ SONiC

Section *

☐ SYSTEM ☐ INTERFACE ☐ FILE ☐ OSPF ☒ FRR

Template Text *

```
vrf SZ_ART_DR
ip route 10.90.0.0/16 10.136.118.24
```

The FRR configlet will be shown at the bottom of the routing section when viewing Rendered Config for the device.

I2_virtual_ext_001_leaf1 Rendered Config Preview

```

368 " neighbor l3rtr timers connect 5",
369 " neighbor l3rtr next-hop-self",
370 " neighbor l3rtr soft-reconfiguration inbound",
371 " address-family l2vpn evpn",
372 " rd 172.16.0.2:2",
373 " route-target import 10000:1",
374 " route-target export 10000:1",
375 " advertise ipv4 unicast",
376 " exit-address-family",
377 " address-family ipv4 unicast",
378 " redistribute connected route-map AllPodNetworks",
379 " no neighbor l3rtr send-community extended",
380 " exit-address-family",
381 "",
382 "!",
383 "vrf SZ_ART_DR",
384 " ip route 10.90.0.0/16 10.136.118.24",
385 ""
386 ]
387 },
388 {
389   "command": "systemctl reset-failed frr.service"
390 },
391 {
392   "command": "/usr/sbin/aos_reset_frr_service"
393 }
394 ], "dhcp": [
395 /

```

Cumulus Configlet Example: Syslog

A file-based configlet adds a text file to the system and replaces the contents of the targeted file with template text from the configlet. As an example it can be of great use when configuring a syslog server with management VRF enabled. To add this configuration to Cumulus devices the file written can require double quotation marks as shown below:

```

cumulus@switch:~$ cat /etc/rsyslog.d/11-remotesyslog.conf
## Copy all messages to the remote syslog server at 192.168.0.254 port 514
action(type="omfwd" Target="192.168.0.254" Device="mgmt" Port="514"
Protocol="udp")

```

To setup such a file-based configlet, precede each double quotation mark in the configuration with three backslashes. Three backslashes are needed because double quotes must be escaped and backslashes need escaping too.

The following example shows how a file-type configlet with double quotation marks should be applied with three backslashes. These are required to escape double-quote.

- Config Style - Cumulus

- Section - **FILE**
- Template Text

```
*. * @192.168.0.253:514 #UDP
*. * @192.168.0.254:514 #UDP
action(type=\\\\"omfwd\\\\" Target=\\\\"192.168.0.254\\\\"
Device=\\\\"mgmt\\\\" Port=\\\\"514\\\\" Protocol=\\\\"udp\\\\"")
```

- Filename

```
/etc/rsyslog.d/11-remotesyslog.conf
```

Generators ^{*}

Config Style ^{*}

☒ Cumulus ☐ NXOS ☐ EOS ☐ Junos ☐ SONiC

Section ^{*}

☐ SYSTEM ☐ INTERFACE ☒ FILE ☐ OSPF ☐ FRR

Template Text ^{*}

```
*. * @192.168.0.253:514 #UDP
*. * @192.168.0.254:514 #UDP
action(type=\\\\"omfwd\\\\" Target=\\\\"192.168.0.254\\\\"
Device=\\\\"mgmt\\\\" Port=\\\\"514\\\\" Protocol=\\\\"udp\\\\"")
```

Filename

```
/etc/rsyslog.d/11-remotesyslog.conf
```

6.5.2.4 Cisco NX-OS Configlet Examples

Cisco NX-OS Configlet Example: Syslog

The following system-level configlet shows how syslog configuration can be applied on NX-OS devices:

- Config Style - **NXOS**
- Section - **SYSTEM**
- Template Text

```
logging server 192.168.0.30
logging facility local3
logging trap warning
```

- Negation Template Text

```
no logging server 192.168.0.30
no logging facility local3
no logging trap warning
```

Generators *

Config Style *

☐ Cumulus
 ☒ NXOS
 ☐ EOS
 ☐ Junos
 ☐ SONiC

Section *

☒ SYSTEM
 ☐ INTERFACE
 ☐ SYSTEM TOP
 ☐ OSPF

Template Text *

```
logging server 192.168.0.30
logging facility local3
logging trap warning
```

Negation Template Text

```
no logging server 192.168.0.30
no logging facility local3
no logging trap warning
```

6.5.2.5 Arista EOS Configlet Examples

EOS Configlet Example: NTP

The following configlet configures NTP servers on Arista EOS devices. The configlet uses property sets for the NTP server IP addresses.

- Config Style - **EOS**
- Section - **SYSTEM**

- Template Text

```
ntp server {{NTP_SERVER_1}}  
ntp server {{NTP_SERVER_2}}
```

- Negation Template Text

```
no ntp server {{NTP_SERVER_1}}  
no ntp server {{NTP_SERVER_2}}
```

Generators ^{*}

Config Style ^{*}

☐ Cumulus ☐ NXOS ☒ EOS ☐ Junos ☐ SONiC

Section ^{*}

☒ SYSTEM ☐ INTERFACE ☐ SYSTEM TOP ☐ OSPF

Template Text ^{*}

```
ntp server {{NTP_SERVER_1}}  
ntp server {{NTP_SERVER_2}}
```

Negation Template Text

```
no ntp server {{NTP_SERVER_1}}  
no ntp server {{NTP_SERVER_2}}
```

EOS Configlet Example: Interface Speed

The following configlet applies ‘speed auto’ to an interface. Scope (devices and interfaces) is set when importing the configlet:

- Config Style - **EOS**
- Section - **INTERFACE**
- Template Text

```
speed auto
```

- Negation Template Text

```
no speed auto
```

Generators *

Config Style *

☐ Cumulus
 ☐ NXOS
 ☒ EOS
 ☐ Junos
 ☐ SONiC

Section *

☐ SYSTEM
 ☒ INTERFACE
 ☐ SYSTEM TOP
 ☐ OSPF

Template Text *

```
speed auto
```

Negation Template Text

```
no speed auto
```

EOS Configlet Example: OSPF

The following OSPF configlet shows how to modify **SPF** and **LSA** timers on EOS. Note the configuration will be added underneath router ospf :

- Config Style - **EOS**
- Section - **OSPF**
- Template Text

```
timers throttle spf {{ospf_spf_delay_msec}}
{{ospf_spf_hold_msec}} {{ospf_spf_max_msec}}
```

- Negation Template Text

```
no timers throttle spf
```

Generators *

Config Style *

☐ Cumulus ☐ NXOS ☒ EOS ☐ Junos ☐ SONiC

Section *

☐ SYSTEM ☐ INTERFACE ☐ SYSTEM TOP ☒ OSPF

Template Text *

```
timers throttle spf {{ospf_spf_delay_msec}}
{{ospf_spf_hold_msec}} {{ospf_spf_max_msec}}
```

Negation Template Text

```
no timers throttle spf
```

6.5.2.6 Juniper Junos Configlet Examples

Junos Configlet Example: NTP

The following system-level configlet shows how NTP configuration can be applied on Junos devices:

- Config Style - **Junos**
- Section - **SYSTEM**
- Template Text

```
system {
  ntp {
    boot-server 10.1.4.1;
    server 10.1.4.2;
  }
}
```


Generators *

Config Style *

☐ Cumulus
 ☐ NXOS
 ☐ EOS
 ☒ Junos
 ☐ SONiC

Section *

☒ SYSTEM

Template Text *

```

system {
  ntp {
    boot-server 10.1.4.1;
    server 10.1.4.2;
  }
}

```

6.5.2.7 Enterprise SONiC Configlet Examples

SONiC Configlet Example: NTP

The following SONiC system-level configlet uses the SONiC `config` command to set the NTP server for SONiC using the mgmt VRF.

- Config Style - **SONiC**
- Section - **SYSTEM**
- Template Text

```

sonic-db-cli CONFIG_DB hset NTP\\\\\\|global vrf mgmt
config ntp add {{ntp_server}}

```

- Negation Template Text

```

config ntp del {{ntp_server}}

```

Generators *

Config Style *

☐ Cumulus ☐ NXOS ☐ EOS ☐ Junos ☒ SONiC

Section *

☒ SYSTEM ☐ FILE ☐ OSPF ☐ FRR

Template Text *

```
sonic-db-cli CONFIG_DB hset NTP\\\\\\|global vrf mgmt  
config ntp add {{ntp_server}}
```

Negation Template Text

```
config ntp del {{ntp_server}}
```

SONiC Configlet Example: SNMP

The following SONiC system-level configlet uses the SONiC `config` command to set the SNMP snmptrap for SONiC using the mgmt VRF.

- Config Style - **SONiC**
- Section - **SYSTEM**
- Template Text

```
config snmptrap modify 2 {{SNMP_SERVER}} -v mgmt -c mypass
```

- Negation Template Text

```
config snmptrap del 2
```

Generators *

Config Style *

☐ Cumulus ☐ NXOS ☐ EOS ☐ Junos ☒ SONiC

Section *

☒ SYSTEM ☐ FILE ☐ OSPF ☐ FRR

Template Text *

```
config snmptrap modify 2 {{SNMP_SERVER}} -v mgmt -c mypass
```

Negation Template Text

```
config snmptrap del 2
```

SONiC Configlet Example: Syslog

The following SONiC system-level configlet uses the SONiC config command to set the Syslog server for SONiC.

- Config Style - **SONiC**
- Section - **SYSTEM**
- Template Text

```
config syslog add {{syslog_host}}
```

- Negation Template Text

```
config syslog del {{syslog_host}}
```

Config Style *

☐ Cumulus ☐ NXOS ☐ EOS ☐ Junos ☒ SONiC

Section *

☒ SYSTEM ☐ FILE ☐ OSPF ☐ FRR

Template Text *

```
config syslog add {{syslog_host}}
```

Negation Template Text

```
config syslog del {{syslog_host}}
```

SONiC Configlet Example: FRR

The following SONiC FRR configlet shows how to add a static route.

- Config Style - **SONiC**
- Section - **FRR**
- Template Text

```
ip route 4.2.2.2/32 {{static_route_next_hop}}
ip route 4.2.2.3/32 {{static_route_next_hop}}
```

The screenshot shows a web-based configuration interface for SONiC. It has three main sections: 'Config Style', 'Section', and 'Template Text'. In the 'Config Style' section, 'SONiC' is selected with a radio button. In the 'Section' section, 'FRR' is selected with a radio button. The 'Template Text' section contains a text area with the following content: 'ip route 4.2.2.2/32 {{static_route_next_hop}}' and 'ip route 4.2.2.3/32 {{static_route_next_hop}}'. A red 'X' icon is visible in the top right corner of the interface.

SONiC Configlet Example: Using `sonic-cli` Commands

The following SONiC system-level configlet uses the `sonic-cli` command to set the `delay-restore` option for SONiC mclag. It is required to use `sudo -u admin` at the beginning, `\\\\` for spaces in each `sonic-cli` command, and to add `< /dev/console` to the end.

- Config Style - **SONiC**
- Section - **SYSTEM**
- Template Text

```
sudo -u admin sonic-cli -c config -c mclag\\\\ domain\\\\ 1 -c delay-restore\\\\\n↪600 < /dev/console
```

- Negation Template Text

```
sudo -u admin sonic-cli -c config -c mclag\\\\ domain\\\\ 1 -c no\\\\\n↪delay-restore < /dev/console
```

Config Style *

☐ Cumulus
☐ NXOS
☐ EOS
☐ Junos
☒ SONIC

Section *

☒ SYSTEM
☐ FILE
☐ OSPF
☐ FRR

Template Text *

```
sudo -u admin sonic-cli -c config -c mclag\\\\ domain\\\\ 1 -c delay-restore\\\\ 600 < /dev/console
```

Negation Template Text

```
sudo -u admin sonic-cli -c config -c mclag\\\\ domain\\\\ 1 -c no\\\\ delay-restore\\\\ 600 < /dev/console
```

6.5.3 Editing Configlet in Global Catalog

Changes to configlets in the global catalog do not affect configlets that have already been imported into blueprint catalogs, thereby preventing unintended changes to blueprints. If the intent is for an existing blueprint to use a modified configlet, follow the steps in *Editing Configlet Generators in Blueprint Catalog*.

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

6.5.4 Deleting Configlet from Global Catalog

Deleting a configlet removes it from the global catalog. Configlets that were previously imported into a blueprint are not affected.

1. Either from the list view (Design / Configlets) or the details view, click the **Delete** button for the configlet to delete.
2. Click **Delete** to delete the configlet from the global catalog and return to the list view.

6.5.5 Troubleshooting

6.5.5.1 Configlets and Config Deviation

If an improperly-configured configlet causes AOS deployment errors (when the command is rejected by the device), a **service config deployment** failure occurs. In this case, follow the steps below to resolve the anomaly.

1. From the blueprint, navigate to **Staged / Catalog / Configlets** and delete the configlet.
2. Click **Uncommitted** and commit the change. The configuration deviation persists showing an empty expected config. This happens because when a deployment fails, there is no golden config. The golden config is the running config of the device after *successful* deployment of AOS-rendered config. The expected config has no relationship to the AOS-rendered config.
3. Click **Dashboard**, then click **Config Dev.** (in the **Deployment Status** section).
4. Click the node name, then select **Accept Changes** to notify AOS that the failure can be ignored.

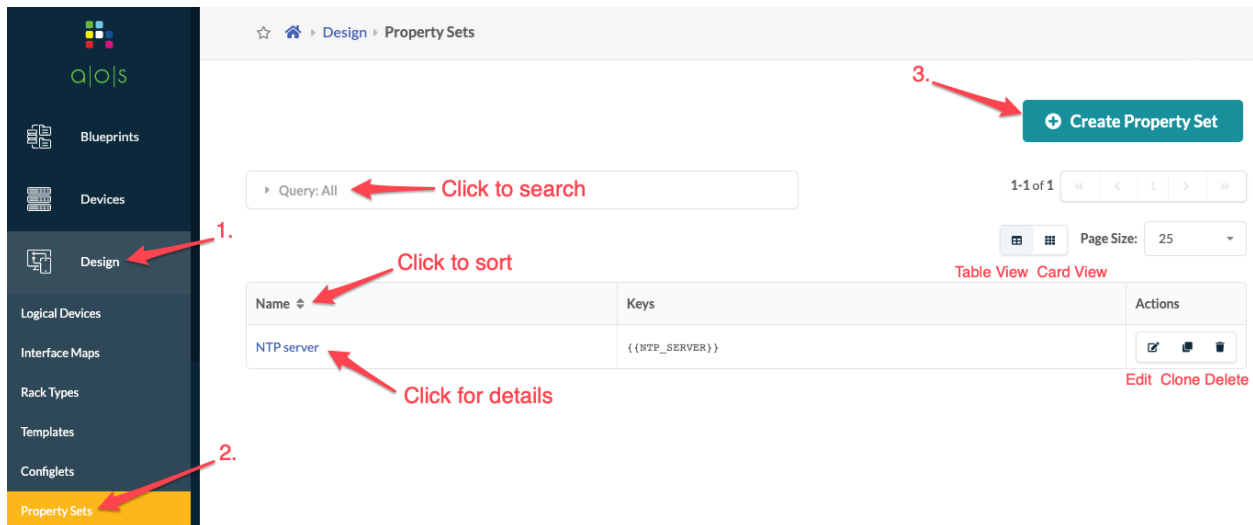
6.6 Property Sets

6.6.1 Property Set Overview

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

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

For *Intent-Based Analytics*, you can parameterize macro level SLAs or individual business units by applying property sets to IBA probe definitions.



To access property sets - from the AOS web interface, navigate to **Design / Property Sets**.

To see details - click a property set name. Property sets include the following details:

Name Identifies the property set

Properties key-value pair(s)

Cloning Property Set

Instead of entering all details for a new property set, you can clone an existing one, give it a unique name and customize it.

6.6.2 Creating Property Set

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

6.6.3 Editing 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 the intent is for a blueprint to use a modified property set, then the property set must be re-imported into the blueprint.

1. From the list view (Design / Property Sets), click the name of the property set to edit.
2. Click the **Edit** button (top-right) .
3. Make your changes.
4. Click **Update** (bottom-right) to update the Property Set.

6.6.4 Deleting Property Set

If a property set is assigned to a *configlet*, it cannot be deleted.

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

6.7 TCP/UDP Port Aliases

6.7.1 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.

To access TCP/UDP ports - from the AOS web interface, navigate to **Design / TCP/UDP Ports**.

The screenshot shows the AOS web interface for managing TCP/UDP Port Aliases. The left sidebar contains a navigation menu with the following items: Blueprints, Devices, Design (selected), Logical Devices, Interface Maps, Rack Types, Templates, Configlets, Property Sets, and TCP/UDP Ports. The main content area is titled 'Design > TCP/UDP Ports'. It features a search bar with the text 'Query: All' and a 'Click to search' annotation. Below the search bar is a table with the following structure:

| Label ↕ | Value | Actions |
|-------------|-------|---|
| NAT gateway | 1025 | Edit Clone Delete |

Annotations on the table include 'Click to sort' pointing to the 'Label' header and 'Table view Card view' pointing to the view toggle buttons. A 'Create Port Alias' button is located in the top right corner, with a '3.' annotation pointing to it. A '1.' annotation points to the 'Design' menu item in the sidebar, and a '2.' annotation points to the 'TCP/UDP Ports' menu item in the sidebar.

Cloning TCP/UDP Port Alias

Instead of entering all details for a new TCP/UDP port alias, you can clone an existing one, give it a unique name, and customize it.

6.7.2 Creating TCP/UDP Port Alias

1. From the list view (Design / TCP/UDP Ports) click **Create Port Alias**, then enter a name and one or more values.
2. Click **Create**. When you add a rule for TCP or UDP protocols to a security policy, the TCP/UDP port alias will appear in the drop-down list.

6.7.3 Editing TCP/UDP Port Alias

1. From the list view (Design / TCP/UDP Ports) click the **Edit** button for the port alias to edit.
2. Make your changes.
3. Click **Update** to update the TCP/UDP port alias and return to the list view.

6.7.4 Deleting TCP/UDP Port Alias

1. From the list view (Design / TCP/UDP Ports) click the **Delete** button for the port alias to delete.
2. Click **Delete** to delete the TCP/UDP port alias from the system.

RESOURCES

7.1 ASN Pools

7.1.1 ASN Pool Overview

You can create resource pools during the design phase or just before you need them during the build phase. When you assign resources to managed devices in your network (blueprint), they are pulled automatically from the pool you specify. In cases where you need to assign a specific network identifier you have the option of assigning a resource individually. To prevent shortages as a network grows, network design best practices recommend defining more resources than are required for the initial design.

Superspines, spines, leafs and Layer 3 servers use BGP to exchange routing information in the underlay. Autonomous system numbers (ASNs) must be assigned to each device, either from the same pool or from different pools. They can also be assigned individually if a specific resource must be assigned to a specific device. ASN pools include the following details:

Pool Name To identify the resource pool.

Total Usage Percentage of ASNs in use for all ranges in the resource pool. (Hover over status bar to see number of ASNs in use and total number of ASNs in the pool.) (New in version 3.3.0)

Range Usage The ASNs included in the range and the percentage that are in use. (Hover over status bar to see number of ASNs in use and total number of ASNs in that range.) (New in version 3.3.0)

Status Indicates if the pool is in use.

Tags (optional) To enable filtering by user-specified categories.

From the web interface, navigate to **Resources > ASN Pools**.

Click to search

Change between table and card view

Click to sort

1. 2. 3.

Create ASN Pool

Query: All

1-3 of 3

Page Size: 25

| Pool Name | Total Usage | Range Usage | Status | Tags | Actions |
|-------------------------------|-------------|-------------|------------|---------|-------------------|
| asn_pool_example | 0.19% | 6.59% | IN USE | default | Edit Clone Delete |
| Private-64512-65534 | 0% | 0% | NOT IN USE | default | Edit Clone Delete |
| Private-4200000000-4294967294 | 0% | 0% | NOT IN USE | default | Edit Clone Delete |

Cloning ASN Pool

Instead of entering all details for a new ASN pool, you can clone an existing one, give it a new name and customize it.

7.1.2 Creating ASN Pool

1. From the web interface, navigate to **Resources > ASN Pools** and click **Create ASN Pool**.
2. Enter a name, (optional) tags, and a range.
 - To specify an additional range, click **Add a range** and enter the range.
3. Click **Create** to create the pool and return to the list view.

When the blueprint is ready, you can assign resources by *updating assignments* in the **Staged > Physical** view.

7.1.3 Editing ASN Pool

If any ASNs in an ASN pool are in use, they cannot be removed from the pool.

1. Either from the list view (Resources > ASN Pools) or the details view, click the **Edit** button for the pool to edit.
2. Make your changes.
 - To add a range, click **Add a range**.
 - To change a range, change the existing range numbers.
 - To remove a range, click the **X** to the right of the range to delete.
3. Click **Update** to update the pool and return to the list view.

7.1.4 Deleting ASN Pool

If any ASNs in an ASN pool are in use, the pool cannot be deleted.

1. Either from the list view (Resources > ASN Pools) or the details view, click the **Delete** button for the pool to delete.

2. Click **Delete** to delete the pool and return to the list view.

7.2 VNI Pools

7.2.1 VNI Pool Overview

You can create resource pools during the design phase or just before you need them during the build phase. When you assign resources to managed devices in your network (blueprint), they are pulled automatically from the pool you specify. In cases where you need to assign a specific network identifier you have the option of assigning a resource individually. To prevent shortages as a network grows, network design best practices recommend defining more resources than are required for the initial design.

Virtual network identifiers (VNIs) are VXLAN network identifiers used in a VXLAN-Aware datacenter fabrics. For more information about VNI usage, see *Create Inter-rack VXLAN/EVPN based Virtual Networks*. VNI pools include the following details:

Pool Name To identify the resource pool.

Total Usage Percentage of VNIs in use for all ranges in the resource pool. (Hover over status bar to see number of VNIs in use and total number of VNIs in the pool.) (New in version 3.3.0)

Range Usage The VNIs included in the range and the percentage that are in use. (Hover over status bar to see number of VNIs in use and total number of VNIs in that range.) (New in version 3.3.0)

Status Indicates if the pool is in use or not.

Tags (optional) To enable filtering by user-specified categories.

From the web interface, navigate to **Resources > VNI Pools**.

The screenshot shows the VNI Pools management interface. On the left is a sidebar with navigation links: Blueprints, Devices, Design, Resources, ASN Pools, and VNI Pools (highlighted). The main content area has a breadcrumb 'Resources > VNI Pools'. At the top right is a 'Create VNI Pool' button. Below it is a search bar with the text 'Query: All'. To the right of the search bar are view toggles (table and card) and a 'Page Size: 25' dropdown. The main table lists VNI Pools with columns: Pool Name, Total Usage, Range Usage, Status, Tags, and Actions. Two rows are visible: 'Default-10000-20000' (status: IN USE) and 'vni_pool_example' (status: NOT IN USE). Red arrows and numbers 1 through 3 point to specific UI elements: 1 points to the VNI Pools link in the sidebar, 2 points to the 'Resources' link in the sidebar, and 3 points to the 'Create VNI Pool' button. Other annotations include 'Click to search' pointing to the search bar, 'Change between table and card view' pointing to the view toggles, and 'Click to sort' pointing to the sort icons in the table headers.

Cloning VNI Pool

Instead of entering all details for a new VNI pool, you can clone an existing one, give it a new name and customize it.

7.2.2 Creating VNI Pool

1. From the web interface, navigate to **Resources > VNI Pools**, then click **Create VNI Pool**.
2. Enter a name, (optional) tags, and a valid range (4096 through 16777214).

- To specify an additional range, click **Add a range**.

3. Click **Create** to create the pool and return to the list view.

When the blueprint is ready, you can assign resources by *updating assignments* in the **Staged > Physical** view.

7.2.3 Editing VNI Pool

If any VNIs in a VNI pool are in use, they cannot be removed from the pool.

1. Either from the list view (Resources > VNI Pools) or the details view, click the **Edit** button corresponding to the pool to edit.
2. Make your changes.
 - To add a range, click **Add a range**.
 - To change a range, change the existing range numbers.
 - To remove a range, click the **X** to the right of the range to delete.
3. Click **Update** to update the pool and return to the list view.

7.2.4 Deleting VNI Pool

If any VNIs in a VNI pool are in use, the pool cannot be deleted.

1. Either from the list view (Resources > VNI Pools) or the details view, click the **Delete** button for the pool to delete.
2. Click **Delete** to delete the pool and return to the list view.

7.3 IP Pools

7.3.1 IP Pool Overview

You can create resource pools during the design phase or just before you need them during the build phase. When you assign resources to managed devices in your network (blueprint), they are pulled automatically from the pool you specify. In cases where you need to assign a specific network identifier you have the option of assigning a resource individually. To prevent shortages as a network grows, network design best practices recommend defining more resources than are required for the initial design.

IP addresses are used in the following situations:

Loopback IPs - Spines - Every spine requires an IP address for the loopback IP. The loopback IP is used as the BGP Router ID. If a spine peers with an external router, the router peers with the spine loopback IP.

Loopback IPs - Leafs - Every leaf requires an IP address for the loopback IP. The loopback IP is used as the BGP router ID. If a leaf peers with an external router, the router peers with the leaf loopback IP.

SVI Subnets - MLAG Domain - A Switch Virtual Interfaces (SVI) subnet for an MLAG domain is used to allocate an IP address between MLAG leaf switches.

Link IPs - Spines <-> Leafs - Link IPs are used between spines and leafs to build the L3-CLOS fabric. These IPs are necessary for BGP peering between spines and leafs, and represent the ‘fabric’ of the network.

Link IPs - To External Router - IP addresses facing the external router are used to statically-route the external router loopback and route across that link.

IP pools include the following details:

Pool Name To identify the resource pool.

Total Usage Percentage of IP addresses in use for all subnets in the resource pool. (Hover over status bar to see number of IP addresses in use and total number of IP addresses in the pool.) (New in version 3.3.0)

Per Subnet Usage The IP addresses included in the subnet and the percentage that are in use. (Hover over status bar to see number of IP addresses in use and total number of IP addresses in that subnet.) (New in version 3.3.0)

Status Indicates if the pool is in use or not.

Tags (optional) To enable filtering by user-specified categories.

From the web interface, navigate to **Resources > IP Pools**.

Click to search

Click to sort

Change between table and card view

1-5 of 5

Page Size: 25

| Pool Name | Total Usage | Per Subnet Usage | Status | Tags | Actions |
|------------------------|-------------|------------------|------------|---------|-------------------|
| ip_pool_example | 0% | 0% | NOT IN USE | | Edit Clone Delete |
| Private-10.0.0.0/8 | 0% | 0% | IN USE | default | |
| Private-172.16.0.0/12 | 0% | 0% | NOT IN USE | default | |
| Private-192.168.0.0/16 | 3.52% | 3.52% | IN USE | default | |

7.3.2 Creating IPv4 Pool

Important: If you configure a switch with an invalid IP block you may receive an error during the deploy phase. The range of IP address blocks is not constrained. It is your responsibility to specify valid IP addresses. For example, specifying the erroneous multicast subnet 224.0.0.0/4 would be accepted, but it would result in an unsuccessful deployment.

Cloning IP Pool

Instead of entering all details for a new IP pool, you can clone an existing one, give it a new name and customize it.

- From the list view (Resources > IP Pools) click **Create IP Pool**, then enter a name and valid subnet.
 - To specify an additional subnet, click **Add a Subnet**.
- Click **Create** to create the pool and return to the list view.

When the blueprint is ready, you can assign resources by *updating assignments* in the **Staged > Physical** view.

7.3.2.1 Warning for Overlapping IP Pools

As of version 3.2.1 if you assign IP pools to a blueprint that already has those IP pools assigned (same range or overlapping range), then a warning is raised in the web interface.

For example, if overlapping subnets are created in an IP pool and then assigned in a blueprint under a security zone for leaf loopback and external router link IP address space, the duplicate IP addresses are identified and an alert is raised.

🏠 > Resources > IP Pools

[+ Create IP Pool](#)

Query: All 1-6 of 6 Page Size: 25

| Pool Name | Subnets | Status | Tags | Actions |
|-----------------|-------------------------------------|---------------------|---------|---|
| Pool-Duplicate | IN USE 172.48.20.24/30 | IN USE | default | Edit Delete |
| Pool_Duplicate1 | IN USE 172.48.20.24/29 | IN USE | default | Edit Delete |

Physical **Virtual** Policies Catalog Settings Tasks

Virtual Networks **Security Zones** Remote EVPN Gateways Virtual Infra Endpoints

[+ Create Security Zone](#)

Query: All 1-5 of 5 Page Size: 25

| VRF Name | Type | VLAN ID | Route Target | VNI ID | 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 |
| orange | EVPN | 210 | 200010:1 | 200010 | DHCP Relay not configured |
| red | EVPN | 200 | 20001:1 | 20001 | 198.51.100.2 fc01:a05:198:51:100::2 |
| yellow | EVPN | 205 | 20005:1 | 20005 | DHCP Relay not configured |

Build

2/2 red: Leaf Loopback IPs

1-1 of 1

Pool Name

☒ Pool-Duplicate

4/4 red: To External Router Link IPs

4/4 red: Virtual Network SVI Subnets

Query: All 1-5 of 5 Page Size: 25

| VRF Name | Type | VLAN ID | Route Target | VNI ID | 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 |
| orange | EVPN | 210 | 200010:1 | 200010 | DHCP Relay not configured |
| red | EVPN | 200 | 20001:1 | 20001 | 198.51.100.2 fc01:a05:198:51:100:2 |
| yellow | EVPN | 205 | 20005:1 | 20005 | DHCP Relay not configured |

Create Security Zone

Build

2/2 red: Leaf Loopback IPs

4/4 red: To External Router Link IPs

1-1 of 1

Pool Name

☒ Pool_Duplicate1

4/4 red: Virtual Network SVI Subnets

Similarly, this feature detects if you re-use IP addresses from an IP pool and manually assign them as loopback for an external router or use them in any other part of the blueprint.

leaf1 Rendered Config Preview

```

33 description Loopback for VRF red
34 ip address 10.0.0.3/32
35 ipv6 address fc01:a05:fab::3/128
36 !
37 interface loopback3
38 vrf forwarding blue
39 description Loopback for VRF blue
40 ip address 10.0.0.5/32
41 ipv6 address fc01:a05:fab::5/128
42 !
43 interface loopback4
44 vrf forwarding yellow
45 description Loopback for VRF yellow
46 ip address 10.0.0.9/32
47 ipv6 address fc01:a05:fab::7/128
48 !
49 interface Ethernet1
50 description facing_spinel:Ethernet1
51 no switchport
52 ip address 172.16.0.1/31
53 no shutdown
54 exit
55 !

```

Edit External Router

Name *

rtr_leaf1_leaf2

IPv4 Address *

10.0.0.9

IPv6 Address

fc01:a05:198:51:100::2

ASN *

65533

Update

Logical Diff Physical Diff Full Nodes Diff Build Errors Warnings

1-2 of 2 Page Size: 25

Warning Message

Node sz:cf208f0a-b3bb-4a0f-93de-fdb283a1c54b,leaf1_loopback: SZ Loopback ipv4 subnet '10.0.0.9/32' overlaps with Loopback '7f982b96-caa6-4eec-89e5-a8db6c96504d' ipv4 subnet '10.0.0.9/32'. 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 7f982b96-caa6-4eec-89e5-a8db6c96504d: Loopback ipv4 subnet '10.0.0.9/32' overlaps with SZ Loopback 'sz:cf208f0a-b3bb-4a0f-93de-fdb283a1c54b,leaf1_loopback' ipv4 subnet '10.0.0.9/32'. 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.

In all such scenarios warning messages about duplicate/overlap IP addresses are raised prompting you to fix them. However, warnings do not prevent you from committing the blueprint without fixing the warnings.

Blueprints > zz-karun-interface_policy.veos.2485377892356-379912618 - interface_policy-veos-virtual > Uncommitted > Warnings

Revert Commit

Dashboard Analytics Staged Uncommitted Active Time Voyager

Logical Diff Physical Diff Full Nodes Diff Build Errors Warnings

1-4 of 4 Page Size: 25

Warning Message

Node sz:2e96dfc3-b0ee-4d75-9fef-2a1d8f7c4168,leaf2_loopback: SZ Loopback ipv4 subnet '172.48.20.25/32' overlaps with Logical Link 'evpn_mlag_001_leaf1<->router0001' ipv4 subnet '172.48.20.24/31'. 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 sz:2e96dfc3-b0ee-4d75-9fef-2a1d8f7c4168,evpn_mlag_001_leaf1<->router0001: Logical Link ipv4 subnet '172.48.20.24/31' overlaps with SZ Loopback 'sz:2e96dfc3-b0ee-4d75-9fef-2a1d8f7c4168,leaf1_loopback' ipv4 subnet '172.48.20.24/32'. 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 sz:2e96dfc3-b0ee-4d75-9fef-2a1d8f7c4168,evpn_mlag_001_leaf1<->router0001: Logical Link ipv4 subnet '172.48.20.24/31' overlaps with SZ Loopback 'sz:2e96dfc3-b0ee-4d75-9fef-2a1d8f7c4168,leaf2_loopback' ipv4 subnet '172.48.20.25/32'. 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 sz:2e96dfc3-b0ee-4d75-9fef-2a1d8f7c4168,leaf1_loopback: SZ Loopback ipv4 subnet '172.48.20.24/32' overlaps with Logical Link 'evpn_mlag_001_leaf1<->router0001' ipv4 subnet '172.48.20.24/31'. 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.

7.3.3 Editing IPv4 Pool

If any IP addresses in a pool are in use, they cannot be removed from the pool.

1. Either from the list view (Resources > IP Pools) or the details view, click the **Edit** button for the pool to edit.
2. Make your changes.
 - To add a subnet, click **Add a subnet**.

- To change a subnet, change the subnet value.
 - To remove a subnet, click the **X** to the right of the subnet to delete.
3. Click **Update** to update the pool and return to the list view.

7.3.4 Deleting IPv4 Pool

If any IP addresses in an IP pool are in use, the pool cannot be deleted.

1. Either from the list view (Resources > IP Pools) or the details view, click the **Delete** button for the pool to delete.
2. Click **Delete** to delete the pool and return to the list view.

7.4 IPv6 Pools

7.4.1 IPv6 Pool Overview

You can create resource pools during the design phase or just before you need them during the build phase. When you assign resources to managed devices in your network (blueprint), they are pulled automatically from the pool you specify. In cases where you need to assign a specific network identifier you have the option of assigning a resource individually. To prevent shortages as a network grows, network design best practices recommend defining more resources than are required for the initial design.

To use IPv6 addressing, IPv6 must be *enabled*. IPv6 is supported on EVPN L2 deployments and L3 deployments. Full feature parity for IPv6 across vendors is not available. Refer to *AOS Feature Matrix* for details. IPv6 pools include the following details:

Pool Name To identify the resource pool.

Total Usage Percentage of IPv6 addresses in use for all subnets in the resource pool. (Hover over status bar to see number of IPv6 addresses in use and total number of IPv6 addresses in the pool.) (New in version 3.3.0)

Per Subnet Usage The IPv6 addresses included in the subnet and the percentage that are in use. (Hover over status bar to see number of IPv6 addresses in use and total number of IPv6 addresses in that subnet.) (New in version 3.3.0)

Status Indicates if the pool is in use or not.

Tags (optional) To enable filtering by user-specified categories.

From the web interface, navigate to **Resources > IPv6 Pools**. A pool is predefined (fc01:a05:fab::/48).

Click to search

Change between table and card view

Click to sort IPv6

1. 2. 3.

Create IPv6 Pool

1-2 of 2

Page Size: 25

| Pool Name | Total Usage | Per Subnet Usage | Status | Tags | Actions |
|-------------------------------|-------------|------------------|------------|---------|-------------------|
| ipv6_pool_example | 0% | 0% | NOT IN USE | | Edit Clone Delete |
| Private- fc01:a05:fab::/48 | 0% | 0% | NOT IN USE | default | Edit Clone Delete |

Cloning IP Pool

Instead of entering all details for a new IP pool, you can clone an existing one, give it a new name and customize it.

7.4.2 Creating IPv6 Pool

1. From the web interface, navigate to **Resources > IPv6 Pools**, then click **Create IPv6 Pool**.
2. Enter a name and valid subnet.
 - To specify an additional subnet, click **Add a Subnet**.
3. Click **Create** to create the pool and return to the list view.

When the blueprint is ready, you can assign resources by *updating assignments* in the **Staged > Virtual** view.

7.4.3 Editing IPv6 Pool

If any IP addresses in a pool are in use, they cannot be removed from the pool.

1. Either from the list view (Resources > IPv6 Pools) or the details view, click the **Edit** button for the pool to edit.
2. Make your changes.
 - To add a subnet, click **Add a subnet**.
 - To change a subnet, change the subnet value.
 - To remove a subnet, click the **X** to the right of the subnet to delete.
3. Click **Update** to update the pool and return to the list view.

7.4.4 Deleting IPv6 Pool

If any IP addresses in an IP pool are in use, the pool cannot be deleted.

1. Either from the list view (Resources > IPv6 Pools) or the details view, click the **Delete** button for the pool to delete.

2. Click **Delete** to delete the pool and return to the list view.

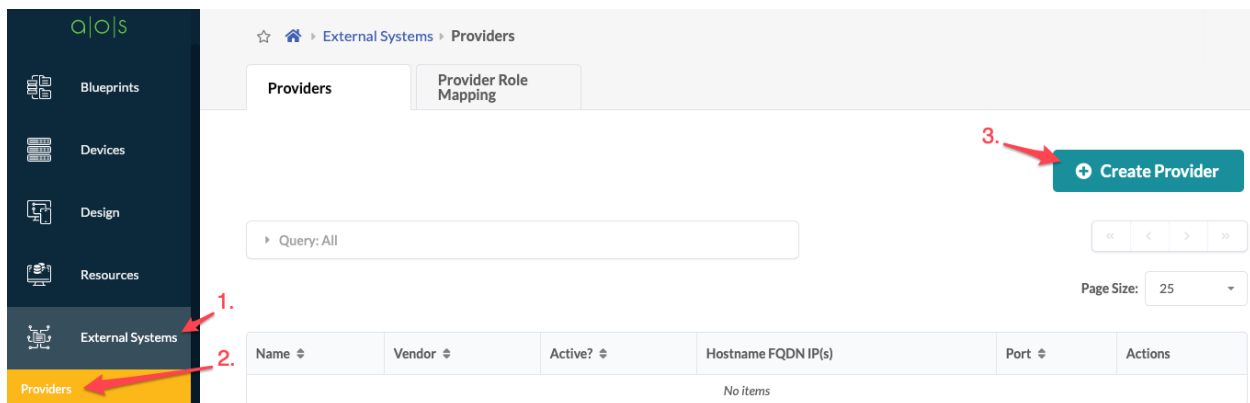
EXTERNAL SYSTEMS

8.1 Providers

8.1.1 Provider Overview

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 authenticate and authorize users with the following protocols: LDAP, Active Directory, TACACS+, and RADIUS. See the **Creating** sections below for more information about individual protocols.

To go to the list view from the web interface, navigate to **External Systems > Providers**.



Cloning Provider

Instead of entering all details for a new provider, you can clone an existing one, give it a new name and customize it.

8.1.2 Creating LDAP Provider

Lightweight Directory Access Protocol (LDAP)

1. From the web interface, 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 AOS server will connect as
- **Password** - The LDAP server user password for the Apstra server to connect as
- **Encryption** - None, SSL/TLS or STARTTLS
- **Advanced Config**
 - **Timeout** (seconds)
 - **Username Attribute Name** - The LDAP attribute from the user entry that Apstra Server uses for authentication. (usually cn or uid)
 - **User Search Attribute Name**
 - **User First Name Attribute Name**
 - **User Last Name Attribute Name**
 - **User Email Attribute Name**
 - **User Object Class Attribute Name**
 - **User Member Attribute Name**
 - **Group Name Attribute Name**
 - **Group DN Attribute Name**
 - **Group Search Attribute Name**
 - **Group Member Attribute Name**
 - **Group Member Mapping Attribute Name**
 - **Group Object Class Attribute Name**

5. You can **Check provider parameters** and **Check login** (to verify authentication with the remote user credentials) before creating the provider.

6. Click **Create** to create the provider and return to the list view.

After configuring and activating a provider, you must *map that provider* to one AOS user role to give access permissions to users with that role. (Only one AOS role can be mapped to a provider group.)

8.1.3 Creating Active Directory Provider

Active Directory (AD) is a database-based system that provides authentication, directory, policy, and other services in a Windows environment.

1. From the web interface, navigate to **External Systems > Providers** and click **Create Provider**.
2. Enter a **Name** (64 characters or fewer), select **AD**, and if you want AD to be the active provider, toggle on **Active?**.
3. For **Connection Settings**, enter/select the following:
 - **Port** - The TCP port used by the server
 - **Hostname FQDN IP(s)** - The fully qualified domain name (FQDN) or IP address of the AD server. For high availability (HA) environments, specify multiple AD servers using the same settings. If the first server cannot be reached, connections to succeeding ones are attempted in order.
4. For **Provider-specific Parameters** enter/select the following, as appropriate:
 - **Groups Search DN** - The AD Distinguished Name (DN) path for the RBAC Groups Organizational Unit (OU)
 - **Users Search DN** - The AD Distinguished Name (DN) path for the RBAC Users Organization Unit (OU)
 - **Bind DN** - The AD Distinguished Name (DN) path for the active server user that the Apstra server will connect as
 - **Password** - The AD server user password for Apstra server to connect as
 - **Encryption** - None, SSL/TLS or STARTTLS
 - **Advanced Config**
 - **Timeout** (seconds)
 - **Username Attribute Name** - The AD attribute from the user entry that the Apstra server uses for authentication. (usually **cn** or **uid**)
 - **User Search Attribute Name**
 - **User First Name Attribute Name**
 - **User Last Name Attribute Name**
 - **User Email Attribute Name**
 - **User Object Class Attribute Name**
 - **User Member Attribute Name**
 - **Group Name Attribute Name**
 - **Group DN Attribute Name**
 - **Group Search Attribute Name**
 - **Group Member Attribute Name**
 - **Group Member Mapping Attribute Name**
 - **Group Object Class Attribute Name**
5. You can **Check provider parameters** and **Check login** (to verify authentication with the remote user credentials) before creating the provider.
6. Click **Create** to create the provider and return to the list view.

After configuring and activating a provider, you must *map that provider* to one AOS user role to give access permissions to users with that role. (Only one AOS role can be mapped to a provider group.)

8.1.4 Creating TACACS+ Provider

Terminal Access Controller Access-Control Systems (TACACS+)

1. From the web interface, navigate to **External Systems > Providers** and click **Create Provider**.
2. Enter a **Name** (64 characters or fewer), select **TACACS+**, and if you want TACACS+ to be the active provider, toggle on **Active?**.
3. For **Connection Settings**, enter/select the following:
 - **Port** - The TCP port used by the server, usually **49**
 - **Hostname FQDN IP(s)** - The fully qualified domain name (FQDN) or IP address of the TACACS+ server. For high availability (HA) environments, specify multiple TACACS+ servers using the same settings. If the first server cannot be reached, connections to succeeding ones are attempted in order.
4. For **Provider-specific Parameters** enter/select the following, as appropriate:
 - **Shared Key** - shared key configured on the server

Caution: Shared key is not displayed when editing a configured TACACS+ provider. If you do not change it, the previously configured shared key is retained. If you test the provider and you have not re-entered the shared key, a null shared key is used for the test and may not work.

- **Auth Mode** - Authentication mode - ASCII (clear-text), PAP (Password Authentication Protocol), or CHAP (Challenge-Handshake Authentication Protocol)
5. You can **Check provider parameters** and **Check login** (to verify authentication with the remote user credentials) before creating the provider.
 6. Click **Create** to create the provider and return to the list view.

8.1.4.1 Configuring 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 **AOS 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 = aos-admins
    }
}
```

(continues on next page)

(continued from previous page)

```
}
}
```

The **aos-admins** group must be mapped to a defined **AOS Role**.

After configuring and activating a provider, you must *map that provider* to one AOS user role to give access permissions to users with that role. (Only one AOS role can be mapped to a provider group.)

8.1.5 Creating RADIUS Provider

Remote Authentication Dial-In User Service (RADIUS). See below for limitations.

1. From the web interface, 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 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 tell AOS which role 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 AOS can map the user to an existing group in AOS the RADIUS server must return the AOS group name as part of the authentication response.

Warning: If the group is unmapped, users cannot log in.

- **Timeout** (seconds) - Defaults to 30 seconds

After configuring and activating a provider, you must *map that provider* to one AOS user role to give access permissions to users with that role. (Only one AOS role can be mapped to a provider group.)

8.1.5.1 RADIUS Limitations

- No support for changing the RADIUS user's password on a remote RADIUS server.
- RADIUS authentication does not control Linux user login via SSH.
- No support for group role-mapping changes.
- Nested groups are not allowed. You must explicitly assign each group to a role.
- When a user logs in, only username and password are required for authenticating against the remote RADIUS server. Log in credentials are not cached. Therefore, when a user logs in, a connection between AOS and the remote RADIUS server is required.

8.1.6 Editing Provider

Important: Any users who are logged into AOS when a setting is changed in an active RBAC provider, are immediately logged out without notification. To continue, the user must log back into the Apstra server. This does not affect users who are defined locally on the Apstra server (for example, **admin**).

1. Either from the list view (External Systems > Providers) or the details view, click the **Edit** button for the provider to edit.
2. Make your changes.
3. Click **Update** (bottom-right) to edit the provider and return to the list view.

8.1.7 Deleting Provider

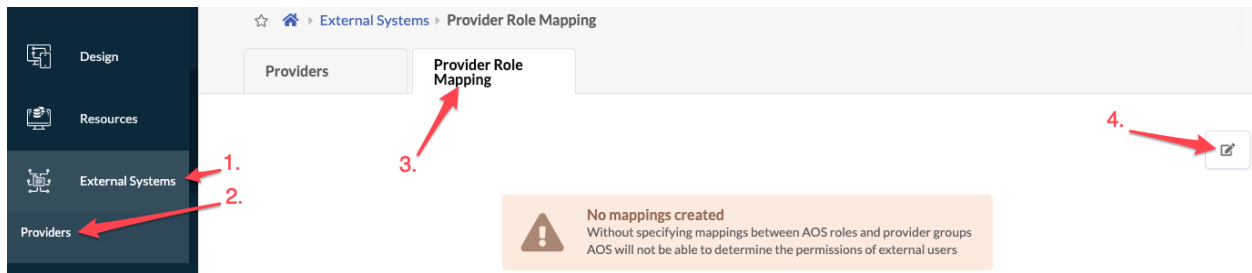
1. Either from the list view (External Systems > Providers) or the details view, click the **Delete** button for the provider to delete.
2. Click **Delete** to delete the provider and return to the list view.

8.1.8 Provider Role Mapping

After configuring an RBAC provider, you must map the provider to one AOS user role to give access permissions to users with that role. You can create, edit and delete provider role mappings. Other details to be aware of include the following:

- Only one provider can be active at a time.
- Only one AOS role can be assigned to a provider group. (As of version 4.0 you will be able to map multiple AOS roles to the same provider group.)
- 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`.

To go to the provider role mapping list view from the web interface, navigate to **External Systems > Providers > Provider Role Mapping**.



8.1.8.1 Creating Role Map

1. From the web interface, navigate to **External Systems > Providers > Provider Role Mapping** and click the **Edit** button (top-right).
2. Click **Add mapping**, select an AOS role from the drop-down list, then enter a provider group. If you map a second AOS role to the same provider group, the first AOS role mapping is discarded and the second AOS role takes its place. The following is an example for mapping the **aos-admins** group that was configured in the TACACS+ configuration above.

Edit Role Mappings

| AOS Role | Provider Group |
|---------------|----------------|
| administrator | aos-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. Click **Update** to create the role map. If the provider that you mapped is the active provider, then users with the mapped role can log in with their usernames and passwords defined in the RBAC server.

8.1.8.2 Editing Role Map

Important: 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 the user **admin**, if **admin** is not logged in locally.

1. From the web interface, 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.

8.1.8.3 Deleting Role Map

1. From the list view (External Systems > Providers > Provider Role Mapping), click the **Edit** button (top-right) and click the **X** next to the mapping to delete.
2. Click **Update** to update the role map.

8.2 External Routers

8.2.1 External Router Overview

External routers represent AOS-managed switches that the network uses for traffic exiting and entering the data center fabric (via BGP). AOS ensures that proper routing is in place by peering leafs or spines with external routers (via ASNs and loopback IPs).

8.2.1.1 Characteristics and Requirements

- External router points are defined per security zone (as opposed to the entire blueprint) (as of AOS version 3.0.0).
- Leaf and external router point connectivity can be via an L2 or L3 interface.
- L2 interfaces can be in a bond such as MLAG or vPC.
- Specify custom import prefix-lists per security zone or per external connectivity point (ECP) (for different external routers).
- Static Routing and IS-IS for external router connectivity is not directly supported.

8.2.1.2 BGP Specific Requirements

- IPv6 peering to external routers is supported (as of AOS version 3.0.0).
- AOS does not support Bidirectional Forwarding Detection (BFD) for external routers (since not all vendors support BFD).
- AOS BGP timers are 1 second for keepalive interval, 3 seconds for hold time, and 5 seconds for connect time.

- The external router role must be assigned to a device (specified in a logical device) to enable it to be deployed as an external router.
- Choose between BGP loopback (eBGP multi-hop) or BGP interface peering.

8.2.1.3 OSPF Specific Requirements

- OSPF is supported on Cisco NX-OS, Arista EOS, and Cumulus Linux devices.
- IPv6 peering to external routers is not supported for OSPF (as of version 3.2.0).
- Bidirectional Forwarding Detection (BFD) for external routers with OSPF is supported. It is disabled by default.
- The default values for Cisco NX-OS, Arista EOS, and Cumulus Linux OSPF Hello/Dead timers can be customized (when assigning external connectivity points in the security zone).
- The external router role must be assigned to a device (specified in a logical device) to enable it to be deployed as an external router.

Note:

- Border leafs are always OSPF ASBRs and always part of a non-backbone area.
 - The external routers are OSPF ABRs and part of the backbone area.
 - OSPF v2 must be supported (OSPFv3 is not supported)
 - OSPF External Router connections are not yet supported for Junos devices.
-

8.2.1.4 Policies

External router connections have one or more external connectivity points (ECPs). ECPs can be fine-tuned with complex custom routing policies to control traffic entering and exiting a pod or security zone. ECPs are defined per L3 Group, rather than the entire blueprint.

You can disable the export of spine-leaf links, L3 edge server links, L2 edge subnets and loopbacks, based on the security zone or per ECP (for different external routers).

Routing Policy Default only (0.0.0.0/0) - at least one default routing policy is expected for each external router in a blueprint (for telemetry)

All (accept all incoming routes)

Overlay Control Protocol (specified in the template)

Static VXLAN External routers can peer with spine or leaf devices configured with external connections.

MP-EBGP EVPN External routers must peer with leaf devices configured with external connections.

8.2.1.5 External Router Config - BGP Example

Below is an example of BGP configuration for an external router that generates a default route facing AOS. It has the following characteristics:

- The ASN of the external router is 100.
- The loopback IP of the external router is 9.0.0.1.
- The external router is connected to two spine switches.

- The ASN of Spine1 is 65416.
- The loopback IP of Spine1 is 172.20.0.4.
- The ASN of Spine2 is 65417.
- The loopback IP of Spine2 is 172.20.0.6.
- L3 fabric link facing spine1 is 10.0.0.1/31.
- L3 fabric link facing spine2 is 10.0.0.3/31.

The prefix-list *PREPEND-FABRIC-PREFIX* prepends the external router's ASN multiple times, which eliminates the fabric preferring the external router for internal fabric prefixes. This is just one resolution and is not the required method.

Listing 1: Linux Quagga External Router Example

```
router bgp 100
  bgp router-id 9.0.0.1
  neighbor 172.20.0.4 remote-as 65416
  neighbor 172.20.0.4 ebgp-multihop 2
  neighbor 172.20.0.4 timers 1 3
  neighbor 172.20.0.4 timers connect 5
  neighbor 172.20.0.4 default-originate
  neighbor 172.20.0.4 soft-reconfiguration inbound
  neighbor 172.20.0.4 update-source lo:1
  neighbor 172.20.0.4 route-map PREPEND-FABRIC-PREFIX out

  neighbor 172.20.0.6 remote-as 65417
  neighbor 172.20.0.6 ebgp-multihop 2
  neighbor 172.20.0.6 timers 1 3
  neighbor 172.20.0.6 timers connect 5
  neighbor 172.20.0.6 default-originate
  neighbor 172.20.0.6 soft-reconfiguration inbound
  neighbor 172.20.0.6 update-source lo:1
  neighbor 172.20.0.6 route-map PREPEND-FABRIC-PREFIX out
!
ip route 172.20.0.4/32 10.0.0.0
ip route 172.20.0.5/32 10.0.0.2
!
route-map PREPEND-FABRIC-PREFIX permit 10
  set as-path prepend 100 100 100 100
```

8.2.1.6 External Router Config - OSPF Example

Below is an example of OSPF configuration for an external router that generates a default route facing AOS. It has the following characteristics:

- The OSPF area of the external router is 51.
- The loopback IP of the external router is 9.0.0.1.
- The external router is connected to two border leaf switches.
- AOS border leaf is an ASBR.
- AOS border leaf is NOT part of a backbone area.
- External router is an ABR attached to the backbone area.
- AOS border leaf to be configured with MTU ignore.

- OSPF network type is broadcast.
- OSPF Hello/Dead interval timers are default.
- L3 fabric link IP on border leaf router is 10.61.60.1/24.
- L3 fabric link IP on external router is 10.61.60.254/24.

Listing 2: Linux Quagga External Router Example

```
ip route 10.0.0.2/32 10.60.61.1 bond0.61
ip route 10.0.0.3/32 10.61.61.1 bond1.61
ip route 10.0.0.4/32 10.60.62.1 bond0.62
ip route 10.0.0.5/32 10.61.62.1 bond1.62
ip route 10.0.0.6/32 10.60.60.1 bond0.60
ip route 10.0.0.7/32 10.61.60.1 bond1.60
ipv6 route ::/0 Null0
!
interface bond0.60
 ip ospf area 51
!
interface bond0.62
 ip ospf area 51
!
interface bond1.60
 ip ospf area 51
!
interface bond1.62
 ip ospf area 51
!
router ospf
 redistribute static route-map STATIC_TO_OSPF
!
ip prefix-list DEFAULT_V4 seq 5 permit 0.0.0.0/0
ipv6 prefix-list DEFAULT_V6 seq 5 permit ::/0
!
route-map STATIC_TO_OSPF permit 10
 match ip address prefix-list DEFAULT_V4
!
route-map STATIC_TO_OSPF permit 20
 match ipv6 address prefix-list DEFAULT_V6
```

8.2.1.7 External Router Details

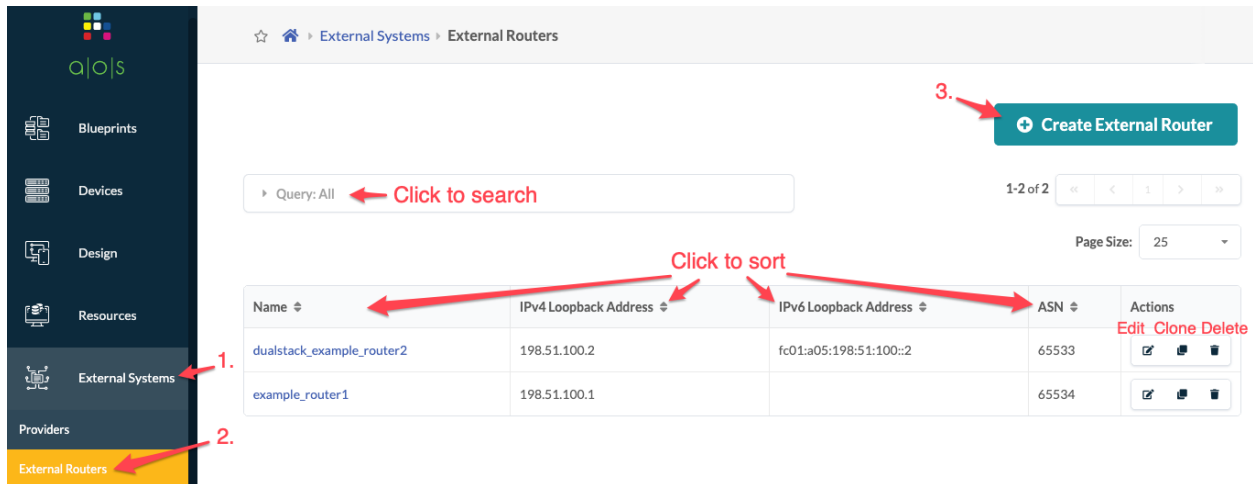
External routers include the following details:

Name 64 characters or fewer

IPv4 Address IPv4 Address of the external router. For BGP loopback peering, AOS uses this for eBGP multi-hop peering.

IPv6 Address (available for AOS version 3.0 and later) Optional IPv6 Address of the external router. For BGP loopback peering, AOS uses this for eBGP multi-hop peering.

ASN AOS automatically generates eBGP peer configuration facing this router Autonomous System Number (ASN).



Cloning External Router

Instead of entering all details for a new external router, you can clone an existing one, give it a new name and customize it.

8.2.2 Creating External Router

1. From the blueprint, navigate to **External Systems / External Routers**, then click **Create External Router**.
2. Enter a name, IPv4 address, IPv6 Address (optional), and ASN.
3. Click **Create** (bottom-right) to create the external router and add it to the global catalog. It is now available for *importing and assigning in a blueprint*.

8.2.3 Editing External Router

Changes to external routers in the global catalog do not affect external routers that have already been used in blueprints, thereby preventing potentially unintended changes to those blueprints. If your intent is for the blueprint to use an updated external router, then you must delete it from the blueprint, re-import it after it has been updated, and re-assign the link to it. See below for the steps for updating an external router.

1. Either from the list view (External Systems / External Routers) or the details view, click the **Edit** button for external router to edit.
2. Make your changes.
3. Click **Update** (bottom-right) to update the external router in the global catalog and return to the list view.

8.2.4 Deleting External Router

1. Either from the list view (External Systems / External Routers) or the details view, click the **Delete** button for the external router to delete.
2. Click **Delete External Router** to delete the external router from the global catalog.

9.1 User Management

Users with the **administrator** role can create, edit, delete and log out locally-defined users, as well as change their passwords.

The screenshot shows the Platform User Management interface. The left sidebar contains a navigation menu with items: Blueprints, Devices, Design, Resources, External Systems, Platform, User Management, and Users. The main content area shows the 'Users' page. At the top right, there is a 'Create User' button. Below it, there is a search bar with the text 'Query: All' and a 'Click to search' annotation. To the right of the search bar, there is a pagination control showing '1-2 of 2' and a 'Page Size: 25' dropdown. Below the search bar, there is a table with columns: Username, Email, Roles, Currently Logged In?, and Actions. The table contains two rows: 'admin' and 'blueprint3-vn'. The 'admin' row has a role of 'administrator' and is 'Yes as local user'. The 'blueprint3-vn' row has roles 'blueprint3-create-vn', 'read-write-resources', and 'blueprint3-make-any-chg', and is 'No'. The 'Actions' column for the 'blueprint3-vn' row contains icons for 'Log Out', 'Edit', 'Clone', and 'Delete'. Annotations include: '1.' pointing to the 'Platform' item in the sidebar, '2.' pointing to the 'Users' item in the sidebar, '3.' pointing to the 'Create User' button, 'Click to sort' pointing to the 'Roles' column header, and 'Use Case' pointing to the 'blueprint3-vn' row.

User profiles include the following details and options:

- Username
- First Name (optional)
- Last Name (optional)
- Email (optional)
- Password
- Roles

Cloning User Profiles

Instead of entering all details for a new user, you can clone an existing one, give it a new name and customize it.

9.1.1 Creating User Profile

1. From the AOS web interface, navigate to **Platform / User Management / Users**, then click **Create User**.
2. Enter a username and password, then select one or more roles. If custom roles have been created, they appear as options along with the predefined roles that ship with AOS. You can see the permissions specified for each of the roles at **Platform / User Management / Roles**.
3. Click **Create** to create the user profile and return to the list view.

9.1.1.1 Use Case 1: Only Create Virtual Networks (not Including Allocating Resources)

To limit a user's role to only creating virtual networks and looking at blueprint details, assign them the role as described in *Role Management: Use Case 3*.

9.1.1.2 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. See *role management use case 3A and 4* for more information.

Create User

Username *

blueprint3-vn

First Name

virtual network management

Last Name

Email

Password *

••••••••••

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
- ☒ blueprint3-make-any-chg

9.1.2 Editing User Profile

1. Either from the list view (Platform / User Management / Users) or the details view, click the **Edit** button for the user profile.
2. Change roles and/or other details.
3. Click **Update** to update the user profile and return to the list view.

9.1.3 Changing User Password

1. From the list view (Platform / User Management / Users) click a username, click the **Change Password** button (top-right), then enter the new password, twice.
2. Click **Change Password** to update the password.

9.1.4 Logging Out User

From the list view (Platform / User Management / Users) click the **Log Out** button for the user.

AOS REST API and User Profiles

In addition to using the AOS web interface to manage user profiles AOS REST API can also be used. Navigate to **Platform / Developers** for documentation and tools. See the **aaa** section for user-related APIs.

9.1.5 Deleting User Profile

User **admin** cannot be deleted.

1. Either from the list view (Platform / User Management / Users) or the details view, click the **Delete** button for the user profile.
2. Click **Delete** to delete the user profile and return to the list view.

9.2 Role Management

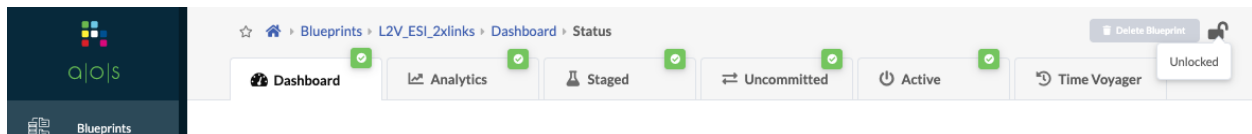
9.2.1 Role Management Overview

Users with the **administrator** role can create, edit and delete user roles (which are assigned to *user profiles*). These roles can also be *mapped* to external groups used by authentication providers such as LDAP, Active Directory, TACACS+, and RADIUS.

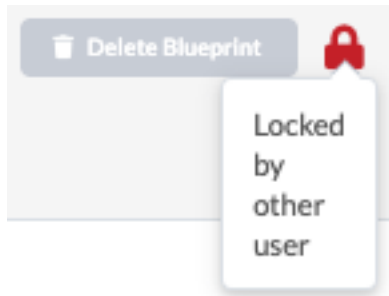
With Enhanced Role Based Access Control introduced in AOS version 3.3, AOS administrators can now create blueprint-specific roles with very specific privileges allowing limited control to associated users. This allows AOS administrators to create more hierarchical roles & protect against accidental changes to the network.

The blueprint locking feature, introduced in AOS version 3.3, prevents restricted users from making changes that effectively are not permitted. In particular, a restricted user should not be able to commit changes made by another user.

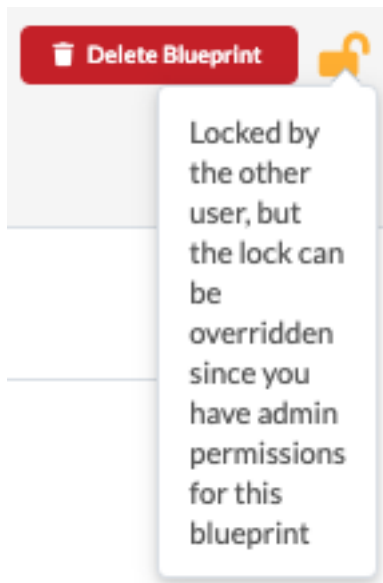
A blueprint with no uncommitted changes is considered “unlocked”.



Let’s say you have a permission to create/update/delete virtual networks, and another user has made uncommitted changes to the blueprint. The blueprint is considered “locked”, and you will not be able to create/update/delete virtual networks until the changes are committed or reverted by the “locking user” who made the uncommitted changes, unless you are the locking user.



An admin user who has “Write/Commit Blueprints” permissions can make any changes to, apply changes for, revert changes for any blueprint.



AOS ships with four predefined user roles (administrator, device_ztp, user, viewer). These user roles cannot be modified. User roles include the following details and options:

- Name
- Type - global permission or per-blueprint permissions
- Global Permissions (read, write, commit, as applicable)
 - Blueprints - blueprints
 - Devices - device profiles, agents, devices
 - Design - configlets, templates, rack types, logical devices, property sets, interface maps
 - Resources - IP pools, external routers, IPv6 pools, ASN pools, VNI pools
 - AAA - sysdb data, AAA providers, roles, audit config, audit events, users
 - Other - streaming, AOS metric logs, ztp, port setting schema, AOS cluster management, virtual infra manager, telemetry service registry, port aliases
- Per-Blueprint Permissions (new in AOS version 3.3.0)
 - Which Blueprints? All or by ID
 - Permissions
 - * Read blueprint

- * Make any changes to staging blueprint (includes managing VNs and their endpoints)
- * Commit changes
- * Manage virtual networks (includes managing VN endpoints)
- * Manage virtual network endpoints

From the AOS web interface, navigate to **Platform / User Management / Roles**.

The screenshot displays the 'Roles' page in the Apstra AOS web interface. The left sidebar shows the navigation menu with 'Platform' (1), 'User Management' (2), and 'Roles' (3) highlighted. The main content area shows a list of roles with their permissions and associated users. Red arrows point to specific roles: 'blueprint1-read-write-commit' (Use Case 1), 'blueprint2-manage-VN-endpoints' (Use Case 2), 'blueprint3-create-vn' (Use Case 3), and 'read-write-resources' (Use Case 3A and 4).

| Role | Permissions | Users |
|--------------------------------|---|---------------|
| administrator | Blueprints: Read/Write/Commit Blueprints Devices: Read/Write Device Profiles, Write/Read Agents, Read/Write Devices Design: Read/Write Configlets, Read/Write Templates, Read/Write Rack Types, Read/Write Logical Devices, Read/Write Property Sets, Read/Write Interface Maps Resources: Read/Write IP Pools, Read/Write External Routers, Write/Read IPv6 Pools, Write/Read ASN Pools, Write/Read VNI Pools AAA: Write/Read Sysdb Data, Write/Read AAA Providers, Write/Read Roles, Read/Write Audit Config, Read Audit Events, Read/Write Users Other: Read/Write Streaming, Read AOS Metric Logs, Write/Read ztp, Read Port Setting Schema, Read/Write AOS Cluster Management, Read Virtual Infra Manager, Read/Write Telemetry Service Registry, Read/Write Port Aliases | admin |
| blueprint1-read-write-commit | blueprint1 Commit changes Read blueprint Make any change to staging blueprint | |
| blueprint2-manage-VN-endpoints | blueprint2 Manage virtual network endpoints | |
| blueprint3-create-vn | blueprint3 Commit changes Read blueprint Manage virtual network endpoints Manage virtual networks | blueprint3-vn |
| blueprint3-make-any-chg | blueprint3 Make any change to staging blueprint | blueprint3-vn |
| device_ztp | Other: Write ztp | |
| read-write-resources | Resources: Read/Write IP Pools, Read/Write External Routers, Write/Read IPv6 Pools, Write/Read ASN Pools, Write/Read VNI Pools | blueprint3-vn |

Cloning User Roles

Instead of entering all details for a new role, you can clone an existing one, give it a new name and customize it.

9.2.2 Creating User Role

1. From the AOS web interface, navigate to **Platform / User Management / Roles**, then 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.

9.2.2.1 Use Case 1: Read, Write and Commit Specified Blueprints

To create a role that gives a user permission to read, write, and commit to specified 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.

Create Role

Name *

blueprint1-read-write-commit

Description

A user with this role can read, write and commit the blueprint named blueprint1.

Type

☐ Global Permissions ☒ Per-Blueprint Permissions

Which Blueprints?

All

☒ OFF

By ID

☐ blueprint2☒ blueprint1☐ blueprint3

Permissions *

Read blueprint

☒ ONMake any change to
staging blueprint☒ ON

Commit changes

☒ ONManage virtual
networks☐ OFFManage virtual
network endpoints☐ OFF

9.2.2.2 Use Case 2: Manage VN Endpoints on Specified Blueprints

To create a role that gives a user permission to only manage virtual network endpoints on specified blueprints, select **Per-Blueprint Permissions**, select one or more blueprint IDs (or **All** for all blueprints), then toggle on **Manage virtual network endpoints**.

Create Role

Name *

blueprint2-manage-VN-endpoints

Description

A user with this role can manage VN endpoints on blueprint2.

Type

☐ Global Permissions ☒ Per-Blueprint Permissions

Which Blueprints?

All

☐ OFF

By ID

☒ blueprint2

☐ blueprint1

☐ blueprint3

Permissions *

Read blueprint

☒ ON

Make any change to
staging blueprint

☐ OFF

Commit changes

☐ OFF

Manage virtual
networks

☐ OFF

Manage virtual
network endpoints

☒ ON

9.2.2.3 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.

Create Role

Name *

blueprint3-create-vn

Description

A user with this role can create virtual networks in blueprint3 only. (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

☐ blueprint2

☐ blueprint1

☒ blueprint3

Permissions *

Read blueprint

☒ ON

 Make any change to
staging blueprint

☐ OFF

Commit changes

☒ ON

 Manage virtual
networks

☒ ON

 Manage virtual
network endpoints

☒ ON

9.2.2.4 Use Case 3A: Create Virtual Networks and Allocate Resources

For the 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**.

9.2.2.5 Use Case 4: Read and Write Resources for 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.

Create Role

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 |
|------------------|--|--|--------|
| Interface Maps | <input type="checkbox"/> OFF | <input type="checkbox"/> OFF | |
| Resources | <input checked="" type="checkbox"/> ON | <input checked="" type="checkbox"/> ON | |
| IP Pools | <input checked="" type="checkbox"/> ON | <input checked="" type="checkbox"/> ON | |
| External Routers | <input checked="" type="checkbox"/> ON | <input checked="" type="checkbox"/> ON | |
| IPv6 Pools | <input checked="" type="checkbox"/> ON | <input checked="" type="checkbox"/> ON | |
| ASN Pools | <input checked="" type="checkbox"/> ON | <input checked="" type="checkbox"/> ON | |
| VNI Pools | <input checked="" type="checkbox"/> ON | <input checked="" type="checkbox"/> ON | |
| Δ Δ Δ | <input type="checkbox"/> OFF | <input type="checkbox"/> OFF | |

9.2.3 Editing User Role

The four predefined user roles (administrator, device_ztp, user, viewer) cannot be modified.

1. Either from the list view (Platform / User Management / Roles) or the details view, click the **Edit** button for the user role.
2. Change permissions, as applicable.
3. Click **Update** to update the role and return to the list view.

AOS REST API and User Roles

In addition to using the AOS web interface to manage user roles (as described below) AOS REST API can also be used. Navigate to **Platform / Developers** for documentation and tools. See the **aaa** section for role-related APIs.

9.2.4 Deleting User Role

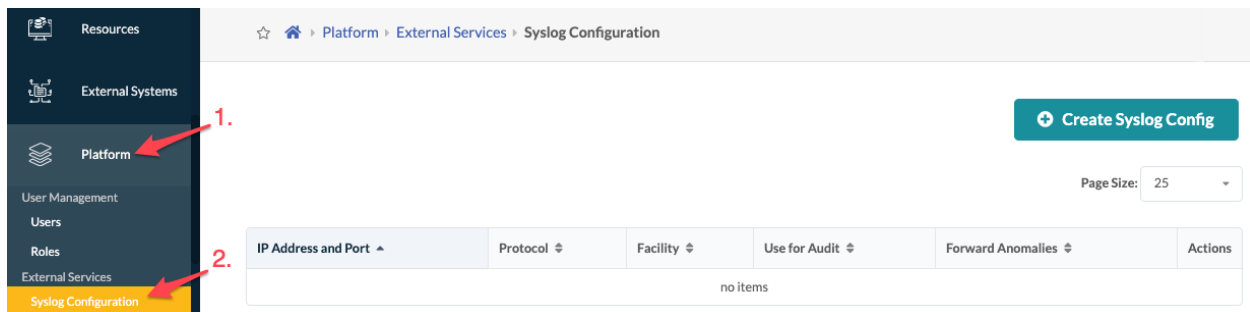
The four predefined user roles (administrator, device_ztp, user, viewer) cannot be deleted.

1. Either from the list view (Platform / User Management / Roles) or the details view, click the **Delete** button for the user role.
2. Click **Delete** to delete the role and return to the list view.

9.3 Syslog Configuration

You can forward audit messages and anomaly messages to one or more external Syslog servers from AOS, then AOS will generate alarms that can be used in an NMS alarming environment.

From the AOS web interface, navigate to **Platform / External Services / Syslog Configuration**.



9.3.1 Configuring Syslog Servers

1. From the AOS web interface, navigate to **Platform / External Services / Syslog Configuration**, then click **Create Syslog Config**.
2. Configure the Syslog server with an **IP Address** or hostname, **Port**, **Protocol** (UDP or TCP), and **Facility** (kern, user, mail, daemon, auth, syslog, lpr, news, uucp, authpriv, ftp, cron, local0, local1, local2, local3, local4, local5, local6, local7).
3. Click **Create** to save the configuration and return to the list view. Messages are not automatically sent to the Syslog server.
4. To configure another Syslog server, repeat the steps above.
5. To enable messages to be sent to the configured server(s), toggle on **Use for Audit** and/or **Forward Anomalies**, as appropriate.

Syslog messages follow Common Event Format (CEF) conventions as shown below:

Listing 1: CEF in Syslog Messages

AOS Log Format:

```
'{timestamp} {host} '
  'CEF:{version}||{device_vendor}||{device_product}||{device_version}||'
  '{device_event_class_id}||{name}||{severity}||{extension}'
```

Where:

```
{version}          : always "0"
{device_vendor}    : always "Apstra"
{device_product}   : always "AOS"
{device_version}   : current AOS version
{device_event_class_id} : "100" for audit logs, "101" for anomaly logs
{name}            : "Audit event" for audit logs, "Alert" for anomaly logs
{severity}         : "medium" for audit logs, "Very-High" for anomaly logs
```

And where {extension} is either :

```
For anomaly logs : msg=<json payload>
For audit logs   : cat=<activity> src=<src_IP> suser=<username> act=<activity_
↪result> cs1Label=<field1_type> cs1=<field1_value> cs2Label=<field2_type> cs2=
↪<field2_value> cs3Label=<field3_type> cs3=<field3_value>
```

Anomaly Log JSON Format

```
u'blueprint_label' : Name of the blueprint the anomaly was raised in.
u'timestamp'       : Unix timestamp when the Anomaly was raised.
u'origin_name'     : Serial Number of the device the anomaly affects.
u'alert'           : The value is a JSON Payload with the actual anomaly (see next_
↪table)
u'origin_hostname' : Hostname of the device the anomaly affects.
u'device_hostname' : Hostname of the device the anomaly affects.
u'origin_role'     : Role of the device the anomaly affects.
u'first_seen'      : Unix timestamp when the Anomaly was raised for the first time.
u'raised'          : Always True.
u'severity'        : The severity level of the anomaly. Always 3.
```

Audit Log Extension Format:

```
cat          : Activity performed. Valid values: "Login", "Logout", "BlueprintCommit",
↪ "DeviceConfigChange", "BlueprintDelete".
src          : Source IP of the client making HTTP requests.
suser        : Who performed the activity.
act          : Outcome of the activity - free-form string. In case of error, include_
↪error string. Ex: Unauthorized
cs1Label     : The string "Blueprint Name"
cs1          : Name of the blueprint on which action was taken.
cs2Label     : The string "Blueprint ID"
cs2          : Id of the blueprint on which action was taken.
cs3Label     : The string "Commit Message". Only exists if user has added a commit_
↪message (optional)
cs3          : Commit Message. Only exists if user has added a commit message_
↪(optional)
deviceExternalId : Id (typically serial number) of the managed device on which_
↪action was taken.
```

(continues on next page)

(continued from previous page)

deviceConfig : Config that is pushed and applied on the device where "#012" is ↵
 ↵used to indicate a line break to log collectors and parsers.

An example of Syslog messages is shown below:

Listing 2: Syslog Message Example

```
Jul 31 03:10:36 aos-server - 2019-07-31T03:10:36.797839+0000 aos-server
CEF:0|Apstra|AOS|3.0.0-151|101|Alert|Very-High|msg={'device_hostname':
'<device hostname unknown>', u'timestamp': 1564542636797839, u'alert':
{u'probe_alert': {u'stage_name': u'Anomaly', u'actual_int': 9711, u'probe_id':
u'46b827c2-be4d-47a5-95f9-1ed0a57224f6', u'key_value_pairs': [{u'value':
u'"Ethernet5"', u'key': u'interface'}, {u'value': u'"2CC260994101"', u'key':
u'system_id'}], u'item_id': u'anomaly,probe\=46b827c2-be4d-47a5-95f9-1ed0a57224f6
,proc\=Anomaly,stage\=out,interface\=Ethernet5,system_id\=2CC260994101',
u'expected_int': 8000}, u'first_seen': 1564542636797795, u'raised': True,
u'severity': 3, u'id': u'4981e646-e665-481b-bada-391689e7ebf3'}, u'origin_name':
u'anomaly,probe\=46b827c2-be4d-47a5-95f9-1ed0a57224f6,proc\=Anomaly,stage\=out,
interface\=Ethernet5,system_id\=2CC260994101'}

Jul 31 03:11:01 aos-server - 2019-07-31T03:11:01.699190+0000 aos-server
CEF:0|Apstra|AOS|3.0.0-151|101|Alert|Very-High|msg={'device_hostname':
'<device hostname unknown>', u'timestamp': 1564542661699190, u'alert':
{u'probe_alert': {u'stage_name': u'Anomaly', u'actual_int': 12890, u'probe_id':
u'46b827c2-be4d-47a5-95f9-1ed0a57224f6', u'key_value_pairs': [{u'value':
u'"Ethernet5"', u'key': u'interface'}, {u'value': u'"2CC260994101"', u'key':
u'system_id'}], u'item_id': u'anomaly,probe\=46b827c2-be4d-47a5-95f9-1ed0a57224f6,
proc\=Anomaly,stage\=out,interface\=Ethernet5,system_id\=2CC260994101',
u'expected_int': 8000}, u'first_seen': 1564542636797795, u'raised': False,
u'severity': 3, u'id': u'4981e646-e665-481b-bada-391689e7ebf3'}, u'origin_name':
u'anomaly,probe\=46b827c2-be4d-47a5-95f9-1ed0a57224f6,proc\=Anomaly,stage\=
out,interface\=Ethernet5,system_id\=2CC260994101'}
```

9.4 Streaming Architecture

AOS can be configured to generate Google Protocol Buffer (protobuf) streams for counter data (perfmon), alerts, and events. 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 customers understand how they can consume the AOS protobuf stream. It is for demonstration purposes only, except for the AOS Telegraf input plugin. This plugin is fully supported by Apstra for customers to use as part of their streaming telemetry solution.

9.4.1 AOSOM-Streaming

The AOS server can be configured to stream perfmon, events and alerts, or any combination thereof. Each data type is sent to a streaming receiver over its own TCP socket. So even if all three data types are configured for the same streaming receiver, there will still be three connections created between the AOS server and the streaming receiver. This also allows for all three types to be sent to three different streaming receivers.

The Aiosom Streaming project provides a packaged solution to collect and visualize telemetry streaming information coming from an AOS Server. This provides a web interface experience and example queries to handle alerts, counters, and AOS events. This open-source project officially lives on Github at <https://github.com/Apstra/aosom-streaming>.

The packaged solution includes:

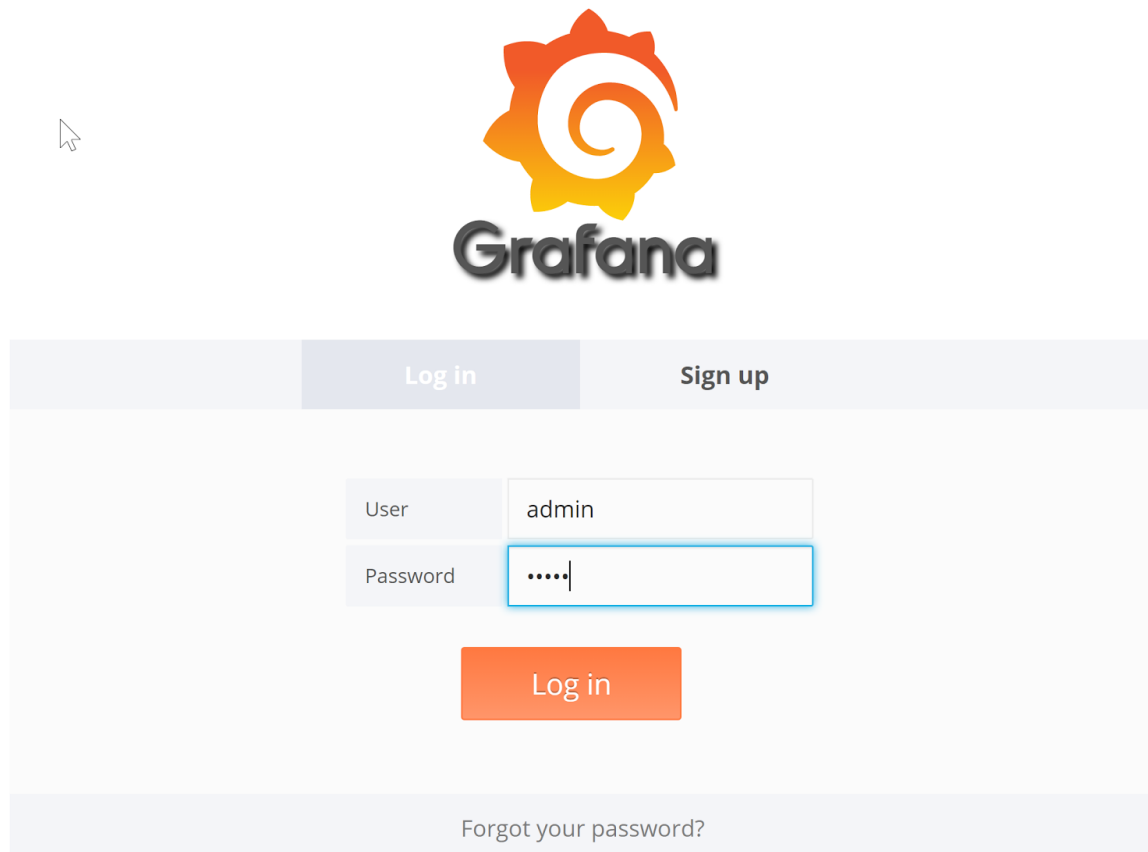
- A graphical Interface based on Grafana (port 3000)
- Prometheus for Counters and Alerts (port 9090)
- Influxdb for Events (port 8086)
- 2 Collectors, one for each database based on Telegraf.

Note: AOSOM streaming is demonstration software, not intended for production environments.

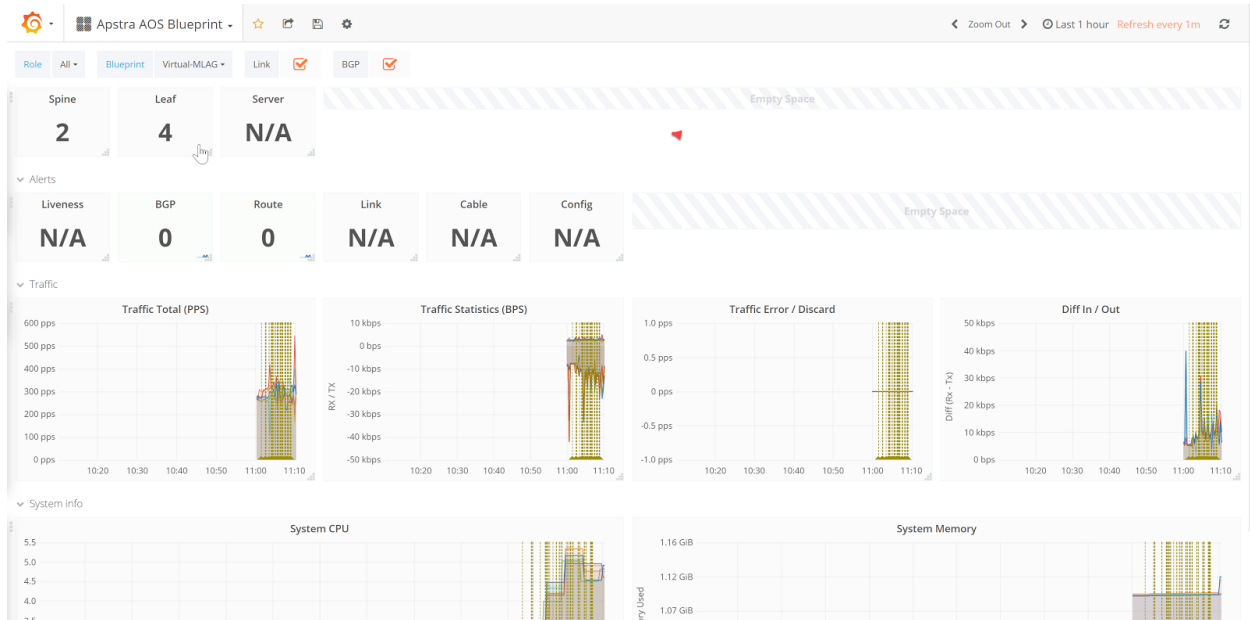
9.4.1.1 Using Aosom-streaming

Grafana Web UI

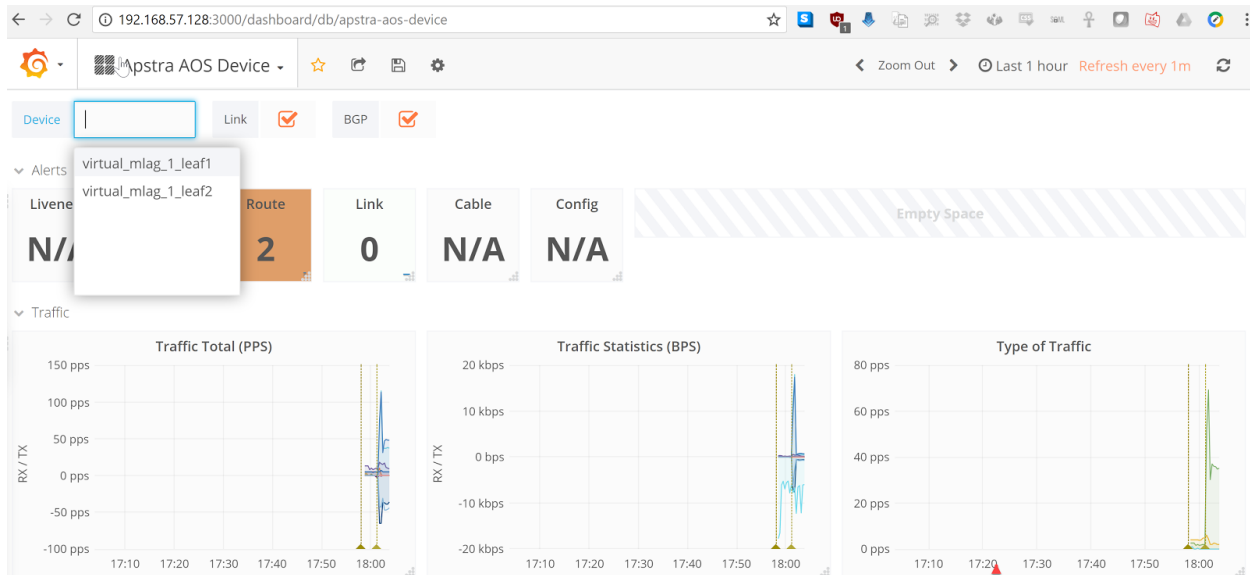
You can browse to the Grafana web UI by visiting **http://<aosom-streaming>:3000** The default username is **admin** and the password is **admin**.



The grafana web UI includes two main sections (top left) - Apstra AOS Blueprint, which describes overall telemetry alerts and traffic throughput, as well as individual devices for interface telemetry. The blueprints will be learned automatically using the AOS 'telegraf' docker container and no further configuration is necessary - it should all be automatic.



In the screenshot above, we can observe traffic in the demo AOS environment, and aggregate CPU, traffic, and errors. Change the dashboard at the to 'Apstra AOS Device' to filter telemetry events based on specific and individual devices. Here we can observe there are 2 active route anomalies in the blueprint, and AOS has received telemetry for two leaf switches.



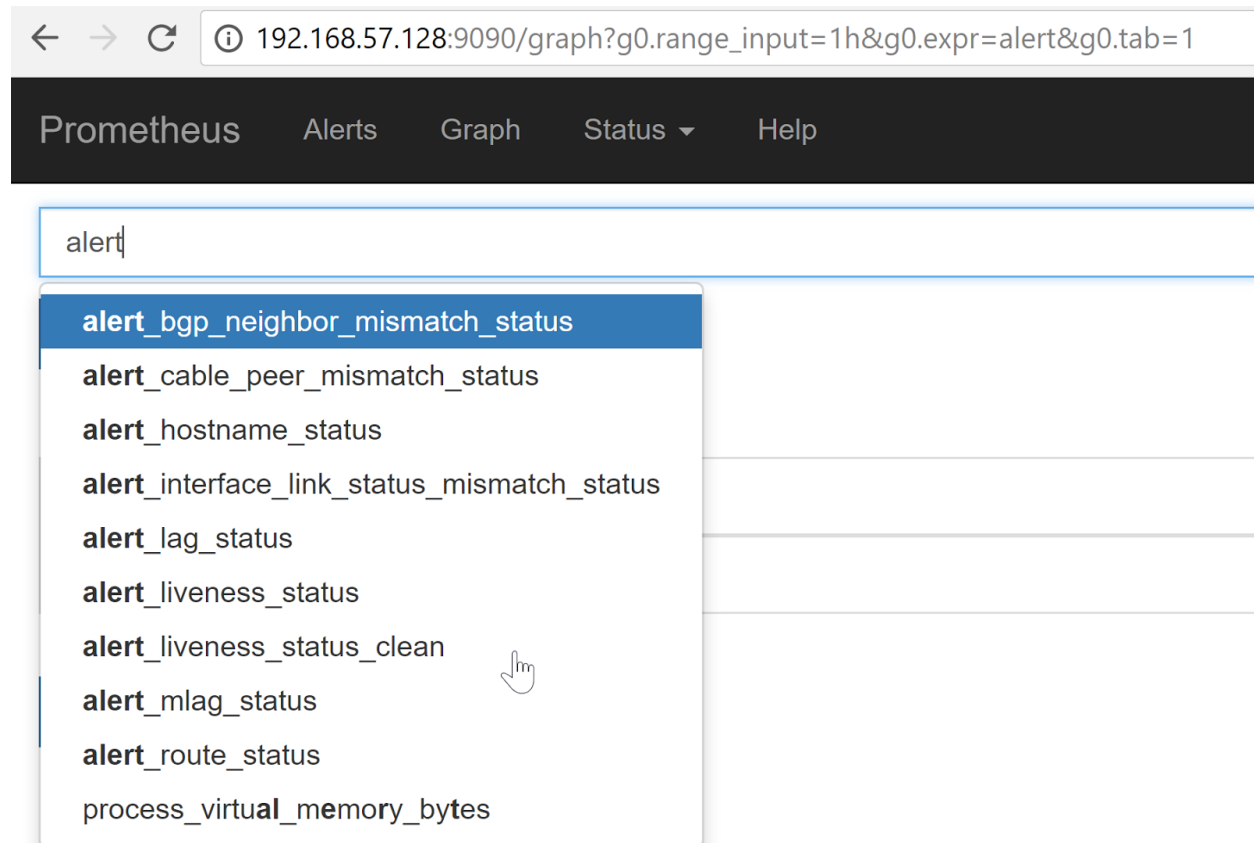
We can scroll down and view device statistics such as CPU and Memory:



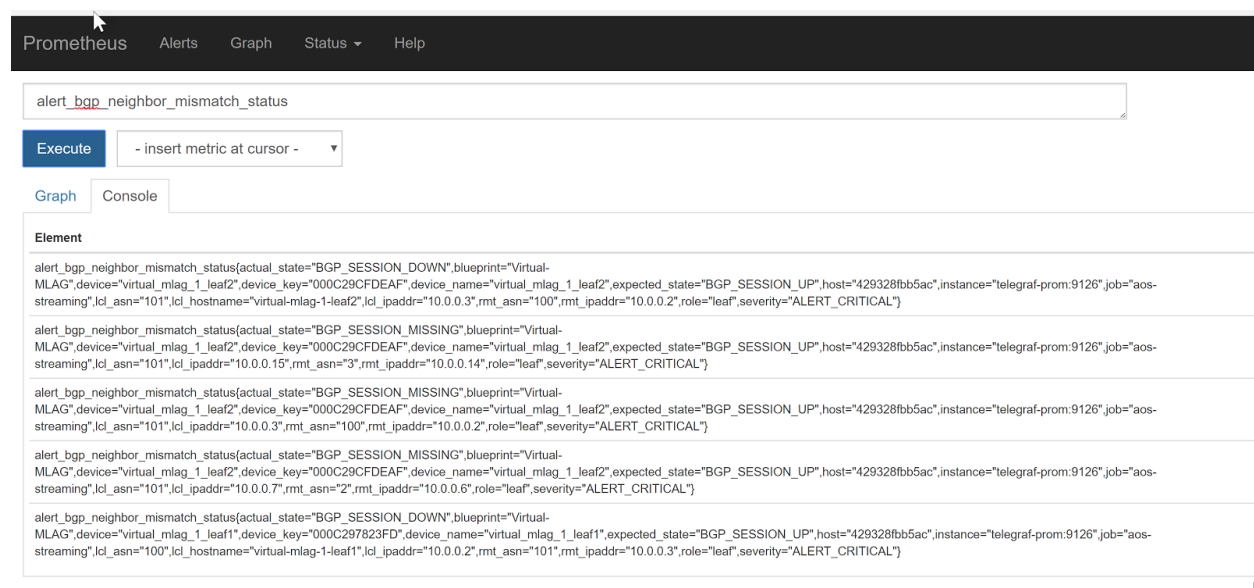
Prometheus Web UI

Prometheus is used for alerts and device telemetry counter storage in the Aosom-streaming appliance. Prometheus is available by browsing to <http://<aosom-streaming>:9090>

Each of the queries are built dynamically by AOS when incoming events appear. Begin typing under 'execute' to see example query names. Starting with 'alert' will tab-complete available alerts that prometheus has received from AOS.



Here is an example of BGP Neighbors being offline.



InfluxDB

InfluxDB is used to store AOS events from telemetry streaming. InfluxDB is available by viewing <http://<aosom-streaming>:8083>

We can show the available influxdb keys with queries, such as ‘show field keys’ or ‘show measurements’.

The screenshot shows the InfluxDB web interface with the query 'show field keys' entered. The results are displayed in three sections, each with a table showing field keys and their types.

| fieldKey | fieldType |
|----------|-----------|
| event | "integer" |

| fieldKey | fieldType |
|----------|-----------|
| event | "integer" |

| 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 with the query 'select * from event_lag_state' entered. The results are displayed in a table with columns: time, blueprint, device, device_key, device_name, event, host, interfacesup, interfacesupcount, lagname, and role.

| time | blueprint | device | device_key | device_name | event | host | interfacesup | interfacesupcount | lagname | role |
|--------------------------------|----------------|------------------------|----------------|------------------------|-------|----------------|--------------|-------------------|-----------------|--------|
| 2017-07-31T23:57:58.524206286Z | "Virtual-MLAG" | "virtual_mlag_1_leaf1" | "000C297823FD" | "virtual_mlag_1_leaf1" | 1 | "0a84241e1366" | "1" | "0" | "port-channel3" | "leaf" |
| 2017-07-31T23:57:58.52422376Z | "Virtual-MLAG" | "virtual_mlag_1_leaf1" | "000C297823FD" | "virtual_mlag_1_leaf1" | 1 | "0a84241e1366" | "1" | "1" | "port-channel3" | "leaf" |
| 2017-07-31T23:57:58.524249294Z | "Virtual-MLAG" | "virtual_mlag_1_leaf1" | "000C297823FD" | "virtual_mlag_1_leaf1" | 1 | "0a84241e1366" | "1" | "2" | "port-channel3" | "leaf" |
| 2017-07-31T23:57:58.524272244Z | "Virtual-MLAG" | "virtual_mlag_1_leaf1" | "000C297823FD" | "virtual_mlag_1_leaf1" | 1 | "0a84241e1366" | "1" | "0" | "port-channel1" | "leaf" |
| 2017-07-31T23:57:58.524299294Z | "Virtual-MLAG" | "virtual_mlag_1_leaf1" | "000C297823FD" | "virtual_mlag_1_leaf1" | 1 | "0a84241e1366" | "1" | "1" | "port-channel1" | "leaf" |

Note: Developing an influx-db application is beyond the scope of this documentation. If you would like to extend capability or develop an application based on telemetry streaming, please let us know and we would be happy to help.

9.4.1.2 Aosom-Streaming configuration

Aosom-Streaming Application setup

Configuring telemetry streaming as part of this project only requires editing `variables.env`, run `make start` file and restarting the containers. No configuration is necessary on the AOS server. Documentation for starting, stopping, and clearing data is available at <https://github.com/Apstra/aosom-streaming>

The telegraf project connects to the AOS API and posts an IP:Port that AOS uses to stream realtime telemetry data back to.

Setting up Aosom-streaming

Copy `variables.default` to `variables.env`:

```
aosom@ubuntu:~/aosom-streaming$ cp variables.default variables.env
```

Modify `variables.env`

`AOS_SERVER` should be the IP address of the AOS server that will send telemetry data to the aosom-streaming server. Username, port and password information can be customized.

“`LOCAL_IP`” should be the IP address assigned to `ens33` (first ethernet interface)

In this case, this is learned via DHCP on this VM. See `ip addr show dev ens33`

GRAFANA configuration options specify the username and password for the grafana web UI.

```
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
```

Set up the project with `make start`, or if 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
```

(continues on next page)

(continued from previous page)

```

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

```

Aosom-Streaming reconfiguration after AOS Server upgrade

Depending on the AOS Server upgrade procedure one or more steps need to be taken in order to ensure a proper streaming connection.

Note: As of AOS 3.2, Apstra has introduced the `sequenced` mode to help detect lost messages on the receiver side. If your previous AOS version was prior to AOS 3.2 or your receiver mode is still `unsequenced` you will need to re-create the receiver endpoint from the AOS Dashboard. Please refer to [Receiver Configuration](#) to learn how to delete a receiver.

AOS Upgrade on the same VM

If the AOS Server has been upgraded using the same VM, and the IP has not changed, then you just need to verify that the current **Telegraf** container image is matching the proper version for this new AOS release.

To verify the current version:

```

admin@aeon-ztps:~$ docker ps
CONTAINER ID IMAGE
4edf204e7be9 apstra/telegraf:latest

```

You can check the different Telegraf versions in the [Apstra Docker Hub](#)

Modify the `docker-compose.yml` file in the server running the streaming container accordingly if required and restart the services:

```
docker-compose up -d
```

Verify that the container is running with the new image by running `docker ps`

Note: Contact Apstra Support if you have any doubt regarding the version to install or if you have any questions about the procedure.

AOS VM to VM upgrade

In this case as the Server IP has changed, you will need to follow the next steps:

1. Modify the `variables.env` to use the new AOS IP, please refer to *Modify variables.env* for details.
2. As in the previous point **AOS Upgrade using the same VM**, check that the **Telegraf** container image is running with the proper version. If required you will need to modify the `docker-compose.yml` file as well to point to the correct docker image. Please check *AOS Upgrade on the same VM* to achieve this part.

9.4.1.3 (Optional) Build Aosom-Streaming VM

These are the simple steps to build your own Aosom-streaming VM - at the end of the day, Aosom-Streaming is only a simple Docker container, and this guide is only setting up a very basic docker server.

Install Ubuntu 16.04.2

Download the Ubuntu 16.04.2 ISO and provision a new VM.

The default username we've chosen is 'aosom' with password 'admin'.

For larger blueprints, Apstra recommends 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 required packages

Based on Ubuntu 16.04.2

```
apt-get update
```

Perform a system update 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 openssl-
↪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-jjsonschema 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-jjsonschema
  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 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 will allow '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.

```

(continues on next page)

(continued from previous page)

```
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, and this is up to your orchestration toolchain. Open `aosom-streaming/docker-compose.yml` and add `restart: always` to each of the service directives. This will ensure docker containers will be 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:
```

(continues on next page)

(continued from previous page)

```
- '4444:4444'
+ restart: always
```

Set up `variables.env` and start container as 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`.

9.4.1.4 Troubleshooting

While most troubleshooting information is included in the Github main page at <https://github.com/Apstra/aosom-streaming>, we can run some simple commands to make sure the environment is healthy

Checking for logs from AOS to Aosom-streaming

Run docker logs aosomstreaming_telegraf-influx_1

We 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
```

Ensuring all containers are running

Run `docker ps` to see and ensure all the expected containers are running:

```
aosom@ubuntu:~/aosom-streaming$ docker ps
CONTAINER ID        IMAGE                      COMMAND                  CREATED             STATUS              PORTS
↪ 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
```

(continues on next page)

(continued from previous page)

```

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

```

9.5 Receiver Configuration

The AOS Telegraf input plugin can be used to receive streaming telemetry from AOS. [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 AOS. The Telegraf platform consists of input and output plugins that customers can choose from for aggregating and storing metrics to different backend databases. The AOS input plugin for Telegraf will deserialize the protobuf stream and create metrics that can then be sent to a particular backend database, e.g., Prometheus, InfluxDB, or Elasticsearch.

9.5.1 Configuring Using Telegraf Plugin

The configuration described here assumes we are using the AOS Telegraf input plugin. Configuring streaming receivers in AOS can be done through the AOS web interface (see below), or handled by the Telegraf plugin by providing it the AOS credentials. A separate AOS account with only the streaming credentials is recommended. If the configuration is done 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 will pull 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. The input and output plugin configurations are shown in the snippet below. The output plugin is configured for the Prometheus client and will listen on port 9126. The input plugin is configured for AOS.

```

# 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

```

(continues on next page)

(continued from previous page)

```
aos_login = "$AOS_LOGIN"  
aos_password = "$AOS_PASSWORD"
```

9.5.1.1 AOS Plugin Configuration

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 AOS to this receiver

The rest of these parameters are only necessary if we want the AOS Telegraf plugin to configure the streaming receivers in AOS via the API.

aos_server - specifies the IP address of the AOS server

aos_port - should always be 443

aos_login - AOS username

aos_password - AOS password

9.5.2 Configuring Using AOS Web Interface

The AOS server can be configured 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, there will be three connections created between the AOS server and the streaming receiver. This also allows for all three types to be sent to three different streaming receivers.

To access receivers - from the AOS web interface, navigate to **Platform / Streaming / Receivers** to see the list view. (In previous AOS versions receivers were called streaming endpoints.)

To sort - click one of the sortable headers. To reverse the sort order, click the header again.

Receivers include the following details:

- Hostname - Hostname
- Port - default: 4444
- Message Type - alerts, events, perfmon
- Sequencing Mode - unsequenced, sequenced

Platform > Streaming > Receivers

Create Receiver

Download messages description Streaming Architecture Receiver Deployment Instructions 1-5 of 5 Page Size: 25

| Hostname and Port | Message Type | Sequencing Mode | Alive | Connection Reset Count | Last Transmitted Message | Last Disconnected | Actions |
|-------------------|--------------|-----------------|-------|------------------------|--------------------------|----------------------|----------|
| 172.20.63.4:4444 | Perfmon | Sequenced | ✓ | 0 | | | [Delete] |
| 172.20.63.4:6666 | Alerts | Sequenced | ✓ | 9 | | 2020-06-04, 03:42:14 | [Delete] |
| 172.20.63.4:4444 | Alerts | Sequenced | ✓ | 11 | | 2020-06-04, 03:25:52 | [Delete] |
| 172.20.63.4:6666 | Perfmon | Sequenced | ✓ | 0 | | | [Delete] |
| 172.20.63.4:4444 | Events | Sequenced | ✓ | 1 | | 2020-06-04, 10:50:04 | [Delete] |

9.5.2.1 Creating Receiver

1. From the AOS web interface, navigate to **Platform / Streaming / Receivers**, then click **Create Receiver**.
2. Enter/select required values.
3. Click **Create** to create the receiver and return to the list view.

9.5.2.2 Deleting Receiver

1. From the AOS web interface, navigate to **Platform / Streaming / Receivers**, then click the delete button corresponding to the receiver to delete.
2. Click **Delete** to delete the receiver from the system and return to the list view.

9.6 Event Log

Information about user login, logout, blueprint changes, and other events are captured in the event log.

From the AOS web interface, navigate to **Platform / Audit / Event Log**.

1.

2.

Export to CSV

Query: All 1-25 of 129 Page Size: 25

| Time | User | User IP Address | Type | Result | Details |
|-------------|-------|-----------------|--------|---------|---------|
| 2 hours ago | admin | 10.1.253.226 | Login | Success | |
| 5 hours ago | admin | 10.1.253.226 | Logout | Success | |
| 5 hours ago | admin | 10.1.253.226 | Login | Success | |

You can view changes to device configurations by clicking **View Config** next to a **DeviceConfigChange** type of event.

Device Config

```

1 {
2   "hostname": [{
3     "filename": "/etc/hostname",
4     "data": [
5       "leaf-2-525400096AF5"
6     ]
7   },
8   {
9     "command": "/bin/hostname -F /etc/hostname"
10  },
11  {
12    "command": "systemctl reset-failed lldpd.service"
13  },
14  {
15    "command": "/usr/sbin/service lldpd restart"
16  }
17 ], "interfaces": [{
18   "filename": "/etc/network/interfaces",
19   "data": [
20     "# This file was generated by AOS. Do not edit by hand.",
21     "#",
22     "# The loopback interface",
23     "auto lo",
24     "iface lo inet loopback",
25     "    address 172.16.0.1/32",
26     "    # Logical VTEP",
27     "    address 203.0.113.1/32",
28     "",
29     "# Fabric interfaces",
30     "auto swp1",
31     "iface swp1",
32     "    address 172.16.0.9/31",
33     "    alias facing_spine2:swp2",
34     "",
35     "# Auto generated"

```

Fig. 1: Device Configuration Change Details

9.6.1 Exporting Event Log to CSV File

1. Click **Export to CSV**.

2. Enter details for information that you want to extract.
3. Click **Save as CSV File**.

9.6.2 Sending Event Log with Syslog

See *Syslog Configuration* for details about sending the event log to an external system with the Syslog protocol.

9.7 AOS Cluster

Demand for compute resources can grow beyond the capacity of a single virtual machine (VM). Multiple VMs can be clustered, specifically for scaling off-box agents and IBA probe processing units (as of version 3.0).

When a worker VM is added to the main Apstra controller VM, it registers with the Apstra server VM through sysdb, collects facts about the VM (such as core/memory/disk configuration and usage), and launches a container on the local VM. The Apstra controller VM reacts to REST API requests, configures the worker VM for joining or leaving the cluster, and keeps track of cluster-wide runtime information. It also reacts to container configuration entities and schedules them to the worker VM.

REST API and Clusters

In addition to managing Apstra server VMs using the web interface (as described below) you can also use REST API. Navigate to **Platform > Developers** for **REST API Documentation** and tools. See the **cluster** section.

Apstra server VMs include the following details:

Static Configuration

Address IP address or Fully-qualified Domain Name (FQDN) of the VM

Name Apstra VM name, such as **controller** (the main Apstra controller) or **node1** (a worker node)

State ACTIVE, MISSING, or FAILED

Roles Controller or worker

Tags The function of the node: iba and/or offbox

Capacity Score Calculated by Apstra

CPU Number of CPUs

Errors If an error exists, a message appears here. For example, the following image shows the error message when an agent process has been restarted because an agent has crashed.

The screenshot displays the Apstra web interface for an AOS Cluster controller. The left sidebar contains navigation links for Blueprints, Devices, Design, Resources, External Systems, Platform (selected), User Management, Users, Roles, External Services, Syslog Configuration, Streaming Endpoints, Event Log, AOS Cluster (highlighted), Developers, Technical Support, and About. The user is logged in as 'admin'.

The main content area shows the configuration for the 'controller' node under 'Platform > AOS Cluster'. It includes a 'Static Configuration' table with the following details:

| | |
|----------------|-------------|
| Address | 172.20.30.3 |
| Name | controller |
| Roles | controller |
| Tags | ibac offbox |
| Capacity Score | 19 |
| CPU | 8 |

Below the configuration, an 'Errors' section shows a message: 'Some agents have crashed and have been restarted recently: aos_controller_1'.

The 'Usage' section provides a summary of resource usage:

- Container Service Usage: 26%
- Containers Count: 5
- Memory Usage: 64%
- CPU Usage: 0%

The 'Disk(s) Usage' table shows usage per logical volume:

| Name | Usage | Used, GB | Total, GB |
|------------------------------|-------|----------|-----------|
| aos-server-vg-var+log | 4% | 0.55 | 11.20 |
| aos-server-vg-root | 35% | 3.27 | 9.32 |
| aos-server-vg-var+lib+aos+db | 0% | 0.08 | 11.20 |
| aos-server-vg-var | 15% | 4.84 | 25.85 |

The 'Containers' section shows a table of running containers:

| Name | State | Memory Usage, Mb | CPU Usage | Cumulative File Size, Mb |
|------------------|----------|------------------|-----------|--------------------------|
| aos_auth_1 | launched | 178.90 | 0% | 45.88 |
| aos_controller_1 | launched | 3639.52 | 1% | 146.64 |
| aos_metadb_1 | launched | 81.30 | 0% | 75.08 |
| aos_sysdb_1 | launched | 389.30 | 1% | 111.51 |
| ibac3f82fc3 | launched | 239.45 | 0% | 16.91 |

Usage

Container Service Usage Current VM container server usage (Percentage)

Containers Count Number of containers

Memory Usage Current VM memory usage (percentage)

CPU Usage Current VM CPU usage (percentage)

Disk Usage Current VM disk usage per logical volume (GB and percentage)

Containers The containers that are running on the node and the resources used by each container

Username Web interface username login credential, such as *admin*

Password Web interface password login credential

From the web interface, navigate to **Platform > AOS Cluster**. You can health check the VMs from here. To see more information, click an address.

1. Platform

2. AOS Cluster

3. Create Node

Click for details

| Address | Name | State | Roles | Tags | Capacity Score | Containers Count | CPU | Memory Usage, Gb | CPU Usage | Disk Usage | Container Service Usage | Actions |
|-------------|------------|--------|------------|---------------|----------------|------------------|-----|------------------|-----------|------------|-------------------------|-------------------|
| 172.20.71.3 | controller | ACTIVE | controller | | 321 | 4 | 4 | 5.11 (16%) | 6% | 6% | 0% | Edit Clone Delete |
| 172.20.71.4 | worker_1 | ACTIVE | worker | iba offbox | 643 | 2 | 4 | 0.88 (2%) | 1% | 9% | 0% | Edit Clone Delete |

Cloning Apstra VM

Instead of entering all details for a new VM, you can clone an existing one, give it a new name and customize it.

9.7.1 Creating Apstra VM

1. *Install Apstra software* on the VMs to be clustered, making sure they are all the same version as the main controller (which acts as the cluster manager). If they are not the same version, the controller will not accept them as part of the cluster.
2. From the list view (Platform > AOS Cluster) click **Add Node**, then enter a name, tags (optional), address (IP or FQDN), username and password.
3. 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**.

9.7.2 Editing Apstra VM

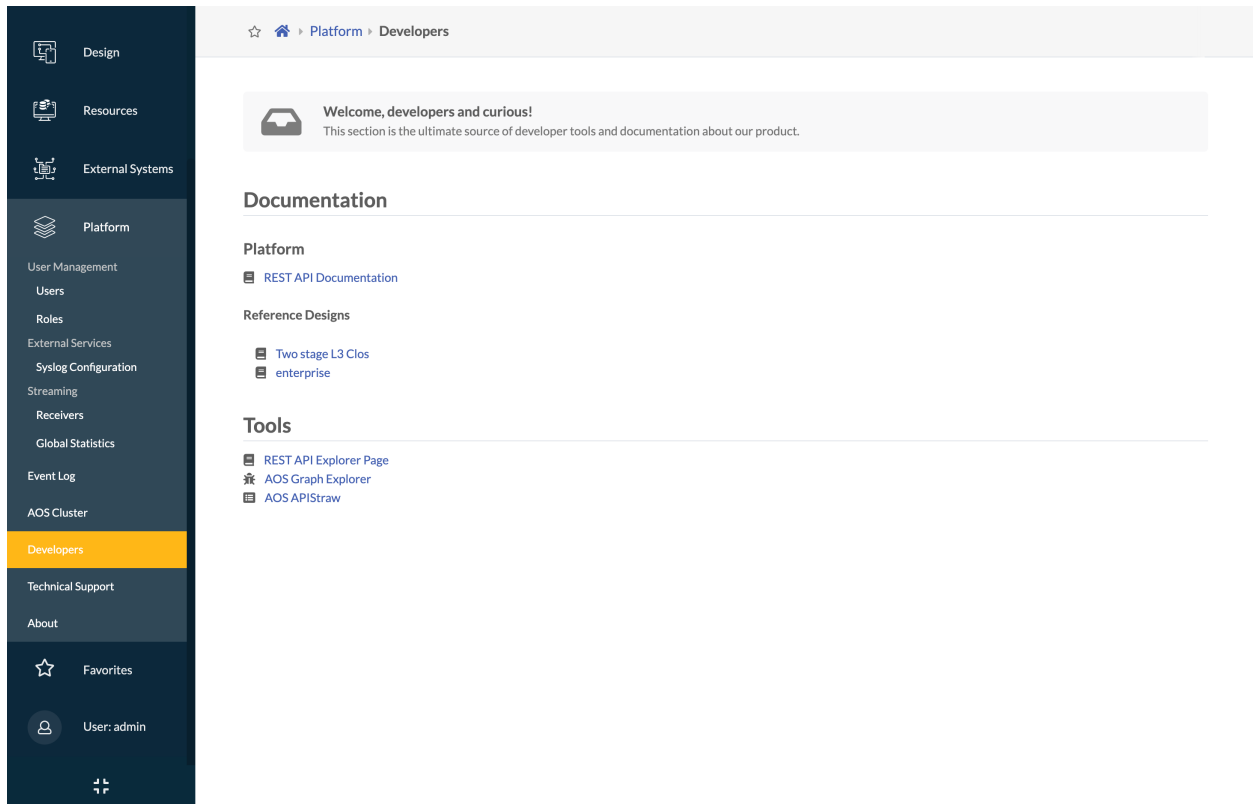
1. Either from the list view (Platform > AOS Cluster) or the details view, click the **Edit** button for the VM to edit.
2. Make your changes.
3. Click **Update** to update the Apstra VM worker node.

9.7.3 Deleting Apstra VM

1. Either from the list view (Platform > AOS Cluster) or the details view, click the **Delete** button for the Apstra VM to delete.
2. Click **Delete** to delete the Apstra VM.

9.8 Developers

Developer documentation and tools are available for AOS users in the **Developers** option in the AOS UI **Platform** menu.



9.8.1 API Documentation

The **Documentation** section includes links to the AOS in-product API documentation.

Platform REST API Documentation includes API documentation for APIs used outside of an AOS Blueprint (e.g. AOS Global Catalog Logical Devices).

Reference Designs Two stage L3 Clos includes API documentation for APIs used in the standard AOS L3 Clos Blueprint (e.g. AOS Blueprint Virtual-Networks).

Reference Designs enterprise is for future functionality and is not currently supported by Apstra.

9.8.1.1 API Examples

Examples for using the Apstra AOS API can be found in one of these sections.

Resource Pools API Reference

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 AOS Web UI.

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

Creating an 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.eyJ1c2VybmFtZSI6ImFkbWluIiwiaWF0Ij01YXR1ZD9hdCI6IjIwMTctMDU0MzFUMDQ6MjI0MjI0MTIwMTgzWiIsInNlc3Npb24iOiJOTliOGVlOS05Y2NjLTRjZTAtYTU5NS0wODI3N2ZkYjA0ZDZifQ.FnJMR3crPoD0-lQRXnpPOJ8TCsRG9Wr-DaddnAIj6ko' - --data-binary '{"display_name": "Example", "ranges": [{"first": 100, "last": 200}], "tags": []}' --compressed
--insecure
```

Listing ASN Pools

```
curl 'https://192.168.25.250/api/resources/asn-pools' -H 'AuthToken:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybWFnZSI6ImFkbWluIiwiaWF0Ij0tLiY
XRlZF9hdCI6IjIwMTctMDU0MzFUMDQ6MjI6MDcuNTIwMTgzWiIsInNlc3Npb24iOiJjOTliO
GVLOS05y2NjLTRjZTAtYTYSNS0wODI3N2ZkYjA0ZDYifQ.FnJMR3crPoD0-lQRXnpPOJ8TCsR
G9Wr-DaddnAIj6ko' --compressed --insecure
```

```
{
  "items": [
    {
      "created_at": "2017-05-30T12:56:07.293082Z",
      "display_name": "Private ASN",
      "id": "c23ea447-8f37-419a-9b1c-c48cc55d5b9c",
      "last_modified_at": "2017-05-30T12:56:07.293082Z",
      "ranges": [
        {
          "first": 65412,
          "last": 65534,

```

(continues on next page)

(continued from previous page)

```

    "status": "pool_element_in_use"
  },
  ],
  "status": "in_use",
  "tags": []
}
]
}

```

Deleting an 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
FtZSI6ImFkbWluIiwiaWY3JlYXRlZF9hdCI6IjIwMTctMDU0MzFUMDA6MjI6MDcuNTIwMTgzZW
lZSI6InNlc3Npb24iOiJ0TlI0GVlOS0S0Y2NjLTRjZTAtYTYSNS0wODI3N2ZkYjA0ZDYifQ.FnJ
MR3crPoD0-lQRXnpPOJ8TCsRG9Wr-DaddnAIj6ko' --compressed --insecure
```

Assigning an ASN Pool to a 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.eyJ1c2Vyby
mFtZSI6ImFkbWluIiwiaWF0Ij01YXRlZF9hdC1eIjwiMTctMDUtMzMUMDA6Mji6MDcuNTI
wMTgzWiIsInNlc3Nb2d4OiJ0LiOGVLOS05Y2NjLTRjZTAtYTYSNS0WODI3N2kYy
A0ZDYifQ.FnJMR3crPoD0-1QRXnpPOJ8TCsRG9Wr-DaddnAIj6ko' --data-binary
 '{"pool_ids":["c23ea447-8f37-419a-9b1c-c48cc55d5b9c"]}' --compressed --insecure
```

A successful ASSIGNMENT returns HTTP 200 OK.

Unassigning an ASN Pool from a blueprint

Removing IP pools from the blueprint requires the user to PUT an empty pool_id list to the blueprint with the payload []:

PUT to the HTTP endpoint https://aos-server/api/blueprints/<blueprint_id>/resource_groups/asn/<pool_name>

With the payload:

```
{ "pool_ids": [] }
```

```
curl
'https://192.168.25.250/api/blueprints/4c1e69c6-97bd-4c99-9504-7818f138b17f/resource_
→groups/asn/spine_asns'
-X PUT -H 'AuthToken: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFt
ZSI6ImFkbWluIiwiaWY3JlYXRlZF9hdCI6IjIwMTctMDU0MzFUMDA6MjI6MDcuNTIwMTgzWi
IsInNlc3Npb24iOiJjOTliOGVlOS05Y2NjLTRjZTAtYTY5NS0wODI3N2ZkYjA0ZDYifQ.FnJ
MR3crPoD0-lQRXnpPOJ8TCsRG9Wr-DaddnAIj6ko' --data-binary '{"pool_ids":[]}'
--compressed --insecure
```

If the request is successful there will be no response.

Listing the ASN Pools assigned to a blueprint

Available ASN Pool resource groups for assignment can be shown with an HTTP GET to https://aos-server/api/blueprints/<blueprint_id>/resource_groups

```
curl
'https://192.168.25.250/api/blueprints/4c1e69c6-97bd-4c99-9504-7818f138b17f/resource_
→groups'
-H 'AuthToken: eyJhbGciOiJIUzI1NWMTctMDU0MzFUMDA6MjI6MDcuNTIwMTgz
WiIsInNlc3Npb24iOiJjOTliOGVlOS05Y2NjLTRjZTAtYTY5NS0wODI3N2ZkYjA0ZD
YifQ.FnJMR3crPoD0-lQRXnpPOJ8TCsRG9Wr-DaddnAIj6ko' --compressed --insecure
| python -m json.tool
```

```
{
  "items": [
    {
      "name": "leaf_asns",
      "pool_ids": [
        "c23ea447-8f37-419a-9b1c-c48cc55d5b9c"
      ],
      "type": "asn"
    },
    {
      "name": "spine_asns",
      "pool_ids": [
        "c23ea447-8f37-419a-9b1c-c48cc55d5b9c"
      ],
      "type": "asn"
    },
    {
      "name": "leaf_loopback_ips",
      "pool_ids": [
        "56e8e0dc-babd-4652-92a5-fc37294a7b26"
      ],
      "type": "ip"
    },
    {
      "name": "mlag_domain_svi_subnets",
      "pool_ids": [
        "ed7d8830-c703-4ac0-8252-77e0f272a677"
      ],
    }
  ]
}
```

(continues on next page)

(continued from previous page)

```

        "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"
    },
    {
        "name": "to_external_router_link_ips",
        "pool_ids": [
            "aae25699-9ad6-4382-8a7b-3f4a8ef582ac"
        ],
        "type": "ip"
    }
]
}

```

API - IP Pools

Creating an IP Pool

JSON Payload for creating an IP Pool:

Listing 3: IP Pool creation

```

{
    "id": "example_ip_pool",
    "display_name": "example_ip_pool",
    "tags": ["default"],
    "subnets": [
        {"network": "10.0.0.0/8"}
    ]
}

```

The **subnets** section requires a list of dictionaries with keyword **network** and value matching a CIDR mask. The subnets cannot overlap with each other in the same pool. That is to say, 192.168.10.0/24 and 192.168.0.0/16 cannot be configured in the same pool.

Tags are optional and are not currently used in AOS. If ID is specified, it will be saved, otherwise an ID will be returned in the HTTP Response after creating the pool.

An HTTP POST to <https://aos-server/api/resources/ip-pools> with JSON payload will reply with the ID of the new IP pool.

```

curl 'https://192.168.25.250/api/resources/ip-pools' -X
POST -H 'AuthToken: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmF

```

(continues on next page)

(continued from previous page)

```
tZSI6ImFkbWluIiwjY3JlYXRlZF9hdCI6IjIwMTctMDUtMzFUMDA6MjI6MDcuNTIwMTgzWi
IsInNlc3Npb24iOiJjOTliOGVlOS05Y2NjLTRjZTAyYTY5NS0wODI3N2ZkYjA0ZDYifQ.Fn
JMR3crPoD0-lQRXnpPOJ8TCsRG9Wr-DaddnAIj6ko' --data-binary '{"display_name":
"example_ip_pool","subnets":[{"network":"10.0.0.0/8"}, {"network":
"192.168.0.0/16"}],"tags":[]}' --compressed --insecure
```

```
{"id": "d0312b4a-017e-4478-8b8d-df0417ce8d3b"}
```

Listing 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
luIiwjY3JlYXRlZF9hdCI6IjIwMTctMDUtMzFUMDA6MjI6MDcuNTIwMTgzWiIsInNlc3Npb24
iOiJjOTliOGVlOS05Y2NjLTRjZTAyYTY5NS0wODI3N2ZkYjA0ZDYifQ.FnJMR3crPoD0-lQRXnpP
OJ8TCsRG9Wr-DaddnAIj6ko' --compressed --insecure | python -m json.tool
```

Listing 4: Listing IP Pools

```
{
  "items": [
    {
      "created_at": "2017-05-30T12:56:37.190419Z",
      "display_name": "External Routers",
      "id": "aae25699-9ad6-4382-8a7b-3f4a8ef582ac",
      "last_modified_at": "2017-05-30T12:56:37.190419Z",
      "status": "in_use",
      "subnets": [
        {
          "network": "192.168.66.0/24",
          "status": "pool_element_in_use"
        }
      ],
      "tags": []
    },
    {
      "created_at": "2017-05-31T03:48:38.562331Z",
      "display_name": "example_ip_pool",
      "id": "d5046aa6-eab2-4990-9816-0a519cela8db",
      "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": []
    }
  ],
}
```

(continues on next page)

(continued from previous page)

```

{
  "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": []
}
]
}

```

Deleting an IP pool

To delete an IP Pool perform an HTTP DELETE to https://aos-server/resources/ip-pools/{pool_id}

A successful DELETE returns HTTP 200 OK and an empty JSON response { }

```
curl
'https://192.168.25.250/api/resources/ip-pools/d0312b4a-017e-4478-8b8d-df0417ce8d3b'
-X DELETE -H 'AuthToken: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFkbWluIiwiaWF0Ij01YXRlZF9hdCI6IjIwMTctMDUtMzMvFUMDA6MjI6MDcuNTIwMTgzWiIsInN1
I3Npb24iOiJ0TlI0GVLOS05Y2NjLTRjZTAyTYYSNS0wODI3N2kxYjA0ZDYifQ.FnJMR3crPoD0
-lQRXnpPQJ8TCsRG9Wr--DaddnAIj6ko --compressed --insecure
```

Assigning an IP pool to a blueprint

To assign an IP pool to the blueprint perform an HTTP PUT to <https://aos-server/blueprints/<blueprint id>/resource groups/ip/<group name>>

For instance, to associate a resource pool **spine_loopback_ips** with a blueprint first obtain the ID of the resource pool, and append it to a list for slot assignation. When updating the IP Pool resource group, specify all pools in the payload at the same time. We cannot add single pools, so PUT them all at once. Instruct AOS to associate IP pool with ID 'ed7d8830-c703-4ac0-8252-77e0f272a677' to the blueprint. You may have to GET existing pool IDs prior to adding a new one to avoid deleting existing pools.

Payload:

```
{ "pool_ids": [ "pool_id1", "pool_id2", "pool_id3" ] }
```

```
curl
'https://192.168.25.250/api/blueprints/4c1e69c6-97bd-4c99-9504-7818f138b17f/resource_
groups/ip/spine_loopback_ips'
-X PUT -H 'AuthToken: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFkbWluIiwiaWF0Ij01OTliOGVLOS05Y2NjLTRjZTAtYTYSNS0wODI3N2ZkYjA0ZDYifQ.FnJMR3crPoD0-lQRXnp
POJ78TCsrRG9Wr-DaddnAij6ko' --data-binary '{"pool_ids":["ed7d8830-c703-4ac0-825
2-77e0f272a677"]}' --compressed --insecure
```

A successful ASSIGNMENT returns an HTTP 200 OK.

Removing an IP pool from a Blueprint

To remove IP pools from the blueprint PUT an empty `pool_id` list to the blueprint with the payload []:

PUT to the HTTP endpoint https://aos-server/api/blueprints/<blueprint_id>/re-source_groups/ip/<allocation_group_name>

With the payload:

```
{ "pool_ids": [] }
```

CURL Example

```
curl
'https://192.168.25.250/api/blueprints/4c1e69c6-97bd-4c99-9504-7818f138b17f/resource_
groups/ip/spine_loopback_ips'
-X PUT -H 'AuthToken: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZ
SI6ImFkbWluIiwiaWF0Ij0tLiOGVlOS05Y2NjLTRjZTAtYTYSNS0wODI3N2ZkYjA0ZDYifQ.FnJMR3cr
PoD0-lQRXnpPOJ8TCsRG9Wr-DaddnAIj6ko' --data-binary '{"pool_ids":[]}'
--compressed --insecure
```

A successful **REMOVAL** returns an empty response: { }

Listing the IP Pools Assigned to a Blueprint

```
curl
'https://192.168.25.250/api/blueprints/4c1e69c6-97bd-4c99-9504-7818f138b17f/resource_
→groups'
-H 'AuthToken: eyJhbGciOiJIUzI1NiMTctMDUzMzFUMDA6MjI6MDcuNTIwMTgzWiIsInNlc3
Npb24iOiJjOTliOGVlOS05Y2NjLTRjZTAtYTY5NS0wODI3N2ZkYjA0ZDYifQ.FnJMR3crPoD
0-lQRXnpPOJ8TCsRG9Wr-DaddnAIj6ko' --compressed --insecure | python -m json.tool
```

```
{
  "items": [
    {
      "name": "leaf_asns",
      "pool_ids": [
        "c23ea447-8f37-419a-9b1c-c48cc55d5b9c"
      ],
      "type": "asn"
    },
    {
      "name": "spine_asns",
      "pool_ids": [
        "c23ea447-8f37-419a-9b1c-c48cc55d5b9c"
      ],
      "type": "asn"
    },
    {
      "name": "leaf_loopback_ips",
      "pool_ids": [
        "56e8e0dc-babd-4652-92a5-fc37294a7b26"
      ],
      "type": "ip"
    },
    {
      "name": "mlag_domain_svi_subnets",
      "pool_ids": [
        "ed7d8830-c703-4ac0-8252-77e0f272a677"
      ],
      "type": "ip"
    },
    {
      "name": "spine_leaf_link_ips",
      "pool_ids": [
        "ed7d8830-c703-4ac0-8252-77e0f272a677"
      ],
      "type": "ip"
    },
    {
      "name": "spine_loopback_ips",
      "pool_ids": [
        "56e8e0dc-babd-4652-92a5-fc37294a7b26"
      ],
      "type": "ip"
    },
    {
      "name": "to_external_router_link_ips",
      "pool_ids": [
        "aae25699-9ad6-4382-8a7b-3f4a8ef582ac"
      ]
    }
  ]
}
```

(continues on next page)

(continued from previous page)

```

    },
    "type": "ip"
  }
]
}

```

Configlet API Reference

For full API documentation, view the Platform API reference under the AOS Web UI. 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 AOS 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.

Listing 5: Configlet schema

```

{
  "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 - Creating a configlet

To create a configlet, POST to <https://aos-server/api/design/configlets> with a valid JSON structure representing the configlet. Creating a configlet this way only allows it to be available for assignment in the AOS Web UI - it is not required in this method for the REST API to assign to a blueprint. See the assigning a configlet section for more details.

A POST will create a new configlet. A PUT will overwrite an existing configlet. PUT requires the URL of the configlet. <https://aos-server/api/design/configlets/{id}>

```

curl -H "AuthToken: EXAMPLE" -d '{"display_name":"DNS","ref_archs":["two_stage_l3clos
↪"],"section":"system","generators":[{"config_style":"eos","template_text":"ip name-
↪server 192.168.1.1","negation_template_text":"no ip name-server 192.168.1.1"}]}' -X_
↪POST "http://aos-server/api/design/configlets"

```

The response will contain the ID of the newly created configlet `{"id": "995446c7-de7d-46bb-a88a-786839556064"}`

API - Deleting a configlet

Deleting a configlet requires an HTTP DELETE to the configlet by URL <http://aos-server/api/design/configlets/{id}>

Listing 6: CURL Example - HTTP DELETE

```
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 - Assigning a 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 - Unassigning a configlet

To unassign a configlet, remove it from the items list by PUT with an empty json post.

Listing 7: CURL PUT example

```
curl "http://aos-server/api/blueprints/e4068e99-813c-4290-b7cc-e145d85a98a8/configlets" \
  -X PUT -H "AuthToken: EXAMPLE" -H "Content-Type: application/json; charset=utf-8" \
  --data ""
```

The response will be an empty json set once the configlet is deleted: `{"items": []}`

Property Set API Reference

For full API documentation, view the Platform API reference under 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.

Listing 8: Property Set schema

```
{
  "items": [
    {
      "label": "string",
      "values": {
        "additionalProp1": "string",
        "additionalProp2": "string",
        "additionalProp3": "string"
      },
      "id": "string"
    }
  ]
}
```

API - Creating a 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 assignation 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 - Deleting a 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}>

Listing 9: CURL Example - HTTP DELETE

```
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 - Assigning a 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" -X PUT -H "AuthToken: EXAMPLE" -H "Content-Type: application/json;
↪ charset=utf-8" --data '{"id": "73223e81-a451-4e7f-91fb-fb476f4b9fc8"}'
```

Response

```
{"id": "73223e81-a451-4e7f-91fb-fb476f4b9fc8"}
```

API - Unassigning a 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}

Listing 10: CURL PUT example

```
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 { }

External Routers API Reference

Routers are assigned to the blueprint similar to resource pools, but are not ‘allocated’ in a similar way internally as they can be shared between blueprints, and no resource allocation is done in the back-end.

The router requires a slot assignment of the name ‘to_external_router_link_ips’ as well.

Create a router

An example payload for an external router:

```
{
  "id": "example_router1",
  "display_name": "example_router1",
  "address": "198.51.100.1",
  "asn": 65534,
  "tags": ["default"]
}
```

Listing 11: Patch router:

```
curl 'https://192.168.25.250/api/resources/external-routers' -H 'AuthToken:
→eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
→eyJ1c2VybmFtZSI6ImFkbWluIiwia3JlYXRlZF9hdCI6IjIwMTctMDU0MzFUMDA6MjI6MDcuNTIwMTgzWiIsInNlc3Npb24iOi
→FnJMR3crPoD0-lQRXnpPOJ8TCsRG9Wr-DaddnAIj6ko' --data-binary '{"display_name":
→"example_router1","address":"198.51.100.1","asn":65534}' --compressed --insecure
```

Assign a router to a blueprint

Routers can be assigned to a blueprint with a PATCH to https://aos-server/api/blueprints/<bp_id>/external-router-assignments with the JSON payload:

```
{"assignments":{"<router_id>":"router1"}
```

Each additional assigned router in the blueprint will be assigned the name of router1, router2, router3, etc. The router ID is learned from the graph.

```
curl 'https://192.168.25.250/api/blueprints/4c1e69c6-97bd-4c99-9504-7818f138b17f/
→external-router-assignments' -X PATCH -H -H 'AuthToken:
→eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
→eyJ1c2VybmFtZSI6ImFkbWluIiwia3JlYXRlZF9hdCI6IjIwMTctMDU0MzFUMDA6MjI6MDcuNTIwMTgzWiIsInNlc3Npb24iOi
→FnJMR3crPoD0-lQRXnpPOJ8TCsRG9Wr-DaddnAIj6ko' '{"assignments":{"7c700fe6-404f-4f86-
→959d-3a1fe33a7938":"router1"}}' --compressed --insecure
```

Unassign a router to a blueprint

Unassigning the router is done by posting *null* to the router node ID to the API endpoint https://aos-server/api/blueprints/<bp_id>/external-router-assignments with JSON payload:

```
{
  "assignments": {
    "7c700fe6-404f-4f86-959d-3a1fe33a7938": null
  }
}
```

```
curl 'https://192.168.25.250/api/blueprints/4c1e69c6-97bd-4c99-9504-7818f138b17f/external-router-assignments' -X PATCH -H 'AuthToken: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFkbWluIiwia3JlYXRlZF9hdCI6IjIwMTctMDUtMzFUMDA6MjI6MDcuNTIwMTgzWiIsInNlc3Npb24iOiJFnJMR3crPoD0-lQRXnpPOJ8TCsRG9Wr-DaddnAIj6ko' -H 'Accept: application/json; charset=utf-8' --data-binary '{"assignments":{"7c700fe6-404f-4f86-959d-3a1fe33a7938":null}}' --compressed --insecure
```

List routers assigned to a blueprint

Obtaining a list of router assignments (and their node ID) is done by viewing the API endpoint with an HTTP GET to https://aos-server/api/blueprints/<bp_id>/external-router-assignments. The response will contain router ID slots for assignment.

```
{
  "assignments": {
    "7c700fe6-404f-4f86-959d-3a1fe33a7938": null
  }
}
```

```
curl 'https://192.168.25.250/api/blueprints/4c1e69c6-97bd-4c99-9504-7818f138b17f/external-router-assignments' -H 'AuthToken: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFkbWluIiwia3JlYXRlZF9hdCI6IjIwMTctMDUtMzFUMDA6MjI6MDcuNTIwMTgzWiIsInNlc3Npb24iOiJFnJMR3crPoD0-lQRXnpPOJ8TCsRG9Wr-DaddnAIj6ko' -H 'Accept: application/json; charset=utf-8' --compressed --insecure
```

```
{
  "assignments": {
    "7c700fe6-404f-4f86-959d-3a1fe33a7938": null
  }
}
```

Interface Descriptions

Besides main parameters of network interfaces like name, speed and port mode, AOS 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 L3 server.

3. The peer interface belongs to leaf, spine, L3 server, external router or L2 server 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`
- `facing_external-router`
- `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 AOS.

AOS REST API - Interface descriptions

The AOS API is able to change the auto-generated interface description. However, there is no such functionality in the AOS 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 - Obtaining interface configuration

To obtain interface configuration, send GET request to <https://aos-server/api/blueprints/{blueprint-id}/nodes/{interface-node-id}>.

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"
}
```


API - Creating or modifying 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.

Request:

```
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 - Deleting 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"
}
```

IBA Probes API Reference

Generic Probe REST API

Here we describe as much of the API as is necessary to understand how to use IBA for someone already roughly familiar with AOS and its API conventions. The excruciating detail necessary for formal API documentation is reserved for the API documentation itself.

We will walk through the API as it is used for the example workflow described in the introduction, demonstrating its general capability by specific example.

Creating a Probe

To create a probe, the operator POSTs to `/api/blueprints/<blueprint_id>/probes` with the following form:

Listing 12: POST for Probe Creation

```
{
  "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',
700 'dst_intf')",
(continues on next page)
```

(continued from previous page)

```

        "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 AOS 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 AOS, 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.

Listing 13: POST for Probe Creation with Predefined Id

```
{
  "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>`.

Changed in version 2.3: HTTP PATCH and PUT methods for probes are available since AOS 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.

Listing 14: PATCH for Probe Metadata Update

```
{
  "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.

Inspection of a Probe

As described earlier, the operator may wish to inspect the various stages of a probe. Recall from the previous section that such stages were implicitly created by being named in the input and output of the various processors.

The API for reading a particular stage is `/api/blueprints/<blueprint_id>/probes/<probe_id>/stages/<stage_name>`

Note: Stage Types

As alluded to previously, 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.

NST, DSTS, and TS are the same as above, but are time-series instead of single values.

Let us 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:

Listing 15: Sample Interface Collector Stage

```
{
  "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
    }
  ]
}
```

(continues on next page)

(continued from previous page)

```

    ]
}

```

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`.

Listing 16: Sample Standard Deviation Stage

```

{
  "type": "n",
  "units": "",
  "value": 1
}

```

As mentioned earlier, 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`.

Listing 17: Sample Standard Deviation Stage

```

{
  "possible_values": [
    "true",
    "false"
  ],
  "type": "ds",
  "units": "",
  "value": false
}

```

As shown, this stage indicates whether or not 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.

It is important to note that 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, the user can discover the configuration used for creation of said processor.

Querying Probe Anomalies

As mentioned earlier, the final stage of our example processor raises an AOS Anomaly (and sets its output to “true”), when the standard deviation of tx_bytes across server-facing interfaces is greater-than 100.

Probe anomalies can be queried 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):

Listing 18: Sample Standard Deviation Stage

```
{
  "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.

Introspecting Processors

As mentioned earlier, the set of processors available to the operator is a function of the platform and the reference design. AOS 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.

Listing 19: Standard Deviation Processor Description from API

```
{
  "description": "Standard Deviation Processor.\n\n Groups as described by group_by,\n↪ then calculates std deviation and\n outputs one standard deviation for each group.\n↪Output is NS.\n Input is an NS or NSTS.\n ",
  "inputs": {
    "in": {
      "required": true,
      "types": [
        {
          "keys": [],
          "possible_values": null,

```

(continues on next page)

(continued from previous page)

```

        "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 version 3.0: Previously, inputs and outputs section didn’t specify whether specific inputs or outputs were required, so the format was changed from:

Warning: This syntax is deprecated and invalid as of AOS version 3.0.

```
"inputs": {
  "in": [
    {
      "data_type": "ns",
      "keys": [
        "system_id"
      ],
      "value_map": null,
      "value_type": "int64"
    }
    ...
  ]
}
```

Streaming Data

Any processor instance in any probe can be configured to have its output stages streamed in the “perfmon” channel of AOS 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.

Listing 20: Format for Generic Probe Messages

```
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 Methods

You can access complete AOS API documentation from the AOS 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.

Creating a 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

Updating a 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

Deleting a Root Cause Identification Instance

```
DELETE /api/blueprints/<blueprint_id>/arca/<arca_id>
No request payload.
```

Return values:

204 - Delete succeeded.

404 - ARCA instance not found.

Getting a 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(SYMPTOM_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 expected 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

Getting list of 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.

AOS Cluster APIs

In addition to using the AOS web interface to check the health of AOS VMs, you can also use AOS REST API.

Health Checking AOS VMs via AOS REST API

From the AOS web interface, 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"
  ]
}
```

Using API From Python

Following is examples of Python3 code using the AOS API.

API User Login

```
import requests, sys

# IP of Cloudlabs AOS Server
aos_server = '172.16.90.3'
username = 'admin'
password = 'aos aos'

# authenticate and get a auth token
url = 'https://' + aos_server + '/api/user/login'
headers = { 'Content-Type':"application/json", 'Cache-Control':"no-cache" }
data = '{ \"username\":\"\" + username + '\", \"password\":\"\" + password + '\" }'
response = requests.request("POST", url, data=data, headers=headers, verify=False)
print('POST',url,response.status_code)
if response.status_code != 201:
    sys.exit('error: authentication failed')
auth_token = response.json()['token']
print(auth_token)
headers = { 'AuthToken':auth_token, 'Content-Type':"application/json", 'Cache-Control'
↪ ': "no-cache" }
```

API Blueprints

```
# get blueprint ID ... assuming there is only one
url = 'https://' + aos_server + '/api/blueprints'
response = requests.request('GET', url, headers=headers, verify=False)
print('GET', url, response.status_code)
blueprint_id = response.json()['items'][0]['id']
blueprint_name = response.json()['items'][0]['label']
print(blueprint_name, blueprint_id)
```

API Blueprints 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)
```

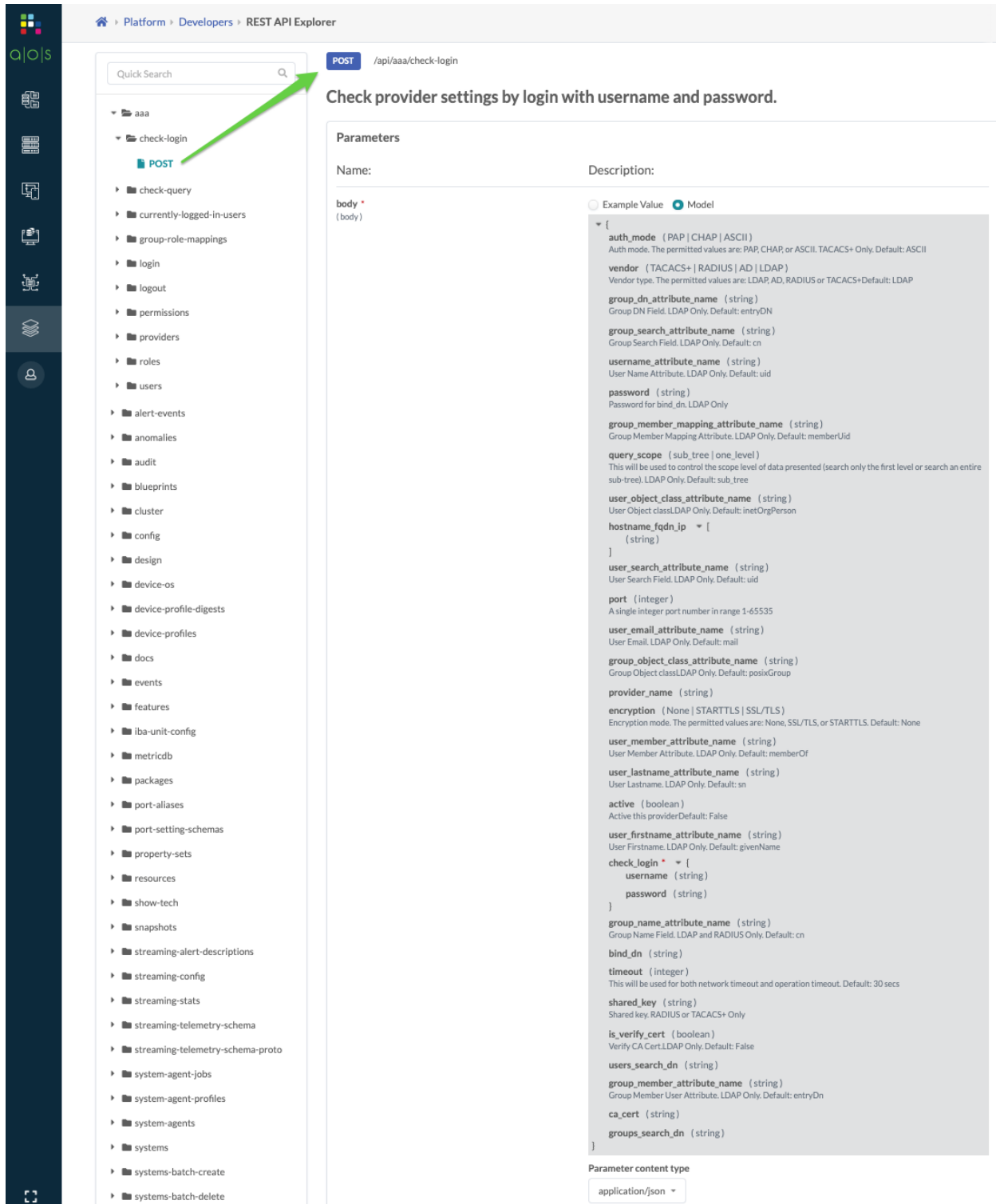

9.8.2 Tools

Documentation for available Apstra AOS Developer Tools can be found in one of these sections.

9.8.2.1 REST API Explorer

With Apstra's REST API explorer, you can browse and search for specific AOS REST API endpoints relevant to both the platform and reference designs.

From the AOS web interface, navigate to **Platform / Developers / REST API Explorer** to see the screen as shown below. The left column contains a list of API categories from which you can browse. You can also search for a specific endpoint by entering a query in the **Quick Search** field. The details view of an endpoint includes information about the URL, method, summary, parameters and responses. The example below shows the model for checking provider settings by login with username and password.



Platform > Developers > REST API Explorer

Quick Search

POST /api/aaa/check-login

Check provider settings by login with username and password.

Parameters

Name: body (body)

Description:

Example Value

```
{
  auth_mode: (PAP | CHAP | ASCII)
  Auth mode. The permitted values are: PAP, CHAP, or ASCII. TACACS+ Only. Default: ASCII
  vendor: (TACACS+ | RADIUS | AD | LDAP)
  Vendor type. The permitted values are: LDAP, AD, RADIUS or TACACS+ Default: LDAP
  group_dn_attribute_name: (string)
  Group DN Field. LDAP Only. Default: entryDN
  group_search_attribute_name: (string)
  Group Search Field. LDAP Only. Default: cn
  username_attribute_name: (string)
  User Name Attribute. LDAP Only. Default: uid
  password: (string)
  Password for bind_dn. LDAP Only
  group_member_mapping_attribute_name: (string)
  Group Member Mapping Attribute. LDAP Only. Default: memberUid
  query_scope: (sub_tree | one_level)
  This will be used to control the scope level of data presented (search only the first level or search an entire sub-tree). LDAP Only. Default: sub_tree
  user_object_class_attribute_name: (string)
  User Object classLDAP Only. Default: inetOrgPerson
  hostname_fqdn_ip: [
    (string)
  ]
  user_search_attribute_name: (string)
  User Search Field. LDAP Only. Default: uid
  port: (integer)
  A single integer port number in range 1-65535
  user_email_attribute_name: (string)
  User Email. LDAP Only. Default: mail
  group_object_class_attribute_name: (string)
  Group Object classLDAP Only. Default: posixGroup
  provider_name: (string)
  encryption: (None | STARTTLS | SSL/TLS)
  Encryption mode. The permitted values are: None, SSL/TLS, or STARTTLS. Default: None
  user_member_attribute_name: (string)
  User Member Attribute. LDAP Only. Default: memberOf
  user_lastname_attribute_name: (string)
  User Lastname. LDAP Only. Default: sn
  active: (boolean)
  Active this providerDefault: False
  user_firstname_attribute_name: (string)
  User Firstname. LDAP Only. Default: givenName
  check_login: {
    username: (string)
    password: (string)
  }
  group_name_attribute_name: (string)
  Group Name Field. LDAP and RADIUS Only. Default: cn
  bind_dn: (string)
  timeout: (integer)
  This will be used for both network timeout and operation timeout. Default: 30 secs
  shared_key: (string)
  Shared key. RADIUS or TACACS+ Only
  is_verify_cert: (boolean)
  Verify CA CertLDAP Only. Default: False
  users_search_dn: (string)
  group_member_attribute_name: (string)
  Group Member User Attribute. LDAP Only. Default: entryDn
  ca_cert: (string)
  groups_search_dn: (string)
}
```

Parameter content type

application/json

9.9 Technical Support

Technical Support is available to all customers with a valid and current license for Juniper Apstra. 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 JTAC Apstra Support may not be able to offer support and will refer you to the appropriate sales team to purchase a current license. See [Juniper JTAC Apstra Support Service Level Agreements \(SLA\)](#).

If you require assistance with registration or to open a technical support case via phone, please call Juniper Customer Care at +1-888-314-5822 (toll free, US & Canada). If you are outside the US or Canada, use +1-408-745-9500 or a country number listed at <https://support.juniper.net/support/requesting-support/>.

9.9.1 Providing Technical Support Data

To aid the support process, please provide Juniper JTAC Apstra Support with diagnostic information from the Apstra environment. Separate *show tech* files are needed from the Apstra server and from each of the affected device agents. You can obtain the show tech files for the Apstra server and connected on-box agents from the web interface (as of version 3.1). The show tech files for off-box agents (and the Apstra server and on-box agents running versions earlier than 3.1) must be obtained from the Apstra controller Linux CLI. See procedures below.

Registered, licensed customers can upload show tech files to their customer case at <https://casemanager.juniper.net/casemanager/>.

9.9.1.1 Show Tech for On-box Agents and Apstra Server (Web Interface)

You can use the web interface to collect show tech files for the controller and connected on-box agents for Cumulus Linux, Arista EOS, and Cisco NX-OS (as of version 3.1); and Enterprise SONiC (as of version 3.3).

1. If a controller Linux CLI username and password have not been configured, you must configure them now, before collecting show tech files.
 1. From the web interface, navigate to **Platform > AOS Cluster** to see the VMs list view.
 2. Click the IP address of the controller.
 3. Click the **Edit** button (top-right) to see the dialog for editing VMs.
 4. Enter the controller Linux CLI username and password.
 5. Click **Update** to complete the update.
2. From the web interface, navigate to **Platform > Technical Support** to see the show tech files list view.
3. Click **Collect Show Tech** to see the dialog for selecting and collecting show tech files.
4. If you want to collect a show tech from the controller, leave **AOS Controller** selected.

Note: For Apstra server controllers with large databases, the operation may timeout. If this happens, you must collect the show tech using CLI. For more information, see the **Show Tech for Apstra Server (CLI)** section below.

5. Check the box for **Managed Devices** to see the list of connected managed devices (devices with installed agents that have been acknowledged).
6. Select up to twenty devices for show tech collection.

Note: The configured device system agent username and password authentication is used to collect device show tech. If you have configured the device to use another authentication (AAA) method like RADIUS and TACACS with a different username and password, you cannot collect show tech from the web interface. You must collect show tech using CLI. For instructions, see CLI sections below for the device vendor.

- Click **Collect** to start the collection process.
- 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 1-2 of 2 Page Size: 25

Job ID: b7de9423-9b48-4064-841b-be9c99a3c9a2
SUCCESS

| Device Address | State | Started | Finished | Logs |
|----------------|---------|----------------------|----------------------|----------|
| 172.20.39.10 | SUCCESS | 2019-08-24, 03:04:13 | 2019-08-24, 03:05:19 | (183 KB) |
| 172.20.39.9 | SUCCESS | 2019-08-24, 03:04:13 | 2019-08-24, 03:05:24 | (199 KB) |
| 172.20.39.6 | SUCCESS | 2019-08-24, 03:04:13 | 2019-08-24, 03:05:24 | (200 KB) |
| 172.20.39.8 | SUCCESS | 2019-08-24, 03:04:13 | 2019-08-24, 03:05:23 | (200 KB) |
| 172.20.39.11 | SUCCESS | 2019-08-24, 03:04:13 | 2019-08-24, 03:05:22 | (200 KB) |

Job ID: 8fc44c61-4085-40ce-ba97-743fb32a7e80
SUCCESS

| Device Address | State | Started | Finished | Logs |
|----------------|---------|----------------------|----------------------|--------|
| AOS Controller | SUCCESS | 2019-08-24, 03:04:08 | 2019-08-24, 03:04:40 | (4 MB) |

- When the show tech files have been downloaded, you may click the **Delete** button for each job to free up disk space.
- Upload show tech files via the Support portal from a computer with the ability to upload.

9.9.1.2 Show Tech for Junos Off-box Agents (CLI)

You can collect off-box agent and Junos device show tech information from the CLI with the `aos_offbox_show_tech_collector` command (as of version 3.3). This command can be run on the Apstra server that the Junos off-box agent is running on.

The `aos_offbox_show_tech_collector` command only supports JUNOS off-box agents.

This command requires the device management IP address(es) and a valid device SSH username and password.

```
admin@aos-server:~$ aos_offbox_show_tech_collector --help
usage: aos_offbox_show_tech_collector [--ips IPS [IPS ...]] [--user USER]
                                     [--password PASSWORD]
                                     [--output-dir OUTPUT_DIR]
aos_offbox_show_tech_collector: error: unrecognized arguments: --help
admin@aos-server:~$
```

- Log into the Apstra server that the Junos off-box agent is running on via SSH.
- Run the `aos_offbox_show_tech_collector` command with the IP(s) of the Junos device(s), and a valid admin username and password.

Listing 21: Generating aos_show_tech for Junos off-box AOS Agent

```
admin@aos-server:~$ sudo aos_offbox_show_tech_collector --ips 172.20.202.6 --user_
↪admin --password admin-password
2020-08-10 12:03:46,116 aos-offbox-172_20_202_6-f
2020-08-10 12:03:46,384 collecting offbox container aos-offbox-172_20_202_6-f_
↪showtech
2020-08-10 12:03:50,873 invoking DI container to collect device 172.20.202.6 show_
↪tech
2020-08-10 12:05:42,155 Done collecting device show_tech
2020-08-10 12:05:42,204 AOS offbox show tech generated at /home/admin/aos_show_
↪tech_20200810_120542_172_20_202_6.tar.gz
admin@aos-server:~$
```

3. The show tech file (e.g. `aos_show_tech_20200810_120542_172_20_202_6.tar.gz`) will be in your user directory. Note, the file will be owned by root and may need a permissions change in order to copy the file off of the Apstra server.

```
admin@aos-server:~$ ls -l aos_show_tech_20200810_120542_172_20_202_6.tar.gz
-rw----- 1 root root 238156 Aug 10 12:05 aos_show_tech_20200810_120542_172_20_
↪202_6.tar.gz
admin@aos-server:~$ sudo chmod a+r aos_show_tech_20200810_120542_172_20_202_6.tar.
↪gz
admin@aos-server:~$ ls -l aos_show_tech_20200810_120542_172_20_202_6.tar.gz
-rw-r--r-- 1 root root 238156 Aug 10 12:05 aos_show_tech_20200810_120542_172_20_
↪202_6.tar.gz
admin@aos-server:~$
```

9.9.1.3 Show Tech for Off-box Agents (CLI)

Show tech logs for installed off-box agents (all Apstra versions) must be obtained from the CLI. You cannot collect them from the web interface.

1. Log into the Apstra server via SSH.
2. From the Apstra server, run the `docker exec` command as described below to generate the show tech file.

```
docker exec -ti aos-off-box-<ip_address>-t aos show tech
```

The Docker container name is *aos-off-box-* plus the IP address of the off-box device agent with the dots replaced with underscores followed by *-t*.

Listing 22: Example: generate show tech file for off-box agent with IP address 172.20.47.6

```
admin@aos-server:~$ docker exec -ti aos-off-box-172_20_47_6-t aos_show_tech
AOS show tech generated at /tmp/aos_show_tech_20200401_181128.tar.gz
admin@aos-server:~$
```

3. Using SCP, run the `docker cp` command as shown below to copy the show tech file from the off-box agent Docker container to the `/tmp` directory of the Apstra server.

Listing 23: Example: copy show tech file from Docker container to the AOS Server

```
admin@aos-server:~$ docker cp aos-off-box-172_20_47_6-t:/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:~$
```

4. Locate the file archive in the /tmp directory and copy the file to a local computer with the ability to upload.
5. *Upload the files* via the Support portal.

9.9.1.4 Show Tech for Apstra Server (CLI)

We recommend that you use the web interface to obtain Apstra server show tech files, but you have the option of using the Apstra server Linux CLI as described below.

1. Log into the Apstra server via SSH.
2. From the Apstra server, run the `sudo aos_show_tech` command to generate and copy the show tech file to the current working directory of the Apstra server.

Listing 24: Generating aos_show_tech for AOS server

```
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
```

(continues on next page)

(continued from previous page)

```
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 (aos_show_tech_20200401_033431.tar.gz for example), and via SCP, copy the file to a local computer with the ability to upload.
4. *Upload the file* via the Support portal.

9.9.1.5 Show Tech for Arista On-box Agents (CLI)

We recommend that you use the *web interface* to obtain on-box agent show tech files, but you have the option of using the Apstra server Linux CLI as described below.

1. SSH to the Arista device.
2. From the device, run the `sudo aos_show_tech --platform eos` command to generate and copy the show tech file to the /tmp directory.

Listing 25: Generating aos_show_tech for Arista AOS Agent

```
l2-virtual-ext-003-leaf1#bash

Arista Networks EOS shell

[admin@l2-virtual-ext-003-leaf1 ~]$ sudo aos_show_tech --platform eos
AOS show tech generated at /tmp/aos_show_tech_20200401_034102.tar.gz
[admin@l2-virtual-ext-003-leaf1 ~]$
```

3. Locate the file archive in the /tmp directory (aos_show_tech_20200401_034102.tar.gz for example), and copy the file to a local computer with the ability to upload via SCP.
4. *Upload the file* via the Support portal.

9.9.1.6 Show Tech for Cisco On-box Agents (CLI)

We recommend that you use the *web interface* to obtain on-box agent show tech files, but you have the option of using the Apstra server Linux CLI as described below.

1. SSH to the Cisco device.
2. From the device, run the `sudo aos_show_tech --platform nxos` command to generate and copy the show tech file to the /tmp directory.

Listing 26: Generating aos_show_tech for Cisco AOS Agent

```
l2-virtual-ext-004-leaf1# guestshell
[admin@guestshell ~]$ sudo aos_show_tech --platform nxos
AOS show tech generated at /tmp/aos_show_tech_20200401_034529.tar.gz
[admin@guestshell ~]$
```

3. Locate the file archive in the /tmp directory on the Cisco device (aos_show_tech_20200401_034529.tar.gz for example), and via SCP, copy the file to a local computer with the ability to upload.
4. *Upload the file* via the Support portal.

9.9.1.7 Show Tech for Cumulus On-box Agents (CLI)

We recommend that you use the *web interface* to obtain on-box agent show tech files, but you have the option of using the Apstra server Linux CLI as described below.

1. SSH to the Cumulus device.
2. From the device, run the `aos_show_tech --platform cumulus` command to generate and copy the show tech file to the `/tmp` directory.

Listing 27: Generating Show Tech for Cumulus Agents

```
admin@l2-virtual-ext-001-leaf1:mgmt-vrf:~$ sudo aos_show_tech --platform cumulus
AOS show tech generated at /tmp/aos_show_tech_20200401_034527.tar.gz
admin@l2-virtual-ext-001-leaf1:mgmt-vrf:~$
```

3. Locate the file archive in the `/tmp` directory on the Cumulus device (`aos_show_tech_20200401_034527.tar.gz` for example), and via SCP, copy the file to a local computer with the ability to upload.
4. *Upload the files* via the Support portal.

9.9.1.8 Show Tech for SONiC On-box Agents (CLI)

We recommend that you use the *web interface* to obtain on-box agent show tech files, but you have the option of using the Apstra server Linux CLI as described below.

1. SSH to the SONiC device.
2. From the device, run the `aos_show_tech --platform sonic` command to generate and copy the show tech file to the `/tmp` directory.

Listing 28: Generating Show Tech for SONiC Agents

```
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:~$
```

3. Locate the file archive in the `/tmp` directory on the device (`aos_show_tech_20200401_034527.tar.gz` for example), and via SCP, copy the file to a local computer with the ability to upload.
4. *Upload the files* via the Support portal.

FAVORITES

Adding favorites enables you to quickly access frequented pages.

Click to add any page to favorites

Click to see saved pages

Click a favorite to go to the page

Click to see all saved pages on user profile

User profile

| Username | admin |
|------------|---------------|
| First Name | admin |
| Last Name | admin |
| Email | not_set |
| Roles | administrator |

Favorites

Query: All 1-3 of 3

Page Size: 25

| Label | URL | Actions |
|---|--|---|
| AOS / Design / Interface Maps | /design/interface-maps | Edit Label Remove |
| AOS / Blueprints / L2 Virtual / Staged / Physical / Build | /blueprints/ccd840a9-c5df-45d0-b113-163e5831653d/staged/physical/build/systems | Edit Label Remove |
| AOS / Platform / Technical Support | /platform/support | Edit Label Remove |

To add a favorite - click the star in the upper-left corner of the page. Leave the default name, or rename it, then click **Add**. The outlined star becomes a shaded star.

To remove a favorite - click the shaded star on the saved page. The star becomes an outline.

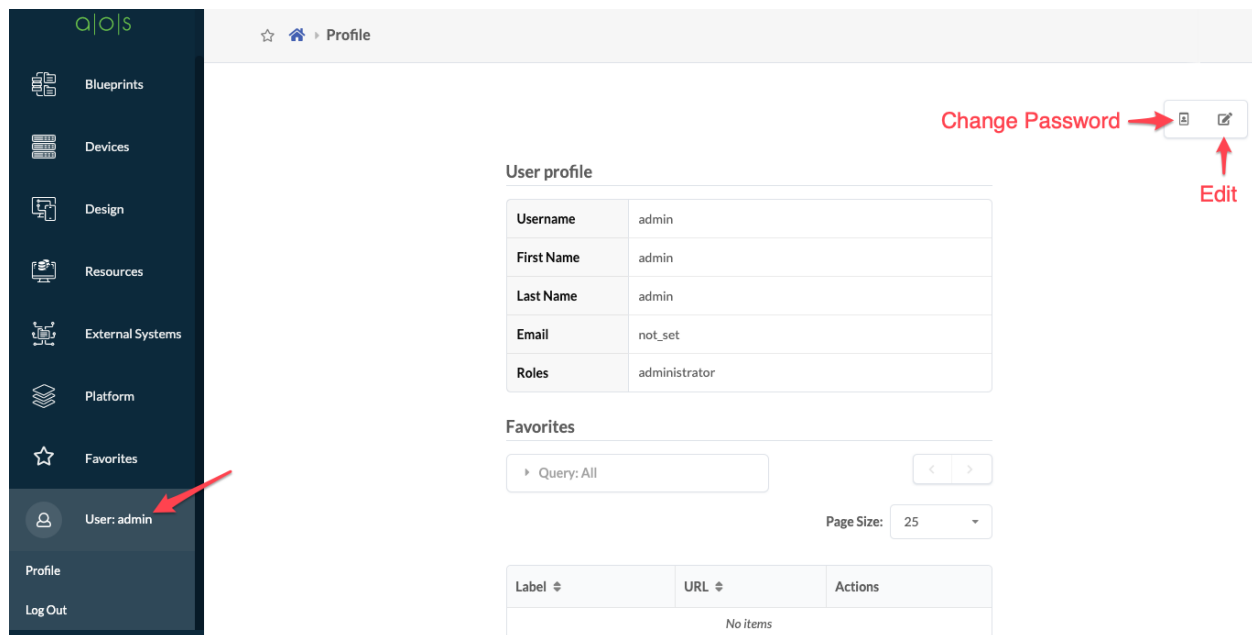
To access favorites - from the AOS web interface, click **Favorites** in the main menu.

To go directly to a favorite page - click the name in the **Favorites** menu. Up to five saved pages appear in the side

menu.

To see all favorites - click **Show more** to go to the user profile where you can link to all favorite pages and change their names.

As a user, you can manage your own password, name, email and favorites.



11.1 Profile

11.1.1 Changing Your Profile Password

1. From any page in the AOS web interface, click your username (bottom-left), then click **Profile** to see your profile page.
2. Click the **Change Password** button (top-right), enter your current password, then enter your new password, twice.
3. Click **Change Password** to update your password and return to your profile.

11.1.2 Changing Your Profile Name/Email

1. From any page in the AOS web interface, click your username (bottom-left), then click **Profile** to see your profile page.

2. Click the **Edit** button (top-right), then change your name and/or email, as applicable.
3. Click **Save** to update your details and return to your profile.

11.1.3 Managing Favorites

From any page in the AOS web interface, click your username (bottom-left), then click **Profile** to see your profile page including all the pages that have been saved as favorites.

- To access a favorite, click its link.
- To change the name of a link, click the **Edit label** button, change the name, then click **Update**.
- To remove a page from your favorites list, click the **Remove** button (trash can), then click **Delete**.

11.2 Log Out

From any page in the AOS web interface, click your username (bottom-left), then click **Log Out**.

12.1 Device Guides

12.1.1 AOS Device Configuration Lifecycle

The device lifecycle consists of various configuration stages and device states as described below.

Important: A good understanding of the AOS device configuration lifecycle (from the moment it is on-boarded to the moment the device is decommissioned) is essential. Apstra strongly recommends the following content to be understood in full.

12.1.1.1 Terminology

In AOS, the following terminology is used to identify the various configuration stages:

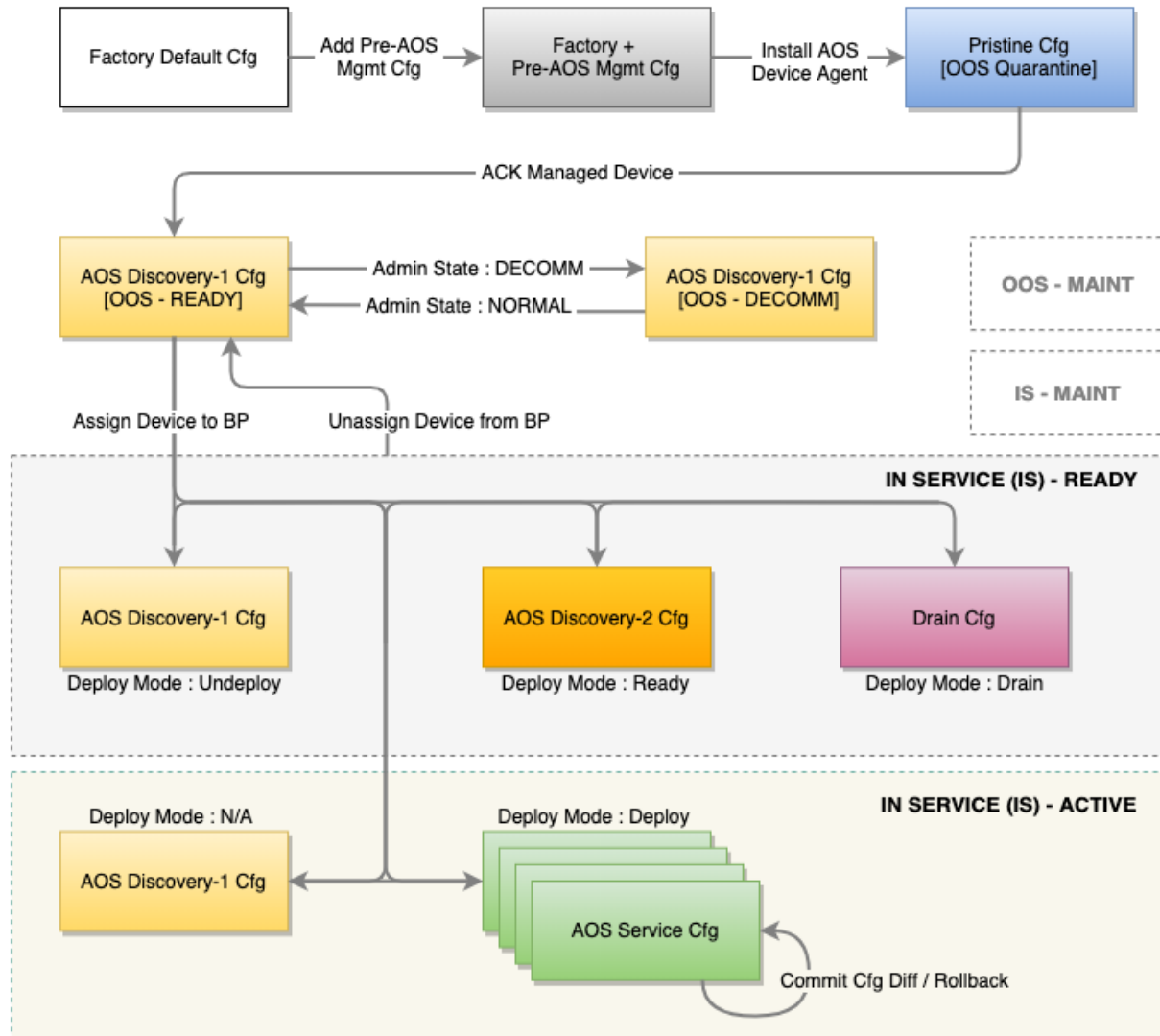
- **Pristine Config:** Consists of pre-existing config plus config added during agent installation. Normally, the pristine config does not change throughout the device's lifecycle.
- **Discovery 1 Config:** Initial basic configuration is added to the device when it is Acknowledged. This includes enabling of LLDP on all interfaces.
- **Discovery 2 Config:** Additional basic configuration is added to the device when it is assigned to a blueprint and deploy mode is "Ready". This includes device hostnames, interface descriptions and port speed / breakout config.
- **Service Config:** Additional configuration required by AOS is added when the device's deploy mode is set to "Deploy". When referring to Service Config this means the accumulation of Discovery 1 Discovery 2 and this additional config.
- **Rendered Config:** Complete AOS rendered configuration for the device, as per the AOS Reference Design.
- **Incremental Config:** Staged changes. The configuration that will be applied when the staged changes are committed.
- **Golden Config:** When AOS successfully commits a change this is followed with a new collection of the running config. This is called the Golden Config, and serves as Intent: Running configuration is continuously matched against this Golden config. Note that when there has been a Deploy failure, Golden Config is unset.

12.1.1.2 Configuration stages: Overview

The following table describes the various config events and resulting config and device states and states of a device as it exists within a blueprint:

| Event | Resulting Device Configuration | Resulting AOS Managed Device State | AOS blueprint Deployment Mode |
|--|---|------------------------------------|-------------------------------|
| New Device | Factory Default Configuration | N/A | Not Assigned |
| Add Pre-AOS [mgmt] Configuration to device | Factory + Pre-AOS | N/A | Not Assigned |
| Install AOS Device System Agent | Pristine Config: Factory + Pre-AOS + 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) | Discovery 2: Discovery 1, plus various basic config | IS-READY | Ready |
| Deploy Device | Service Config: Discovery 2, plus full AOS 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 | AOS 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.

Warning: Any configuration present on the device before agent installation will become part of the Pristine Config and therefore become part of the devices' entire configuration lifecycle. If a correction is required, this will be service impacting. See below for more detail on Pristine Config.

12.1.1.3 Configuration stages: Detail

New Device (Factory Default)

The lifecycle of a device begins with the **factory default** configuration stage.

Add Pre-AOS Config (User-required)

Certain base configuration, such as for *connectivity and AOS Agent installation*, must be included in the entire config lifecycle. This **User-required** config can be bootstrapped with *Apstra ZTP*. The minimum config can also be added with scripts or other methods.

Important: Adding configuration to the device at the pre-AOS stage should be limited to changes required for connectivity or AOS Agent installation, or any config that is known to be required **throughout the device's lifecycle**, for example Banners or NTP / SNMP / syslog server IP addresses. Configuration required that is not rendered by AOS can be added using *Configlets*.

Install Agent (Pristine)

When an AOS device agent is installed on a device (or in the case of an offbox agent, installed server-side) the device connects and registers to AOS in the **Quarantined** state. A partial config is applied and any “Pre-AOS Config” that has already been added will become part of the **Pristine configuration**. This pristine configuration is the basis for all subsequent device configuration.

Warning: For Cumulus Linux, enabling configuration services will replace all config in `/etc/network/interfaces`.

Acknowledge Device (Discovery 1 / Ready)

Acknowledging a device puts it in the **Ready** state and signals the intent to have AOS manage the device. In this **Discovery 1** stage minimal base configuration essential to AOS Agent operation is added to the pristine config. Discovery 1 applies a *complete* configuration (a.k.a. “Full config push”), overwriting all existing configuration to ensure config integrity.

- All interfaces are rendered with interface speeds for the assigned Device Profile.
- All interfaces are `no shutdown` to allow you to view LLDP neighbor information.
- All interfaces are moved to L3 mode (default) to prevent the device from participating in the fabric.
- Cumulus: DHCP relay configuration is removed and wiped.
- Cumulus: Quagga configuration is removed and wiped.
- Cumulus: Hostname is learned when agent first starts up, or re-used if agent ran previously. Hostnames can also be learned via DHCP before the device is acknowledged.

Important: Devices that have been acknowledged cannot simply be deleted - as there is still an active agent on the device talking to the AOS server, they would re-appear within seconds. For details on removing a device from AOS, see the *device decommissioning guide*.

Assign Device (Discovery 2 / Ready)

Assigning a device to a blueprint and setting its Deploy Mode to **Ready** puts it in the **Discovery 2** configuration stage. The device has been staged, but not yet committed (deployed) to the active blueprint. Discovery 2 applies a

complete configuration (aka. “Full config push”) to ensure config integrity. The discovery2 configuration brings up network interfaces and configures interface descriptions and validates telemetry, such as LLDP, to ensure it is properly wired and configured. This configuration is non-disruptive to other services in the fabric. Links are up, but they are configured in L3-mode to prevent STP/L2 operations.

- Hostname is configured per blueprint intent.
- All interface descriptions are changed per blueprint intent.
- Interfaces are rendered with blueprint interface speeds.
- No routing or BGP is configured.
- No L3 information is configured on interfaces.
- Fabric MTU is modified for spines to 9050 bytes.
- Cumulus: Quagga configuration is still empty (`etc/quagga/Quagga.conf`), but the file is created if it does not exist
- Cumulus: DHCP configuration is not modified

Deploy Device (Rendered / Active)

Warning: The first time a device is assigned, the Deploy Mode is set to “Deployed” and the blueprint is committed, AOS triggers a **full config push** for the device, effectively overwriting the complete running config with the Pristine Configuration then adding the full rendered AOS Configuration. Any config that is not part of the AOS rendered config is discarded.

When a device is committed, it becomes **Active**, and AOS 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 AOS server per Jinja) to ensure configuration integrity. This is the full end-state of AOS. A full configuration has been pushed, all interfaces are running, and routing within IP fabric is configured. Full configuration rendering, intent-based telemetry, and standard service operations occur here.

- Hostname is configured per blueprint intent.
- All interface descriptions are changed per blueprint intent.
- Interfaces are rendered with blueprint interface speeds.
- Interface VLANs, LAGS, MLAG, VXLAN, etc are managed.
- All L3 information is rendered.
- BGP configuration is fully rendered for all BGP peering information.
- DHCP configuration is configured for any required DHCP relay agents.
- Cumulus: FRR configuration is fully rendered (`/etc/frr/frr.conf`), including all BGP peering information.
- The device is added to the graph database.

After the full configuration is successfully deployed to the Device AOS will take a snapshot of the Device Configuration (e.g. `show running-config`) and store it as the **Golden Configuration**.

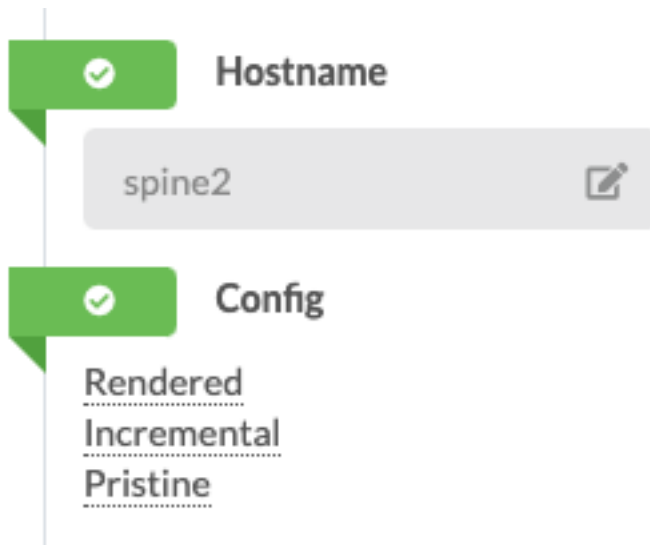
Warning: Adding extra configuration to AOS at this time will result in a configuration deviation anomaly, a difference between the current Device Configuration and the stored **Golden Configuration**. AOS will fail subsequent deployment tasks until the deviation is resolved. Correct the anomaly to proceed.

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

A running configuration can be modified in multiple ways. To modify a config that is not part of the reference design, use *Configlets*.

Stage Device Update (Incremental / Active)

Staging changes to a running blueprint creates an **Incremental** configuration. You can preview the incremental config (including Cumulus as of AOS version 3.3.0) before committing the changes to the network. From the staged blueprint, select the device, and click **Incremental** in the **Config** section (lower-right). (Config previews for **Rendered** and **Pristine** (as of AOS version 3.2.1) are also accessible from here). When the changes have been committed, the Incremental Config is empty.



Commit Device Again (Rendered-Updated / Active)

Whenever a change is committed to a blueprint that affects the device's configuration, a partial config updates the rendered config.

12.1.1.4 Configuration Deviations

After each **successful** config deploy AOS collects the running config and stores it internally as the **Golden** configuration. Intent is the cornerstone of AOS. 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 remember:

- Golden Config is updated upon each *successful* configuration deployment

- When config deployment fails for some reason, Golden config is not set. This means both a config deviation and deployment failure anomaly is raised.
- Running configuration telemetry is continuously collected and matched against the Golden config. Any difference results in a Deviation anomaly.
- Configuration Anomalies can be ‘suppressed’ using the “Accept Changes feature”. This does **NOT** mean the change is added to Golden config or Intent.

See *Configuration Deviation* for details.

12.1.1.5 Device Offline (Unavailable)

A managed device (one that has been acknowledged) that is not connected to the AOS 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.

12.1.1.6 Manually Applying 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.

Important: A full configuration push should be done 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.

12.1.1.7 Deploy Modes

Managed devices in blueprints can be in one of several modes. See *Changing Deploy Mode on One Device* for steps for changing deploy modes.

Not Set The initial state of a device. The device is not active in the fabric.

Deploy A deployed device is an active device in the fabric.

Ready When a device is assigned to a blueprint it is in ready mode; discovery 2 configuration is added (hostnames, interface descriptions, port speed / breakout configuration). It is not yet active in the fabric. Changing from deploy to ready removes AOS-rendered configuration.

Drain Draining a device for physical maintenance enables it to be taken out of service gracefully without impacting existing TCP flows. Depending on the device being drained, AOS 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 Cumulus, all bond interfaces towards L2 servers in the blueprint are deleted and member ports are removed from `/etc/network/interfaces`. As a result LAG/MLAG anomalies are generated.

For Network L3 Switches

Use 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 maintenance 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. See the [device draining guide](#) for more information.

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.

12.1.2 Adding Device

Please have an understanding of the [AOS device configuration lifecycle](#) before working with devices.

Note: If you are adding a device to replace a decommissioned faulty one (for RMA) and you plan to use the same management IP address, please make sure that the original device has been [decommissioned](#) before the new device is assigned, as AOS expects each device to have a unique management IP address.

1. Install [device agent\(s\)](#) onto the device(s); then they will appear as managed devices in the out-of-service quarantined state.
2. [Acknowledge device\(s\)](#) to bring them under AOS management in the out-of-service ready state.
3. After [creating the blueprint](#), assign [interface map\(s\)](#) (device profiles) to leafs and spines.
4. Assign [device system ID\(s\)](#) to the devices and confirm that deploy mode(s) are set to **Deploy** to bring them in-service in the ready state.
5. [Commit](#) the device assignment(s) to the blueprint to bring them in-service in the active state.

☆
🏠
>
Devices
>
Managed Devices

+ Stock Device

Query: All
1-5 of 5

Page Size: 25

| 0 selected | Device Key ↕ | Device Profile ↕ | Operation Mode ↕ | Management IP ↕ | AOS Version ↕ | Hostname ↕ | Location ↕ | OS ↕ | Acknowledged? ↕ | State ↕ | Blueprint ↕ | Comms ↕ |
|--------------------------|--------------|------------------|------------------|-----------------|------------------|-----------------------|------------|----------------|-----------------|-----------|-------------|---------|
| <input type="checkbox"/> | 525400F58B8D | Cumulus VX | FULL CONTROL | 172.20.31.11 | AOS_3.3.0_OB.730 | spine1 | | Cumulus 3.7.11 | ✓ | IS-ACTIVE | pod1 | 🟢 |
| <input type="checkbox"/> | 5254001D9962 | Cumulus VX | FULL CONTROL | 172.20.31.12 | AOS_3.3.0_OB.730 | spine2 | | Cumulus 3.7.11 | ✓ | IS-ACTIVE | pod1 | 🟢 |
| <input type="checkbox"/> | 525400A8DDEC | Cumulus VX | FULL CONTROL | 172.20.31.13 | AOS_3.3.0_OB.730 | evpn-mlag-001-leaf1 | | Cumulus 3.7.11 | ✓ | IS-ACTIVE | pod1 | 🟢 |
| <input type="checkbox"/> | 5054007B2554 | Arista vEOS | FULL CONTROL | 172.20.31.14 | AOS_3.3.0_OB.730 | evpn-single-001-leaf1 | | EOS 4.22.3M | ✓ | IS-ACTIVE | pod1 | 🟢 |
| <input type="checkbox"/> | 525400FA2152 | Cumulus VX | FULL CONTROL | 172.20.31.15 | AOS_3.3.0_OB.730 | evpn-mlag-001-leaf2 | | Cumulus 3.7.11 | ✓ | IS-ACTIVE | pod1 | 🟢 |

12.1.3 Deploying Device

As explained in *AOS device configuration lifecycle*, **Deploy** is one of several modes any device can have. This topic explains in more detail what happens when a device mode is set to Deploy.

Generally, by “Deploy” we mean changing the device mode to **Deploy** AND committing that change to the blueprint. Also note it is entirely possible (and in fact common practice) to have a committed blueprint with no deployed devices. Users can deploy devices as and when required, be it in batches, one by one, or all in one go.

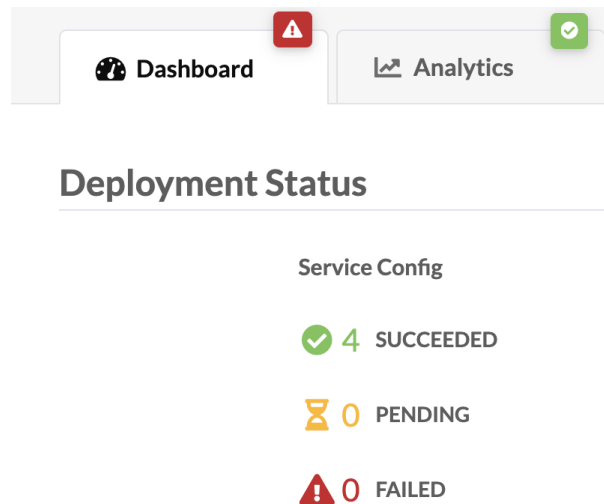
The implications of deploying a device can be summarized into two main categories:

1. The **full** AOS rendered configuration is applied to the device.
2. Anomalies associated to this device are now raised on the dashboard.

Immediately upon successful deployment, AOS collects the full running configuration. AOS continuously collects and matches the running config against this “Golden Config”.

Protocol related anomalies like BGP or LLDP are only raised if devices at both ends are deployed.

A device’s deployment status can be seen on the dashboard and should become visible as *SUCCEEDED* quickly after the deploy mode is committed to the blueprint:



AOS may raise several anomalies immediately after deploying. This is because Intent now dictates certain states which AOS has not been able to verify yet. One could say AOS ‘assumes’ an anomaly until it is verified to be ok. At this point, telemetry data is being checked against Intent, and these anomalies will clear again if there is a match. This can take some time in some cases, eg. BGP sessions that come up and routes that are being advertised.

12.1.3.1 Vendor specifics

Implications of deploying a device can vary across different vendors. Below are known NOS characteristics that result in differences in the way anomalies are raised by AOS.

Juniper Junos

1. “show interface” commands do not list interfaces on ports that do not have a transceiver plugged in. This means AOS can not raise any *Interface Down* anomalies for these interfaces. Such interfaces can be recognized using the ‘show virtual-chassis vc-port’, and will have a status of ‘Absent’.

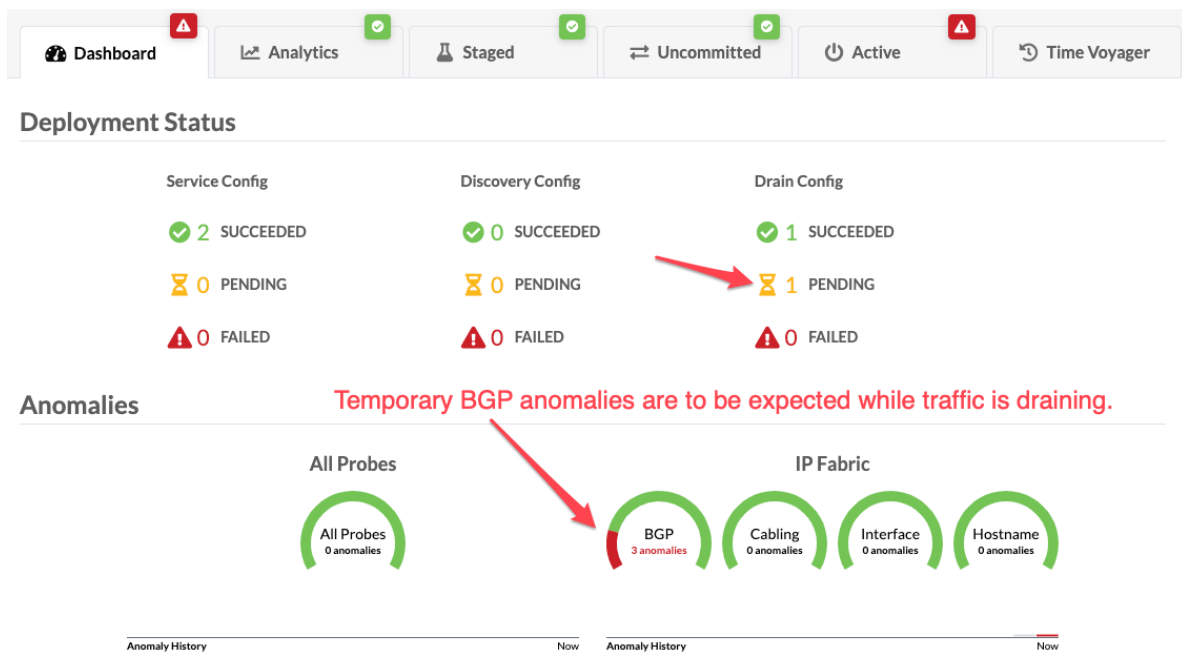
2. If a Virtual Network endpoint is configured on a leaf interface, AOS expects an EVPN type 3 route for that interface. If this interface is down, Junos does not advertise the RT-3, resulting in a “Missing Route” anomaly in AOS. If this anomaly is undesirable it is recommended to remove the interface from the VN until the interface is up.

12.1.4 Draining Device Traffic

New in version 3.3.0: AOS supports draining traffic from ESI leaf pairs.

Devices can be gracefully taken out-of-service for maintenance (or decommissioning) by draining them of traffic.

1. *Change the deploy mode* on the device to **Drain**.
2. *Commit* the change. While TCP sessions drain (which could take some time, especially for EVPN blueprints) BGP anomalies are expected. When drain configuration is complete, the temporary anomalies will be resolved.



12.1.4.1 Monitoring Traffic Draining

While a device is draining you can monitor its progress from the AOS web interface from various locations:

- From the **Deployment Status** section of the blueprint dashboard (Drain Config).

Deployment Status

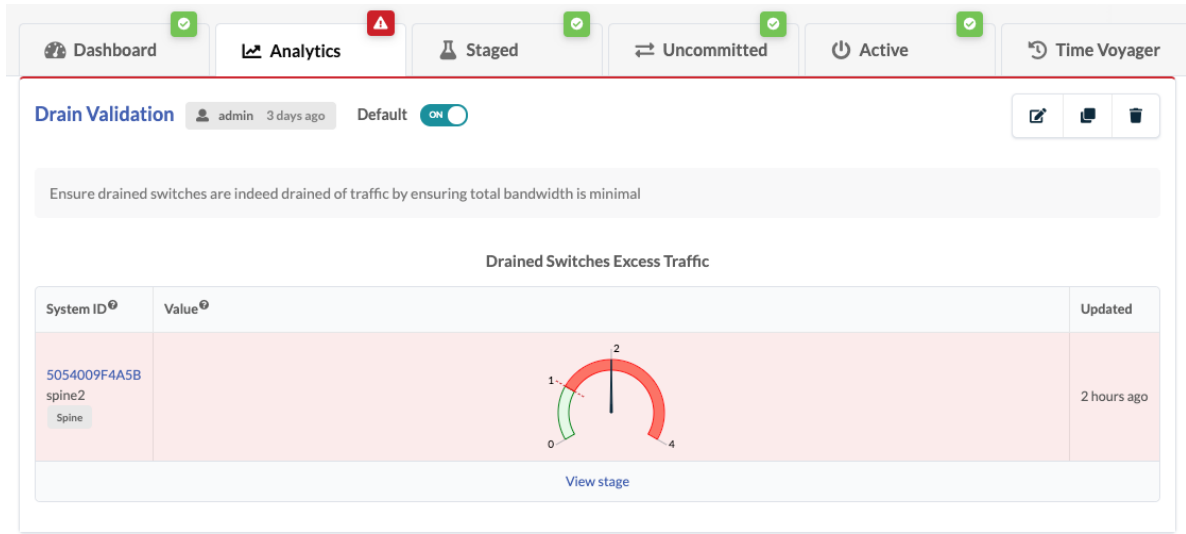
| Service Config | Discovery Config | Drain Config |
|----------------|------------------|---------------|
| ✓ 3 SUCCEEDED | ✓ 0 SUCCEEDED | ✓ 1 SUCCEEDED |
| ⌚ 0 PENDING | ⌚ 0 PENDING | ⌚ 0 PENDING |
| ⚠ 0 FAILED | ⚠ 0 FAILED | ⚠ 0 FAILED |

- From **Active / Physical** in the **Status** panel (Deployment Status: Drain).

The screenshot shows the Apstra Status panel with the **Physical** view selected. The top navigation bar includes **Dashboard**, **Analytics**, **Staged**, **Uncommitted**, **Active**, and **Time Voyager**. The main navigation bar includes **Physical**, **Virtual**, **Policies**, **Catalog**, **Settings**, **Query**, **Anomalies**, and **Root Causes**. The **Nodes: All** and **Links: All** filters are set. The **Topology** section shows **Nodes**, **Links**, and **Racks** tabs, with **Grouped** selected. The **Layer** is set to **Anomalies: All Services**. The **Selected Rack** and **Selected Node** are both set to **All**. The **Show Servers?** checkbox is unchecked. The **External Systems** section shows **External Routers** with a status indicator. The **pod1** section shows a list of nodes: **spines**, **I2_virtual_ext_001**, **I2_virtual_ext_002**, **I2_virtual_ext_003**, and **I2_virtual_ext_004**, each with a status indicator. The **Status** panel on the right shows a list of anomalies with their counts and names. A red arrow points to the **Deployment Status: Drain** entry, which has a count of 1/0/0.

| Count | Anomaly Name |
|---------|------------------------------|
| 0 | Anomalies: All Services |
| 0 | Anomalies: BGP |
| 0 | Anomalies: Cabling |
| 0 | Anomalies: Config |
| 0 | Anomalies: Hostname |
| 0 | Anomalies: Interface |
| 0 | Anomalies: LAG |
| 0 | Anomalies: Liveness |
| 0 | Anomalies: MLAG |
| 0 | Anomalies: Probes |
| 0 | Anomalies: Route |
| 3/0/1/2 | Deploy Mode |
| 0/0/0 | Deployment Status: Discovery |
| 1/0/0 | Deployment Status: Drain |
| 3/0/0 | Deployment Status: Service |
| 0 | Traffic Heat |

- From **Analytics / Dashboards** when the predefined **Drain Validation** dashboard is *instantiated*. (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 still in the process of draining.



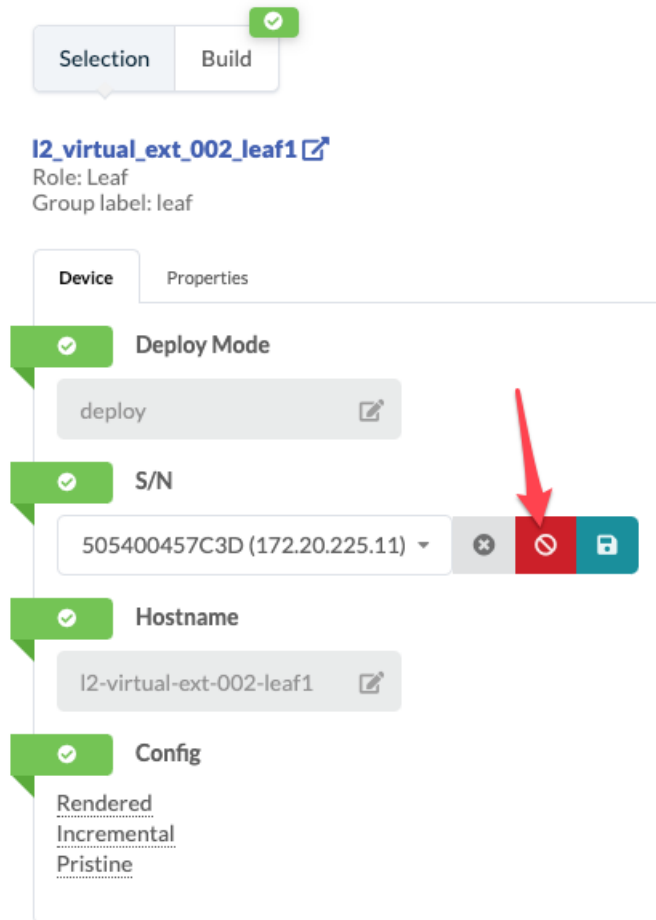
After performing device maintenance, simply change the deploy mode back to **Deploy** and *commit* the change to bring the device back into active service.

12.1.5 Removing Device

After installing agents on devices and acknowledging them, they are managed by AOS and can be assigned to blueprints. You can unassign devices from blueprints and remove them from AOS management, as needed. See below for details.

12.1.5.1 Removing Device from Blueprint

1. *Remove the device assignment* from the blueprint.



2. *Commit* the change to remove the assignment. The device has been disassociated from the blueprint. It is still managed by AOS and it is ready and available to be assigned to the same blueprint or another one.

12.1.5.2 Removing Device from AOS Management

Devices that are no longer managed by AOS are disconnected from the AOS server and removed from the AOS database.

Important: Please follow the order of operations for successful device removal.

1. *Remove the device assignment* from the blueprint (if it had been previously assigned).
2. *Commit* the change to remove the device assignment from the blueprint, as applicable.
3. Set the *admin state* of the device to **DECOMM**.
4. *Uninstall the device agent*.

Note: Even after decommissioning and uninstalling the agent, the device still includes some interface configs that were added by AOS during discovery, including no shutdown, interface speeds for the device profile, and

L3 mode config (e.g. no switchport), in addition to the original pristine config.

5. *Delete the device agent.*
 6. *Delete the device* from the managed devices list.
-

Note: If you are decommissioning a faulty device and replacing it with another one for RMA using the same management IP address, please make sure that the original device has been decommissioned before the new device is assigned, as AOS expects each device to have a unique management IP address.

12.1.5.3 Replacing Device

If you are removing a device that will be replaced, follow the steps above for removing a device from AOS management, then follow the steps for *adding a device*.

12.1.6 Device AAA support

AOS supports Device AAA (authentication, authorization and accounting) framework including RADIUS and TACACS+ on certain platforms. This chapter details the minimum requirements for correct AOS AAA implementations.

There are two methods to apply AAA configurations:

1. **Configlets** This is the recommended and preferred method. This method means the configuration is added to a configlet which is then imported into a blueprint. Drawback is that local credentials need to be available from AOS so that the device can be added and the configlet can be applied. See *Configlets and Property Sets* for details.
2. **User Required configuration** It is possible to ensure pre-existing configuration is retained when a device is brought under AOS management. This is done by ensuring the config is in place before the device is brought under AOS management. See *Configuration Lifecycle*.

Note: The use of AAA framework such as TACACS+ and RADIUS is optional, and correct implementation is the responsibility of the end user.

Warning: When using AAA framework we recommend adding a local AOS user to the devices: in the event AAA authentication or authorization fails when AOS performs a full configuration push, manual recovery (config push) is required.

Important: When upgrading AOS, Device Agent, or NOS, user **must** delete Device AAA/TACACS configlets from the Blueprint before the upgrades. User can re-apply them after the upgrades.

12.1.6.1 Supported Platforms

AOS supports AAA implementations on the following platform:

Cisco NXOS

The below example NXOS configuration has been tested to work correctly with AOS. 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
```

Important: For NXOS, Apstra has observed cases in which a remote user was erratically removed from the device, causing authentication and authorization failures. AOS requires the user (role 'network-admin') to exist on the device in order to manage the device. If not, AOS functions like Agent installation, Telemetry collection and device configuration may fail. The only currently known workaround is to use local authentication.

Arista EOS

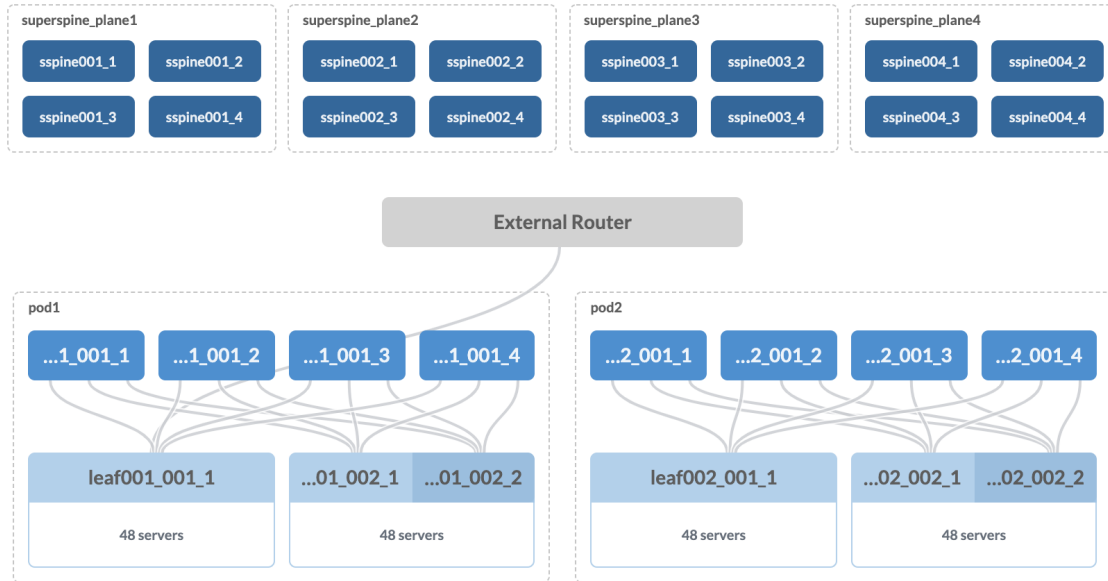
AOS supports AAA configuration on EOS devices.

Important: For EOS, Apstra has observed cases in which AOS Device System Agent upgrade failed while copying files from the AOS server to the device with TACACS+ AAA configured. This commonly happens if TACACS+ is used to use a custom password prompt. To prevent this type of failure, the user should temporarily disable all TACACS+ AAA to where device authentication uses an admin-level username and password for any AOS Device System Agent operations including upgrade.

12.2 5-stage Clos Architecture

12.2.1 5-Stage Clos Overview

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 superspines. Each 5-stage topology consists of one or more planes. Each plane consists of one or more superspines. See below for an example.



Careful planning and consideration are required to build large 5-stage Clos networks. Please take into account the limitations listed below. For assistance with designing and validating 5-stage topologies, please contact [Technical Support](#).

12.2.1.1 5-Stage Clos Limitations

- You cannot change a 3-stage topology to a 5-stage topology.
- You can deploy Arista EOS, Cisco NX-OS, Cumulus and Juniper Junos devices as superspines. You cannot deploy SONiC devices as superspines.
- 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.
- Using segmentation / *group-based policies*.
- Mixing pods with L2 and L3 server connections is not supported.
- Predefined IBA probes on 5-stage topologies have best effort support.
- Root Cause Analysis is not supported.
- IPv6 / IPv4 support:
 - IPv6 in the underlay is not supported.
 - IPv6 for applications is not supported.
 - The entire fabric across all pods must be either all IPv4, all IPv6 or all dual-stack
- Unsupported external connectivity implementations:
 - One *External router* connecting to multiple pods
 - Mixing L2 and L3 external connectivity
 - EVPN with external routers on superspines
 - External routers on spines and leafs in the same pod
- Unsupported blueprint modifications:
 - Add or remove superspine devices

- Add or remove superspine planes
- Add or remove pods (Note it is possible to replace all racks within a pod)

12.2.1.2 5-Stage Clos and EVPN

EVPN networks can be extended across multiple pods within the same blueprint (as of version 3.2) to provide the following added value:

- Scaling: provide any-to-any connectivity for applications distributed across multiple pods.
- Workloads Redistribution: Load-balance applications by migrating a group of applications from one pod to another pod while preserving application IP and MAC addresses.
- Maintenance: Perform pod maintenance by migrating all applications from one pod to another, while preserving the application IP and MAC addresses.
- Active / Standby applications across sites / pods: Deploy A/S applications across multiple pods to provide high availability at pod level, or as part of application migration tasks.
- Facilitate external connectivity for a virtual network from a remote pod without external connectivity.

5-stage Clos networks support the Junos QFX series of switches (as of version 3.3.0). You can use the ESI redundancy protocol (as of version 3.3.0), 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](#).

Just like in other Apstra-managed networks, required configuration is rendered to bring up multi-pod networks, and with proprietary *Intent-based Networking* technology the networks are validated to ensure they operate as designed.

You can create cross-pod virtual networks in the same manner as for 3-stage networks. See [Virtual Networks](#) for a comprehensive guide.

12.2.2 Creating 5-Stage Clos Network

Creating a 5-stage Clos network follows the same workflow as for [3-stage Clos networks](#), 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](#) (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](#) (Design > Interface Maps) that map the logical devices to the correct [device profiles](#); 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](#), 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*, *IPv4 addresses*, *IPv6 addresses*) needed in the network.
6. *Create a blueprint* using the pod-based template that you created in the previous step.
7. Build the 5-stage Clos network in the same manner as *building a 3-stage Clos network*.

12.2.3 Modifying 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. Typically, topology changes are achieved with *rack changes*.

12.3 Integrations

12.3.1 VMware vSphere Integration

12.3.1.1 VMware vSphere Integration Overview

With AOS vCenter integration, VM visibility is provided in virtualized environments. This feature helps to troubleshoot any VM connectivity issues. Any inconsistencies between virtual network settings (VMware Port Groups) and physical networks (AOS Virtual Networks) that might affect VM connectivity are flagged.

To do this, AOS identifies the ESX/ESXi hosts and thereby the VMs connected to AOS-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.

AOS also connects to vCenter to collect information about VMs, ESX/ESXi hosts, port groups and VDS. This collection is done as an AOS extensible telemetry collector. The collector runs in an off-box agent and uses pyVmomi to connect to vCenter. On first connect, it downloads all of the necessary information and thereafter polls vCenter every 60 seconds for new updates. The collector updates the discovered data into the AOS Graph Datastore allowing VM queries and alerts to be raised on physical/virtual network mismatch.

Supported Versions

As of AOS version 3.3.0 VMware vSphere/vCenter integration is currently available for the following versions of VMware:

- vCenter Server/vSphere 6.7
- vCenter Server/vSphere 6.5

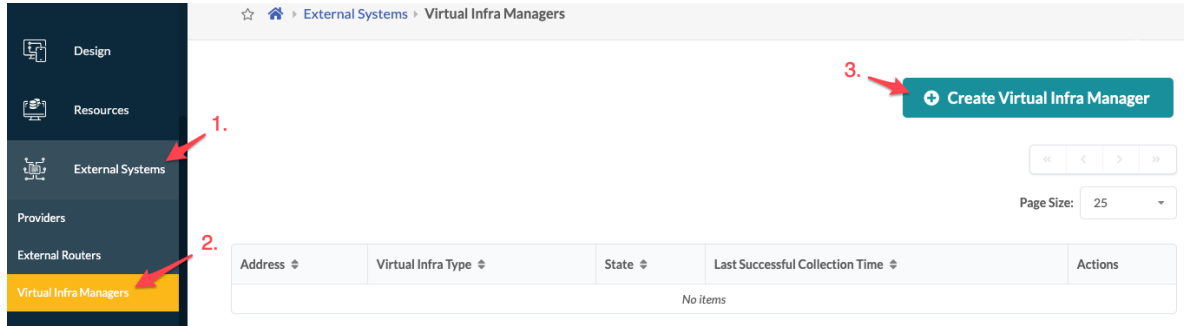
Limitations

- vCenter integration does not support DVS port group with VLAN type Trunking

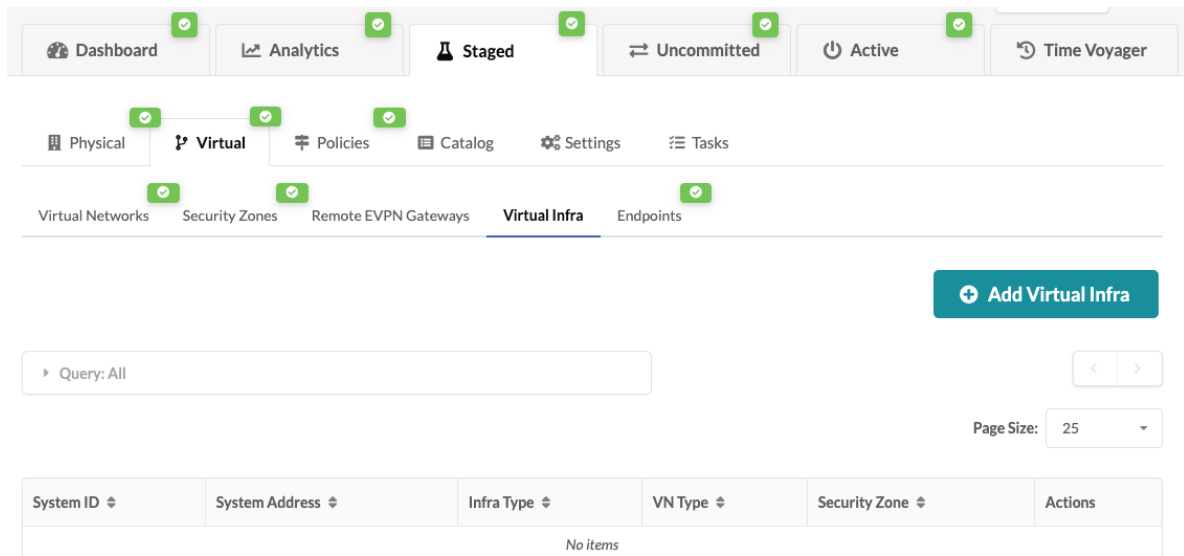
12.3.1.2 Enabling vSphere Integration

To enable vSphere Integration, only **Read** permission is required.

1. From the AOS web interface, navigate to **External Systems / Virtual Infra Managers**, then click **Create Virtual Infra Manager**.



2. Enter the vCenter IP address (or DNS name), select **VMware vCenter Server**, then enter a username and password.
3. Click **Create** to launch an offbox container running vCenter. While the container is connecting, it is in a **DISCONNECTED** state. When the container successfully connects, the state changes to **CONNECTED**.
4. When vCenter is connected, from the blueprint, navigate to **Staged / Virtual / Virtual Infra**, then click **Add Virtual Infra**.



5. Select the vCenter Server from the **Virtual Infra Manager** drop-down list, then click **Create** to stage the change.
6. When you are ready to deploy, commit the changes from the **Uncommitted** tab.

12.3.1.3 VM Visibility

When virtual infra is being managed by AOS, you can query their VMs by name. From the blueprint, navigate to **Active / Query / VMs** and enter search criteria. VMs include the following details:

Hosted On The ESX host that the VM is on

VM IP The IP address as reported by vCenter after installation of VM tools. If the IP address is not available this field is empty. AOS displays the VM IP address if the IP address is available on installation VM tools on the VM.

Leaf:Interface The leaf and the interface ESX host is connected to

Port Group Name:VLAN ID The VNIC's portgroup and the VLAN ID associated with the portgroup

MAC Addresses MAC address of the VNIC

Virtual Infra Address IP address of the vCenter the VM is part of

12.3.1.4 Validating Virtual Infra Integration

You can validate virtual infra with intent-based analytics. 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*

Hypervisor MTU Threshold Check Probe

Hypervisor and Fabric LAG Config Mismatch Probe

Hypervisor and Fabric VLAN Config Mismatch Probe

Hypervisor Missing LLDP Config Probe

VMs without Fabric Configured VLANs Probe

Virtual Infra Redundancy Check Dashboard *Hypervisor Redundancy Checks Probe*

For more information, see *Analytics Dashboard* and *Instantiating Predefined Probe*.

12.3.1.5 vCenter Policy-Based Auto-Remediation

vCenter Policy-Based Auto-Remediation Overview

Beginning in AOS 3.1, AOS provides automatic remediation of virtual network anomalies without user intervention (as of AOS version 3.1). This helps by reducing operational cost where a network operator does not 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 an AOS Blueprint. Remediation of anomalies is done in accordance with this policy.

AOS provides a policy-based auto-remediation approach to automatically notify you if there is a mismatch between vSphere DPG (VMware Port Groups) and AOS VN in a particular Blueprint, or if there is a VLAN mismatch between Virtual Infra and AOS Fabric, or if there is a mismatch in LAG configuration on hypervisors and the corresponding leaf ports. AOS provides automatic guided remediation of such anomalies.

Enabling VXLAN vCenter Policy-Based Auto-Remediation

1. From the blueprint, navigate to **Staged / Virtual / Virtual Infra**, then 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 AOS fabric.

- **VLAN** (rack-local) Chooosed VLAN if the VLAN footprint on local vSphere does not extend to other ToR leafs in a fabric.

5. Select the **Security zone**: If VN type is **rack-local** only the default security zone is allowed.

6. Click **Create**.

After enabling the VLAN remediation policy as inter-rack, AOS searches for matching local VLANs in all ToRs connecting any member host (i.e hypervisor) 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 not found, a new inter-rack VN is created in the specified security zone.

Constraints and Validations

Please find some of the constraints and validations that take place before the remediation happens:

- When remediation policy is set to VLAN, that is rack-local, security zone can only be the default one.
- If VLAN ID for virtual network spanning multiple hypervisors is the same, then it is assumed to be a single layer 2 broadcast domain. For such scenarios, the VLAN remediation policy must be set to VXLAN as for any missing VLAN anomalies it is checked on all the ToR leafs connected to different hypervisors having virtual network with the same VLAN ID. If this is mistakenly chosen as VLAN type, validation errors are thrown.
- AOS throw errors if different types of remediation policies (For example, if one is VXLAN type and other is VLAN type) are found attached to different Virtual Infrass (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.

vCenter Policy-Based Auto-Remediation Features

AOS 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 AOS fabric.
- AOS does not allow the deletion of the security zone that is being referenced in remediation policy.

Note: For an EVPN enabled fabric, it is recommended to have VN type as inter-rack or VXLAN in a specific security zone.

Remediation Steps - Resolve Probe Anomalies

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. AOS 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. AOS 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.

12.3.1.6 Disabling 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. Navigate to the **Uncommitted** tab and commit the deletion.
3. From the **External Systems** menu (left-side), navigate to **Virtual Infra Managers** and click the **Delete** button for the virtual infra to disable.

12.3.2 VMware NSX-T Integration

12.3.2.1 Overview

You can integrate NSX-T with Apstra software (as of version 3.1) to help deploy fabric VLANs that are needed for deploying NSX-T in the data center or for providing connectivity between NSX-T overlay networks and fabric underlay networks. You can accelerate NSX-T deployments by making sure the fabric is ready in terms of LAG, MTU and VLAN configuration as per NSX-T transport node requirements. This feature also helps network operators with fabric visibility in terms of seeing all the NSX-T VMs, VM ports, and physical gateway ports. NSX-T integration helps identify issues on the fabric and on the virtual infrastructure. It eliminates manual config validation tasks between the NSX-T Nodes side and the ToR switches.

Supported Versions

As of version 3.3.0 VMware NSX-T integration is currently available for the VMware NSX-T Data Center 2.5 version.

Limitations

- NSX-T Edge integration is currently supported on bare metal deployments only. NSX-T Edge is not supported on VMs.
- Having more than one NSX-T virtual infra in a blueprint is not supported. We recommend only one virtual infra per blueprint.
- NSX-T integration does not support DVS port group with VLAN-type trunking.

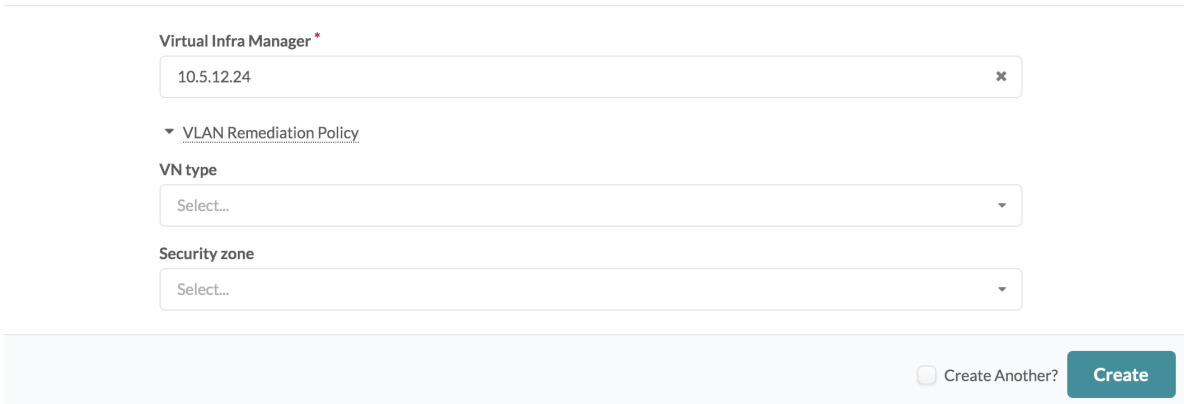
12.3.2.2 Enabling NSX-T Integration

We recommend that you *create a user profile* dedicated to managing NSX-T integration activities.

1. From the web interface, navigate to **External Systems > Virtual Infra Managers** and click **Create Virtual Infra Manager**.
2. In the dialog that appears enter the NSX-T manager IP address (or DNS name), select the virtual infra type **VMware NSX-T Manager** and enter a username and password.
3. Click **Create** to create the virtual infra manager and return to the list view. When the connection is successful, the connection state changes from **DISCONNECTED** to **CONNECTED**.
4. From the blueprint, navigate to **Staged > Virtual > Virtual Infra** and click **Add Virtual Infra**.

5. Select the virtual infra manager from the 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.

Add Virtual Infra Manager



Virtual Infra Manager *

10.5.12.24

▼ VLAN Remediation Policy

VN type

Select...

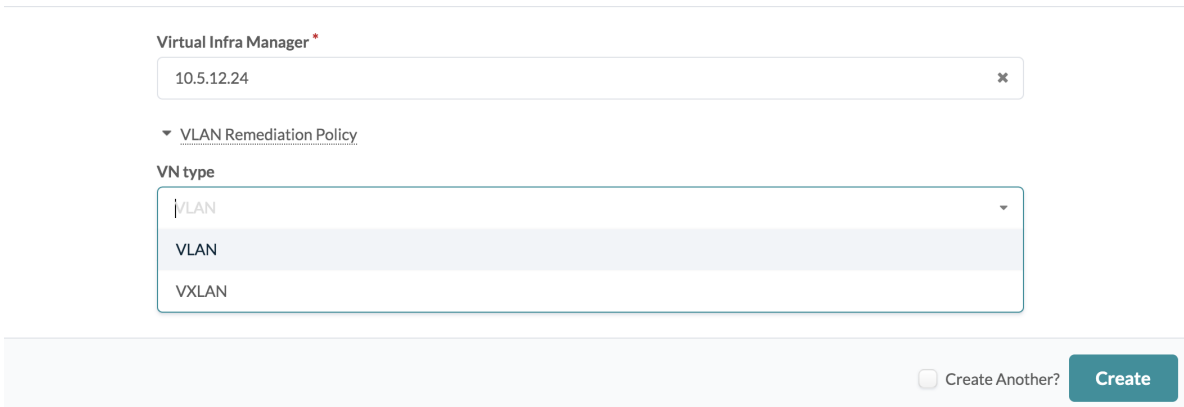
Security zone

Select...

☐ Create Another? **Create**

6. Select the VN type and routing zone.
 - If VLAN (rack-local) is selected you can't change the default routing zone.
 - If VXLAN (inter-rack - when VN extends to different ToRs in the fabric) is selected you can select a different routing zone.

Add Virtual Infra Manager



Virtual Infra Manager *

10.5.12.24

▼ VLAN Remediation Policy

VN type

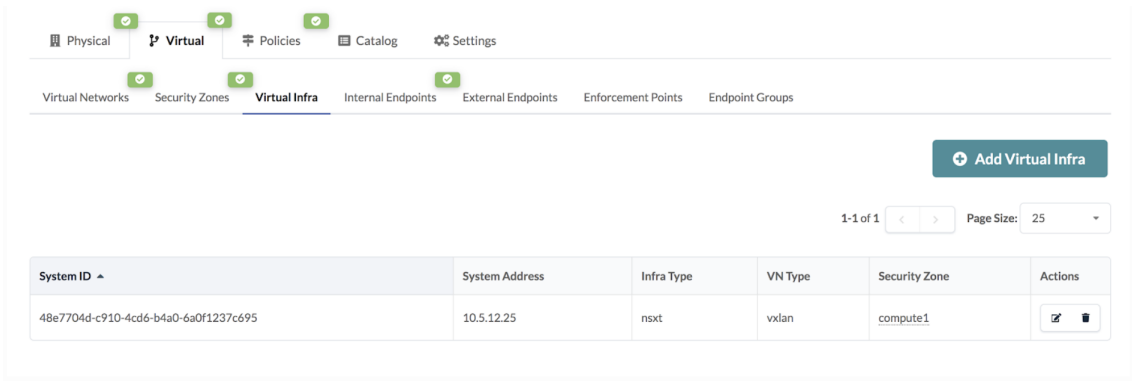
VLAN

VLAN

VXLAN

☐ Create Another? **Create**

7. Click **Create** to create the virtual infra manager and return to the list view. The new virtual infra manager appears in the list.



8. Click **Uncommitted** (top) to see changes, then click **Commit** (top-right) to add the NSX-T manager to the active blueprint.

12.3.2.3 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 leafs. From the blueprint, navigate to **Active > Query** and click **VMs**.

Blueprints > Menlo HQ NSX-T Lab

Dashboard Analytics Staged Uncommitted Active

Physical Virtual Policies Settings Query Anomalies Root Causes

MAC ARP VMs

Search criteria: All 1-11 of 11 Page Size: 25

| VM Name | Hosted On | Hypervisor Hostname | Hypervisor Version | VM IPs | Leaf:Interface | Port Group Name:VLAN ID | MAC Addresses | Virtual Infra Address |
|---------------------------------|---|---------------------|--------------------|---------------|--|--|-------------------|-----------------------|
| Nutanix%2FNSX lab Ext Router VM | nsxtedgehost01 (nsxtedgehost01.dc1.apstra.com) | nsxtedgehost01 | ESXi:6.7.0 | | | | | 10.5.12.25 |
| PoSbareMetalApp30 | nsxtedgehost01 (nsxtedgehost01.dc1.apstra.com) | nsxtedgehost01 | ESXi:6.7.0 | | | | | 10.5.12.25 |
| PoSbareMetalDB20 | nsxtedgehost01 (nsxtedgehost01.dc1.apstra.com) | nsxtedgehost01 | ESXi:6.7.0 | | | | | 10.5.12.25 |
| PoSVMApp60 | nsxtcomputehost01 (srv104-26a) | srv104-26a | ESXi:6.7.0 | 172.16.3.60 | nsxcompute_001_leaf1:Ethernet23/1 nsxcompute_001_leaf1:Ethernet23/4 | PoSApplications:300 PoSApplications:300 | 00:50:56:ad:62:e9 | 10.5.12.25 |
| PoSVMD80 | nsxtcomputehost01 (srv104-26a) | srv104-26a | ESXi:6.7.0 | 172.16.2.80 | nsxcompute_001_leaf1:Ethernet23/1 nsxcompute_001_leaf1:Ethernet23/4 | PoSDatabases:200 PoSDatabases:200 | 00:50:56:ad:b2:8c | 10.5.12.25 |
| advwebtier020 | nsxtedgehost01 (nsxtedgehost01.dc1.apstra.com) | nsxtedgehost01 | ESXi:6.7.0 | 192.168.10.20 | nsxedge_001_leaf1:Ethernet1/22 | benefitswebtieradv:None | 00:50:56:ad:f0:a1 | 10.5.12.25 |

VMs include the following details:

VM Name The Virtual Machine name which is hosted on NSX managed hypervisor.

Hosted On The ESXi host on which Virtual Machine is hosted.

Hypervisor Hostname The hypervisor hostname on which Virtual Machine is hosted and is connected to the leaf TORs in a fabric.

Hypervisor Version The software version of OS running on the hypervisor.

VM IP The IP address as reported by NSX-T after the installation of VM tools. If the IP address is not available this field is empty. AOS displays VM IP if the IP address is available on installation VM tools on the VM.

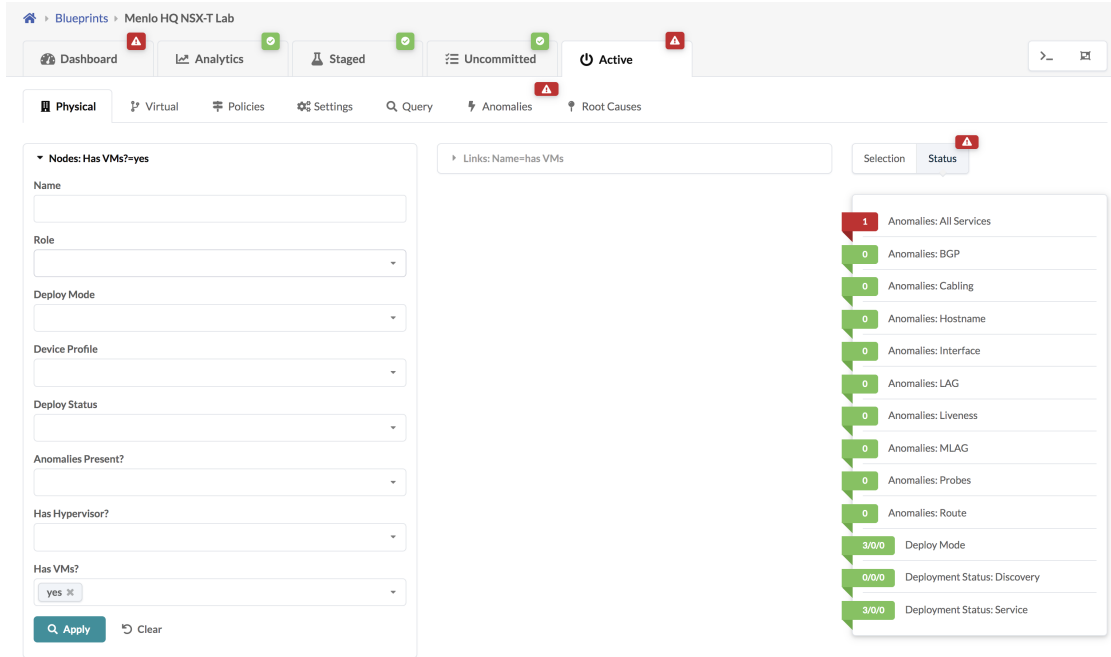
Leaf:Interface System ID for the interface on the leaf to which ESXi host is connected and on which VM resides.

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 AOS 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.



12.3.2.4 Validating Virtual Infra Integration

You can validate virtual infra with intent-based analytics. 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*

Hypervisor MTU Threshold Check Probe

Hypervisor and Fabric LAG Config Mismatch Probe

Hypervisor and Fabric VLAN Config Mismatch Probe

Hypervisor Missing LLDP Config Probe

VMs without Fabric Configured VLANs Probe

Virtual Infra Redundancy Check Dashboard *Hypervisor Redundancy Checks Probe*

For more information, see *Analytics Dashboard* and *Instantiating Predefined Probe*.

12.3.2.5 Disabling Virtual Infra Integration

Virtual infra integrations are disabled by deleting them from the blueprint and external systems.

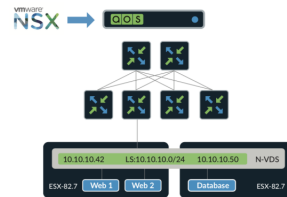
1. From the blueprint, navigate to **Staged > Virtual > Virtual Infra** and click the **Delete** button for the virtual infra to disable.
2. Navigate to the **Uncommitted** tab and commit the deletion.
3. From the **External Systems** menu (left-side), navigate to **Virtual Infra Managers** and click the **Delete** button for the virtual infra to disable.

12.3.3 VMware NSX-T Inventory mapping to AOS Virtual Infrastructure

12.3.3.1 Overview

AOS is able to 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. AOS can integrate with NSX-T to provide AOS admins visibility into the application workloads (aka VMs) running and alert them about any inconsistencies that would affect workload connectivity. **AOS Virtual Infrastructure visibility** helps provide underlay/overlay correlation visibility and use IBA analytics for overlay/underlay.

One cannot view the NSX Inventory in AOS until the NSX-T manager is associated to a blueprint.



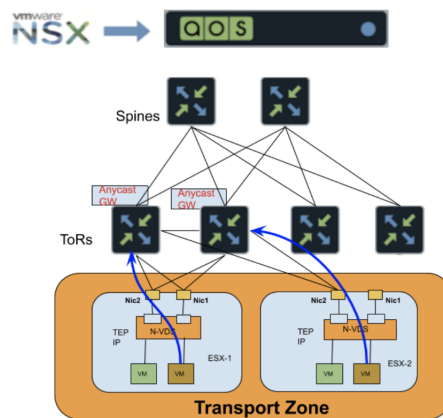
As per above screenshot inventory collection for NSX-T is done via AOS extensible telemetry collector.

12.3.3.2 NSX-T Networking Terminology and correlation

NSX-T uses the following terminology for their control plane and data plane components. Also please find respective correlation with respect to AOS.

TRANSPORT ZONES

Transport Zones (TZ) are used to 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 AOS fabric corresponds to a VLAN based transport zone as per the screenshots below:

Create Virtual Network

vn3000 default x

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: link Tag Type: ☐ Unassigned ☐ Untagged ☒ VLAN Tagged

Assigned To

Query: All 1-2 of 2 Page Size: 25

| 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*: vlan-tz

Description:

N-VDS Name*: N-VDS-VLAN-Uplink

N-VDS Mode: ☒ Standard ☐ Enhanced Datapath

Traffic Type: ☐ Overlay ☒ VLAN

Uplink Teaming Policy Names:

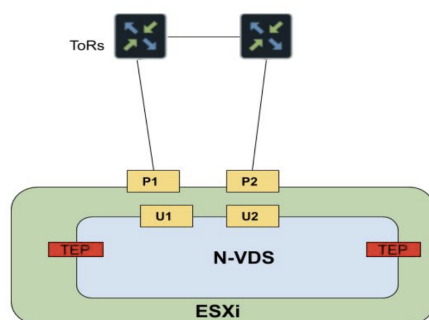
CANCEL **ADD**

N-VDS

It is a NSX managed virtual distributed switch which provides the underlying forwarding and is the data plane of the transport nodes.

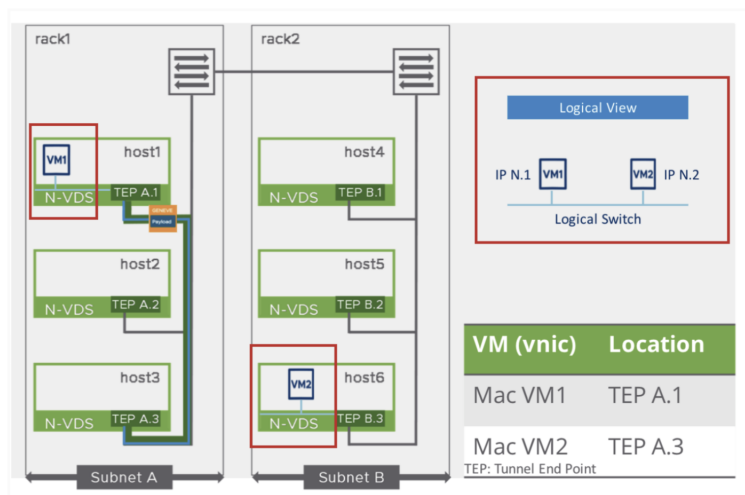
A few notables about N-VDS virtual switches include:

- 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 pnics or LAGs



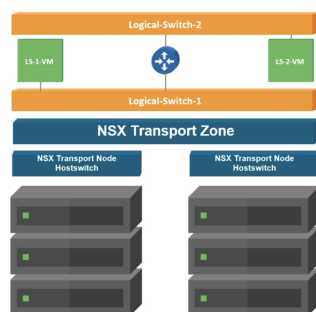
Here TEP are Tunnel Endpoints used for the NSX overlay networking (geneve encapsulation/decapsulation). P1/P2 are pNICs mapped to the uplink profile(U1/U2).

N-VDS are instantiated at the Hypervisor level and can be thought of Virtual switch connected to the ToR physical leafs as below:



Transport Node

It is a node capable of participating in an NSX-T Data Center overlay or VLAN networking.



VMs hosted on different Transport nodes communicate seamlessly across the overlay network. A transport node can belong to:

- Multiple VLAN transport zones.

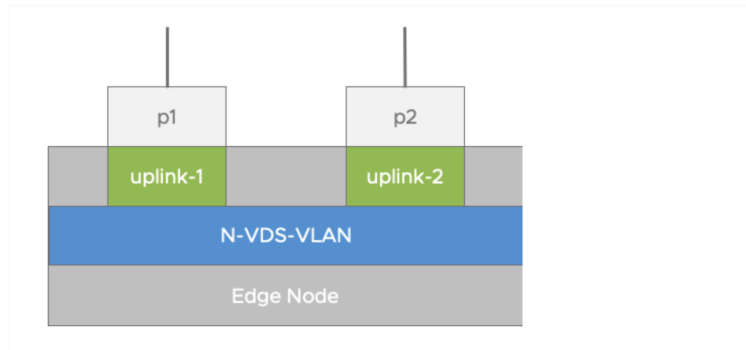
- At most one overlay transport zone with a standard N-VDS.

This can be compared to setting end hosts(servers) in an AOS 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. AOS fabric must be configured with the corresponding VLAN Virtual Network.



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.

12.3.3.3 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 leaves. In NSX-T Data Center, ESXi hosts are prepared as Transport Node which allows nodes to exchange traffic for virtual networks on AOS 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 leaves towards the end servers.
- In NSX-T integration with AOS 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 AOS 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 AOS 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 AOS Fabric.

12.3.3.4 Exact Model details and Relationship

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

In AOS to obtain NSX-T API response for respective hypervisor hosts and understand the correlation you can use graph query. To open the GraphQL Explorer, you can 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 AOS under **Active** tab with **Has Hypervisor = Yes** option as below:

To obtain respective hostname for the transport nodes below query can be used:

Request:

```
{
  hypervisor_nodes {
    hostname
  }
}
```

(continues on next page)

(continued from previous page)

```
}
}
```

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 AOS fabric in terms of LAG and LACP config.

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 AOS Fabric.

NSX-API Request/Response to check MAC address for the Transport node interfaces.

Request:

```
{
  pnic_nodes {
    id mac_address
  }
}
```

Response:

```
{
  "data": {
    "pnic_nodes": [
      {
        "id": "1e2162c3-9ce6-4f35-afc2-217bb48ced49",
        "mac_address": "52:54:00:88:41:28"
      },
      {

```

(continues on next page)

(continued from previous page)

```

        "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 AOS Fabric towards the NSX-T Transport Node. It is the MAC address of the ESXi host pNICs having LAG bond towards ToR leafs in AOS fabric.

Please find below NSX-API Request/Response to check Switch name attribute of transport node's transport zone.

Request:

```

{
  pnic_nodes {
    id switch_id
  }
}

```

Response:

```

{
  "data": {
    "pnic_nodes": [
      {
        "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:

NSX-API Request/Response to check Transport node's interface.

Request:

```

{
  pnic_nodes {
    id label
  }
}

```

Response:

```

{
  "data": {
    "pnic_nodes": [
      {
        "id": "82586be7-2998-401f-82ba-11afa5bb9730",
        "label": "eth2"
      },
    ],
  }
}

```

(continues on next page)

(continued from previous page)

```

    {
      "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.

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": {
    "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"
      }
    ]
  }
}

```


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_intf": "swp4"
      },
      {
        "id": "0043d742-405a-454f-9e9b-695d5dd14608",
        "neighbor_intf": "swp3"
      },
      {
        "id": "b91a5725-7500-489b-a454-e05d7c311525",
        "neighbor_intf": "eth0"
      }
    ]
  }
}
```

NSX-API Request/Response to check the MTU attribute of Transport node's interface.

Request:

```
{
  pnic_nodes {
    id mtu
  }
}
```

Response:

```
{
  "data": {
    "pnic_nodes": [
      {
        "id": "82586be7-2998-401f-82ba-11afa5bb9730",
        "mtu": 1600
      },
      {
        "id": "0043d742-405a-454f-9e9b-695d5dd14608",
        "mtu": 1600
      }
    ]
  }
}
```

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

Please find below NSX-API Request/Response to check VNIC mac address attribute. 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"
      }
    ]
  }
}
```

Please find below 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"
      }
    ]
  }
}
```

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 AOS 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{
```

(continues on next page)

(continued from previous page)

```
id mtu
}
}
```

Response:

```
{
  "data": {
    "vnic_nodes": [
      {
        "id": "9553390b-754e-45ef-8976-e63396d554ee",
        "mtu": 1600
      },
      {
        "id": "a00bb649-5032-462f-97e7-b6c4f5f1ac86",
        "mtu": 1600
      }
    ]
  }
}
```

So Virtual Interface i.e NSX VTEP and vswitch should have mtu of 1600 as per screenshot above.

Port channel policy

- **Label:** Name attribute of the host switch uplink lag profile
- **Mode:** Mode attribute of host switch uplink lag profile
- **Hashing_algorithm:** Load balance algorithm attribute of host switch uplink lag profile

An uplink profile is mapped in a Transport node on the NSX-T side with policies for the links from the hypervisor hosts to NSX-T logical switches.

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
      }
    ]
  }
}
```

(continues on next page)

(continued from previous page)

```

    },
    {
      "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:

Please find below 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 {

```

(continues on next page)

(continued from previous page)

```

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:

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-8360-076d9daddd1b",
        "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.

Please find below 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 AOS 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
      }
    ],
  }
}
```

(continues on next page)

(continued from previous page)

```
{
  {
    "id": "18b81c81-8ae1-46b1-83ca-05cd5b364a1c",
    "vni": 67584
  }
],
  "id": "rack-based-blueprint-9dfa0044"
}
```

12.3.4 NSX-T Security Policy Integration

12.3.4.1 Introduction

The Micro-segmentation capabilities in NSX-T help secure communication within virtual workloads and microservices. This is attained via fine-grain security policies that map to workflows between applications and business logic.

However, many data centers have workloads that have not been virtualized, or cannot be virtualized, running on physical servers. To secure such segmented physical workloads, the security policies have to be implemented resulting in a divergence in policy enforcement between the virtual and the physical workload domain. Since each administrative domain is independent and not in sync with others this can lead to operational overhead, duplication of effort and inefficiency in application delivery and network utilization.

NSX-T micro-segmentation mainly helps with the network security model for virtual workloads. However, for seamless network connectivity, there is a definite advantage in extending the same to secure physical workloads. To achieve the goal of the consistent policy applied regardless of location and workload type it becomes important to apply the security policy at the network edge.

Apstra AOS manages network security and workload isolation through Group Based Policy (GBP). These policies are translated into ACLs for each supported vendor NOS. These ACLs are installed on the associated L3 interfaces of the Top-of-Rack switches, controlling traffic flows outgoing or incoming the L3 interfaces.

New in version 3.2: Apstra currently provides limited availability feature support for NSX-T security policy integration with AOS GBP. For details please refer to Limitations section. Using this feature it allows extending virtual domain segmentation policy to physical workloads. This further helps maintain a single policy engine, hence achieving consistency and operational efficiency, while simultaneously preserving the autonomous nature of the teams and systems.

12.3.4.2 Use Cases

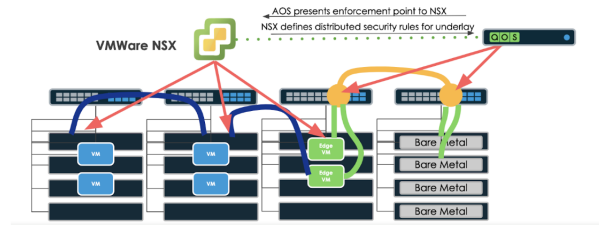
Please find the respective use cases related to security policies application and meeting higher security standard in a data center as below:

1. Allows configuring security policies for the complete network (including NSX-T and bare-metal) all on one single policy engine through the NSX-T Manager portal. This helps leverage NSX-T as the unified policy engine authoring plane with policy enforcement appropriately done at either the hypervisor, edge node, bare metal, ToR switch or 3rd party firewall level.
2. Enforce security policy close to the source, which helps conserve bandwidth and meet security standards.

AOS NSX-T policy integration feature helps extend NSX distributed policies into the underlay fabric. Security policy can be applied for traffic exiting NSX-T destined to physical or virtual workloads, and for traffic exiting physical/virtual workloads destined to NSX-T virtual workloads:

1. AOS can be registered as enforcement point on NSX-T.

- Using NSX-T API/UI can extend security policies for Bare Metal outbound traffic close to the source as the first hop (i.e. Top of Rack). AOS takes care of the configuration and validation, for any types of hardware switches and NOS.



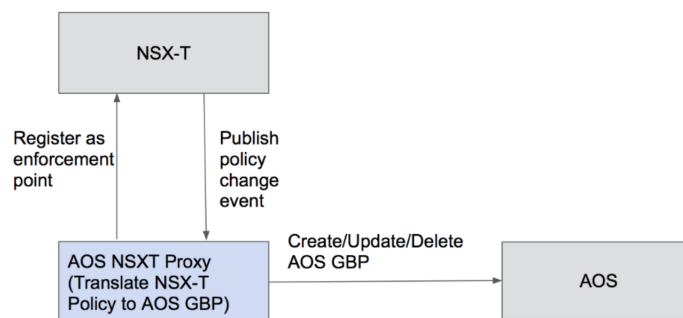
Note: AOS currently enforces security policies using ACLs on the ToR switch L3 interfaces, therefore applying security for routed traffic only (not switched traffic).

12.3.4.3 Overview

The AOS NSX-T policy integration can be delivered through a standalone proxy container. AOS must be registered as an enforcement point with NSX-T. This can be achieved in two ways:

- Have the proxy container register AOS as an enforcement point with NSX-T through configurable parameters, this requires read/write NSX-T user credential.
- NSX-T admin can register AOS as enforcement point directly through NSX-T GUI for better security. This way only read-only NSX-T user credential with limited privileges is needed for proxy container.

Once the proxy container starts, it handles security policy events published by NSX-T policy translates them to AOS GBP (Group-Based Policy), and applies them to the AOS managed fabric. The following diagram is a logical view of the integration:



12.3.4.4 Setup AOS NSX-T Proxy

Currently we only support running NSX-T proxy server on AOS server.

Prerequisites

- On AOS, create (or reuse) a user account for NSX-T. NSX-T Manager uses this account to initiate connections to AOS for policy synchronization and enforcement. This should have read/write permission on blueprint and

commit permission based on commit policy.

2. Before you run the proxy server, you must configure the nginx on AOS to redirect traffic published by NSX-T to the proxy server by adding the following `nsxt` section below the existing `graphcons` to nginx config file located at `/etc/aos/nginx.conf.d/nginx.conf` on your AOS server, you will need `sudo` to edit the file:

Listing 1: `/etc/aos/nginx.conf.d/nginx.conf`

```
location /graphcons {
    auth_request /api/aaa/authorize;
    auth_request_set $auth_status $upstream_status;
    proxy_pass http://webcons/graphcons;
}

location ~ /nsxt {
    # nsxt policy proxy server internal path, no need for aos' auth
    proxy_pass http://127.0.0.1:9999$request_uri;

    proxy_buffering off;
    proxy_set_header Host $http_host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
}
```

After updating the nginx config, have the nginx container reload the configuration:

Listing 2: Reload nginx container

```
admin@aos-server:~$ docker container exec aos_nginx_1 nginx -s reload
```

Note: The port specified for `proxy_pass` in nginx NSX-T configuration block above (which is 9999 in this case) needs to be consistent with the host port specified when you run the proxy server docker container.

Please find below two approaches to run the proxy server:

Approach 1

- Contact Apstra Support to download the debian package.
- Install debian package:

Listing 3: Installing debian package

```
admin@aos-server:~$ sudo dpkg -i aos-nsxt-proxy-0.1.0-develop.deb
[sudo] password for admin:
Selecting previously unselected package aos-nsxt-proxy.
(Reading database ... 110393 files and directories currently installed.)
Preparing to unpack aos-nsxt-proxy-0.1.0-develop.deb ...
Unpacking aos-nsxt-proxy (0.1.0-develop) ...
Setting up aos-nsxt-proxy (0.1.0-develop) ...
0db06dff9d9a: Loading layer [=====]
====>] 105.5MB/105.5MB
f32868cde90b: Loading layer [=====]
```

(continues on next page)

(continued from previous page)

```

=====>] 24.08MB/24.08MB
7b76d801397d: Loading layer [=====
=====>] 8.005MB/8.005MB
2c8d31157b81: Loading layer [=====
=====>] 146.4MB/146.4MB
a637c551a0da: Loading layer [=====
=====>] 576.4MB/576.4MB
bb9c02680a15: Loading layer [=====
=====>] 17.5MB/17.5MB
c9d608035aef: Loading layer [=====
=====>] 71.26MB/71.26MB
715450468940: Loading layer [=====
=====>] 4.608kB/4.608kB
799a7872c8c7: Loading layer [=====
=====>] 6.483MB/6.483MB
28eb210d7514: Loading layer [=====
=====>] 970.8kB/970.8kB
17150087d781: Loading layer [=====
=====>] 76.66MB/76.66MB
78e8360892ea: Loading layer [=====
=====>] 1.536kB/1.536kB
Loaded image: nsxt-proxy:0.1.0-develop
admin@aos-server:~$

```

- Check version of debian package:

Listing 4: Debian version

```

admin@aos-server:~$ sudo dpkg -l aos-nsxt-proxy
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/halF-conf/Half-inst/trig-aWait
|/Trig-pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name                      Version                Architecture           Description
+++-=====
=====
ii  aos-nsxt-proxy              0.1.0-develop          all                     AOS NSXT policy proxy service
admin@aos-server:~$

```

- Provide the required configuration options on AOS server in `/etc/aos/nsxt/ config.yaml`, which can include minimum parameters as an example below:

Listing 5: Config parameters

```

aos:
server: 172.20.76.3
blueprint_id: rack-based-blueprint-09e517dc
proxy:
register_enforcement_point: True
enforcement_point_name: test-ep
enforcement_point_thumbprint:
484F0E72F7202E976972A27DE00F9A07778C3F5C5BB9E216A2BF9C2D05177AD1
nsxt:
url: https://10.5.12.24
username: admin
password: admin

```

Please note that `enforcement_point_thumbprint` can be obtained as per the below command:

Listing 6: Fingerprint from nginx certificate

```
admin@aos-server:~$ sudo openssl x509 -noout -fingerprint -sha256 -inform
pem -in /etc/aos/nginx.conf.d/nginx.crt | tr -d ':'
[sudo] password for admin:
SHA256 Fingerprint=
9B1D62BEB A2618F50C7DD5D806F87DC14B4320E5610D838A7F4BD07B9394BB18
admin@aos-server:~$
```

- Start proxy:

Listing 7: Start proxy

```
sudo systemctl start aos-nsxt-proxy
```

Now that you have the proxy server running, you could run the following commands for further validation:

- Check proxy status:

Listing 8: Check status

```
admin@aos-server:~$ sudo systemctl status aos-nsxt-proxy
* aos-nsxt-proxy.service - AOS NSX-T policy proxy service
Loaded: loaded (/etc/systemd/system/aos-nsxt-proxy.service; disabled;
vendor preset: enabled)
Active: active (running) since Thu 2020-02-06 18:24:47 UTC; 6 days ago
Process: 26849 ExecStartPre=/bin/bash -c (docker stop aos-nsxt-proxy
-container 2> /dev/null;
Main PID: 26891 (docker)
Tasks: 22 (limit: 4915)
CGroup: /system.slice/aos-nsxt-proxy.service
        └─26891 /usr/bin/docker run --env DEBUG=1 --publish 9999:8080
        --name aos-nsxt-proxy-container --volume /etc/aos/ns
Feb 12 21:50:00 aos-server docker[26891]: File "/usr/local/lib/
python3.7/site-packages/aiohttp-3.5.4-py3.7-linux-x86_64.egg
Feb 12 21:50:00 aos-server docker[26891]: headers=self.headers)
Feb 12 21:50:00 aos-server docker[26891]: aiohttp.client_exceptions.
ContentTypeError: 0, message='Attempt to decode JSON with
Feb 12 21:50:00 aos-server docker[26891]: 2020-02-12 21:50:00,134: INFO
: aiohttp.access:233 : 172.17.0.1 [12/Feb/2020:21:50:
Feb 12 21:50:58 aos-server docker[26891]: 2020-02-12 21:50:58,367: INFO
: nsxt_proxy.nsxt_policy_translator:590 : Updating ex
Feb 12 21:50:58 aos-server docker[26891]: 2020-02-12 21:50:58,368: DEBUG
: nsxt_proxy.aos:145 : GET https://10.5.12.125/api/b
```

- View proxy logs:

Listing 9: View logs

```
sudo journalctl -u aos-nsxt-proxy
```

Approach 2

After successfully configuring the nginx on AOS, follow the steps below to bring up the proxy server:

- Download the NSX-T proxy server image file to your AOS instance. Contact Apstra Support for docker image.

- On AOS, load the docker image:

Listing 10: View logs

```
admin@aos-server:~$ docker load -i nsxt-proxy-0.1.0-develop.tar
Loaded image: nsxt-proxy:0.1.0-develop
admin@aos-server:~$
```

- Obtain the fingerprint from nginx certificate and copy the 64 digit fingerprint and use it as the value of `--proxy-enforcement-point -thumbprint` in the next command.

Listing 11: Nginx fingerprint

```
admin@aos-server:~$ sudo openssl x509 -noout -fingerprint -sha256 -inform pem -in /etc/aos/nginx.conf.d/nginx.crt | tr -d ':'
```

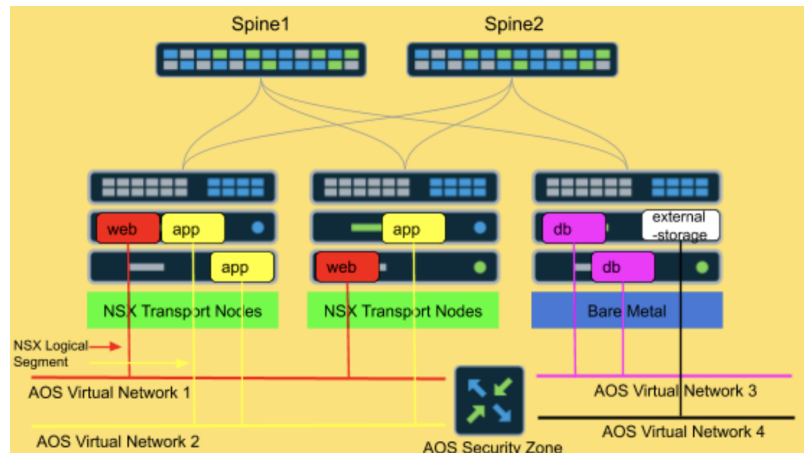
- Start your proxy server, here is an example, you will need to update aos and NSX-T endpoint and credential:

Listing 12: Start proxy server

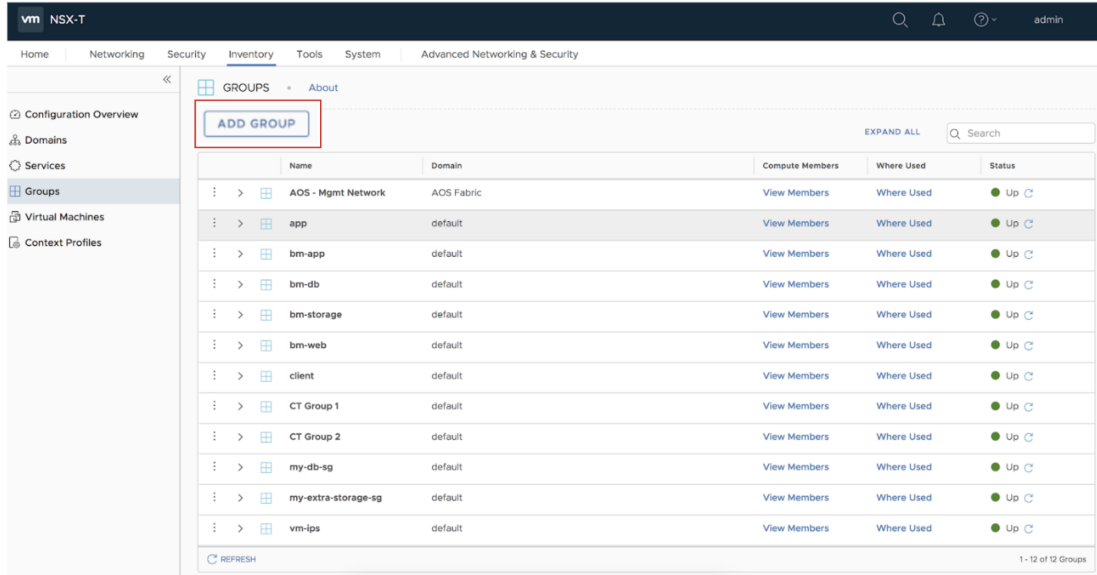
```
admin@aos-server:~$ docker run --publish 9999:8080 nsxt-proxy:0.1.0-develop
--aos-server '172.20.76.3' --aos-username 'admin' --aos-password 'admin'
--aos-blueprint-id 'rack-based-blueprint-09e517dc' --aos-security-zone 'Red'
--proxy-enforcement-point-name 'test-ep' --proxy-enforcement
-point-thumbprint 'F9D85A1B5CBF21217DA6DD60F76B47E52D784118F43A90B289D1A2E0
ABA2F70B' --proxy --commit-policy 'staging' --nsxt-url 'https://10.5.12.24'
--nsxt-username 'admin' --nsxt-password 'VMware!VMware!'
```

12.3.4.5 Create Distributed firewall(DFW) policy on NSX-T

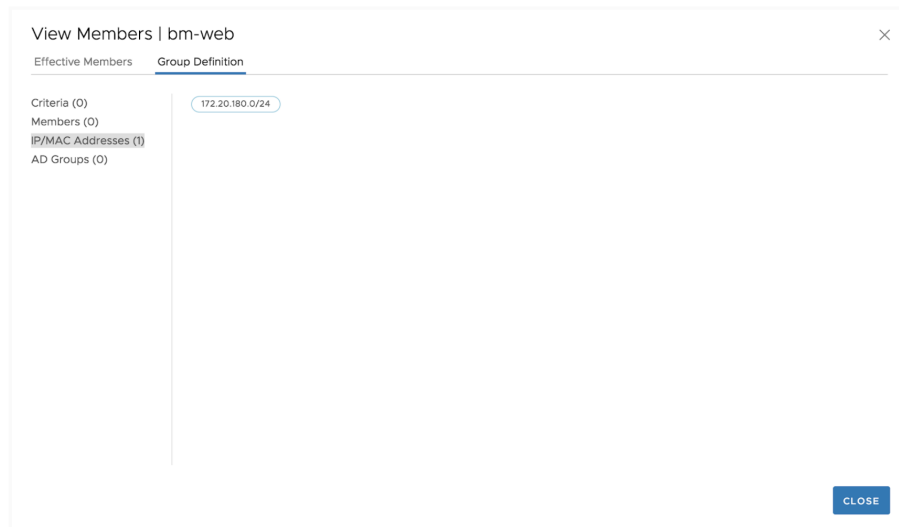
In the tutorial, we will use a 3-tier application as an example below to go through the workflow:



- Create security group under **Inventory > GROUPS > ADD GROUP**

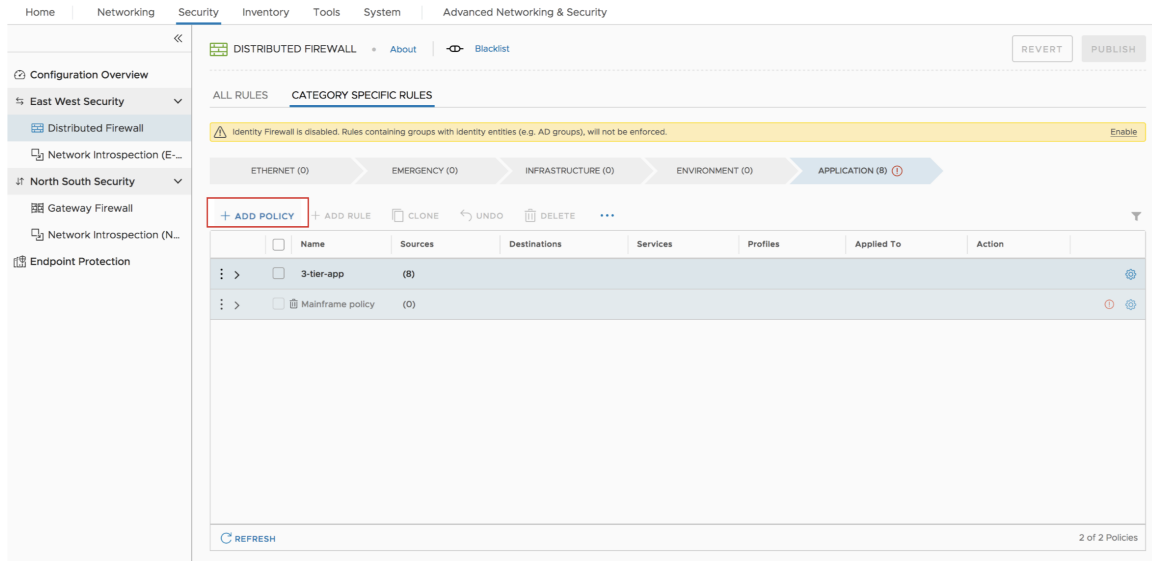


Groups include different objects that are added both statically and dynamically and can be used as the source and destination field of a firewall rule. They can be configured to contain a combination of virtual machines, IP sets, MAC sets, logical ports, logical switches, AD user groups, and other nested groups. Dynamic inclusion of groups can be based on the tag, machine name, OS name, or computer name. Also, you can define tags in groups that are used by NSX-T to facilitate the grouping of VM workloads.

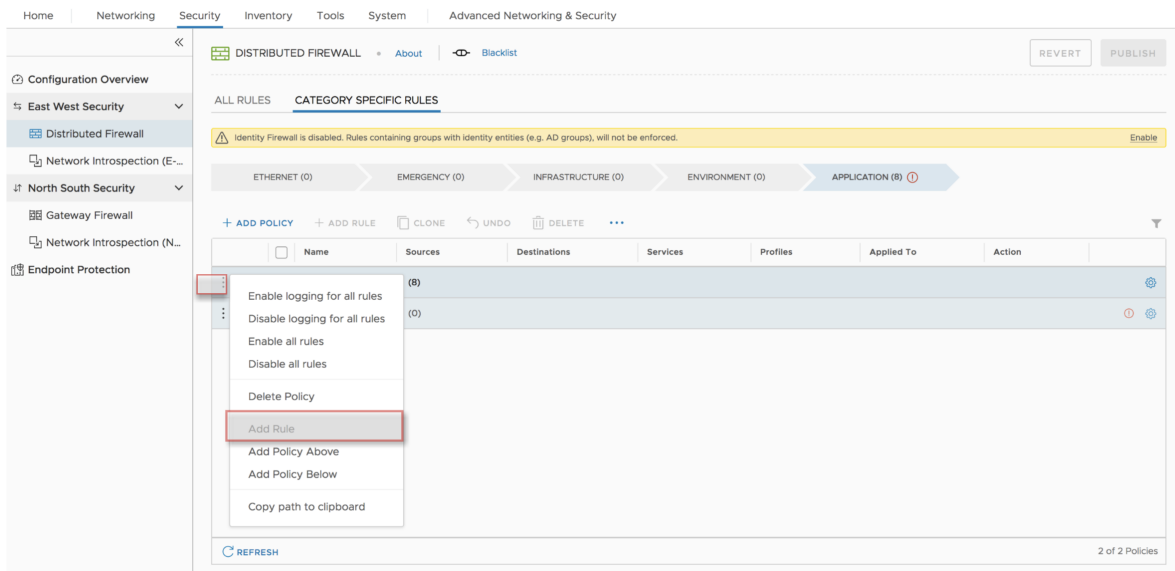


Group definition can be created using the IP/MAC address and other criteria as per the above screenshot which can be used as the source and destination fields.

- Create DFW policy with rules under **Security > Distributed Firewall > ADD POLICY**



- To add security rules, click on the edit button (highlighted by red rectangle below) next to the newly created policy and then select **Add Rule**:



- After clicking Add rule please select the source Group and destination Group as per below screenshots:

Set Source

Rule > New Rule


Negate Selections ☐ No | Negated selections will be shown as Example Group

my-web-sg X

ADD GROUP EXPAND ALL

| | Name | Compute Members | Status |
|-------------------------------------|---------------------|------------------------------|--------|
| <input type="checkbox"/> | my-app-sg | View Members | Up |
| <input type="checkbox"/> | my-db-sg | View Members | Up |
| <input type="checkbox"/> | my-extra-storage-sg | View Members | Up |
| <input checked="" type="checkbox"/> | my-web-sg | View Members | Up |
| <input type="checkbox"/> | PoSApplications | View Members | Up |
| <input type="checkbox"/> | PoSdatabases | View Members | Up |

1 REFRESH 1 - 6 of 6 Groups

CANCEL

APPLY

Set Destination

Rule > New Rule


Negate Selections ☐ No | Negated selections will be shown as Example Group

my-db-sg X

ADD GROUP EXPAND ALL

| | Name | Compute Members | Status |
|-------------------------------------|---------------------|------------------------------|--------|
| <input type="checkbox"/> | my-app-sg | View Members | Up |
| <input checked="" type="checkbox"/> | my-db-sg | View Members | Up |
| <input type="checkbox"/> | my-extra-storage-sg | View Members | Up |
| <input type="checkbox"/> | my-web-sg | View Members | Up |
| <input type="checkbox"/> | PoSApplications | View Members | Up |
| <input type="checkbox"/> | PoSdatabases | View Members | Up |

1 REFRESH 1 - 6 of 6 Groups

CANCEL

APPLY

- Select service for the NSX-T rule as per below example:

Set Service ×

Rule > New Rule

DNS-UDP X

ADD NEW SERVICE

| | Name | Service Entries | Tags | Status |
|-------------------------------------|--------------------|-----------------------------------|------|--------|
| <input type="checkbox"/> | DHCP-Server | UDP (Source: Destination: 67) | 0 | Up |
| <input type="checkbox"/> | DHCPv6 Client | UDP (Source: Destination: 546) | 0 | Up |
| <input type="checkbox"/> | DHCPv6 Server | UDP (Source: Destination: 547) | 0 | Up |
| <input type="checkbox"/> | Directory Services | TCP (Source: Destination: 5725) | 0 | Up |
| <input type="checkbox"/> | DNS | TCP (Source: Destination: 53) | 0 | Up |
| <input checked="" type="checkbox"/> | DNS-UDP | UDP (Source: Destination: 53) | 0 | Up |
| <input type="checkbox"/> | EdgeSync service | TCP (Source: Destination: 0) | 0 | Up |

1 REFRESH 1 - 50 of 409 Services

CANCEL APPLY

- Finally, click Publish to publish the policy to AOS enforcement point after the source/destination and service is selected for the new rule as below:

NSX-T

Home Networking Security Inventory Tools System Advanced Networking & Security

Configuration Overview

East West Security

Distributed Firewall

Network Interception (E-...)

North South Security

Gateway Firewall

Network Interception (N-...)

Endpoint Protection

DISTRIBUTED FIREWALL About Blacklist

1 Total Unpublished Change REVERT PUBLISH

ALL RULES CATEGORY SPECIFIC RULES

Identity Firewall is disabled. Rules containing groups with identity entities (e.g. AD groups), will not be enforced. Enable

ETHERNET (0) EMERGENCY (0) INFRASTRUCTURE (0) ENVIRONMENT (0) APPLICATION (8)

+ ADD POLICY + ADD RULE CLONE UNDO DELETE ... 1 Unpublished Change

| | Name | Sources | Destinations | Services | Profiles | Applied To | Action | |
|----------------|----------|-----------|--------------|----------|----------|------------|--------|----|
| 3-tier-app (8) | | | | | | | | |
| | New Rule | my-web-sg | my-db-sg | DNS-UDP | Any | DFW | Allow | ⚙️ |
| | New Rule | my-web-sg | my-db-sg | DNS | Any | DFW | Allow | ⚙️ |
| | New Rule | my-web-sg | my-db-sg | CM-HTTPS | Any | DFW | Allow | ⚙️ |
| | New Rule | my-web-sg | my-db-sg | HTTPS | Any | DFW | Allow | ⚙️ |
| | New Rule | my-web-sg | my-app-sg | Any | Any | DFW | Allow | ⚙️ |
| | New Rule | my-app-sg | my-db-sg | Any | Any | DFW | Allow | ⚙️ |
| | New Rule | my-db-sg | my-app-sg | Any | Any | DFW | Allow | ⚙️ |
| | New Rule | Any | my-app-sg | Any | Any | DFW | Reject | ⚙️ |

REFRESH 2 of 2 Policies

12.3.4.6 Commit Policy for NSX-T proxy

The commit policy controls how the proxy makes changes to the blueprint. The default commit policy is 'yield'. It waits for the blueprint to not have any outstanding changes (from other GUI or API users), does the changes to staging blueprint and **auto commit** the changes to the active blueprint. This is useful, and necessary when fabric interconnects multiple VMs on hypervisor hosts, with one proxy associated with NSX-T.

The Distributed firewall (DFW) policy is automatically committed immediately to the blueprint if there are no existing uncommitted changes by other sources (i.e. user other than the policy server is making a change to the blueprint at the same time). If there are other uncommitted changes, the proxy server waits until the uncommitted change is committed or the timeout (default is 5 minutes). In case the other user commits the change within the 5 minutes, the

proxy server will then commit the policy change automatically. If it exceeds 5 minutes, the policy enforcement fails with the timeout exception message.

When the policy is published, the NSX-T proxy server translates it to AOS Group Based Policies (GBP). The deployed blueprint shows a policy translated from the rule that was just created.

To see details, navigate to **Staged / Policies / Security Policies** for the respective policy as below:

The screenshot displays the NSX-T Edge configuration page for a Staged Security Policy. The interface includes a top navigation bar with tabs for Dashboard, Analytics, Staged, Uncommitted, and Active. Below the navigation bar, there are view toggles for Expanded View and Compact View. The main content area is divided into sections: Common Parameters, Application Points, and Rules.

Common Parameters:

| | |
|-------------|--|
| Name | 97192f90-4e28-11ea-92b4-99b759 |
| Description | Policy: default.23c23a0-28c9-11ea-b243-052a2bf9c24a. Rule: 97192f90-4e28-11ea-92b4-99b759ee2005, sequence number 1 |
| Enabled | <input checked="" type="checkbox"/> |
| Tags | NSX, BulkObject-2976-4228-b554-050008484e4c |

Application Points:

| | |
|-------------------------------|------------------------------------|
| Source Application Point | External Endpoint Group: my-web-sg |
| Destination Application Point | Internal Endpoint Group: my-app-sg |

Rules:

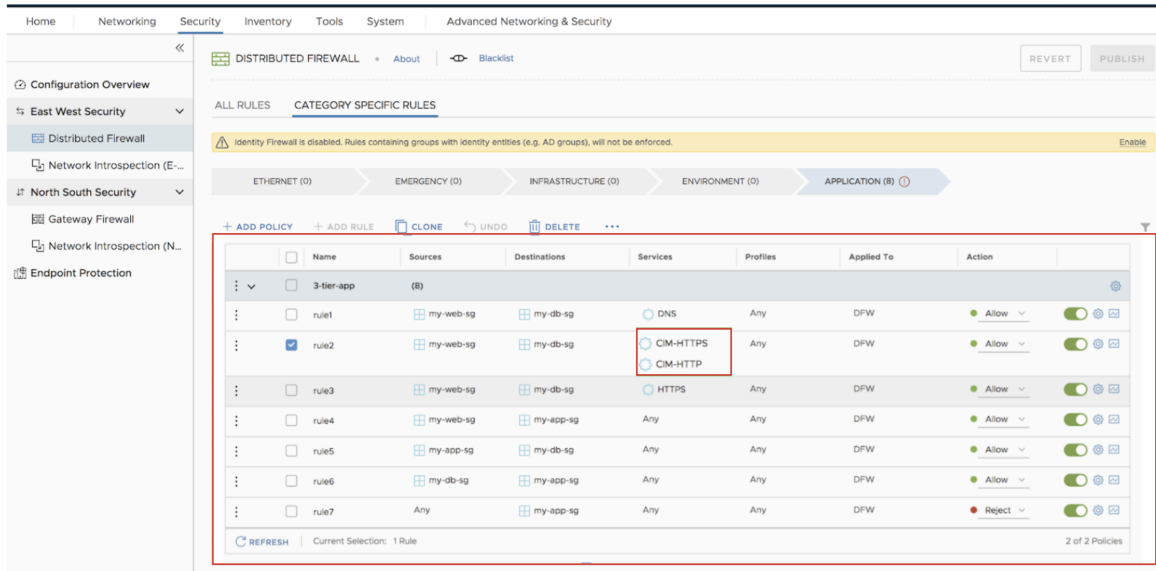
Query: All 1-1 of 1

| Name | Description | Protocol | Action | Source Port | Destination Port |
|-----------|--------------------------------------|----------|--------|-------------|------------------|
| service_7 | 97192f90-4e28-11ea-92b4-99b759ee2005 | TCP | Permit | any | 53 |

12.3.4.7 Policy Translation Details

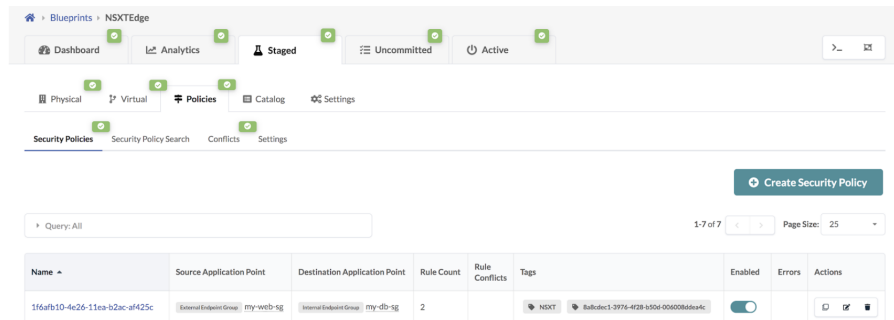
- Each rule in an NSX-T policy is translated into an AOS GBP policy, and each service in an NSX-T rule is translated into an AOS policy rule. This is because of the discrepancy between NSX-T policy and AOS GBP:
 - In the NSX-T policy, you can specify the security group as a source/destination for each rule. The security group is very flexible and can be anything among IPSet, Logical Switch, NSGroup, Logical Port and MAC Set. As in AOS GBP, you can only specify groups (Internal Endpoint / External Endpoint / Internal Endpoint Group / External Endpoint Group / Virtual Network / Security Zone) as a source/destination for an entire policy, it doesn't support per-rule source/destination granularity.
 - In NSX-T policy, it allows the user to specify one to many services when the user creates a security rule, each service will be translated to an AOS GBP rule.

For example:

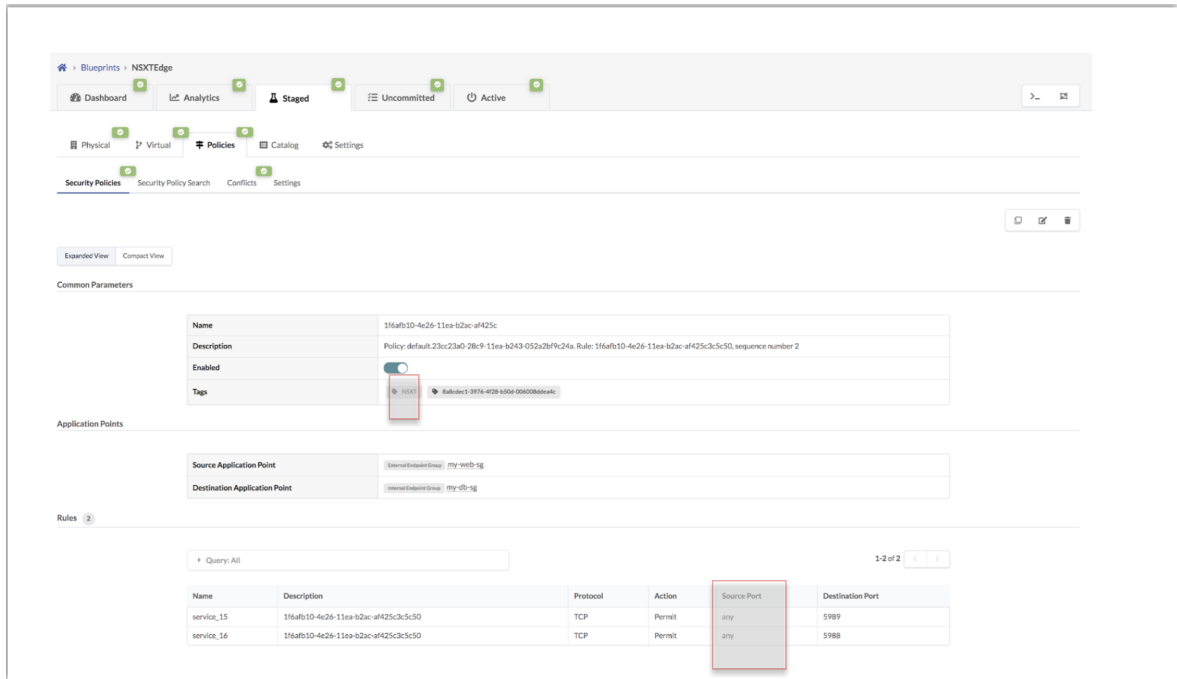


3-tier-app Policy highlighted in red rectangle in the image above has 7 rules: **rule1** to **rule7**. When the policy is published, the proxy server translates each rule in **3-tier-app** Policy to an AOS Policy. 7 policies in total are in AOS (highlighted in red rectangle above). The **rule2** in NSX-T (highlighted by red elliptical circle above) has two services: CIM-HTTPS and CIM-HTTP, which will be translated into two rules in AOS (highlighted by red rectangle above).

Please find below screenshot for **rule2** being translated to AOS GBP:

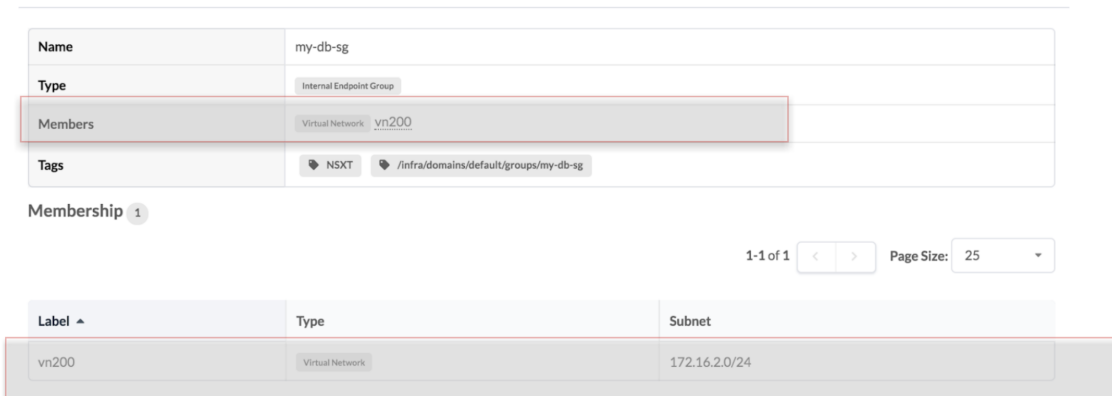


- If the source/destination in an NSX-T rule is set to ANY, it is translated into an AOS policy with a user-specified Security Zone as source/destination. If the Security Zone is not specified, it defaults to the default Security Zone. For example, in the below example since the source port is defined to Any this is tagged with a user-specified Security Zone named NSXT.



- How do we translate the source/destination security group in an NSX-T rule to source/destination in the AOS policy? It's done by fetching the security group member IP addresses from NSX-T, and compare them with CIDR of existing AOS virtual networks within a specified **Security Zone**. If member IPs are within AOS VN, **Internal Endpoint Group** is created with member IPs as **Internal Endpoints**. In case the member IPs are AOS VN IPv4 subnet, source/destination is specified as that VN. If member IPs are not found within any of existing VN, source/destination in an NSX-T rule is created with member IPs as **External Endpoints**.

Endpoint Group Preview



Since the member IPs for the **my-db-sg** group match the CIDR of existing **AOS virtual networks (vn200)** within the specified **Security Zone(NSXT)** here it is set as an **Internal Endpoint**.

- When a rule is translated to an AOS GBP with both source and destination as **External Endpoint Group**, such a policy will be skipped by AOS during translation because it's not logical to apply such policy as there is no enforcement point to enforce such policy in AOS which is external in terms of fabric.

12.3.4.8 EVPN Support

To translate the source/destination security group from NSX-T, look up the member IPs and compare them to AOS existing virtual networks' IPv4 subnet. For EVPN where virtual networks in different VRF share the same IPv4 subnet, specify one Security Zone where the security policies will be deployed. Configure this parameter via the proxy server configuration file i.e. "config.yaml" as per the example below:

Listing 13: Proxy server configuration file to specify security zone

```
root@aos-server:/etc/aos/nsxt# cat config.yaml
aos:
  server: 10.5.12.125
  username: admin
  password: admins@Apstra1
  blueprint_id: 13ca99ca-848d-40b6-86cd-7b31ccc9f7ec
  security_zone: NSXT

proxy:
  server: 0.0.0.0
  server_port: 8080
  commit_policy: yield
  register_enforcement_point: True
  enforcement_point_name: aos
  enforcement_point_thumbprint:
    ↪6B779B5750CDEFF61357F923B3803AB1FFA682503CA8E976CB9E6846F8A533A8

nsxt:
  url: https://10.5.12.25
  username: admin
  password: admins@Apstra1
  poll_interval: 120
root@aos-server:/etc/aos/nsxt#
```

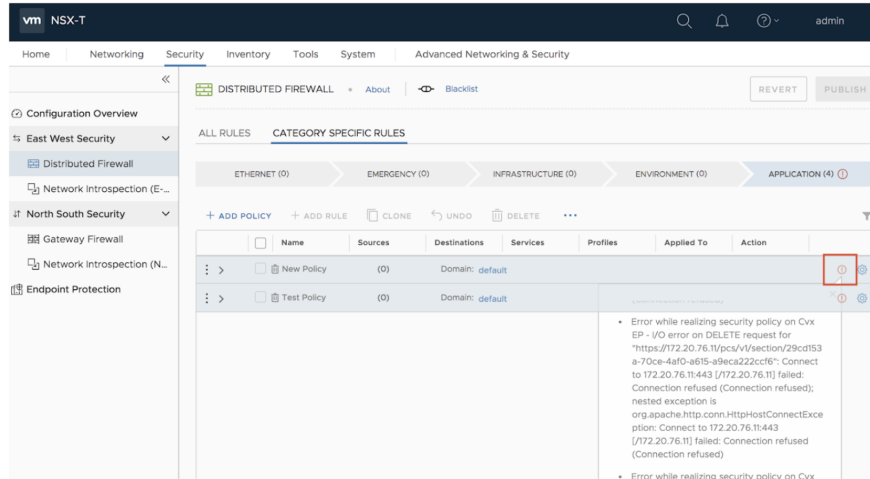
Please find below different scenarios NSX-T policy is translated into an AOS GBP policy on the basis source/destination fields (i.e.IP subnets) and matched across different Security Zones once the Security Zone parameter provided.

- If Security Zone provided, search Virtual Networks within specified Security Zone
- If Security Zone is not provided, search Virtual Networks across all Security Zones and find the matching one. If more than one Security Zone is found for either source/destination, error out. If only one is found, use the matching one for ANY translation and Internal/External Endpoint Group determination (assume no overlapping IPs).
- If more than one Security Zones found for both source and destination, error out. Kindly note that GBP doesn't support inter-Security Zone rule as for ANY translation it will be ambiguous since source and destination can't be in more than one Security Zone.

Note: We do support looking for IP subnets across Security Zones for scenarios where IP subnets are non overlapping.

12.3.4.9 Error Handling

When the policy translation fails, exceptions with detailed error messages will be raised and returned to NSX-T policy. NSX-T will mark the policy with realization error. To view the detailed error message click the red exclamation mark:



12.3.4.10 AOS Policy Conflicts

If conflicts can be resolved by AOS, the proxy server proceeds with deploying the blueprint. If not, then the proxy server returns failure with a detailed message about the conflicts.

Please find below example where policy conflicts are resolved by AOS:

Blueprints > NSXTEdge

Dashboard Analytics Staged Uncommitted Active

Physical Virtual Policies Catalog Settings

Security Policies Security Policy Search Conflicts Settings

Create Security Policy

Query: All 1-7 of 7 Page Size: 25

| Name | Source Application Point | Destination Application Point | Rule Count | Rule Conflicts | Tags | Enabled | Errors | Actions |
|--------------------------------|------------------------------------|------------------------------------|------------|----------------|---|---------|--------|-------------|
| 1f6afb10-4e26-11ea-b2ac-af425c | External Endpoint Group: my-sec-1g | Internal Endpoint Group: my-sec-1g | 2 | Autoreolving | NSXT, Balcluster-3976-4f28-b50d-006008bde4c | On | | [C] [X] [I] |
| 2ac03ac0-28c9-11ea-b243-052a2b | Internal Endpoint Group: my-sec-1g | External Endpoint Group: my-sec-1g | 1 | | NSXT, Balcluster-3976-4f28-b50d-006008bde4c | On | | [C] [X] [I] |
| 2bb28a00-28c9-11ea-b243-052a2b | External Endpoint Group: my-sec-1g | Internal Endpoint Group: my-sec-1g | 1 | | NSXT, Balcluster-3976-4f28-b50d-006008bde4c | On | | [C] [X] [I] |
| 29c6300-28c9-11ea-b243-052a2b | Security Zone: NSXT | External Endpoint Group: my-sec-1g | 1 | | NSXT, Balcluster-3976-4f28-b50d-006008bde4c | On | | [C] [X] [I] |
| 2863900-28c9-11ea-b243-052a2b | Security Zone: NSXT | Internal Endpoint Group: my-sec-1g | 1 | | NSXT, Balcluster-3976-4f28-b50d-006008bde4c | On | | [C] [X] [I] |
| 92208c90-4e23-11ea-b2ac-af425c | External Endpoint Group: my-sec-1g | Internal Endpoint Group: my-sec-1g | 1 | | NSXT, Balcluster-3976-4f28-b50d-006008bde4c | On | | [C] [X] [I] |
| 71912f90-4e28-11ea-92b4-99b759 | External Endpoint Group: my-sec-1g | Internal Endpoint Group: my-sec-1g | 1 | | NSXT, Balcluster-3976-4f28-b50d-006008bde4c | On | | [C] [X] [I] |

12.3.4.11 Limitations

- No scale testing supported.
- This feature is only supported with NSX-T Data Center version: 2.4.1.
- Rule translation: Any to Any rule is not supported, having such a rule will result in enforcement failure. As explained before, ANY is translated to the specified security zone. If there is an Any to Any rule in NSX-T, the proxy server tries to translate it to a GBP policy with source and destination groups both being the same security zone as specified, which is not supported by AOS and will not pass policy validation, hence the translation will fail.

- Limited EVPN support: source/destination in NSX-T security policy rule is translated into AOS GBP internal/external endpoint group by searching against Virtual Networks within Security Zone if Security Zone is provided. If not provided, all Virtual Networks across Security Zones are searched against, if only one matched, the matched one will be used for internal/external determination. If more than one matched, error is thrown, which results in enforcement failure.

12.4 Other Guides

12.4.1 Apstra IBA Getting Started Tutorial

This tutorial gives the Apstra Customer with an existing, deployed AOS Blueprint instructions to get started with AOS Intent Based Analytics (IBA) with pre-defined probes. In addition, this document provides instructions on configuring the “sending” of IBA information from AOS to other systems using protocols such as Syslog.

There are two steps to be taken before installing the IBA Probes:

1. Install AOS CLI. This CLI is an Apstra development effort to provide with extract functionalities to the Apstra Server, including Analytics.
2. Install Custom Collector package. This is required in order to collect Telemetry data from the devices.

12.4.1.1 Install AOS CLI

To install AOS CLI please follow the *AOS CLI step by step guide*.

12.4.1.2 Install Custom Collector Package

To use IBA probes, you must install custom collector packages with the device system agent. These are installed via the web interface. Extract the specific platform Custom Collector (Cumulus, NXOS, Arista) Custom Collector: `aosstdcollectors_custom_<platform>-0.1.0.post<version>-py2-none-any.whl` file from the latest AOS SDK package downloaded from the portal <https://support.juniper.net/support/downloads/?p=apstra-fabric-conductor>.

In addition, if the environment does not have Internet access, manually add the following Python library package files that are requirements for the Custom Collector. They can be downloaded from the python official repository. Please make sure to download the correct version as described below:

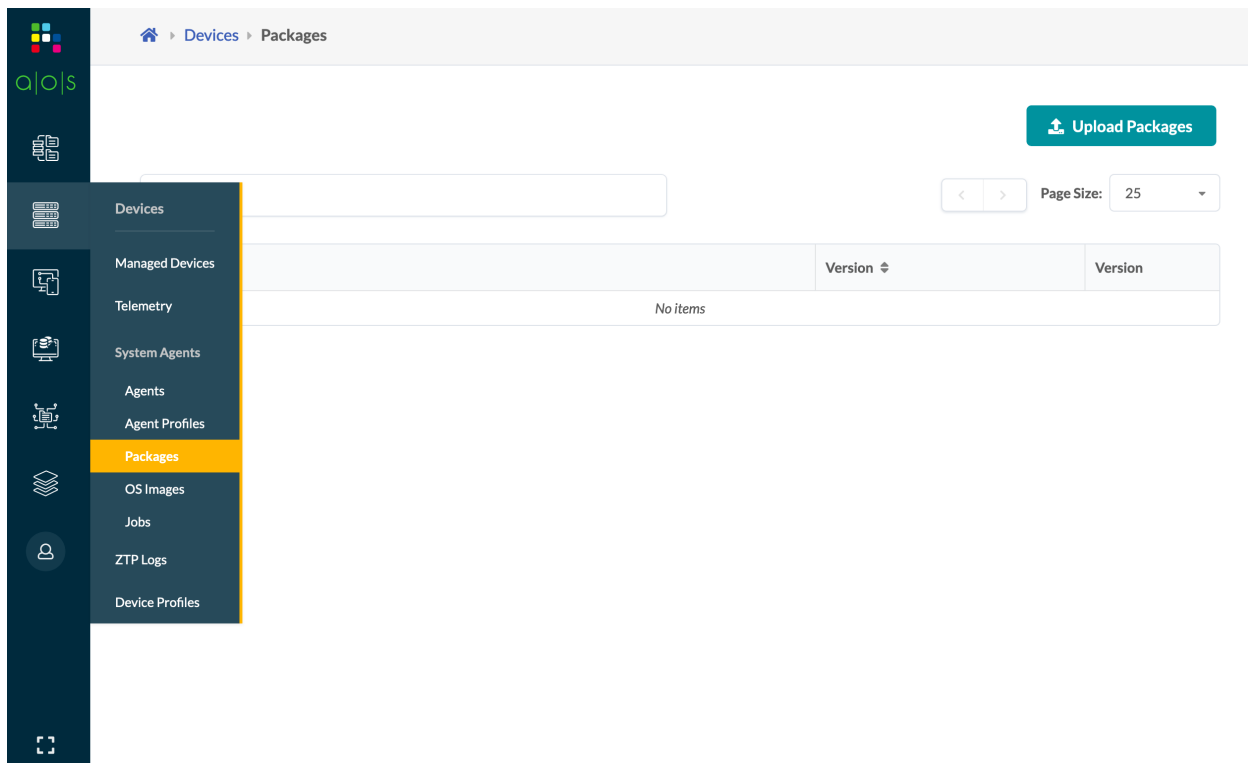
- `netaddr-0.7.17-py2.py3-none-any.whl`
- `gtextfsm-0.2.1.tar.gz`
- `pyeapi-0.8.2.tar.gz`

These three packages are installed in the Apstra Server as well as in the selected devices along with the platform custom Collector as plugins (they can be found in the device folder `/tmp/plugins`).

To upload these packages to the Apstra Server, navigate to **Devices > System Agents > Packages**.

If you have manually uploaded all the required packages, you can verify that they have been properly installed by checking on Devices / Agents and clicking on the specific device’s agent that you would like to check.

From the Config section you can verify what you have selected to be installed, and scrolling down, in the Telemetry section, you can verify what it has been installed.



Note: If Apstra has Internet connection you will not be able to verify that netaddr, gtextfsm and pyeapi has been installed from the UI, only the custom Collector could be checked. You can verify it by logging into your device and checking on the /tmp/plugins folder as described above.

As an example for Arista EOS with an Apstra Server without Internet connection (otherwise the above three packages - netaddr, gtextfsm, pyeapi - will be automatically installed) click the “Upload Packages” button and select the “netaddr-0.7.17-py2.py3-none-any.whl”, “gtextfsm-0.2.1.tar.gz”, “pyeapi-0.8.2.tar.gz”, and “aosstdcollectors_custom_eos-0.1.0.post10-py2-none-any.whl” files.

Click “Upload” and the Package files will be uploaded to the Apstra Server.

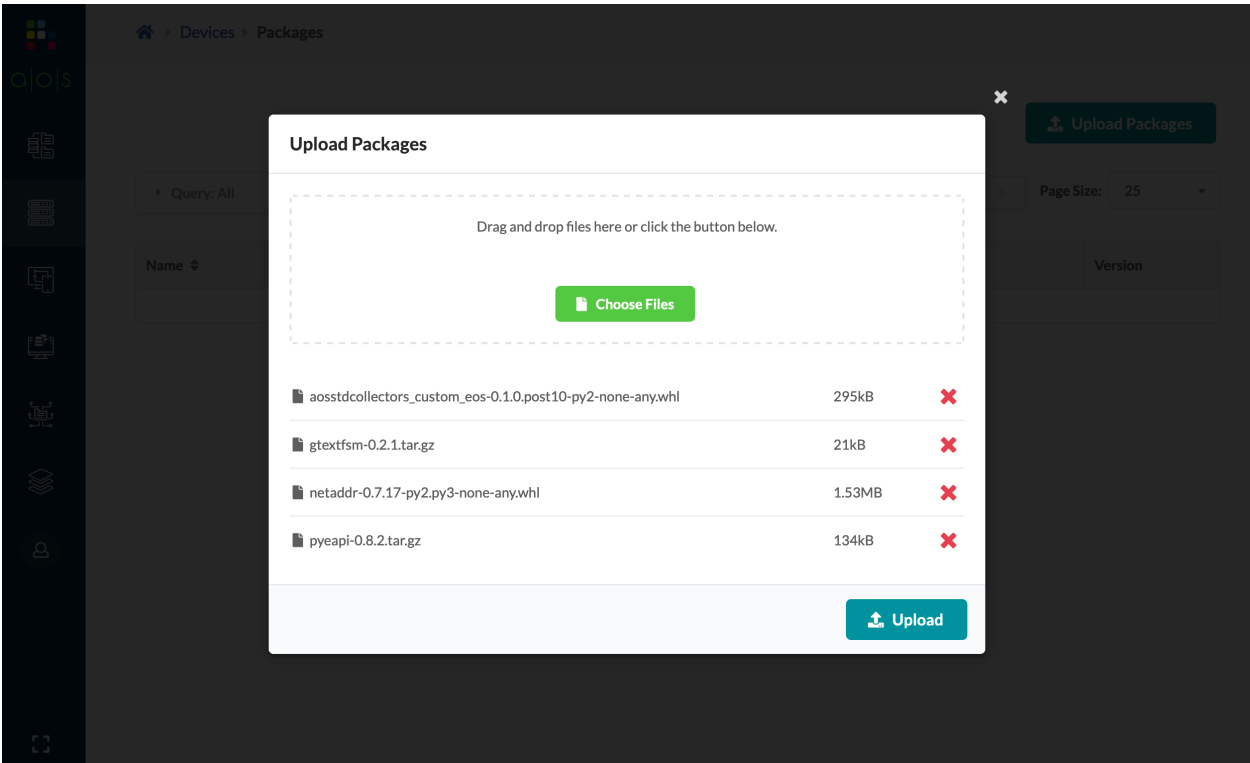
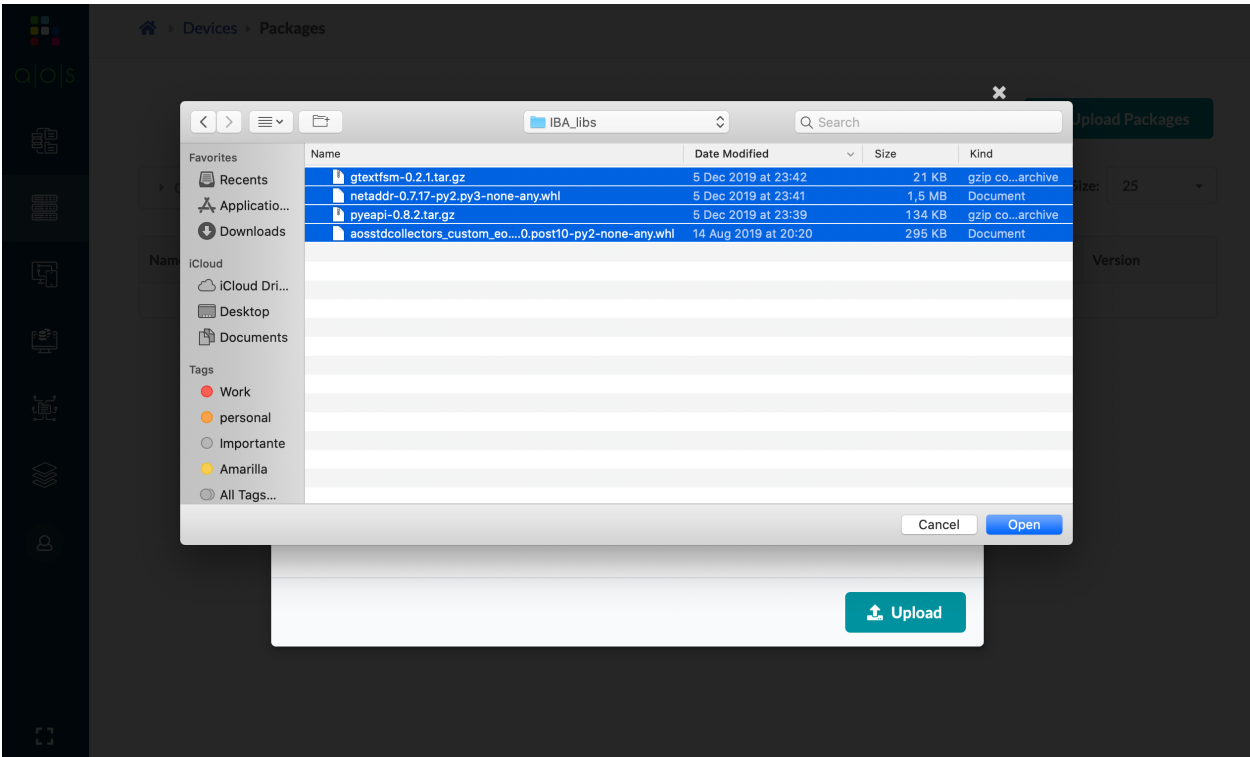
To enable the package on the Device the two packages need to be enabled for a new Agent Profile. To add a new Agent Profile go to Devices / System Agents / Agent Profiles. Click “Create Agent Profile”. Enter “eos” for the “Platform” field. Since the usernames and passwords are already set, these fields do not need to be enabled.

Scroll down and under “Packages” the four packages that were uploaded will be available. Check all four packages to associate these with the Agent Profile then click “Create”.

After the required System Agent profile is created with the Custom Collector Package, it needs to be mapped to each System Agent. To enable the package on the Device, from Devices / System Agents / Agent select the device and under the device, provide appropriate Agent Profile beneath System Agent Profile option as per the screenshot below:

Note: It is recommended that when a System Agent is created, the appropriate System Agent Profile is mapped to enable Custom Collector Package. This helps take care of package update (i.e addition of more packages to profile) or installation etc. on all the associated System Agents by editing just one Agent profile.

You can verify now that the packages has been successfully installed from the Devices / Agents tab.



Devices > Agent Profiles

Create Agent Profile

Profile Parameters

Name *
EOS-IBA

Platform

Username
☐ Set username?

Password
☐ Set password?

Open Options 0

| Key | Value |
|------------|-------|
| No options | |

☐ Create Another? **Create**

Devices > Agent Profiles

Create Agent Profile

| Key | Value |
|------------|-------|
| No options | |

Add an option

Packages 4

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

| <input checked="" type="checkbox"/> | Name ↕ | Version ↕ |
|-------------------------------------|-----------------------------|--------------|
| <input checked="" type="checkbox"/> | aosstdcollectors-custom-eos | 0.1.0.post10 |
| <input checked="" type="checkbox"/> | gtextfsm | 0.2.1 |
| <input checked="" type="checkbox"/> | netaddr | 0.7.17 |
| <input checked="" type="checkbox"/> | pyeapi | 0.8.2 |

4 selected

☐ Create Another? **Create**

Config will display the packages that the user has specified to be installed.

Telemetry will show the packages that have been correctly installed.

For more information about System Agents and Agent Profiles, please refer to the [Agent Profile Documentation](#).

Alternatively, the following AOS-CLI procedure to “bulk” update Device System Agents with an Agent Profile can be used.

12.4.1.3 Bulk Update System Agent Profile

In AOS-CLI (build 423 or later), there is a bulk edit option to “update” the Agent Profile in System Agents based on IP/ID or OS type (os_type) (i.e “eos”).

Use the “system-agents update-profile” command to update System Agents by IP range with a specific Agent Profile.


When setting the --profile option, AOS-CLI shows available Agent Profiles. Use the up and down arrow keys to select.








```
aos> system-agents update-profile --ip 172.20.120.6-11 --profile
EOS-IBA eos
```

For example.








```
aos> system-agents update-profile --ip 172.20.120.6-11 --profile 692bb0bb-c5e0-4d7e-
a70c-c24b0d5650a8
Successfully updated agent 172.20.120.9 with given profile
```

(continues on next page)


a|o|s

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 aosstdcollectors-custom-eos==0.1.0.post10 |

Status


a|o|s

Home > Devices > Agents > 80f490f0-b909-435e-93ce-c6f28176a5d2

| Job ID ↕ | Job Type ↕ | State ↕ | Started ↕ | Finished ↕ | Log |
|----------|------------|---------|----------------------|----------------------|---|
| 7 | INSTALL | SUCCESS | 2020-03-16, 01:38:16 | 2020-03-16, 01:43:02 |  |
| 6 | INSTALL | SUCCESS | 2020-03-14, 09:12:23 | 2020-03-14, 09:17:08 |  |
| 5 | INSTALL | SUCCESS | 2020-03-14, 07:15:59 | 2020-03-14, 07:20:00 |  |
| 4 | UNINSTALL | SUCCESS | 2020-03-14, 07:14:20 | 2020-03-14, 07:15:46 |  |
| 3 | CHECK | SUCCESS | 2020-03-14, 07:09:56 | 2020-03-14, 07:10:32 |  |
| 2 | INSTALL | SUCCESS | 2020-03-13, 02:22:15 | 2020-03-13, 02:27:22 |  |
| 1 | INSTALL | SUCCESS | 2020-03-09, 04:18:45 | 2020-03-09, 04:22:38 |  |

Telemetry Status

| | |
|--------------------|--|
| Status Message | Packages installed successfully |
| Packages Installed | pyeapi==0.8.2 netaddr==0.7.17 gtextfsm==0.2.1 aosstdcollectors-custom-eos==0.1.0.post10 |

(continued from previous page)

```

Successfully updated agent 172.20.120.6 with given profile
Successfully updated agent 172.20.120.11 with given profile
Successfully updated agent 172.20.120.7 with given profile
Successfully updated agent 172.20.120.10 with given profile
Successfully updated agent 172.20.120.8 with given profile
aos>

```

12.4.1.4 Apstra IBA Probes

IBA probes can be implemented using two methods:

- Create them or instantiate predefined ones from the web interface. See [Probes](#) for more information.
- Install them from AOS CLI as described below.

Probe Installation

The pre-defined IBA Probes in this document are installed via the Apstra AOS-CLI utility.

All probes described in this document are included in AOS-CLI build 412 or later. Probe .j2 files may be made available if the probe file is not built into the AOS-CLI build.

Some of these IBA Probes require an updated service registry. Download the latest AOS SDK and extract the `json-schemas.tar.gz` file. Copy the file to the `/home/admin` directory of the Apstra Server so it is available in the AOS-CLI `/mytmp` directory.

```

aos> service-registry import-from --file /mytmp/json-schemas.tar.gz
Successfully imported service registry entry for interface_details
Successfully imported service registry entry for route_count
Successfully imported service registry entry for multicast_groups
Successfully imported service registry entry for sfp
Successfully imported service registry entry for resource_usage
Successfully imported service registry entry for mlag_domain
Successfully imported service registry entry for stp
Successfully imported service registry entry for vtep_counters
Successfully imported service registry entry for vlan
Successfully imported service registry entry for evpn_type5
Successfully imported service registry entry for ping
Successfully imported service registry entry for vxlan_info
Successfully imported service registry entry for pim_neighbor_count
Successfully imported service registry entry for lldp_details
Successfully imported service registry entry for evpn_type3
Successfully imported service registry entry for multicast_info
Successfully imported service registry entry for bgp_vrf
Successfully imported service registry entry for traceroute
Successfully imported service registry entry for vrf
Successfully imported service registry entry for table_usage
Successfully imported service registry entry for vxlan_address_table
Successfully imported service registry entry for acl_stats
Successfully imported service registry entry for device_info
Successfully imported service registry entry for power_supply
Successfully imported service registry entry for interface_buffer
Successfully imported service registry entry for pim_rp
Successfully imported service registry entry for anycast_rp
Successfully imported service registry entry for bgp_iba

```

(continues on next page)

(continued from previous page)

```
Successfully imported service registry entry for interface_iba
aos>
```

Use the `probe create` AOS-CLI command to create the IBA Probes. You will be prompted for additional options.

```
aos> probe create
--blueprint          Id of the blueprint
--file              Filename of json file with probe data. Choose from dropdown or
↳ specify custom path
--skip-service-check [Optional] By default, required telemetry services are checked
↳ and enabled on target
--check-status       [Optional] Wait for probe to become operational. Default: False
--service-interval   When skip-service-check is False and service is not
↳ alreadypresent, this indicates
```

Use `--blueprint` and tab-completion to select the Blueprint ID.

```
aos> probe create --blueprint 67cd936d-c2de-49f8-8708-df465f0cdc68
                               L2 Virtual  two_stage_l3clos
```

Using `--file` and tab-completion will list available probes supplied with AOS-CLI. Scroll through the list with the up and down arrow keys.

```
aos> probe create --blueprint 67cd936d-c2de-49f8-8708-df465f0cdc68 --file
                                                                    evpn.j2
                                                                    sfp.j2
                                                                    memory_usage_
↳ threshold_anomalies.j2
                                                                    bandwidth_
↳ utilization_history.j2
                                                                    power_supply_
↳ anomalies.j2
                                                                    virtual_infra_
↳ vlan_mismatch.j2
                                                                    hardware_vtep_
↳ counters_enabled.j2
```

Some probes will need additional Probe template variables.

```
aos> probe create --blueprint 67cd936d-c2de-49f8-8708-df465f0cdc68 --file /usr/local/
↳ lib/python2.7/site-packages/aos_cli/resources/probes/memory_usage_threshold_
↳ anomalies.j2
--skip-service-check [Optional] By default, required telemetry services are checked
↳ and enabled on target
--check-status       [Optional] Wait for probe to become operational. Default: False
--service-interval   When skip-service-check is False and service is not
↳ alreadypresent, this indicates
--process            Probe template variable
--os_family          Probe template variable
```

IBA Probes Examples

The following section describes the process to install some of the most interesting probes which are not available by default.

Packet Drops

Packet drop IBA probes detect an abnormal amount of packet drops on interfaces of devices managed by Apstra based on interface telemetry collected by Device System Agents.

| Filename | Description |
|--------------------------|---|
| pkt_discard_anomalies.j2 | Detect Fabric interfaces having sustained packet discards |

To install the `pkt_discard_anomalies.j2` IBA Probe:

```
aos> probe create --blueprint 67cd936d-c2de-49f8-8708-df465f0cdc68 --file /usr/local/
↳lib/python2.7/site-packages/aos_cli/resources/probes/pkt_discard_anomalies.j2
Ensuring needed telemetry services for probe are enabled...
Successfully created probe f472ba21-d60f-44dc-9f5d-8318c8b9c07b in blueprint 67cd936d-
↳c2de-49f8-8708-df465f0cdc68
aos>
```

SFP Optics

SFP optic IBA probes detect high and/or low warning thresholds in SFP RX power, TX power, temperature, voltage, or current for compatible optical modules on interfaces of devices managed by Apstra based on SFP telemetry collected by Device System Agents.

| File-name | Description |
|-----------|---|
| sfp.j2 | Detect high and/or low warning thresholds in SFP RX Power, TX Power, Temperature, Voltage, or Current |

To install the `sfp.j2` IBA Probe:

```
aos> probe create --blueprint 67cd936d-c2de-49f8-8708-df465f0cdc68 --file /usr/local/
↳lib/python2.7/site-packages/aos_cli/resources/probes/sfp.j2
Ensuring needed telemetry services for probe are enabled...
Enabled service sfp on device l2-virtual-002-leaf1:172.20.60.11
Enabled service sfp on device l2-virtual-001-leaf1:172.20.60.9
Enabled service sfp on device spine2:172.20.60.8
Enabled service sfp on device spine1:172.20.60.6
Enabled service sfp on device l2-virtual-003-leaf1:172.20.60.10
Enabled service sfp on device l2-virtual-004-leaf1:172.20.60.7
Successfully created probe b0c32a46-636c-4f82-b026-e9925b696625 in blueprint 67cd936d-
↳c2de-49f8-8708-df465f0cdc68
aos>
```

Switch Memory Leak

Switch Memory Leak IBA probes detect abnormal memory leaks in specified processes on devices managed by AOS based on system telemetry collected by Device System Agents.

| 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 |

Note: The Switch Memory Leak IBA probes require device user credentials set in the Device System Agent configuration that has login and access to the device BASH prompt.

Note: This IBA probe is only for Arista EOS devices.

The `memory_usage_threshold_anomalies.j2` IBA Probe requires additional “Probe template variables” for `os_family` and `process`.

```
aos> probe create --blueprint 67cd936d-c2de-49f8-8708-df465f0cdc68 --file /usr/local/
↳lib/python2.7/site-packages/aos_cli/resources/probes/memory_usage_threshold_
↳anomalies.j2
  --skip-service-check [Optional] By default, required telemetry services are_
↳checked and enabled on target
  --check-status       [Optional] Wait for probe to become operational. Default:_
↳False
  --service-interval   When skip-service-check is False and service is not_
↳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`.

```
aos> probe create --blueprint 67cd936d-c2de-49f8-8708-df465f0cdc68 --file /usr/local/
↳lib/python2.7/site-packages/aos_cli/resources/probes/memory_usage_threshold_
↳anomalies.j2 --os_family eos --process fastcapi
Ensuring needed telemetry services for probe are enabled...
Enabled service resource_usage on device l2-virtual-002-leaf1:172.20.60.11
Enabled service resource_usage on device l2-virtual-001-leaf1:172.20.60.9
Enabled service resource_usage on device spine2:172.20.60.8
Enabled service resource_usage on device spine1:172.20.60.6
Enabled service resource_usage on device l2-virtual-003-leaf1:172.20.60.10
Enabled service resource_usage on device l2-virtual-004-leaf1:172.20.60.7
Successfully created probe 6a258d83-1053-42ad-935c-0550cc500b7d in blueprint 67cd936d-
↳c2de-49f8-8708-df465f0cdc68
aos>
```

```
aos> probe create --blueprint rack-based-blueprint-10990707 --file /usr/local/lib/
↳python2.7/site-packages/aos_cli/resources/probes/memory_usage_threshold_anomalies.
↳j2 --os_family eos --process configagent
Ensuring needed telemetry services for probe are enabled...
Successfully created probe ed2c6be1-b4b1-4e1b-bd07-da431e89eeec in blueprint rack-
↳based-blueprint-10990707
aos>
```

Note: “FastCapi” as service process is valid only for EOS version 4.18. For the newer version of EOS i.e 4.20 and later only ConfigAgent is valid. Please 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, redo the `probe create` command for the other process.

The IBA Probe name may be modified to include the process name.

To install the `system_memory_usage_threshold_anomalies.j2` IBA Probe:

```
aos> probe create --blueprint 67cd936d-c2de-49f8-8708-df465f0cdc68 --file /usr/local/
↳lib/python2.7/site-packages/aos_cli/resources/probes/system_memory_usage_threshold_
↳anomalies.j2
Ensuring needed telemetry services for probe are enabled...
Successfully created probe a669ccf8-cba7-414b-ad46-a7d4b4ca3928 in blueprint 67cd936d-
↳c2de-49f8-8708-df465f0cdc68
aos>
```

Fault Tolerance

| File-name | Description |
|-----------------------------|--|
| spine_fault_tolerance.j2 | Monitors 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 | Monitors 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. |

Note: These (2) probes require AOS-CLI build 430 or later.

To install the `spine_fault_tolerance.j2` IBA Probe:

```
aos> probe create --blueprint bf7a322c-ee3a-4dcf-aa20-df0560f538da --file /usr/local/
↳lib/python2.7/site-packages/aos_cli/resources/probes/spine_fault_tolerance.j2 --
↳number_of_faulty_spines_to_be_tolerated 1
Successfully created probe 0f0e9bf7-d9b3-43d7-906e-a9f0675e68f2 in blueprint bf7a322c-
↳ee3a-4dcf-aa20-df0560f538da
aos>
```

Note: `number_of_faulty_spines_to_be_tolerated` has to be specified.

To install the `lag_link_fault_tolerance.j2` IBA Probe:

```
aos> probe create --blueprint bf7a322c-ee3a-4dcf-aa20-df0560f538da --file /usr/local/
↳lib/python2.7/site-packages/aos_cli/resources/probes/lag_link_fault_tolerance.j2
Successfully created probe 45ce5fe8-555f-41a9-b0ae-267125669d3f in blueprint bf7a322c-
↳ee3a-4dcf-aa20-df0560f538da
aos>
```

After all the IBA Probes are installed, they will be available in the Blueprint under **Analytics**.

12.4.1.5 Sending IBA Info with Syslog

See *Syslog Configuration* for details about using Syslog to send messages to Syslog servers.

0 selected

| Name | Anomalies | State | Updated by | Tags | Enabled | Actions |
|--|--------------|-------------|-------------------------|------|-------------------------------------|--|
| Device system health | No anomalies | Operational | System a day ago | | <input checked="" type="checkbox"/> | Edit Copy Delete |
| ECMP Imbalance (Fabric Interfaces) | No anomalies | Operational | admin 9 minutes ago | | <input checked="" type="checkbox"/> | Edit Copy Delete |
| Headroom | No anomalies | Operational | admin a few seconds ago | | <input checked="" type="checkbox"/> | Edit Copy Delete |
| Hypervisor MTU threshold check | No anomalies | Operational | System a day ago | | <input checked="" type="checkbox"/> | Edit Copy Delete |
| Hypervisor and Fabric LAG config mismatch | No anomalies | Operational | System a day ago | | <input checked="" type="checkbox"/> | Edit Copy Delete |
| Hypervisor and Fabric Vlan config mismatch | No anomalies | Operational | System a day ago | | <input checked="" type="checkbox"/> | Edit Copy Delete |
| Hypervisor redundancy checks | No anomalies | Operational | System a day ago | | <input checked="" type="checkbox"/> | Edit Copy Delete |
| VMs without Fabric configured VLANs | No anomalies | Operational | System a day ago | | <input checked="" type="checkbox"/> | Edit Copy Delete |

Active Tasks: 0

12.4.2 Enable VXLAN Routing on Cumulus Tomahawk

Cumulus devices equipped with Tomahawk and Tomahawk+ ASICs do not natively support Routing In and Out of Tunnels (RIOT). Therefore, switch ports for VXLAN routing must be configured to use hyperloop interfaces. Hyperloop interfaces recirculate packets through the ingress pipeline. As packets enter the VXLAN tunnel they are encapsulated, as they exit the VXLAN tunnel they are decapsulated.

To set up the hyperloop interface, you'll create a device profile, make sure your logical device has an unused port (as applicable), create an interface map, update your managed devices, assign the new device profile to the device, and deploy the device.

12.4.2.1 Creating a Device Profile for Hyperloop

You can create a device profile from scratch, but it's more efficient to *clone* an existing one and change details that are different.

1. Choose a device profile to clone from that is based on the device to be used for the hyperloop interface, and add 'hyperloop' to the name to distinguish it.

Devices / Device Profiles

Create Device Profile

Search criteria: Manufacturer = Accton or Edgecore

1-5 of 5 Page Size: 25

Clone Device Profile

Summary

Label *
Edgecore AS7712-32X-hyperloop

Number of slots *
0

Start from ID
0

Create

- In the **Ports** section of the device profile that you are cloning, click the port that you want as the hyperloop port.

Clone Device Profile

Summary

Selector

Capabilities

Ports

Panel #1 Select port(s) to fill in the details

Port breakout Autonegotiation

Edit selected port #32

Connector type *
qsfp28

Transformations

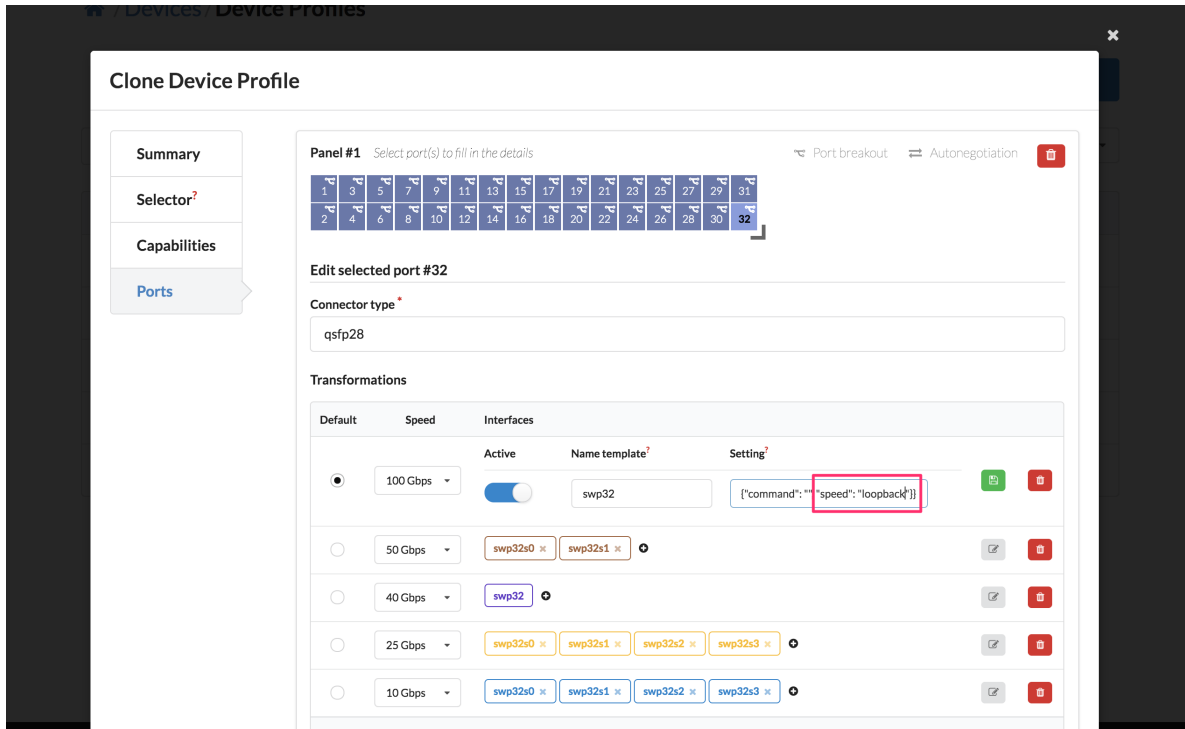
| Default | Speed | Interfaces |
|----------------------------------|----------|---------------------------------|
| <input checked="" type="radio"/> | 100 Gbps | swp32 |
| <input type="radio"/> | 50 Gbps | swp32s0 swp32s1 |
| <input type="radio"/> | 40 Gbps | swp32 |
| <input type="radio"/> | 25 Gbps | swp32s0 swp32s1 swp32s2 swp32s3 |
| <input type="radio"/> | 10 Gbps | swp32s0 swp32s1 swp32s2 swp32s3 |

Add new transformation

Warning: The remaining ports must be set to their native / default port speeds. If they are not, RIOT (and therefore VXLAN routing) will not work. For example, if you're using a 4 x 25G SFP28 breakout and you configure ports 1 and 2 for hyperloop, then ports 3 and 4 must be set to the default of 25G. If you set them to 10G, you will have connectivity, but hyperloop will not work.

3. In the **Transformation** section, click the **Edit** button corresponding to that port.
4. In the **Settings** field, change the value for “speed” to "loopback", then click **Create**.

In the example below, for “full speed” transformation (100Gbps on QSFP28 port), the value for “speed” was changed from the default speed of "100G" to "loopback".

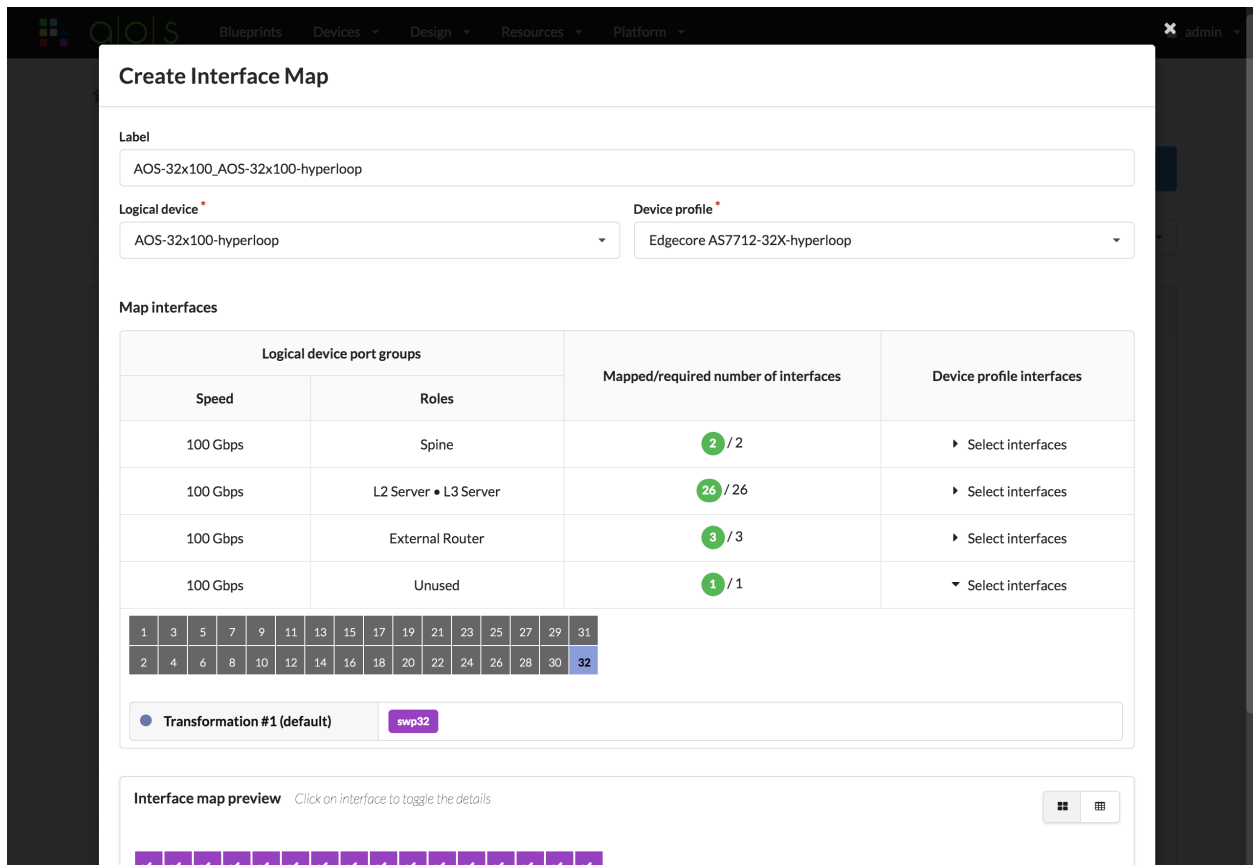
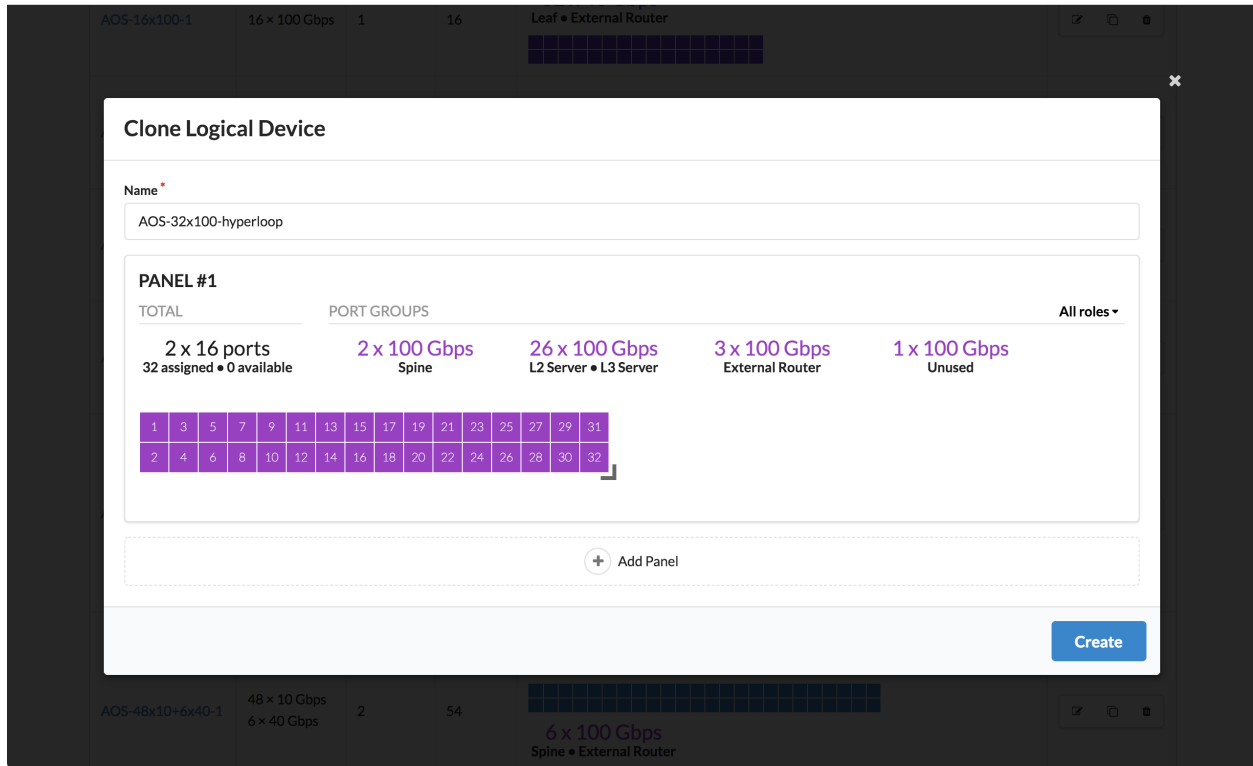


12.4.2.2 Updating Logical Device for Hyperloop

If you have a *logical device* that includes the hyperloop port, make sure the port is configured for an **Unused** role.

12.4.2.3 Creating an Interface Map for Hyperloop

Create an *Interface Map* using the new device profile.



12.4.2.4 Updating Managed Devices for Hyperloop

1. Navigate to the managed device that will use hyperloop.

o|o|s Blueprints Devices Design Resources Platform admin

/ Devices / Managed Devices / 771232X1633033

Info Telemetry

Expanded View Compact View

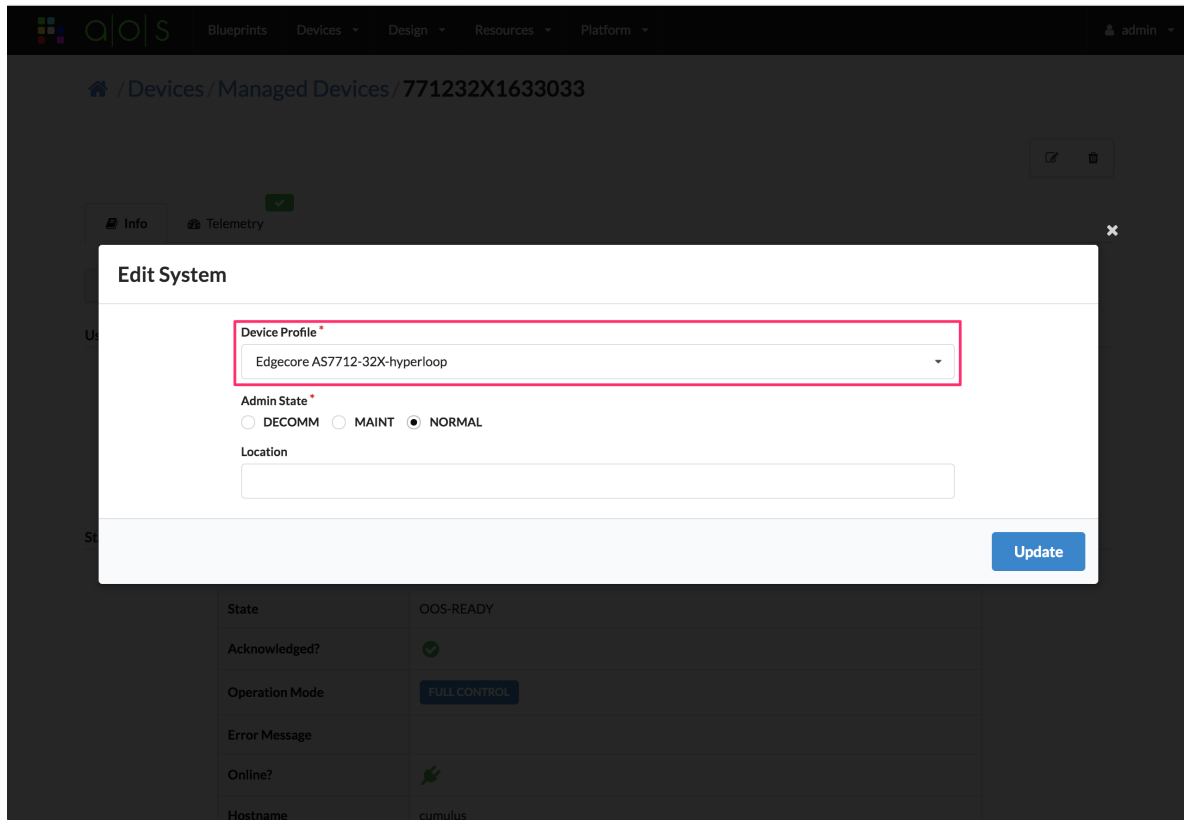
User Config

| | |
|----------------|---------------------|
| Device Profile | Edgecore AS7712-32X |
| Admin State | normal |
| Location | |

Status

| | |
|----------------|--------------|
| State | OOS-READY |
| Acknowledged? | ✓ |
| Operation Mode | FULL CONTROL |
| Error Message | |
| Online? | ✓ |
| Hostname | cumulus |

2. Click the **Edit** button (top-right) and select the new hyperloop device profile from the drop-down list.



3. Click **Update** to save your changes.

12.4.2.5 Assigning Hyperloop Device Profile

In the blueprint, *assign the new device profile* to the appropriate device, then deploy it.

12.4.2.6 Verifying Loopback Port

After deploying the device, verify that the hyperloop port is set to “loopback” by looking at the device file `/etc/cumulus/ports.conf`.

```
admin@border-1-leaf:mgmt-vrf:~$ sudo cat /etc/cumulus/ports.conf
sudo: unable to resolve host border-1-leaf
1=100G
2=100G
3=100G
4=100G
5=100G
6=100G
7=100G
8=100G
9=100G
10=100G
11=100G
12=100G
13=100G
14=100G
```

(continues on next page)

(continued from previous page)

```
15=100G
16=100G
17=100G
18=100G
19=100G
20=100G
21=100G
22=100G
23=100G
24=100G
25=100G
26=100G
27=100G
28=100G
29=100G
30=100G
31=100G
32=loopback
admin@border-1-leaf:mgmt-vrf:~$
```

12.4.3 Adding Access Layer Switches

As of AOS version 3.3.0, Apstra includes the possibility to create and manage access switches in data center networks.

You can follow our existing guidelines for most of the stages, there are some more options and design considerations to take into account as described in the following points.

Important: 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, please refer to the *Juniper Apstra Technology Previews* page or contact *Juniper Support*.

12.4.3.1 Design

Logical Devices

A new role has been introduced for ports facing access switches, **access** Role.

Create port group

Number of ports *

24

1

24

Speed *

10 Gbps

Connected To *

- ☐ Superspine
- ☐ Spine
- ☐ Leaf
- ☒ Access
- ☐ L2 Server
- ☐ L3 Server
- ☐ External Router
- ☐ Peer
- ☐ Unused

Create Port Group

Configure this role on leaf switches facing an access switch, and configure leaf and server ports in the access switch logical device.

To learn about how to create logical devices please refer to our logical device documentation [logical devices](#).

Rack Types

Once the logical devices with the required access ports have been created and the interface map is defined, you can create a new rack and add the access switches.

Create Rack Type

1. Select Access Switches

Leafs

Access Switches

Servers

Topology

Logical Devices

Access Switch

Name *

access_switch1

Instance count (includes linked server groups) *

1

Logical Device *

AOS-8x10-1

+

Add link

+

Add new access switch

leaf_1_1

access_switch1_1

2. Configure Access Switch properties

Specify Access Switch name

Number of access switches

Configure links between Access Switch and Leaf

Add new switch type

☐ Create Another?

Create

To configure the access switch links, click **Add link** as shown in the above picture. Access switches can be single-homed only.

Name the link connection and select LAG Mode or No LAG and specify the number of links.

Link

Name *

access-switch1_leaf1

Leaf *

Leaf1

LAG Mode

☐ LACP (Active) ?

☐ LACP (Passive) ?

☐ Static LAG (no LACP) ?

☒ No LAG ?

Physical link count per individual switch (8 available) *

1

Link speed *

10 Gbps

To learn more about Rack Types please refer to our rack types documentation [rack types](#).

Templates

To create a template with access switches please follow our templates documentation [templates](#).

Configlets

Access switches configlet is not supported as of AOS version 3.3.0. Please contact our support or sales team if you require configlets in access switches.

12.4.3.2 Blueprints

Blueprints are mostly built in the same way as in a regular data center network with no access switches. You can follow our documentation [blueprint creation](#).

As of AOS version 3.3.0 the following operations can be performed in the access switches.

To learn more about AOS operations please visit our documentation [Staged](#).

Changing Link Speed

As in the leaf switches towards the servers, AOS allows you to change the link speed between an access switch and a leaf or a server.

1. From the blueprint, navigate to **Staged / Physical / Links**, then click the **Change link speeds** button.
2. Select the desired link speed.
3. Commit the change in the blueprint.

The screenshot shows the Apstra AOS interface. The breadcrumb navigation is: Blueprints > access-switch_BP > Staged > Physical > Build > Resources. The main menu includes Dashboard, Analytics, Staged, Uncommitted, Active, and Time Voyager. The Physical section is active, showing a 'Links' tab. A 'Change link speeds' button is highlighted in a red box. The interface also shows a 'Selection' panel on the right with a 'Build' button and a list of resources: ASNs - Spines (2/2), ASNs - Leafs (2/2), Loopback IPs - Spines (2/2), Loopback IPs - Leafs (2/2), and Link IPs - Spines<->Leafs (8/8). Below the 'Change link speeds' button, there is a table with columns: Name, Role, Speed, Logical Link, Port Channel ID, Endpoint 1 (Name, Role, Interface, IPv4, IPv6), and Name. The table contains three rows of data for access switches and one row for a spine switch.

| Name | Role | Speed | Logical Link | Port Channel ID | Endpoint 1 | Name |
|--|-----------------------|-------|--------------|-----------------|-----------------------|-------|
| rack_access_001_leaf1<->rack_access_001_access1(access-leaf1)[1] | Leaf to Access Switch | 10G | access-leaf1 | N/A | rack_access_001_leaf1 | Leaf |
| rack_access_002_leaf1<->rack_access_002_access1(access-leaf1)[1] | Leaf to Access Switch | 10G | access-leaf1 | N/A | rack_access_002_leaf1 | Leaf |
| spine1<->rack_access_001_access1(access-leaf1)[1] | Spine | 10G | N/A | N/A | spine1 | Spine |

Change Link Speeds

| 0 selected | Name | Role | Speed | Logical Link | Port Channel ID | Endpoint 1 | | | Endpoint 2 | | |
|--------------------------|--|-----------------------|---------|--------------|-----------------|-----------------------|------|-----------|-------------------------|--------|-----------|
| | | | | | | Name | Role | Interface | Name | Role | Interface |
| <input type="checkbox"/> | rack_access_001_leaf1<->rack_access_001_access1(access-leaf1)[1] | Leaf to Access Switch | 10 Gbps | access-leaf1 | N/A | rack_access_001_leaf1 | Leaf | xe-0/0/2 | rack_access_001_access1 | Access | xe-0/0/0 |
| <input type="checkbox"/> | rack_access_002_leaf1<->rack_access_002_access1(access-leaf1)[1] | Leaf to Access Switch | 10 Gbps | access-leaf1 | N/A | rack_access_002_leaf1 | Leaf | xe-0/0/2 | rack_access_002_access1 | Access | xe-0/0/0 |

100 Mbps
1 Gbps
10 Gbps

Update

Adding New Link

You can add an access-server or leaf-access link similar to adding a link between a leaf and a server.

1. From the blueprint, navigate to **Staged / Physical / Topology**.
2. Select the desired device.
3. Click **Add links** and select the configuration.
4. Commit the change in the blueprint.

You will be able to change the link design in the access switch towards the leaf or the server.

The screenshot displays the Apstra GUI interface for managing network topology. The top navigation bar includes tabs for Dashboard, Analytics, Staged, Uncommitted, Active, and Time Voyager. The 'Staged' tab is selected, and the 'Physical' view is active. The 'Topology' section shows a link between 'rack_access_001_leaf1' and 'rack_access_001_access1'. The 'Add links' button is highlighted in red. The right sidebar shows the configuration for 'rack_access_001_access1', including the 'Deploy Mode' (S/N) and 'Device Info' (Management IP: 172.20.100.11, OS: Junos 19.4R1.10, Operation Mode: FULL CONTROL).

Select the desired ports as well as the LAG mode.

Add Access Switch to Leaf Links ✕

Select devices and their interfaces to create a link:

Leaf *

rack_access_001_leaf1 ✕

Leaf: rack_access_001_leaf1
Device profile: Juniper vQFX

1 2 3 4 5 6 7 8 9 10 11 12

Port #2 Tr. #1 (10 Gbps, default)

xe-0/0/1

Access: rack_access_001_access1
Device profile: Juniper vQFX

1 2 3 4 5 6 7 8 9 10 11 12

Links

Logical Link Name* ⓘ *

access-leaf1 ✕

LAG Mode *

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

1-1 of 1 < >

| Type | Speed | Access Switch | | Logical Link | LAG Mode | Leaf / Se |
|----------|-------|-------------------------|-----------|--------------|----------|--------------------|
| | | Name | Interface | | | Name |
| Existing | 10G | rack_access_001_access1 | xe-0/0/0 | access-leaf1 | No LAG | rack_access_001_le |

Add

Deleting Link

From the same view as before where you have selected the device:

1. Change the **Neighbors view** to **Links view**.
2. Click the **Delete** button (trash can).
3. Commit in the blueprint.

Selected Rack rack_access_001 ✕

Selected Node rack_access_001_access1 (Access) ✕

Topology Label Name ▼

Neighbors View
Links View

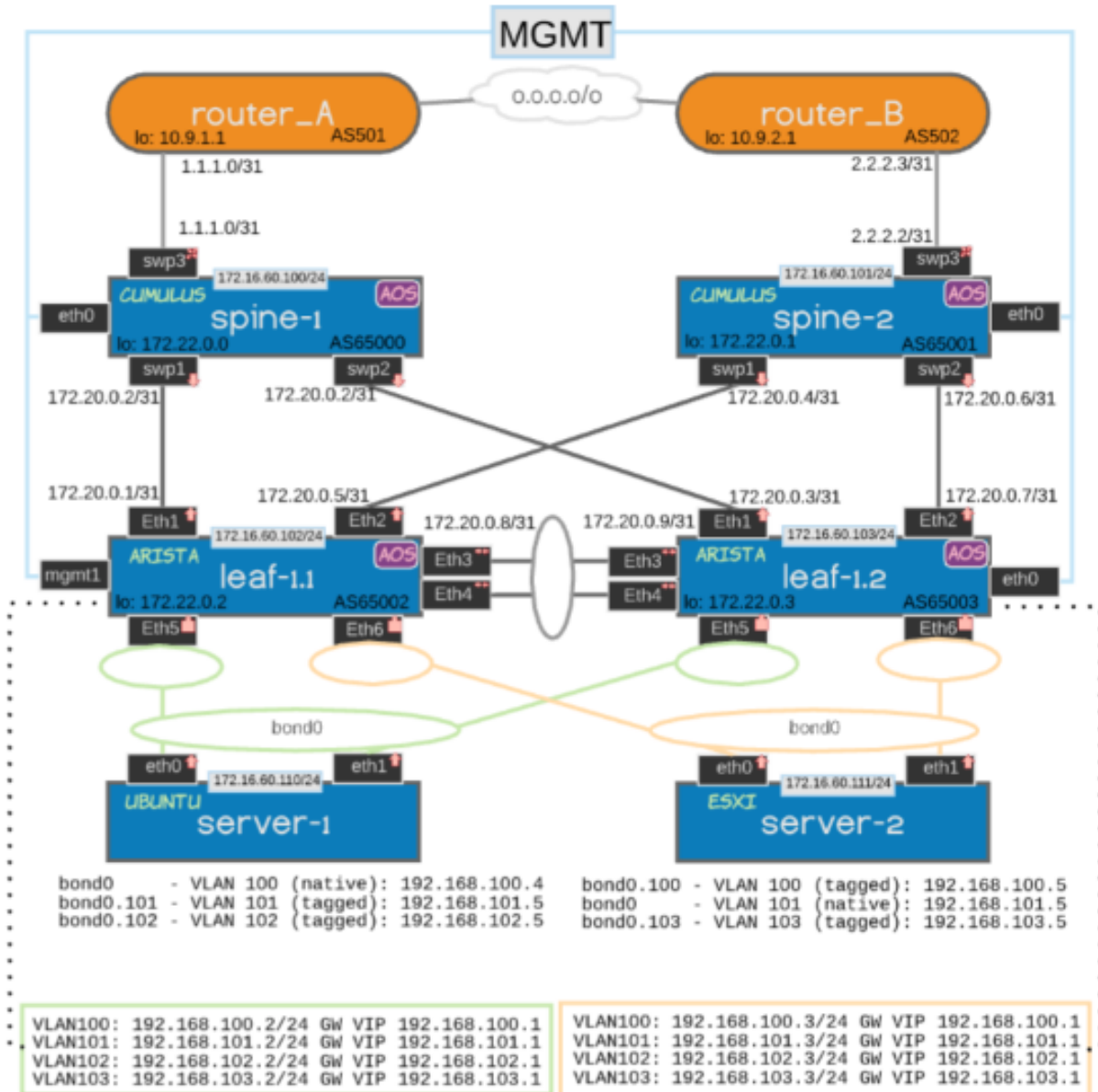
+ Add links
 Edit links

▸ Query: All
1-1 of 1 < >
Page Size: 25 ▼

| | Name | Role | Speed | Logical Link ⓘ | Port Channel ID | Endpoint 1 | | | | Endpoint 2 | | | | Actions |
|--------------------------|--|-----------------------|-------|-----------------------------|-----------------|-------------------------|--------|-----------|----------|-----------------------|------|-----------|----------|---------------|
| | | | | | | Name | Role | Interface | Lag Mode | Name | Role | Interface | Lag Mode | |
| <input type="checkbox"/> | rack_access_001_leaf1<->rack_access_001_access1[access-leaf1][1] | Leaf to Access Switch | 10G | access-leaf1 ⓘ | N/A | rack_access_001_access1 | Access | xe-0/0/0 | No LAG | rack_access_001_leaf1 | Leaf | xe-0/0/2 | No LAG | |

12.4.4 Creating a LAG on a Server

Creating a LAG on a server requires a little bit of network configuration and troubleshooting ability. This document helps to describe both.



12.4.4.1 Prerequisites

Ubuntu 14.04 LTS Installation of packages 'ifenslave' and 'vlan' on top of stock 14.04 LTS server install – apt -y install ifenslave vlan

- *ifenslave* gives ubuntu network stack ability to use 'bond' interfaces and loads bond driver when it is invoked
- *vlan* gives ubuntu the ability to specify vlans (.VLANID suffix on interfaces) and loads vlan driver when invoked

A server connected to two LACP-capable switches running intel 'e1000' network driver nics (virtio is not supported for lacp)

12.4.4.2 Server Configuration

There may be some bug/race condition that prevents bond interfaces from coming up properly in Ubuntu 14.04. Manual up/down scripts may be required for manual ifenslaving.

<https://bugs.launchpad.net/ubuntu/+source/ifenslave-2.6/+bug/14153023>

Below, this configuration contains workarounds.

Listing 14: /etc/network/interfaces

```
auto eth0
iface eth0 inet dhcp

allow-hotplug eth1
iface eth1 inet manual
    up ifenslave bond0 $IFACE
    down ifenslave -d bond0 $IFACE
    bond-master bond0

allow-hotplug eth2
iface eth2 inet manual
    up ifenslave bond0 $IFACE
    down ifenslave -d bond0 $IFACE
    bond-master bond0

auto bond0
iface bond0 inet static
    address 192.168.100.4
    netmask 255.255.255.0
    bond-mode 802.3ad
    bond-slaves eth1 eth2
    bond-lacp-rate fast
    bond-xmit_hash_policy layer3+4
    bond-miimon 100
    bond-downdelay 200
    bond-updelay 200

auto bond0.101
iface bond0.101 inet static
    address 192.168.101.4
    netmask 255.255.255.0

auto bond0.102
iface bond0.102 inet static
    address 192.168.102.4
    netmask 255.255.255.0
```

12.4.4.3 Validation

- Ensure that ‘partner mac’ is not 00:00:00:00:00:00 This implies there is no LACP peer
- Ensure that all slave interfaces are ‘up’
- Check Transmit hash policy is layer3+4 for performance
- Check LACP rate Fast on one side and slow on the other side may be an issue

```
root@ubuntu1:~# cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)

Bonding Mode: IEEE 802.3ad Dynamic link aggregation
Transmit Hash Policy: layer3+4 (1)
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0
Down Delay (ms): 0

802.3ad info
LACP rate: fast
Min links: 0
Aggregator selection policy (ad_select): stable
Active Aggregator Info:
    Aggregator ID: 1
    Number of ports: 2
    Actor Key: 33
    Partner Key: 1
    Partner Mac Address: 02:0c:29:34:0d:f2

Slave Interface: eth1
MII Status: up
Speed: 10000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 00:0c:29:72:2b:44
Aggregator ID: 1
Slave queue ID: 0

Slave Interface: eth2
MII Status: up
Speed: 10000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 00:0c:29:72:2b:4e
Aggregator ID: 1
Slave queue ID: 0
```

There are a few quick tests we can run to ensure a bond is functioning normally.

12.4.4.4 Procfs diagnostics

Listing 15: Operational state of a bond

```
root@ubuntu1:~# cat /sys/devices/virtual/net/bond0/operstate
up
```

Listing 16: Checking members of bonds

```
root@ubuntu1:~# cat /sys/devices/virtual/net/bond0/bonding/slaves
eth1 eth2
```

Listing 17: Ensuring bond is 802.3ad lacp (not ad-select or other lb)

```
root@ubuntu1:~# cat /sys/devices/virtual/net/bond0/bonding/mode
802.3ad 4
```

Listing 18: Check LACP rate

```
root@ubuntu1:~# cat /sys/devices/virtual/net/bond0/bonding/lacp_rate
fast 1
```

Listing 19: Checking bond speed (aggregate)

```
root@ubuntu1:~# cat /sys/devices/virtual/net/bond0/speed
20000
```

12.4.5 L3 Server Configurations

12.4.5.1 Introduction

Users will need to update `/etc/aos/aos.conf` file with the desired `device_profile_id` based on the server device facts which includes server version and supported interface configuration. The management interface on the server should also be mentioned in the config file.

Servers running ubuntu 16.04 or 18.04 will need special handling depending on predictable interface naming scheme feature. With this feature, the interface names on the server would be lot different than the traditional names we are used to seeing (like `eth0`, `eth1`, `eth2` ...).

AOS provides a base device profile matching the output from `ifconfig` from an Ubuntu 16.04 (Xenial) and Ubuntu 18.04 (Bionic) servers.

```

ubuntu@ubuntu: ~
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eno1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 0c:c4:7a:e5:1a:cc brd ff:ff:ff:ff:ff:ff
    inet 10.4.12.113/24 brd 10.4.12.255 scope global dynamic eno1
        valid_lft 7016sec preferred_lft 7016sec
    inet6 fe80::680d:4bc4:ddc6:4d32/64 scope link
        valid_lft forever preferred_lft forever
3: eno2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 0c:c4:7a:e5:1a:cd brd ff:ff:ff:ff:ff:ff
4: ens3f0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 90:e2:ba:ed:a3:ac brd ff:ff:ff:ff:ff:ff
5: ens3f1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 90:e2:ba:ed:a3:ad brd ff:ff:ff:ff:ff:ff
ubuntu@ubuntu:~$

```

Below is a snippet of the configuration file with these values updated for a Ubuntu 16.04 (Xenial) server based on the above ifconfig dump. (note the fields 'interface' and 'device_profile_id').

Listing 20: /etc/aos/aos.conf

```

[controller]
# <metadb> provides directory service for AOS. It must be configured properly
# for a device to connect to AOS controller.
metadb = tbt://172.20.63.3:29731
# <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 = eno1
# 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 =

[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
# 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

[logrotate]

```

(continues on next page)

(continued from previous page)

```
# 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

[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

# <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 = Generic_Server_1RU_4x10G_Xenial
```

AOS provides pre-built device profiles for servers with different interface configurations. The complete list of device profiles pre-built in AOS can be accessed from the GUI at <https://<AOS-vm-ip>/#/devices/device-profiles>. The same applies to interface maps as well, from GUI at <https://<AOS-vm-ip>/#/design/interface-maps>. The below picture illustrates a subset of device profiles available to choose from the GUI.

Query: All 1-25 of 138 Page Size: 25

| Name | Manufacturer | Hardware Model | Modular? | OS Family | OS Version | Actions |
|------------------------------|-----------------|-----------------|----------|-----------|----------------------|------------------------|
| Accton 5712-54X-O | Accton | 5712-54X-O.* | no | Cumulus | {3\,5-7}\d+} | [Edit] [Copy] [Delete] |
| Accton 6712-32X-O | Accton | 6712-32X-O.* | no | Cumulus | {3\,5-7}\d+} | [Edit] [Copy] [Delete] |
| Accton-AS5712-54X_SONIC | Edgecore Accton | 5712-54X-O.* | no | SONIC | *dirty.* | [Edit] [Copy] [Delete] |
| Accton-AS5712-54X_SONIC_BRCM | Edgecore Accton | 5712-54X-O.* | no | SONIC | *2.1.0-Generic.* | [Edit] [Copy] [Delete] |
| Accton-AS5835-54T_SONIC | Edgecore Accton | 5835-54T-O.* | no | SONIC | .* | [Edit] [Copy] [Delete] |
| Accton-AS5835-54X_SONIC | Edgecore Accton | 5835-54X-O.* | no | SONIC | .* | [Edit] [Copy] [Delete] |
| Accton-AS7712-32X_SONIC | Edgecore Accton | 7712-32X-O.* | no | SONIC | .* | [Edit] [Copy] [Delete] |
| Arista DCS-7050CX3-32S | Arista | DCS-7050CX3-32S | no | EOS | 4\.(18 20 21 22)\..* | [Edit] [Copy] [Delete] |
| Arista DCS-7050QX-32 | Arista | DCS-7050QX-32 | no | EOS | 4\.(18 20 21 22)\..* | [Edit] [Copy] [Delete] |
| Arista DCS-7050QX-32S | Arista | DCS-7050QX-32S | no | EOS | 4\.(18 20 21 22)\..* | [Edit] [Copy] [Delete] |
| Arista DCS-7050QX-32S | Arista | DCS-7050QX-32S | no | EOS | 4\.(18 20 21 22)\..* | [Edit] [Copy] [Delete] |

Following example lists 4 different types of `device_profile_ids` based on server type for a 4x1G interface configuration.

Table 1: sample `device_profile_id` and `interface_map` for 4x1G

| OS_Type | Version | Device_profile_id | Interface_map |
|-----------|---------|--------------------------------|--|
| Centos | 7.5 | Generic_Server_1RU_4x1G_Centos | Generic_Server_1RU_4x1G_Centos_AOS-4x1-1 |
| *Ubuntu | 14.04 | Generic_Server_1RU_4x1G | Generic_Server_1RU_4x1G_AOS-4x1-1 |
| **Ubuntu | 16.04 | Generic_Server_1RU_4x1G_Xenial | Generic_Server_1RU_4x1G_Xenial_AOS-4x1-1 |
| ***Ubuntu | 18.04 | Generic_Server_1RU_4x1G_Bionic | Generic_Server_1RU_4x1G_Bionic_AOS-4x1-1 |

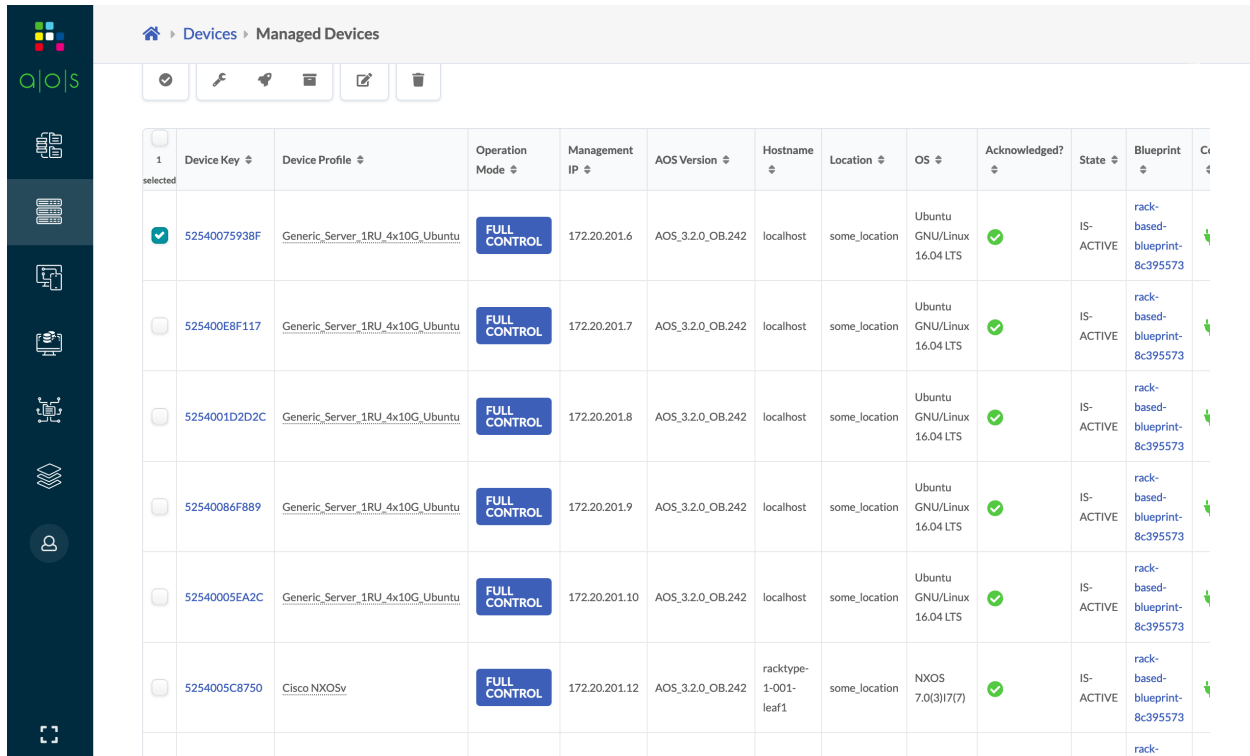
```
* Choose this device profile if running ubuntu 14.04 Trust which is the default
* Choose this device profile if running ubuntu 16.04 Xenial or
  18.04 Bionic with predictable interface naming scheme disabled
** Choose this device profile if running ubuntu 16.04 Xenial with predictable
   interface naming schema enabled and interface naming scheme matches with the
   above image's ifconfig dump
*** Choose this device profile if running ubuntu 18.04 Bionic with predictable
   interface naming schema enabled and interface naming scheme matches with the
   above image's ifconfig dump
```

Users are encouraged to create new device profiles and `interface_maps` from the GUI based on the interface names available on your servers running ubuntu 16.04 or 18.04. AOS supports different server interface map configuration which includes - 1x10G, 1x1G, 1x25G, 1x40G, 2x10G, 2x1G, 2x40G, 4x10G, 4x1G.

Attention: Once the aos.conf file has been modified with the desired fields and values, the AOS device agents need to be restarted for the new values to be reflected in the AOS controller as well as in the GUI.

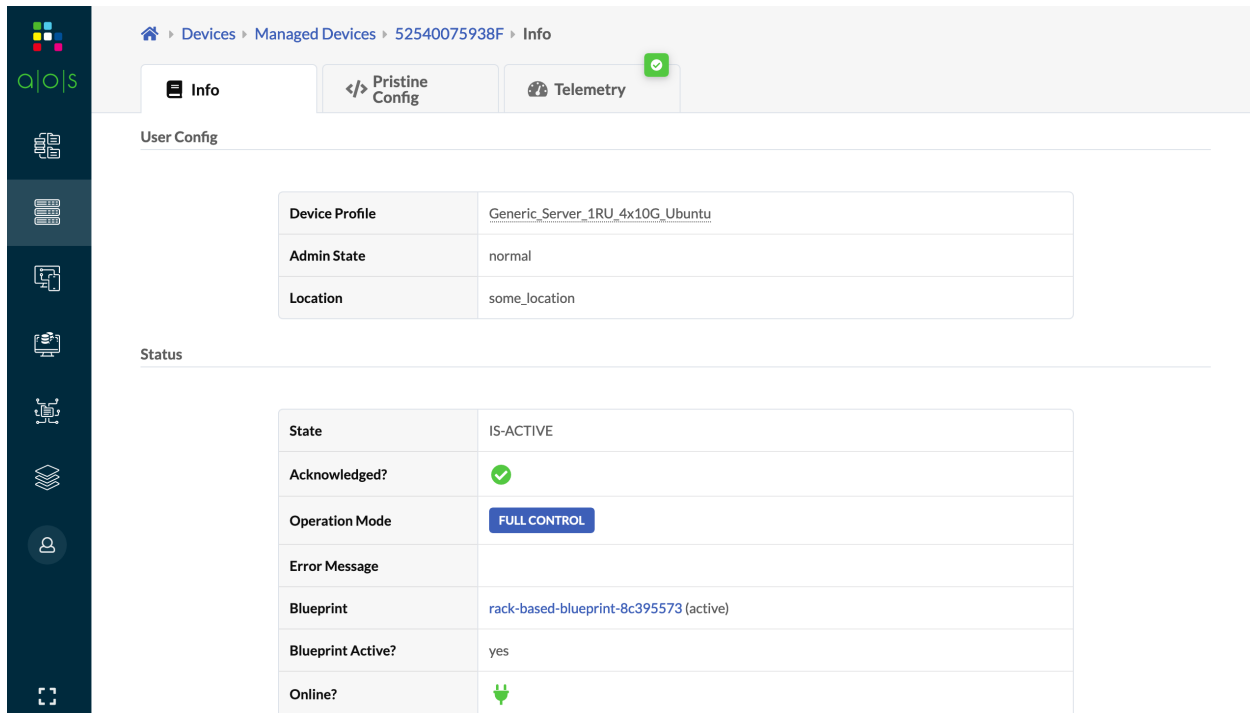
12.4.5.2 Managed Devices

Users can choose the servers from the managed devices option in the GUI and choose a specific server of interest and view the device details as shown below. In this example lets pick the second device in the list which is Ubuntu Xenial 16.04.



| 1 | Device Key | Device Profile | Operation Mode | Management IP | AOS Version | Hostname | Location | OS | Acknowledged? | State | Blueprint | Configuration |
|----------|--------------|---------------------------------|----------------|---------------|------------------|----------------------|---------------|----------------------------|---------------|-----------|-------------------------------|---------------|
| selected | 52540075938F | Generic_Server_1RU_4x10G_Ubuntu | FULL CONTROL | 172.20.201.6 | AOS_3.2.0_OB.242 | localhost | some_location | Ubuntu GNU/Linux 16.04 LTS | ✓ | IS-ACTIVE | rack-based-blueprint-8c395573 | ↓ |
| | 525400E8F117 | Generic_Server_1RU_4x10G_Ubuntu | FULL CONTROL | 172.20.201.7 | AOS_3.2.0_OB.242 | localhost | some_location | Ubuntu GNU/Linux 16.04 LTS | ✓ | IS-ACTIVE | rack-based-blueprint-8c395573 | ↓ |
| | 5254001D2D2C | Generic_Server_1RU_4x10G_Ubuntu | FULL CONTROL | 172.20.201.8 | AOS_3.2.0_OB.242 | localhost | some_location | Ubuntu GNU/Linux 16.04 LTS | ✓ | IS-ACTIVE | rack-based-blueprint-8c395573 | ↓ |
| | 52540086F889 | Generic_Server_1RU_4x10G_Ubuntu | FULL CONTROL | 172.20.201.9 | AOS_3.2.0_OB.242 | localhost | some_location | Ubuntu GNU/Linux 16.04 LTS | ✓ | IS-ACTIVE | rack-based-blueprint-8c395573 | ↓ |
| | 52540005EA2C | Generic_Server_1RU_4x10G_Ubuntu | FULL CONTROL | 172.20.201.10 | AOS_3.2.0_OB.242 | localhost | some_location | Ubuntu GNU/Linux 16.04 LTS | ✓ | IS-ACTIVE | rack-based-blueprint-8c395573 | ↓ |
| | 5254005C8750 | Cisco NXOSv | FULL CONTROL | 172.20.201.12 | AOS_3.2.0_OB.242 | racktype-1-001-leaf1 | some_location | NXOS 7.0(3)17(7) | ✓ | IS-ACTIVE | rack-based-blueprint-8c395573 | ↓ |

The user config section shows a default device profile that AOS picks which need not match exactly with the actual server capabilities. The user can edit this and choose a device profile that suitably matches their server configurations based on the guidelines in the aforementioned table “sample device_profile_id” and “interface_map for 4x1G”.



Home > Devices > Managed Devices > 52540075938F > Info

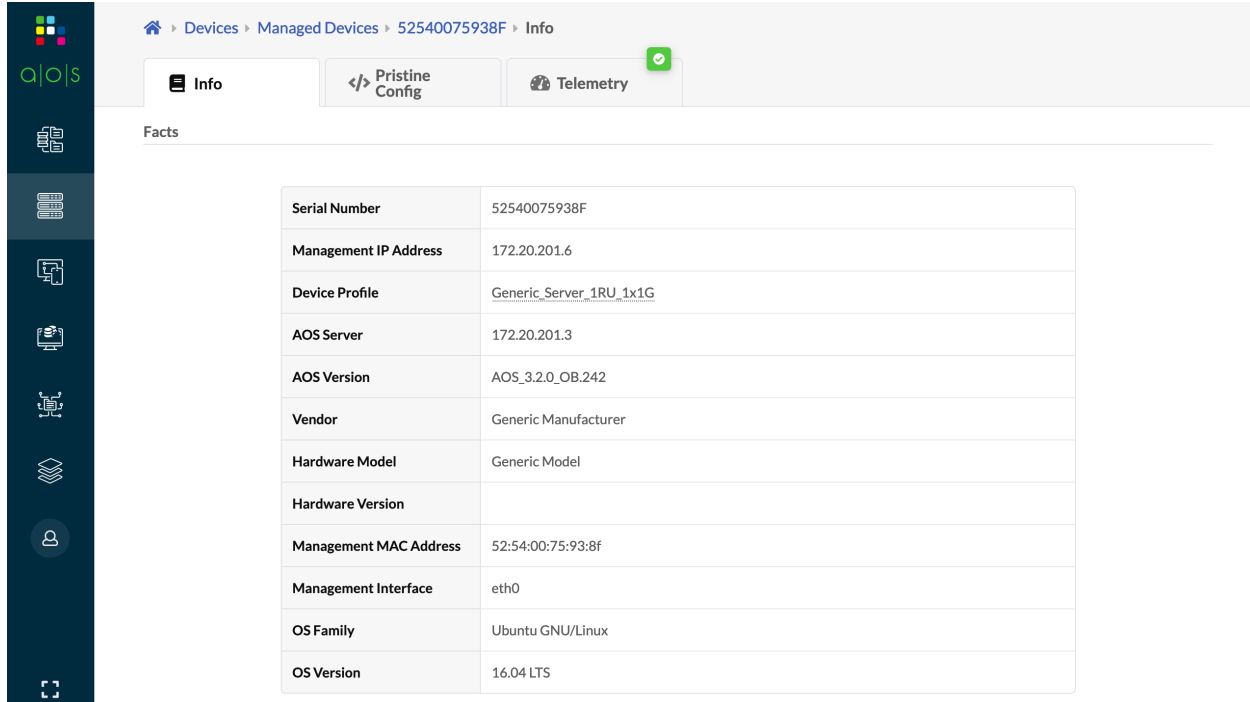
Info | Pristine Config | Telemetry

User Config

| | |
|----------------|---------------------------------|
| Device Profile | Generic_Server_1RU_4x10G_Ubuntu |
| Admin State | normal |
| Location | some_location |

Status

| | |
|-------------------|--|
| State | IS-ACTIVE |
| Acknowledged? | |
| Operation Mode | FULL CONTROL |
| Error Message | |
| Blueprint | rack-based-blueprint-8c395573 (active) |
| Blueprint Active? | yes |
| Online? | |



Home > Devices > Managed Devices > 52540075938F > Info

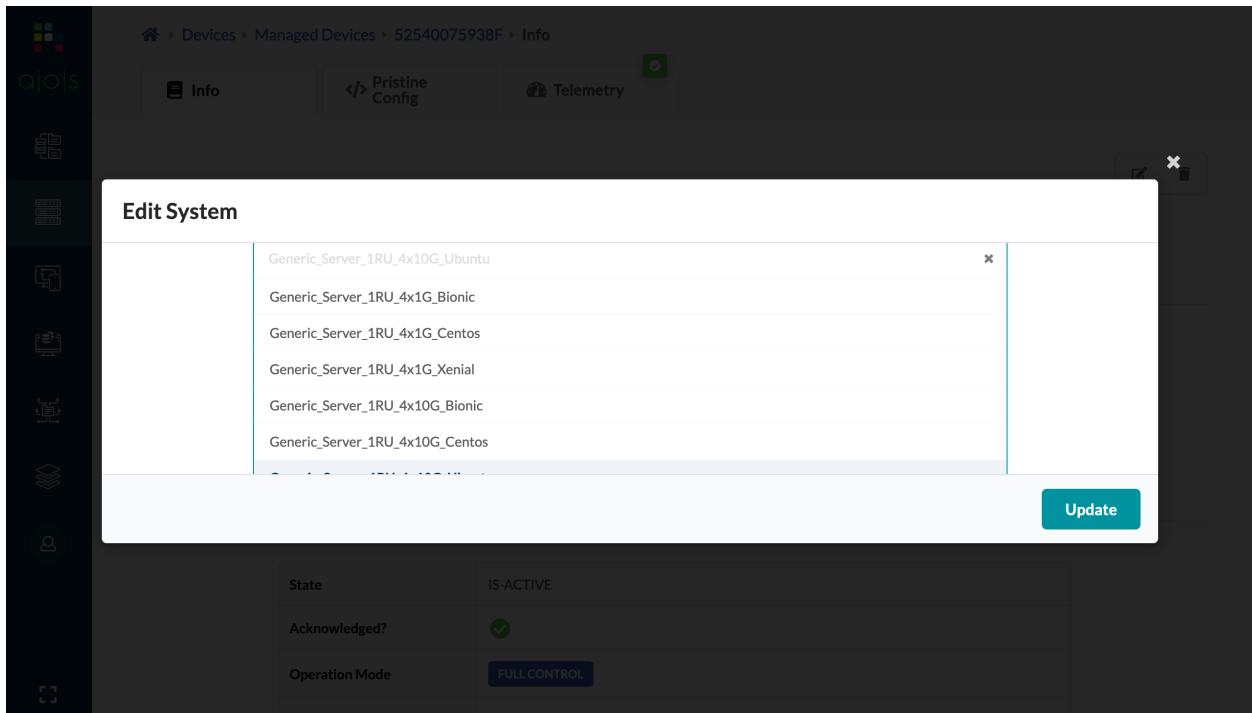
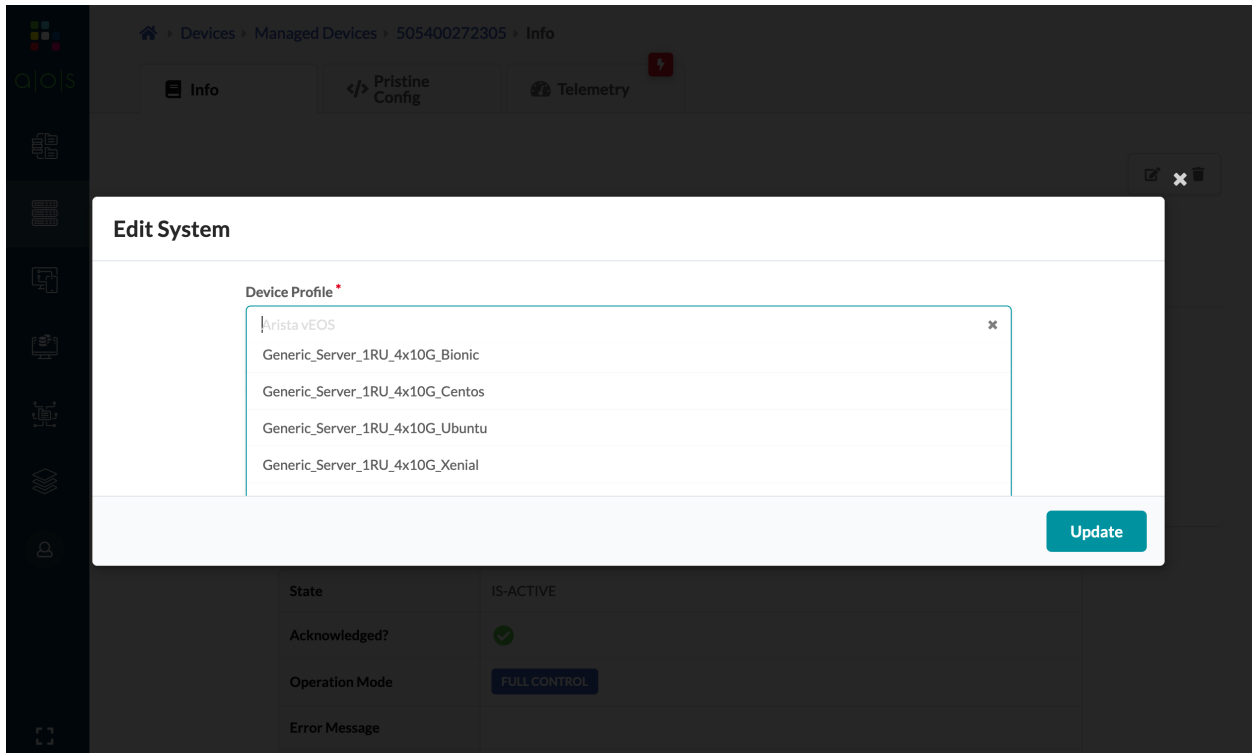
Info | Pristine Config | Telemetry

Facts

| | |
|------------------------|-------------------------|
| Serial Number | 52540075938F |
| Management IP Address | 172.20.201.6 |
| Device Profile | Generic_Server_1RU_1x1G |
| AOS Server | 172.20.201.3 |
| AOS Version | AOS_3.2.0_OB.242 |
| Vendor | Generic Manufacturer |
| Hardware Model | Generic Model |
| Hardware Version | |
| Management MAC Address | 52:54:00:75:93:8f |
| Management Interface | eth0 |
| OS Family | Ubuntu GNU/Linux |
| OS Version | 16.04 LTS |

The section following the user config is the device facts which AOS device agents have reported. If the `/etc/aos/aos.conf` file on the device has been modified with the appropriate management interface name and device_profile_id (as highlighted in the Introduction section), then the updated values will show up here.

Users can click on the edit button next to the User config section to edit the fields as shown below.



Attention: It is at a minimum required to make sure the correct management interface name is mentioned in the `/etc/aos/aos.conf` file on the server and the User config section has the correct `device_profile_id` (default value showing in AOS GUI may not match).

12.4.5.3 Cloning device-profiles

For users who want to create new device profiles from the GUI (especially for Ubuntu Xenial and Bionic servers where the target platform has completely different interface names), can follow the workflow as illustrated below.

For this discussion lets pick 4x10G Xenial server which the user wishes to change to cater to their needs of interface naming. Once the interface names are identified (as mentioned in the Introduction section), the user can choose a `Generic_Server_1RU_4x10G_Xenial` device profile from the pre-built list and choose to clone it as shown below.

Updating the device profile ports may not be allowed because it is referenced by [Generic_Server_1RU_4x10G_Xenial_AOS-4x10-1](#) interface map.

Expanded View Compact View

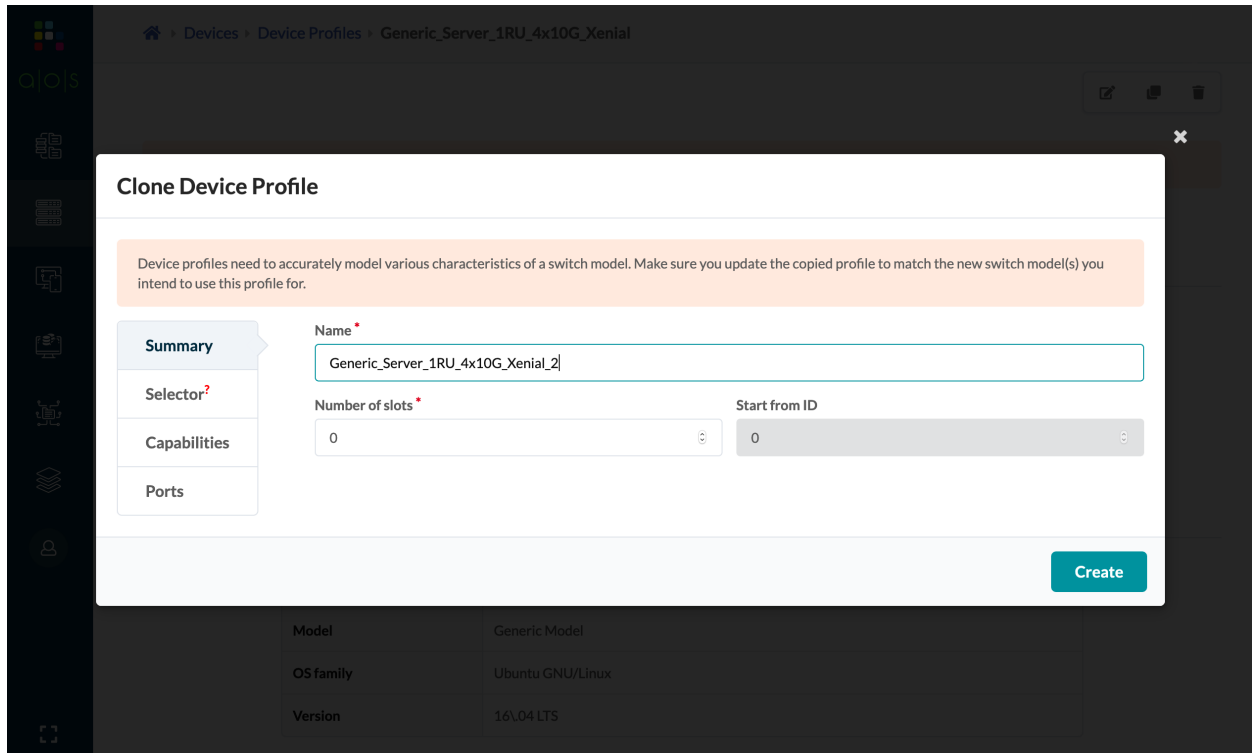
Summary

| | |
|---------------|---------------------------------|
| Name | Generic_Server_1RU_4x10G_Xenial |
| Modular? | no |
| Slot count | 0 |
| Ports preview | ■ ■ ■ ■ |

Selector[?]

| | |
|--------------|----------------------|
| Manufacturer | Generic Manufacturer |
| Model | Generic Model |
| OS family | Ubuntu GNU/Linux |
| Version | 16\04 LTS |

The user can rename the new device profile to a desired naming convention, lets say `Generic_Server_1RU_4x10G_Xenial_2`.



Clone Device Profile

Device profiles need to accurately model various characteristics of a switch model. Make sure you update the copied profile to match the new switch model(s) you intend to use this profile for.

Summary

Name ^{*}

Generic_Server_1RU_4x10G_Xenial_2

Number of slots ^{*}

0

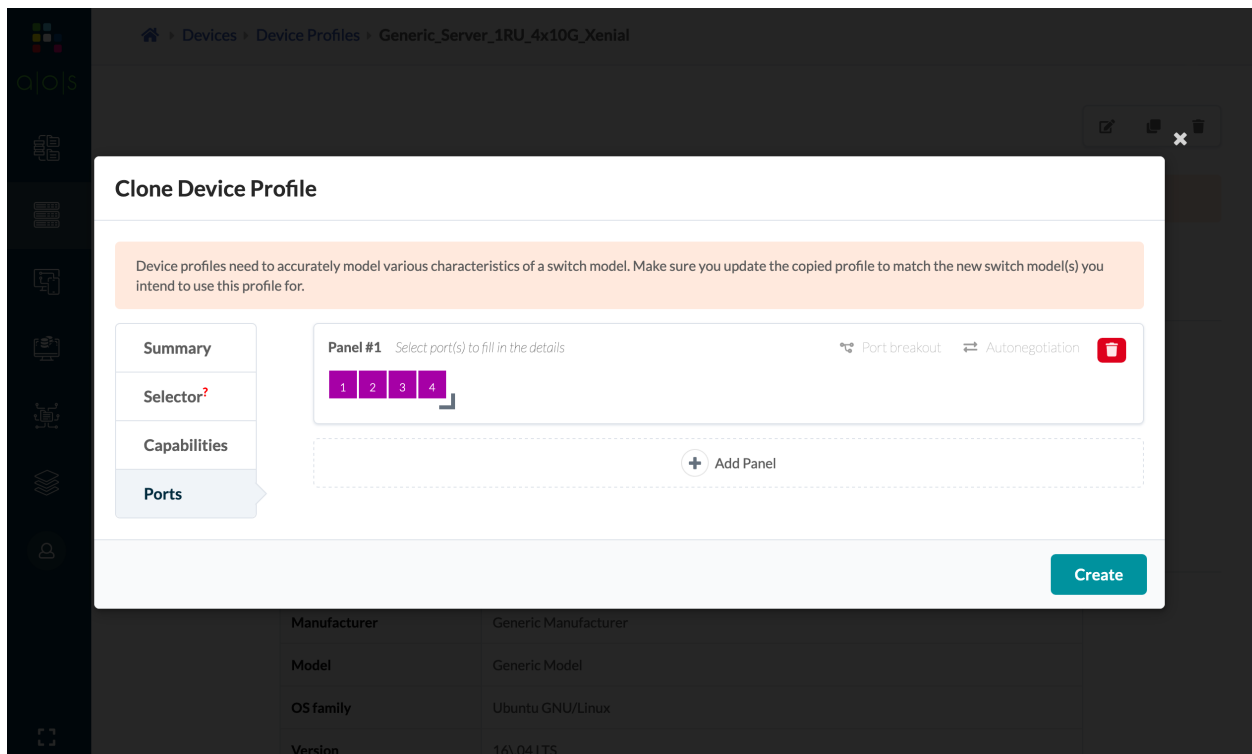
Start from ID

0

Create

| | |
|-----------|------------------|
| Model | Generic Model |
| OS family | Ubuntu GNU/Linux |
| Version | 16.04 LTS |

Now the user can edit the values of the Ports by clicking on the ports panel and then clicking on the individual port numbers (colored violet). The user can now click on the edit button to rename the interface name to a desired value, lets say `enp0s3`, and click on the save button. This step can be repeated for all 4 ports available for this 4x10G device profile. The same procedure applies to other configurations as well.



Clone Device Profile

Device profiles need to accurately model various characteristics of a switch model. Make sure you update the copied profile to match the new switch model(s) you intend to use this profile for.

Summary

Selector[?]

Capabilities

Ports

Panel #1 Select port(s) to fill in the details

Port breakout Autonegotiation

1 2 3 4

+ Add Panel

Create

| | |
|--------------|----------------------|
| Manufacturer | Generic Manufacturer |
| Model | Generic Model |
| OS family | Ubuntu GNU/Linux |
| Version | 16.04 LTS |

The newly created device_profile will show up in the list of device profiles in the GUI at <https://<AOS-vm->

ip>##/devices/device-profiles.

12.4.6 Building a Virtual Lab

You can build your own virtual lab with an instance of the Apstra server and a small group of virtual network devices. Global Support provides limited support for Customer virtual labs.

12.4.6.1 AOS Virtual Lab on EVE-NG

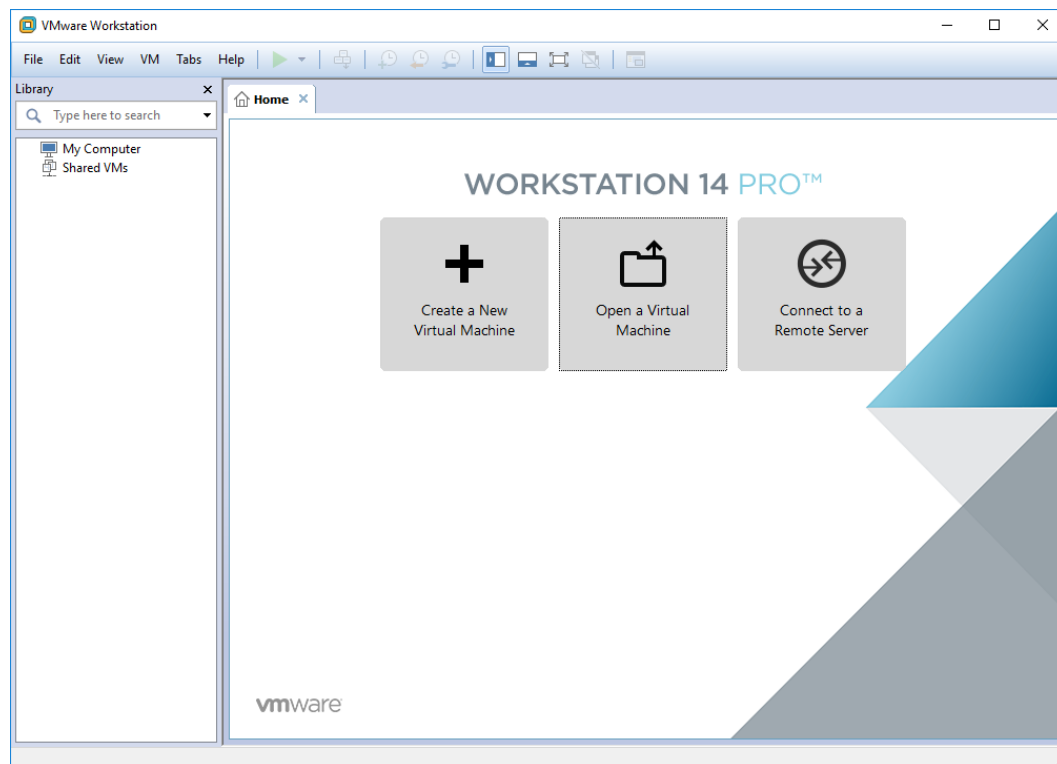
This guide describes the process to set up a virtual lab environment with Apstra AOS and virtual network devices using the Emulated Virtual Environment Next Generation (EVE-NG).

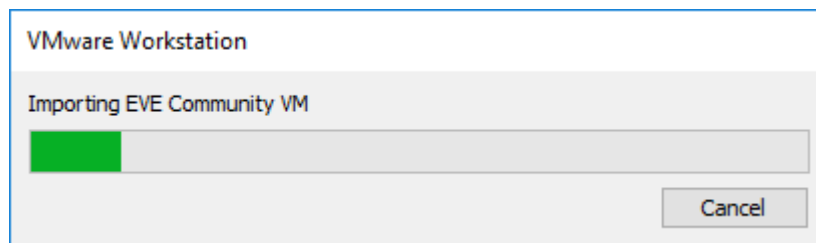
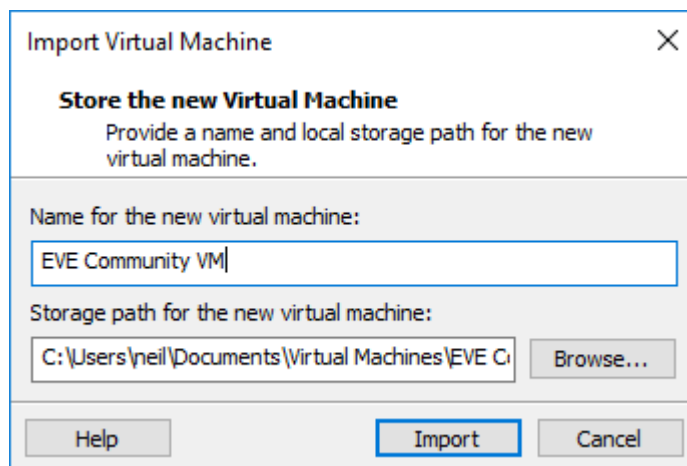
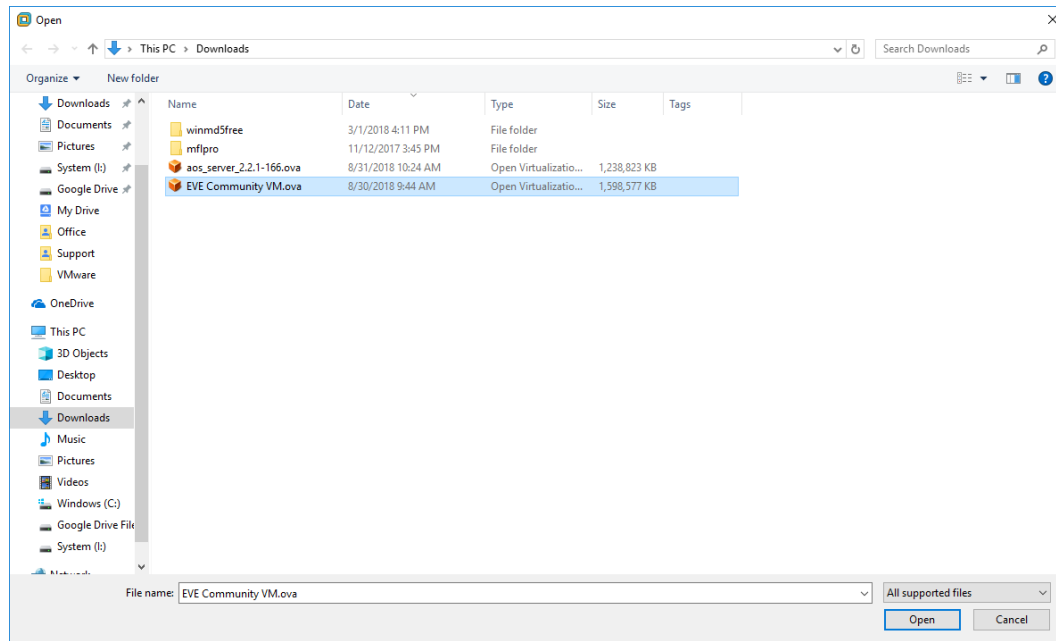
Installing EVE-NG

EVE-NG is a virtual machine which can be obtained from the EVE-NG website (<http://www.eve-ng.net/>).

The following documentation describes installing EVE-NG on VMware Workstation.

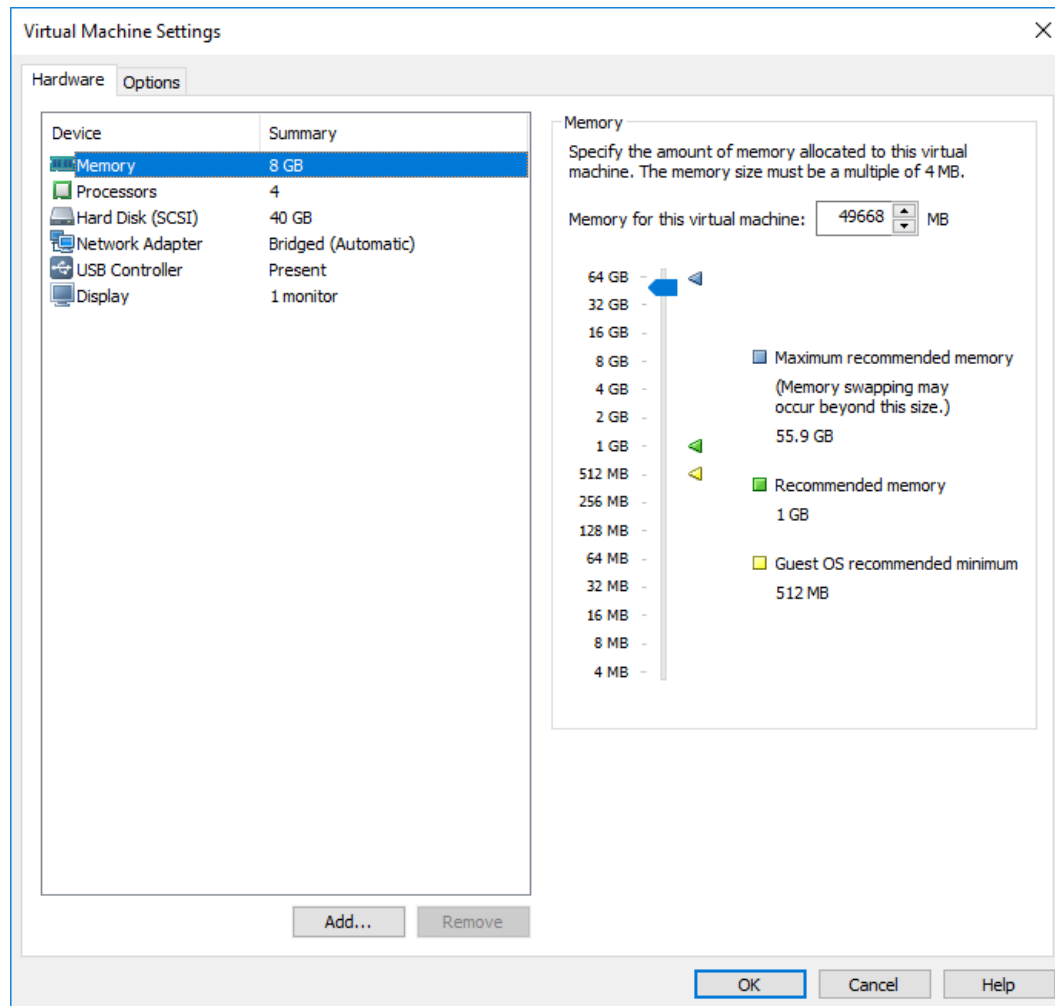
Step 1 Download the EVE-NG OVA and import it as a new Virtual Machine.





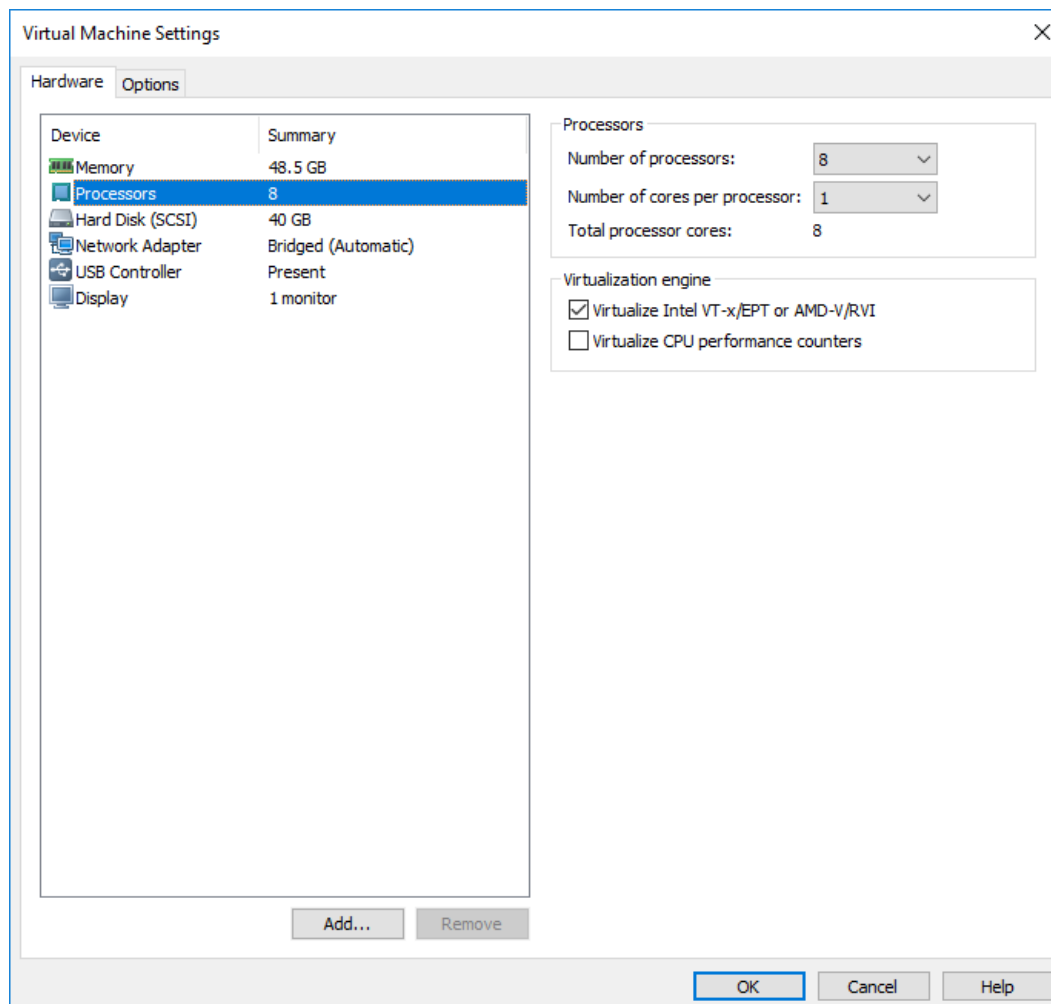
Step 2

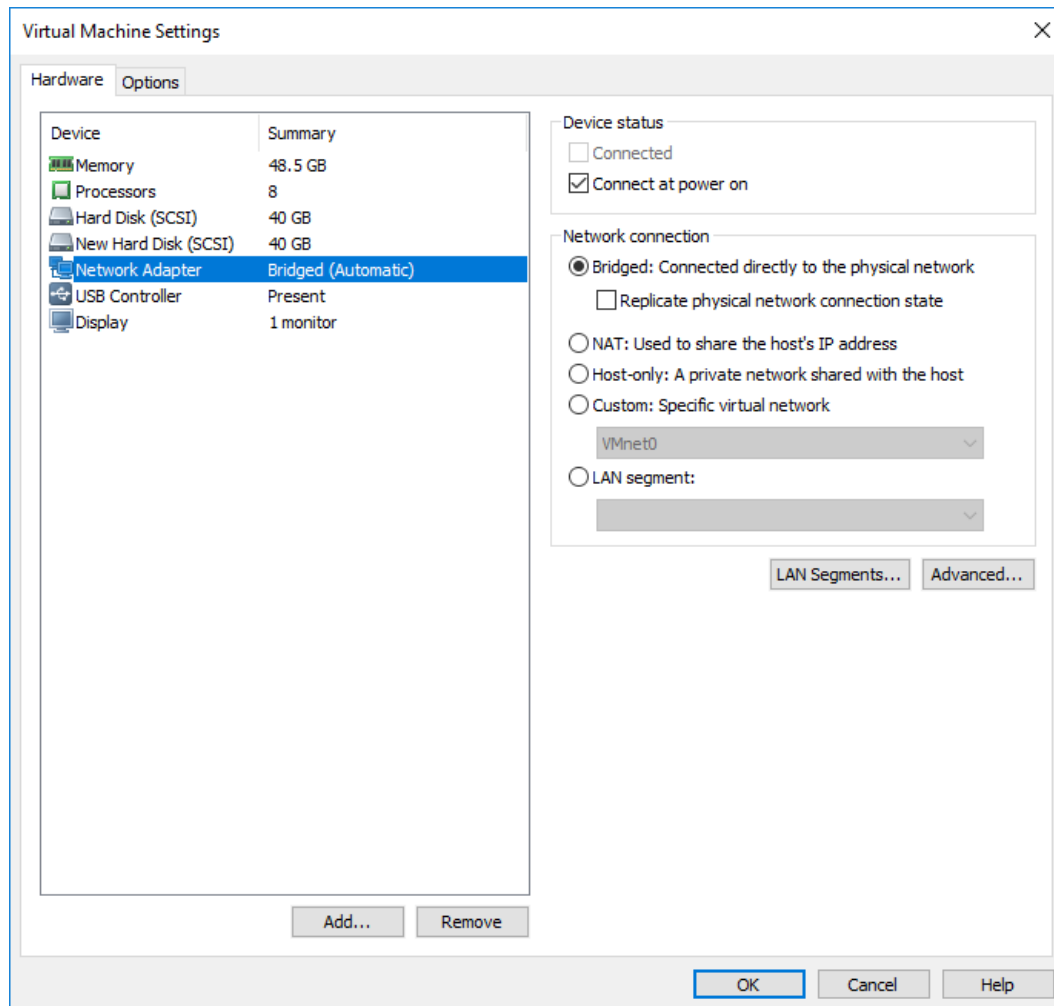
Use as much memory as possible for the virtual machine. This memory will be shared for the AOS VM and virtual network devices.

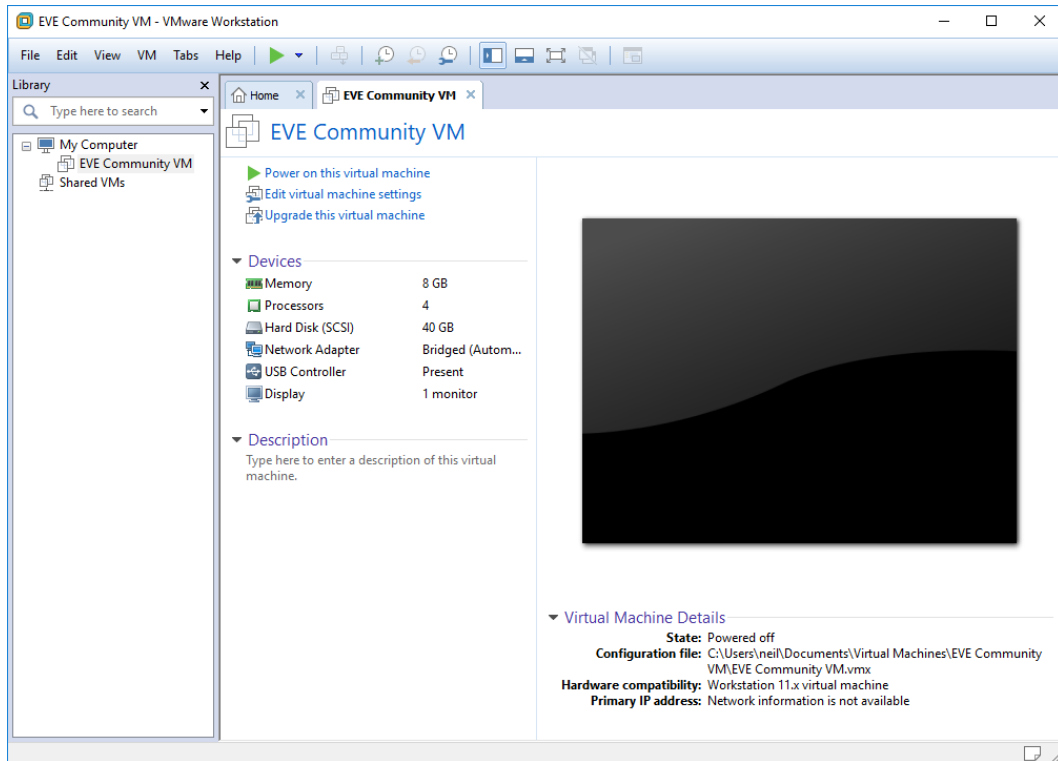


Use as many processors as available.

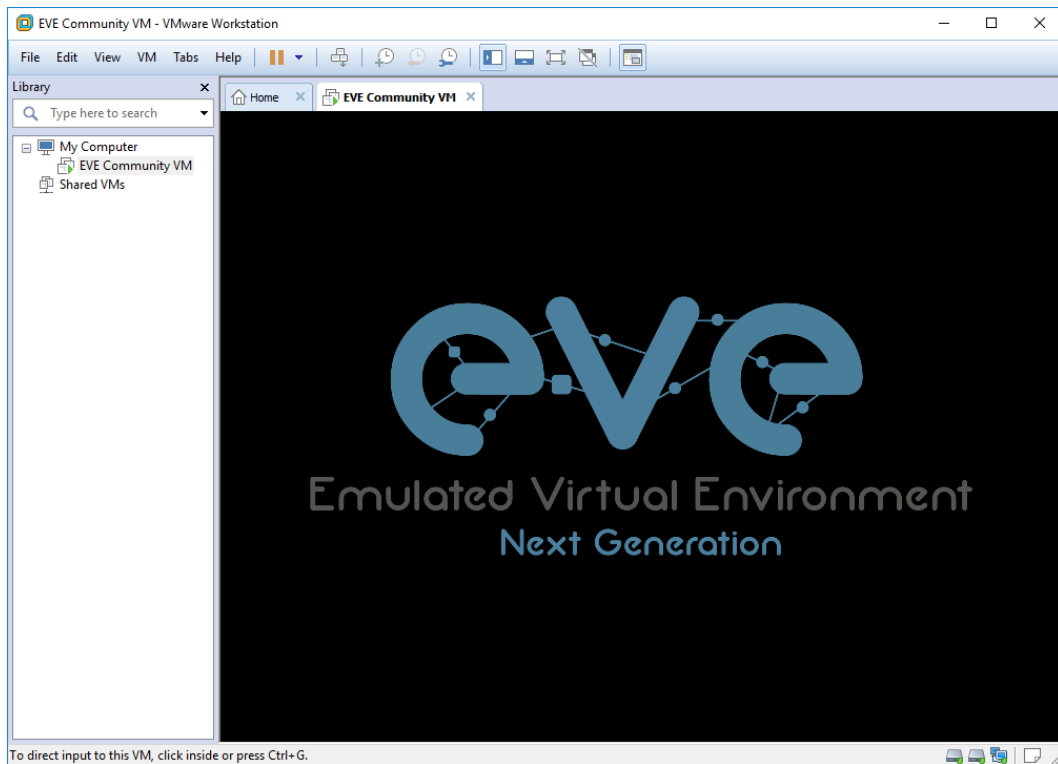
IMPORTANT ... Make sure to enable “Virtualize Intel VT-x/EPT or AMD-V/RVI.



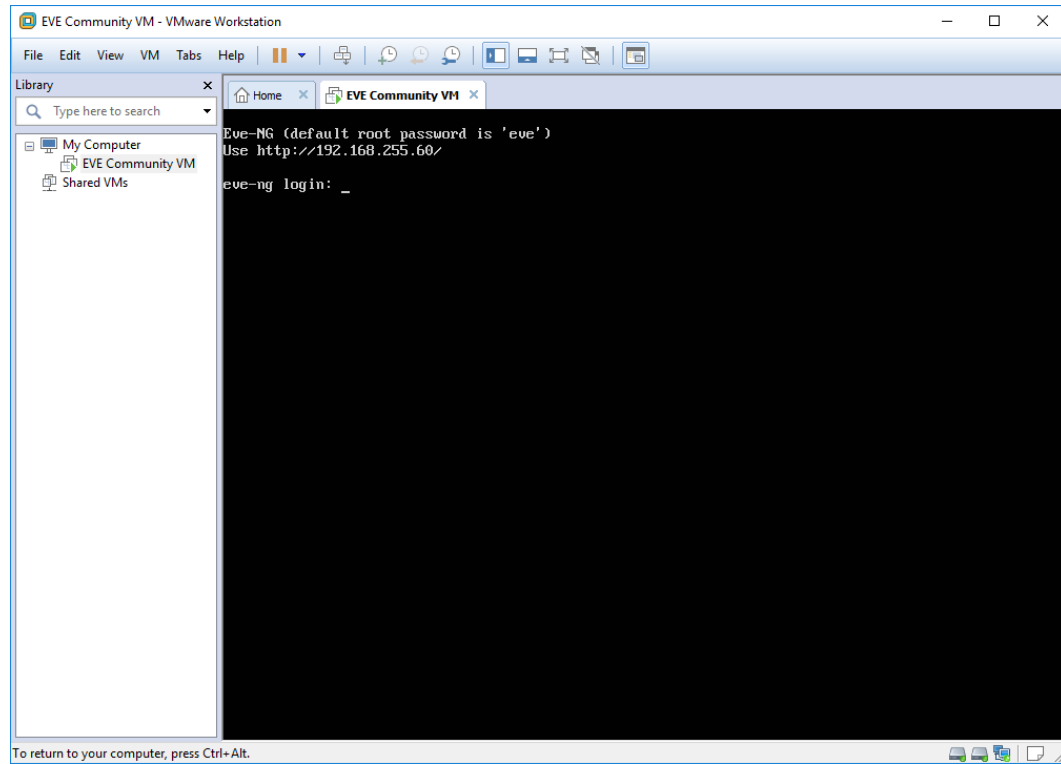




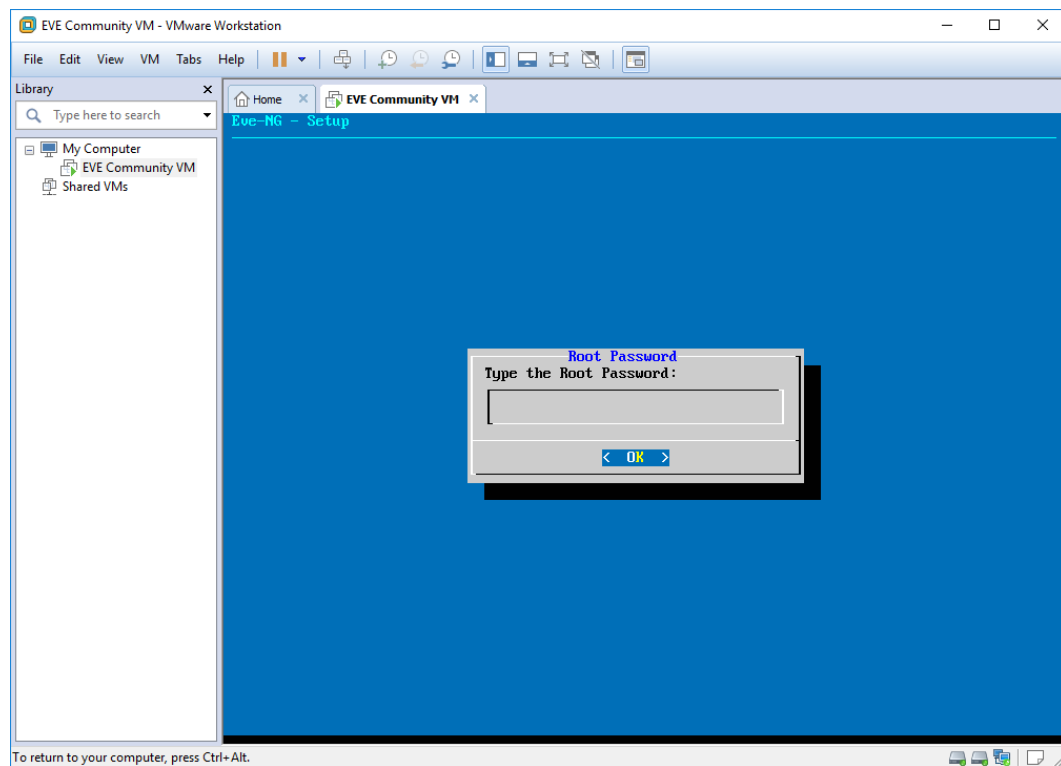
Step 3 Start the EVE-NG VM.



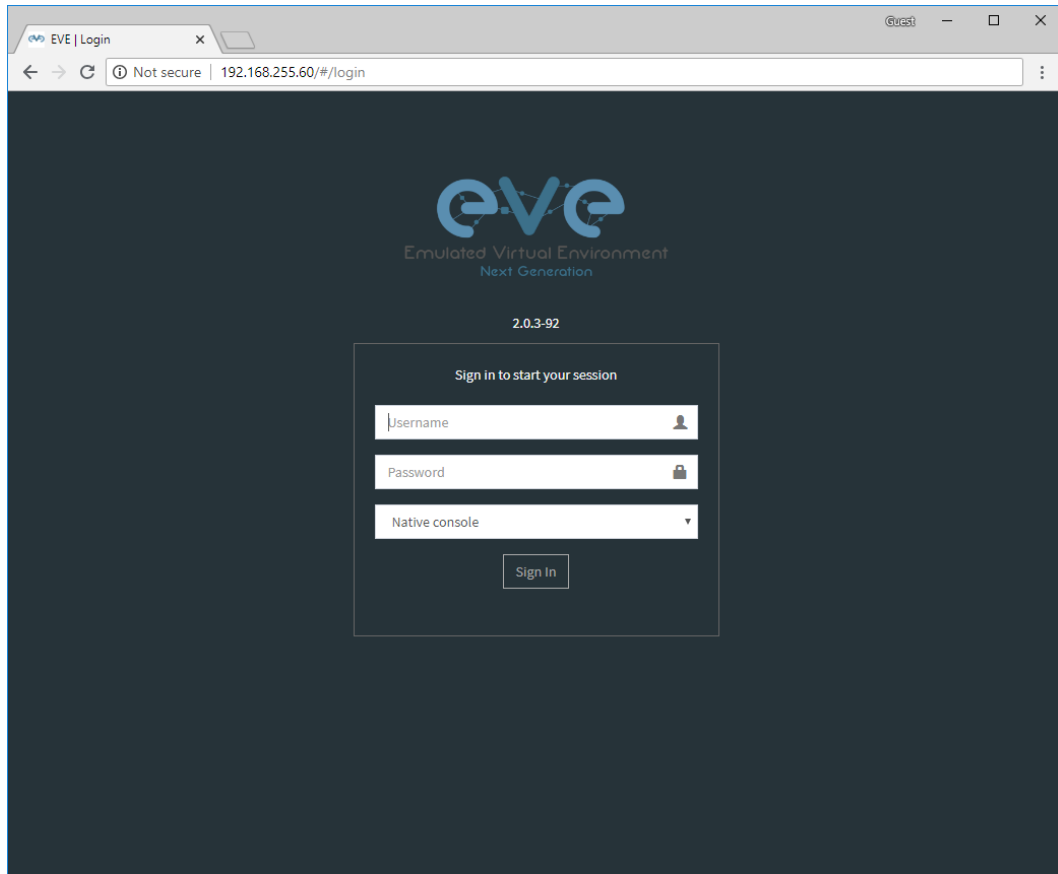
Step 4 After starting the VM, the console will display the EVE-NG Linux login prompt. Log in as username `root`, password `eve`.



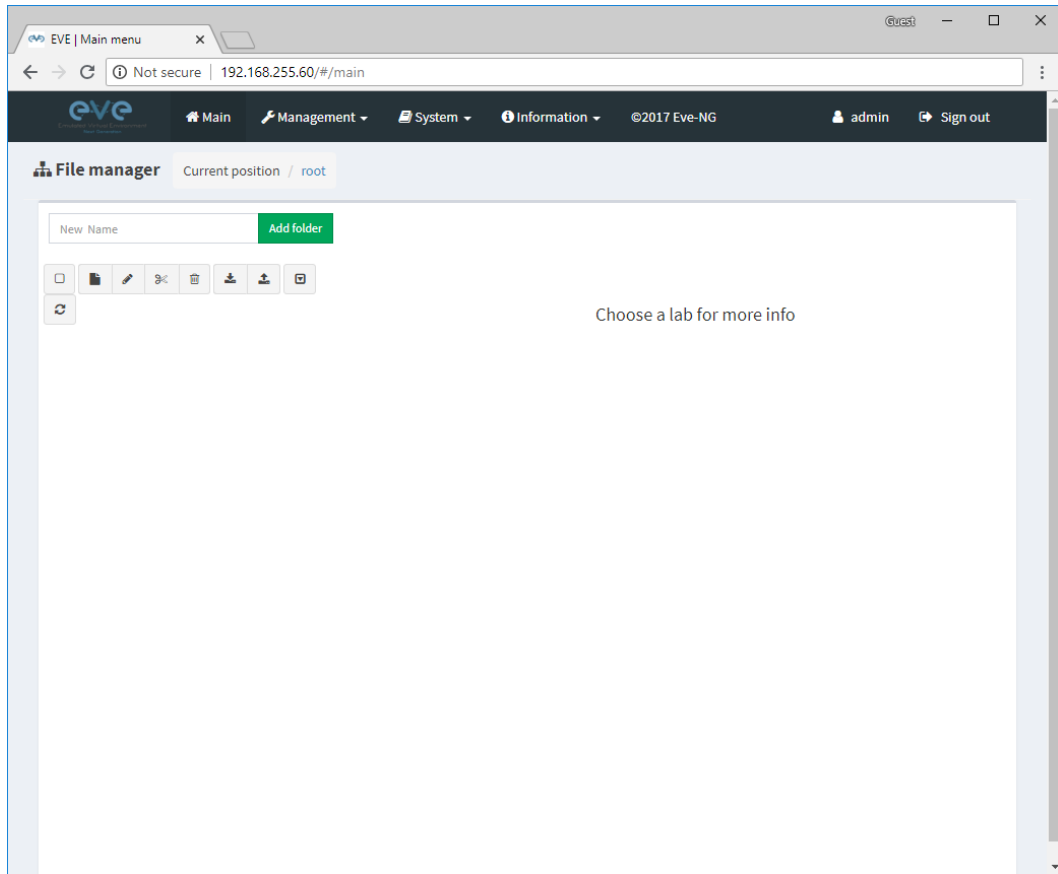
Step 5 EVE-NG will prompt to set a new root password.



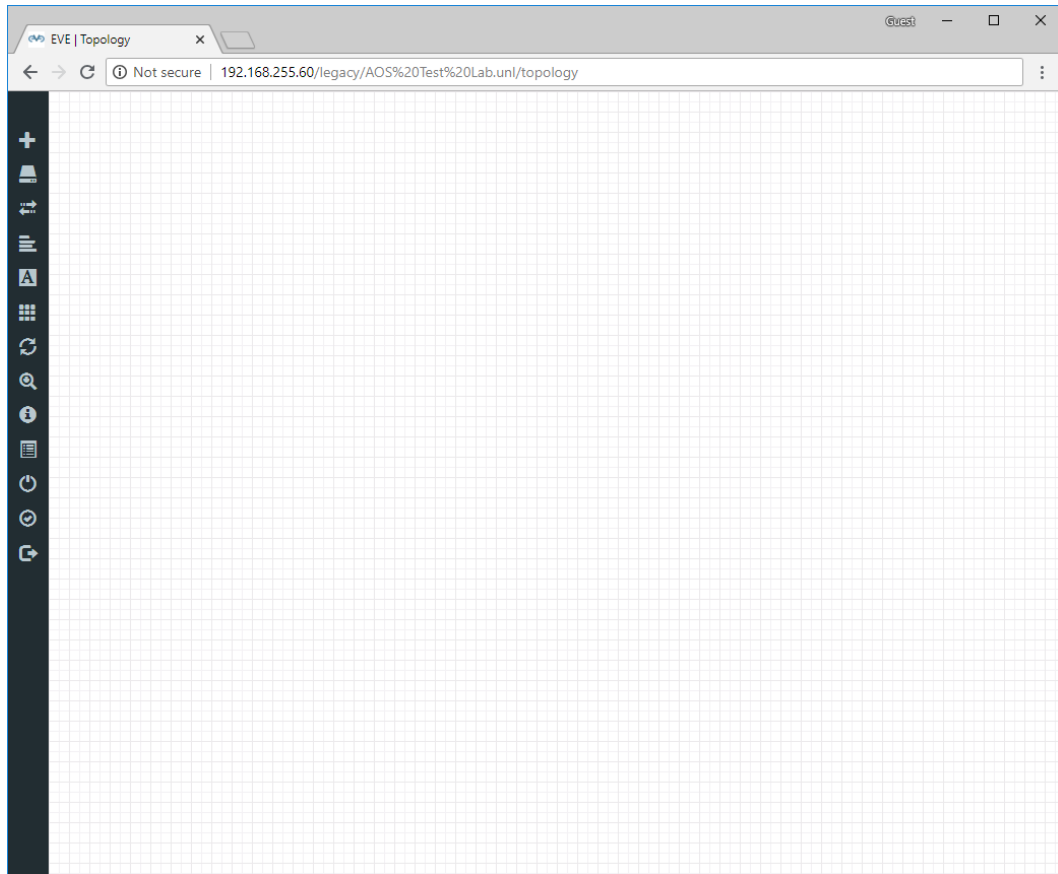
Step 6 After booting the EVE-NG VM, using a web browser, navigate to the IP address of EVE-NG and log in a username admin, password eve.



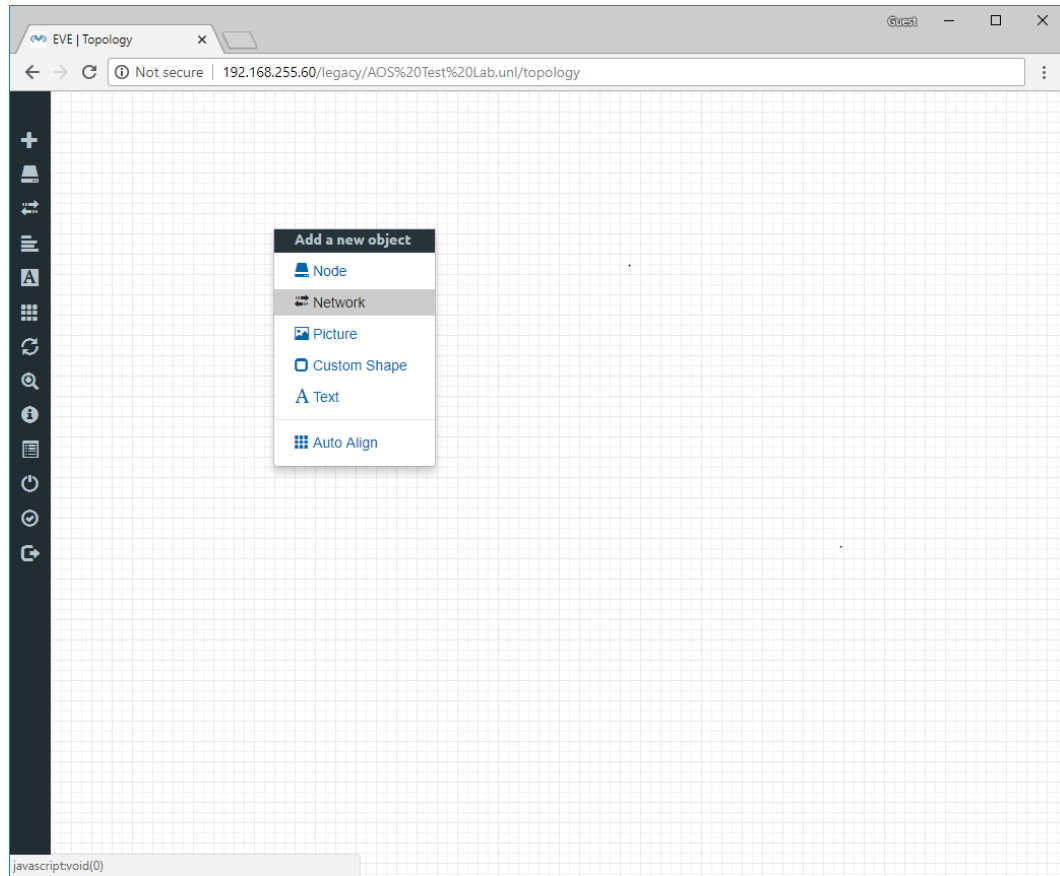
Step 7 Click the “create a new lab” icon.



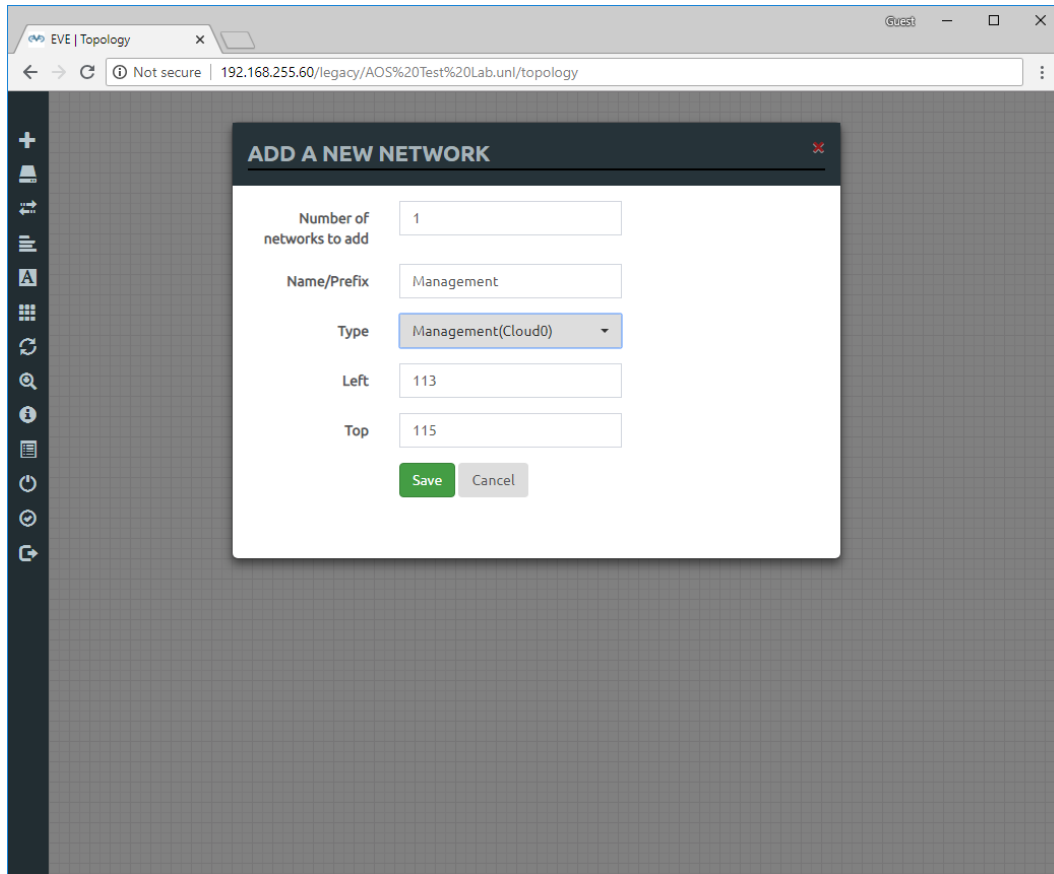
Step 8 You will see a blank lab screen.



Step 9 First, right-click and add a new “Network”.



Step 10 Adding a network of Type “Management (Cloud0)” will connect this network to the interface of the EVE-NG VM.

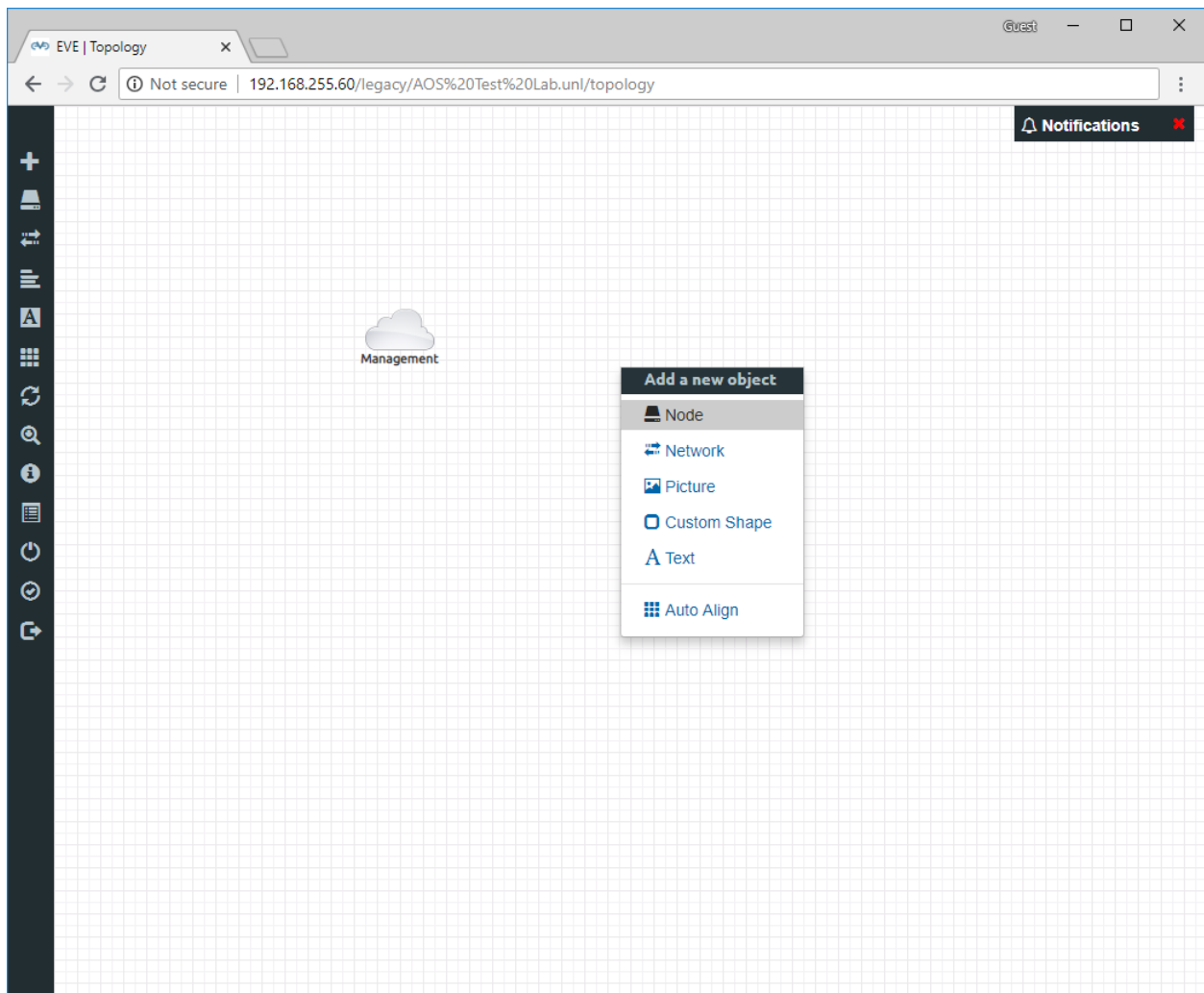


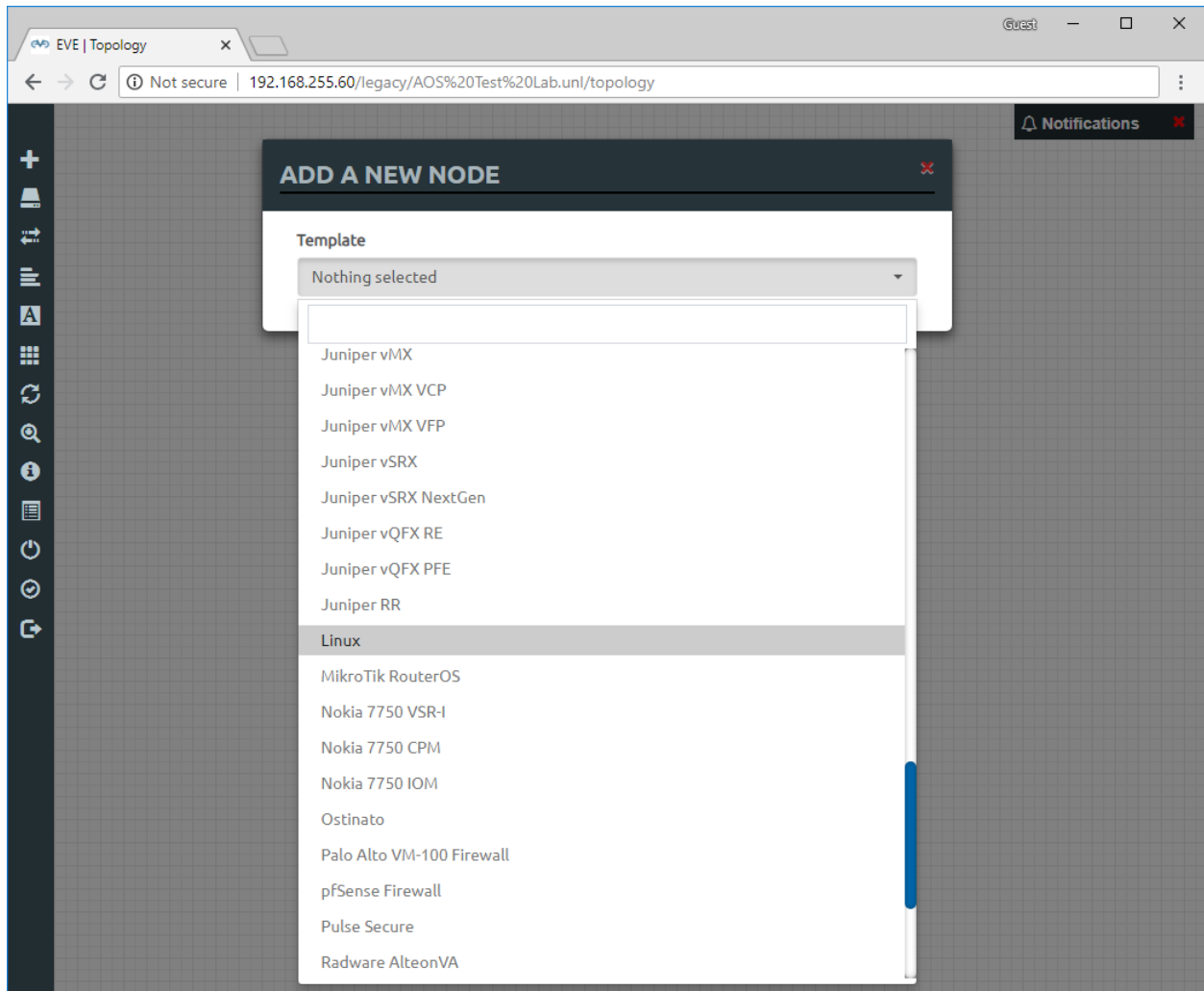
Installing AOS VM

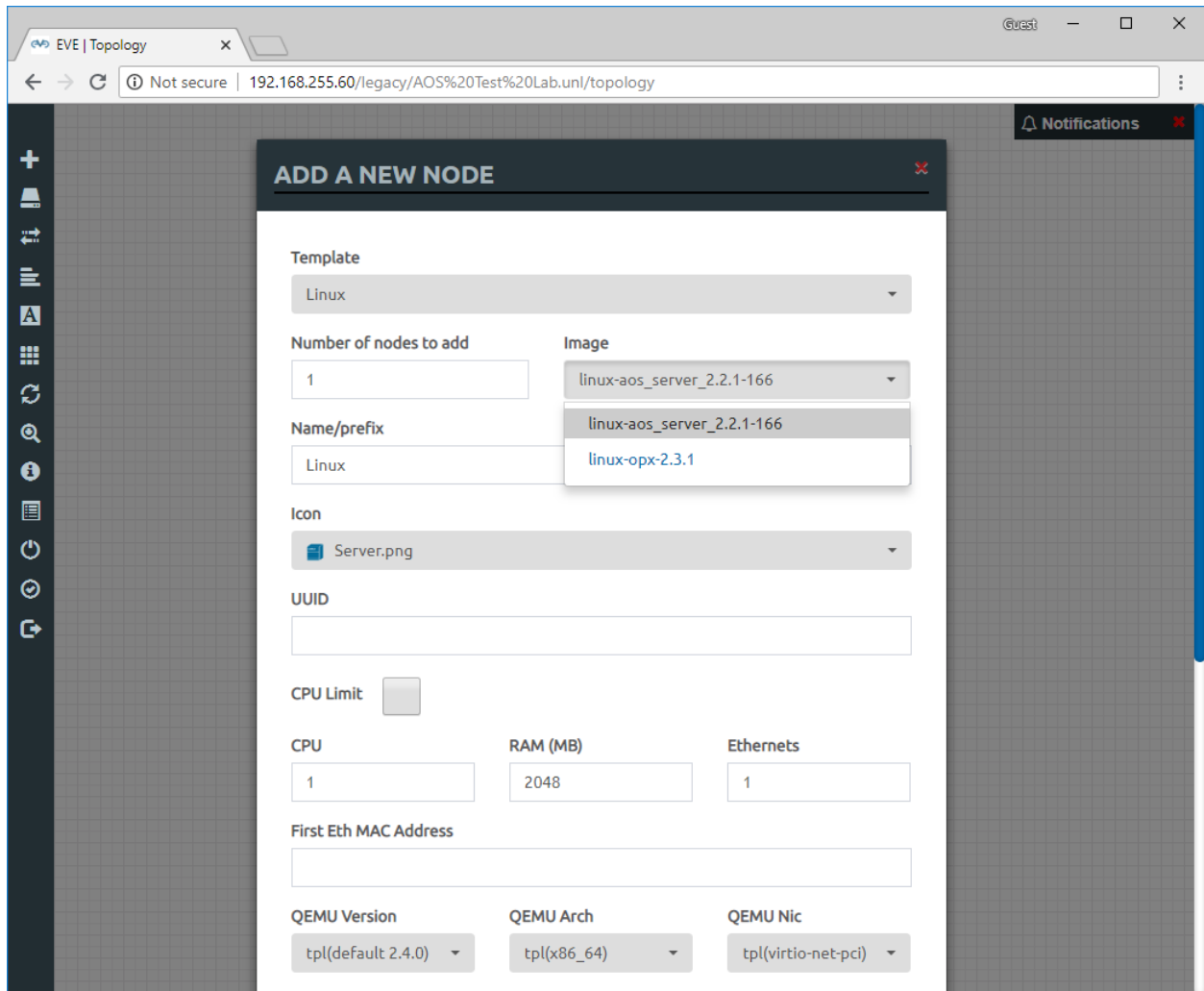
Step 1 To add an AOS Server, upload the “.qcow2.gz” AOS KVM image (e.g. aos_server_2.2.1-166.qcow2.gz) to the EVE-NG server. Then decompress the image with the `gzip -d` command. Make a directory under the `/opt/unetlab/addons/qemu/` directory beginning with `linux-`.

```
root@eve-ng:~# ls -l aos_server_2.2.1-166.qcow2.gz
-rw-r--r-- 1 root root 1195424229 Aug 31 20:34 aos_server_2.2.1-166.qcow2.gz
root@eve-ng:~# mkdir -p /opt/unetlab/addons/qemu/linux-aos_server_2.2.1-166
root@eve-ng:~# gzip -d aos_server_2.2.1-166.qcow2.gz
root@eve-ng:~# cp aos_server_2.2.1-166.qcow2 /opt/unetlab/addons/qemu/linux-aos_
server_2.2.1-166/hda.qcow2
root@eve-ng:~# /opt/unetlab/wrappers/unl_wrapper -a fixpermissions
```

Step 2 From the EVE-NG lab, right-click and add a new Node.



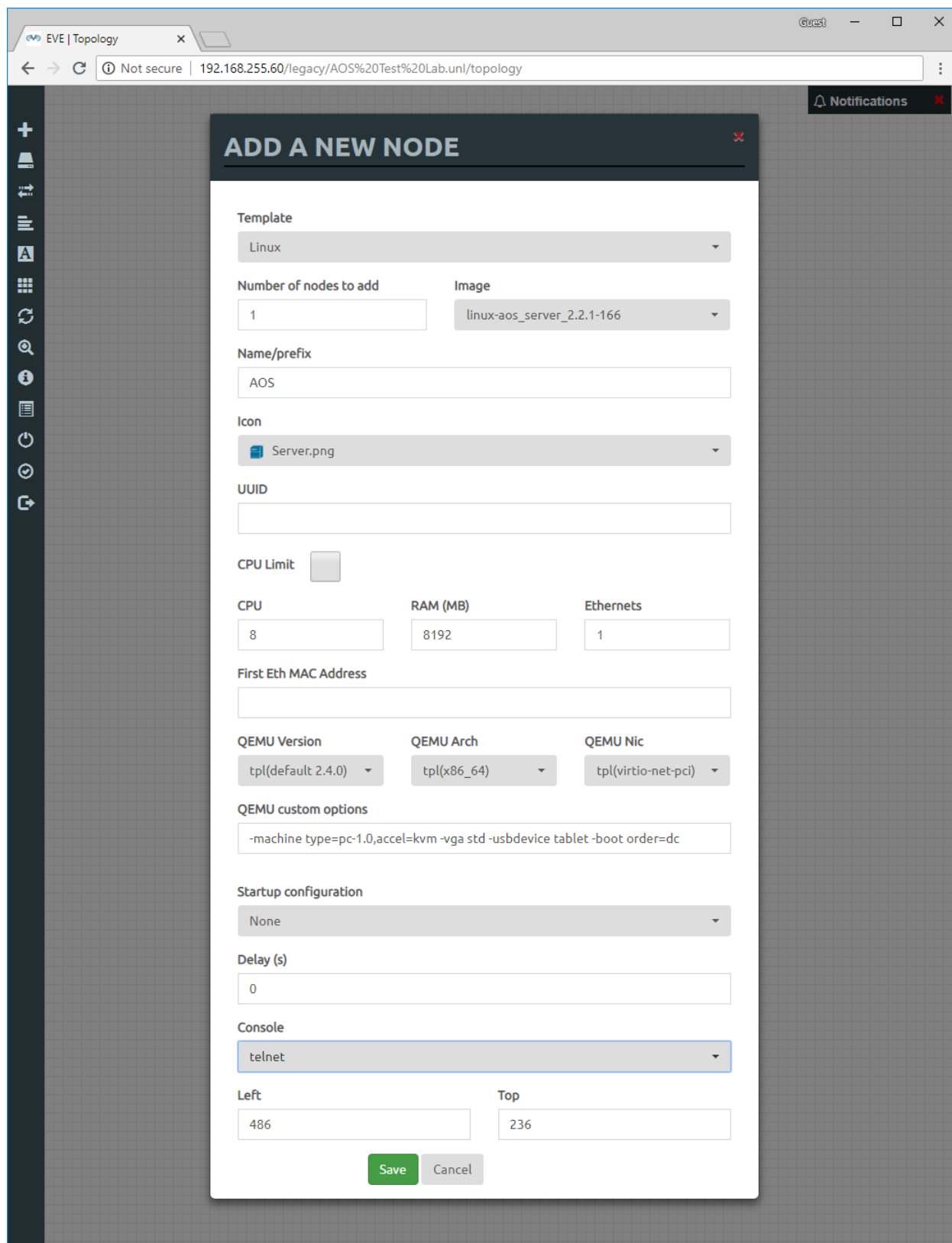




Step 3 Set the following VM CPU/RAM:

- CPU: 8
- RAM: 8192 MB (16384 MB for AOS 3.0 or later)

Step 4 Change the Console to “telnet”.



ADD A NEW NODE

Template: Linux

Number of nodes to add: 1

Image: linux-aos_server_2.2.1-166

Name/prefix: AOS

Icon: Server.png

UUID:

CPU Limit: ☐

CPU: 8

RAM (MB): 8192

Ethernets: 1

First Eth MAC Address:

QEMU Version: tpl(default 2.4.0)

QEMU Arch: tpl(x86_64)

QEMU Nic: tpl(virtio-net-pci)

QEMU custom options: -machine type=pc-1.0,accel=kvm -vga std -usbdevice tablet -boot order=dc

Startup configuration: None

Delay (s): 0

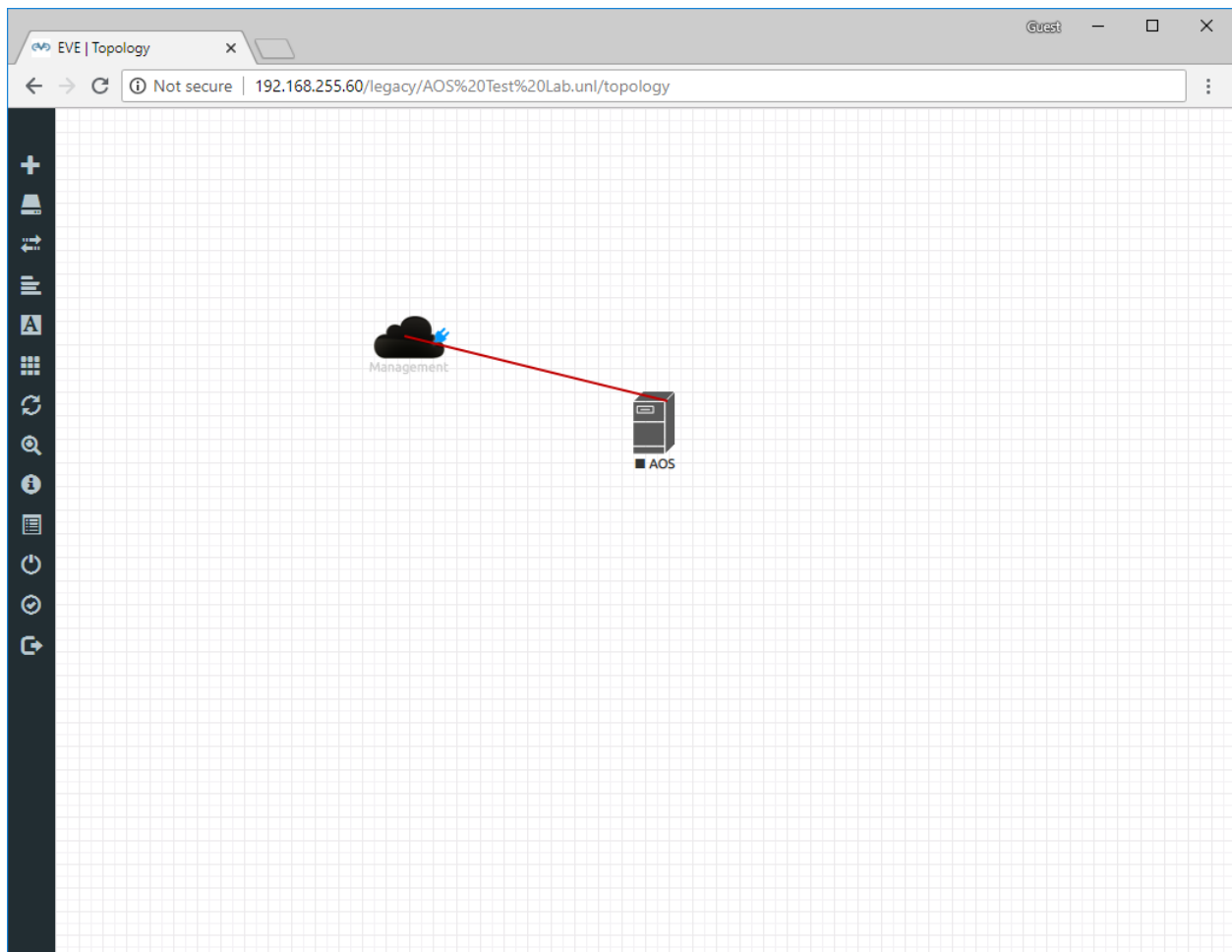
Console: telnet

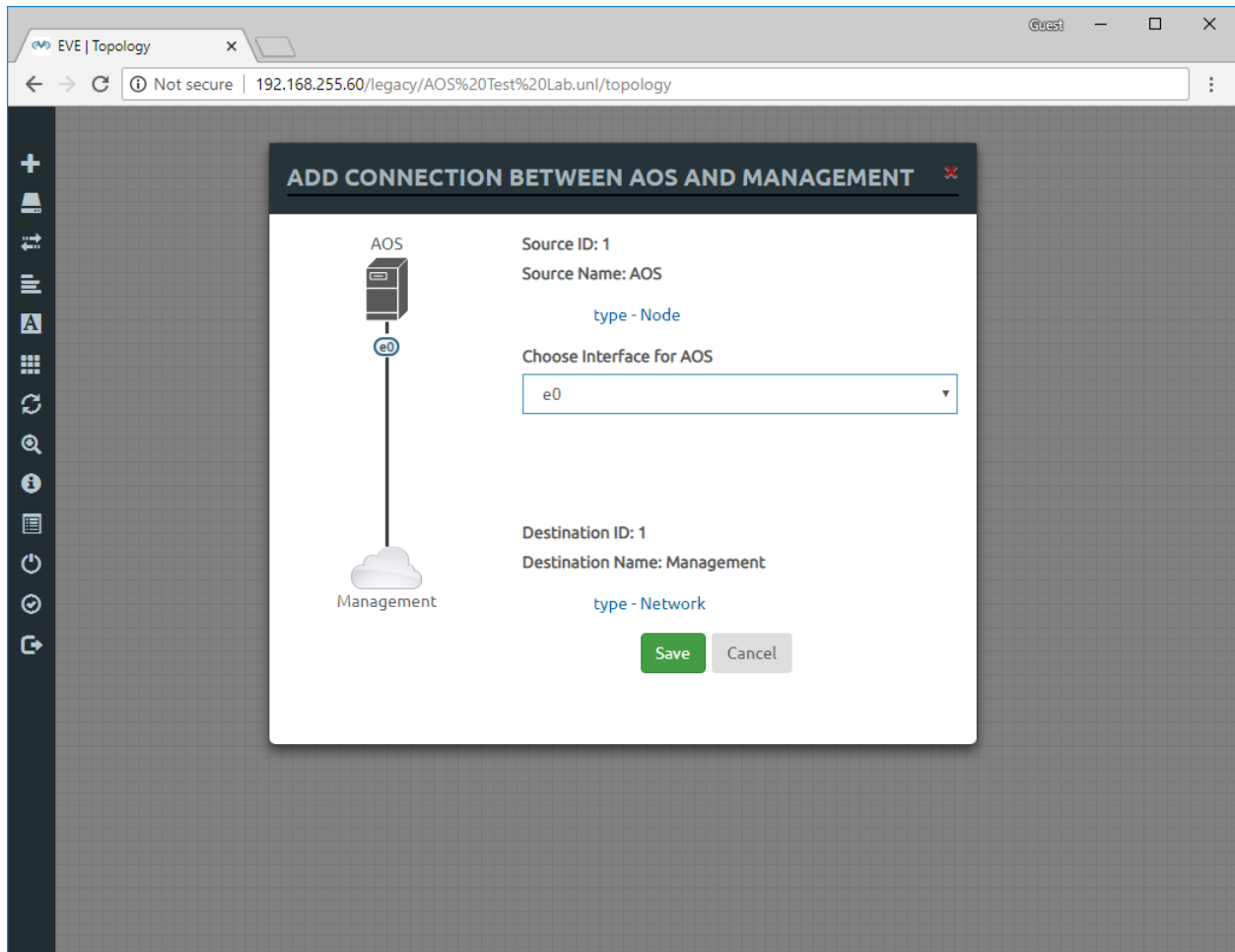
Left: 486

Top: 236

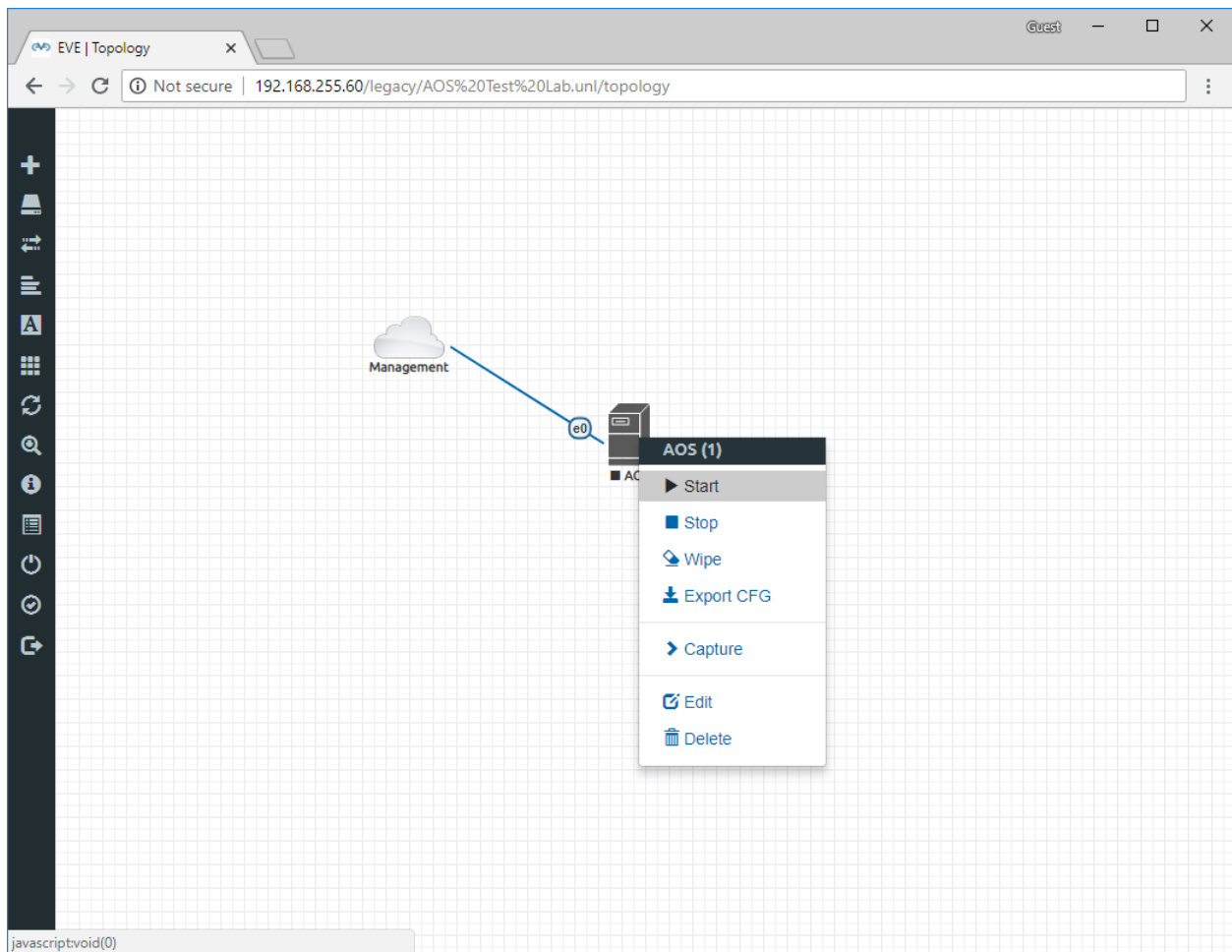
Save Cancel

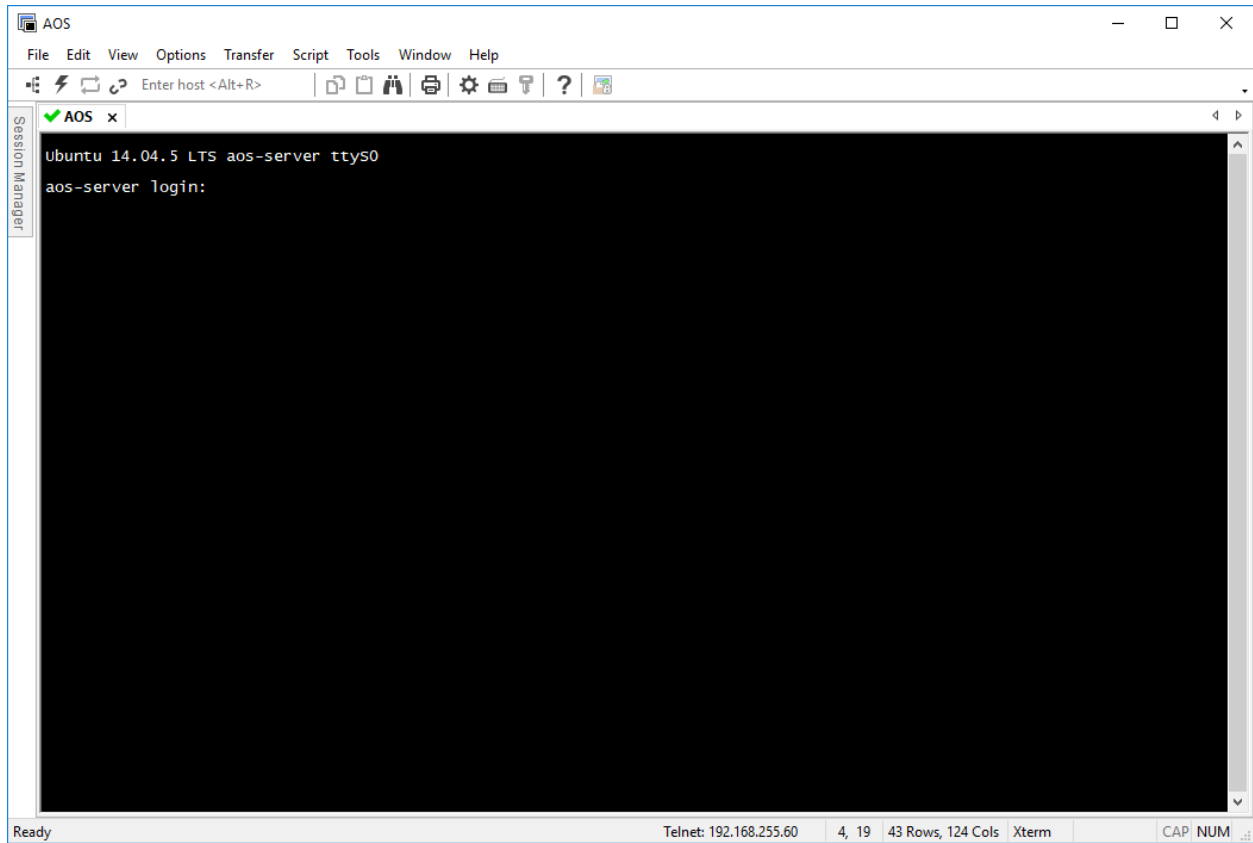
Step 5 Connect the “e0” interface to the Management Network.



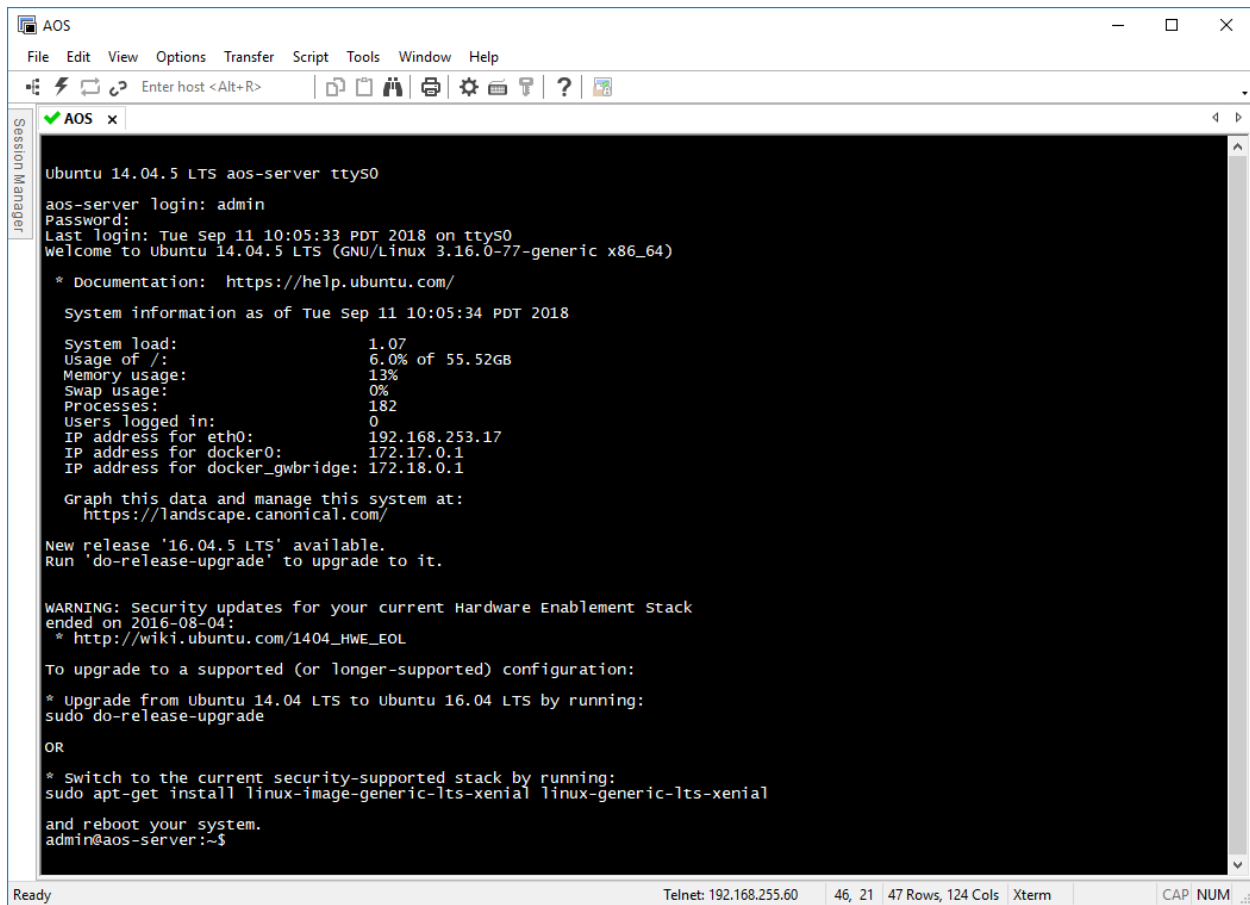


Step 6 Start the VM and connect to its console.





Step 7 Follow *Installing Apstra server* for setting up the Apstra server VM.



```
AOS
File Edit View Options Transfer Script Tools Window Help
Enter host <Alt+R>
Session Manager
AOS x
Ubuntu 14.04.5 LTS aos-server ttys0
aos-server login: admin
Password:
Last login: Tue Sep 11 10:05:33 PDT 2018 on ttys0
Welcome to Ubuntu 14.04.5 LTS (GNU/Linux 3.16.0-77-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

System information as of Tue Sep 11 10:05:34 PDT 2018

System load:          1.07
Usage of /:            6.0% of 55.52GB
Memory usage:         13%
Swap usage:           0%
Processes:            182
Users logged in:      0
IP address for eth0:   192.168.253.17
IP address for docker0: 172.17.0.1
IP address for docker_gwbridge: 172.18.0.1

Graph this data and manage this system at:
https://landscape.canonical.com/

New release '16.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

WARNING: Security updates for your current Hardware Enablement Stack
ended on 2016-08-04:
 * http://wiki.ubuntu.com/1404_HWE_EOL

To upgrade to a supported (or longer-supported) configuration:

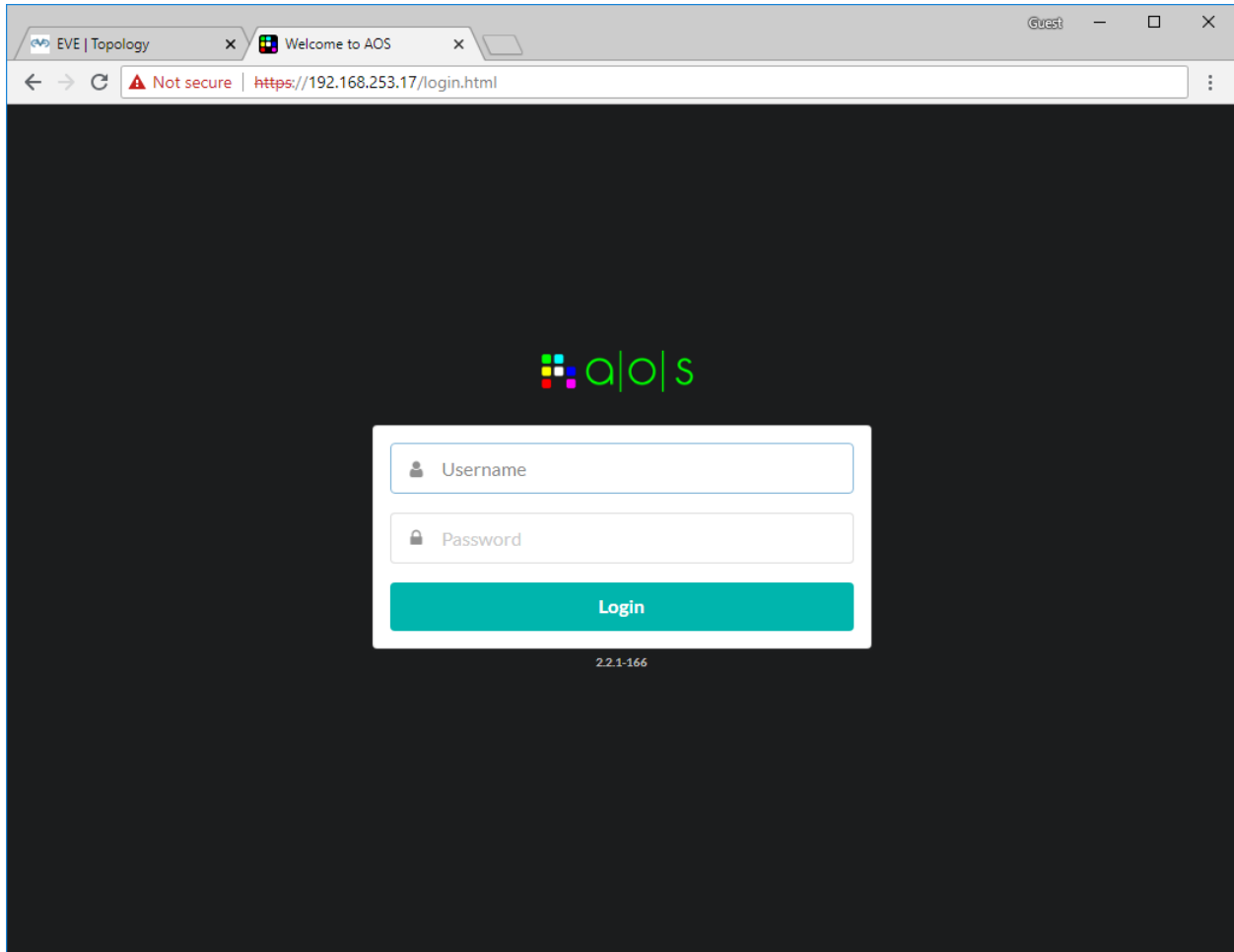
 * Upgrade from Ubuntu 14.04 LTS to Ubuntu 16.04 LTS by running:
sudo do-release-upgrade

OR

 * Switch to the current security-supported stack by running:
sudo apt-get install linux-image-generic-lts-xenial linux-generic-lts-xenial
and reboot your system.
admin@aos-server:~$
```

Ready Telnet: 192.168.255.60 46, 21 47 Rows, 124 Cols Xterm CAP NUM

Step 8 You should now be able to connect to the IP address of the Apstra server with a web browser.



Installing Virtual Network Devices

Note: It is up to the user to obtain Virtual Network Device images. Apstra Support cannot provide these images.

Add Arista EOS Devices

Step 1 Obtain the Arista vEOS 4.20.1F VMDK (vEOS-lab-4.20.1F.vmdk) and “Aboot” (Aboot-veos-serial-8.0.0.iso) images from Arista.

Note: Please check the [AOS Feature Matrix](#) for the latest recommended version of Arista EOS. Arista EOS is licensed software and cannot be provided by Apstra Support.

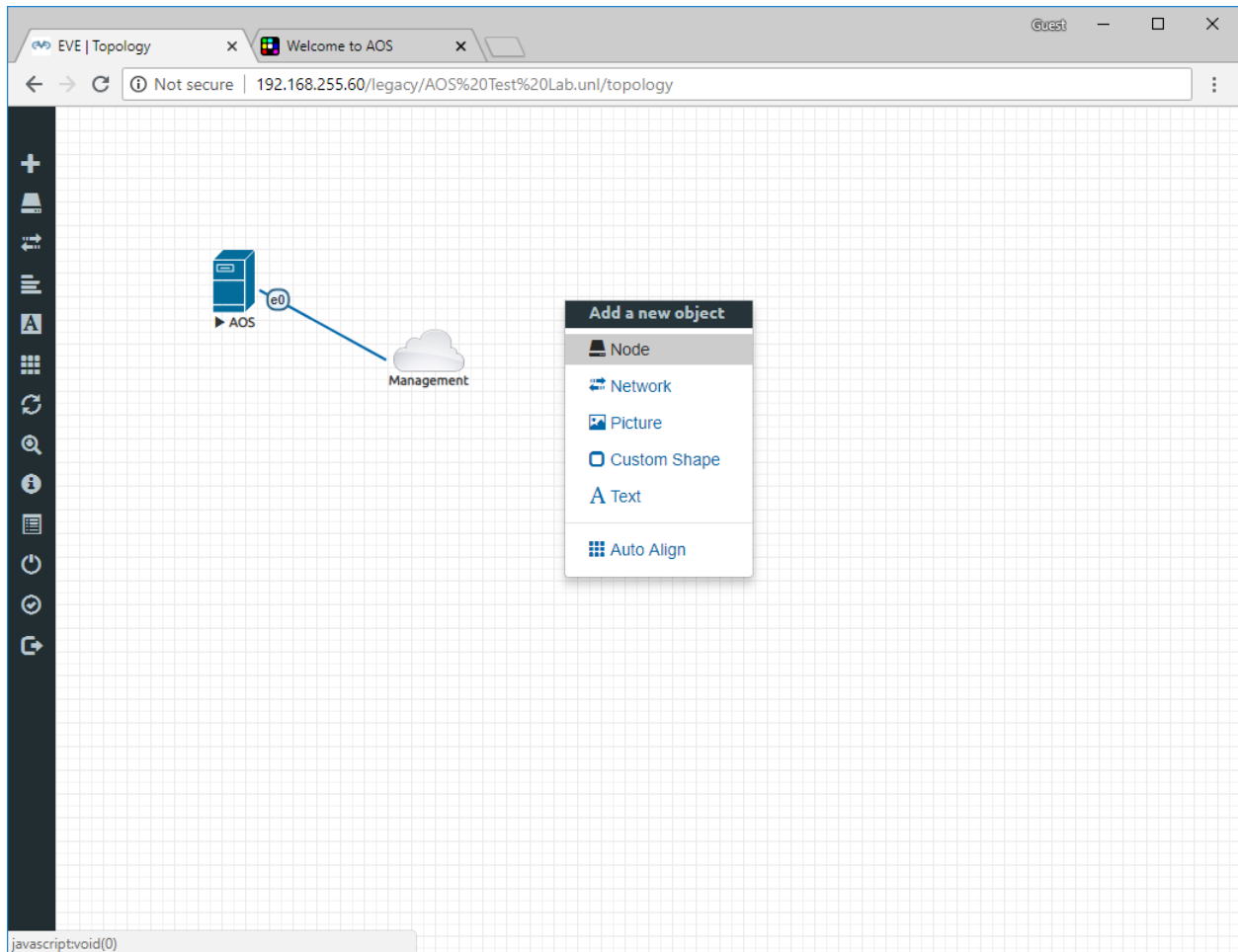
```
root@eve-ng:~# ls -l Aboot-veos-serial-8.0.0.iso vEOS-lab-4.20.1F.vmdk
-rw-r--r-- 1 root root 5242880 Aug 31 20:35 Aboot-veos-serial-8.0.0.iso
-rw-r--r-- 1 root root 662044672 Aug 31 20:34 vEOS-lab-4.20.1F.vmdk
root@eve-ng:~# mkdir -p /opt/unetlab/addons/qemu/veos-4.20.1F
root@eve-ng:~# /opt/qemu/bin/qemu-img convert -f vmdk -O qcow2 vEOS-lab-4.20.1F.vmdk /
  ↪ /opt/unetlab/addons/qemu/veos-4.20.1F/hda.qcow2
```

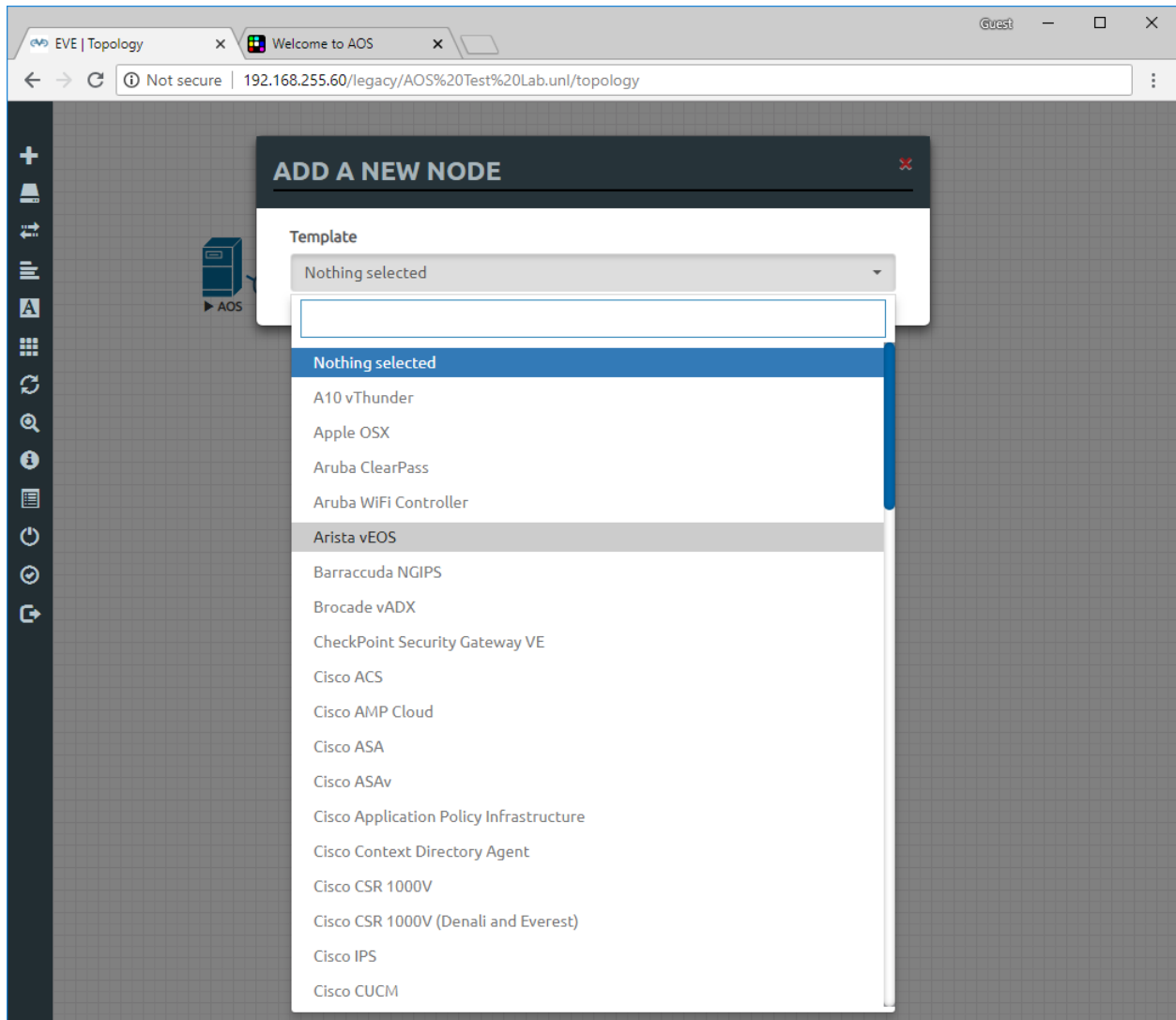
(continues on next page)

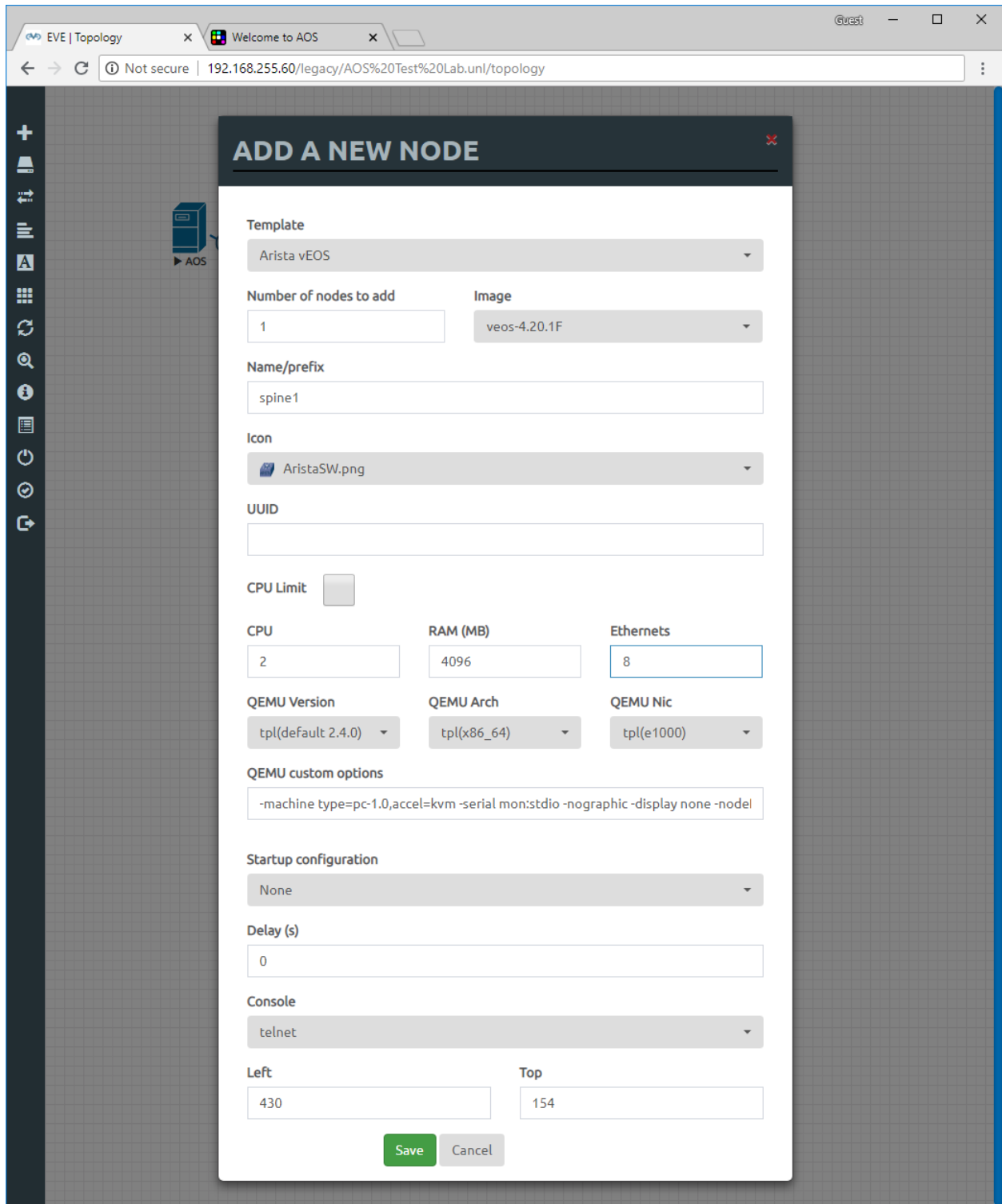
(continued from previous page)

```
root@eve-ng:~# cp Aboot-veos-serial-8.0.0.iso /opt/unetlab/addons/qemu/veos-4.20.1F/  
↪cdrom.iso  
root@eve-ng:~# /opt/unetlab/wrappers/unl_wrapper -a fixpermissions
```

Step 2 From the EVE-NG lab, right-click and add a new Node.







ADD A NEW NODE

Template: Arista vEOS

Number of nodes to add: 1

Image: veos-4.20.1F

Name/prefix: spine1

Icon: AristaSW.png

UUID:

CPU Limit: ☐

CPU: 2

RAM (MB): 4096

Ethernets: 8

QEMU Version: tpl(default 2.4.0)

QEMU Arch: tpl(x86_64)

QEMU Nic: tpl(e1000)

QEMU custom options: -machine type=pc-1.0,accel=kvm -serial mon:stdio -nographic -display none -nodef

Startup configuration: None

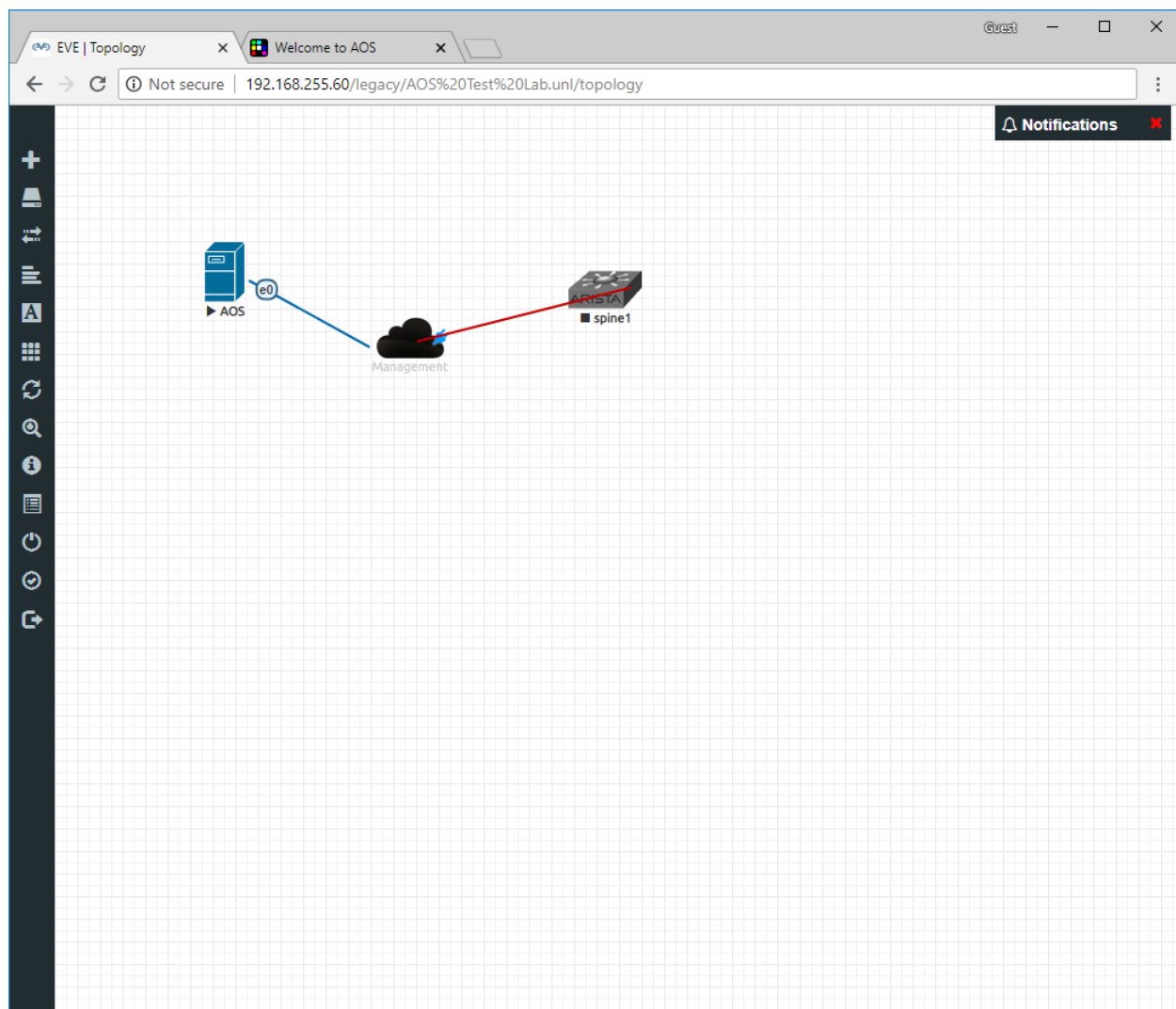
Delay (s): 0

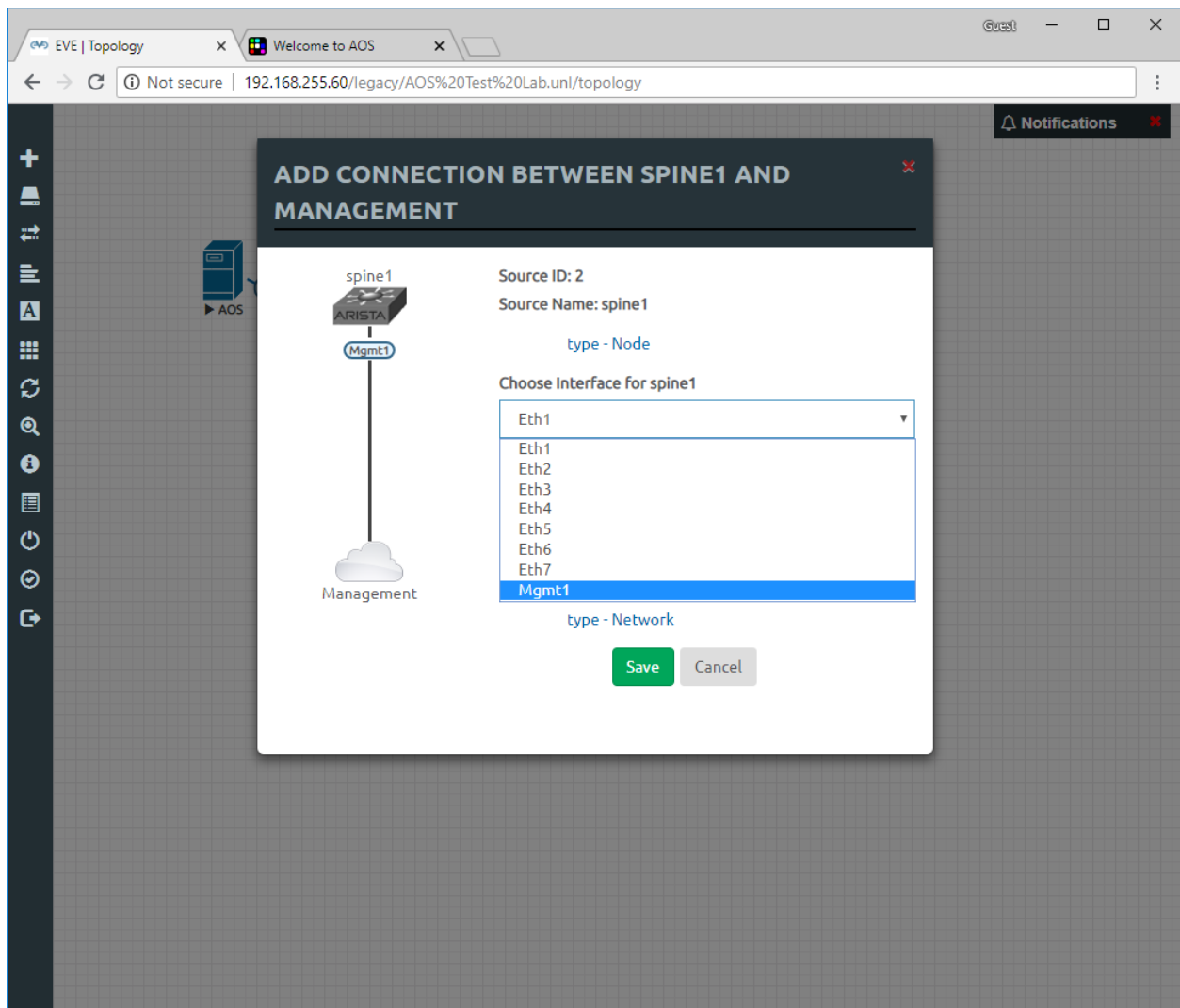
Console: telnet

Left: 430

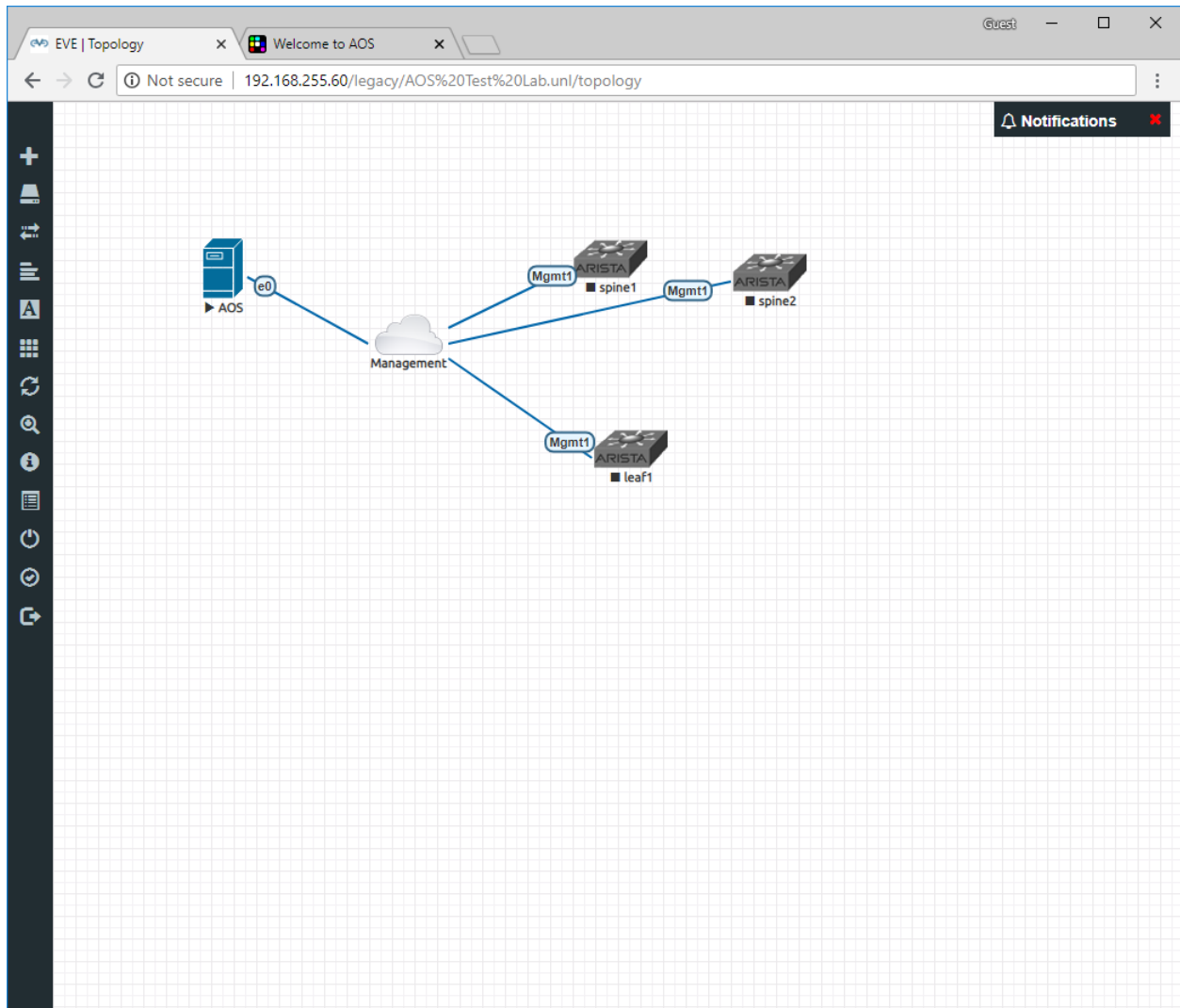
Top: 154

Save **Cancel**

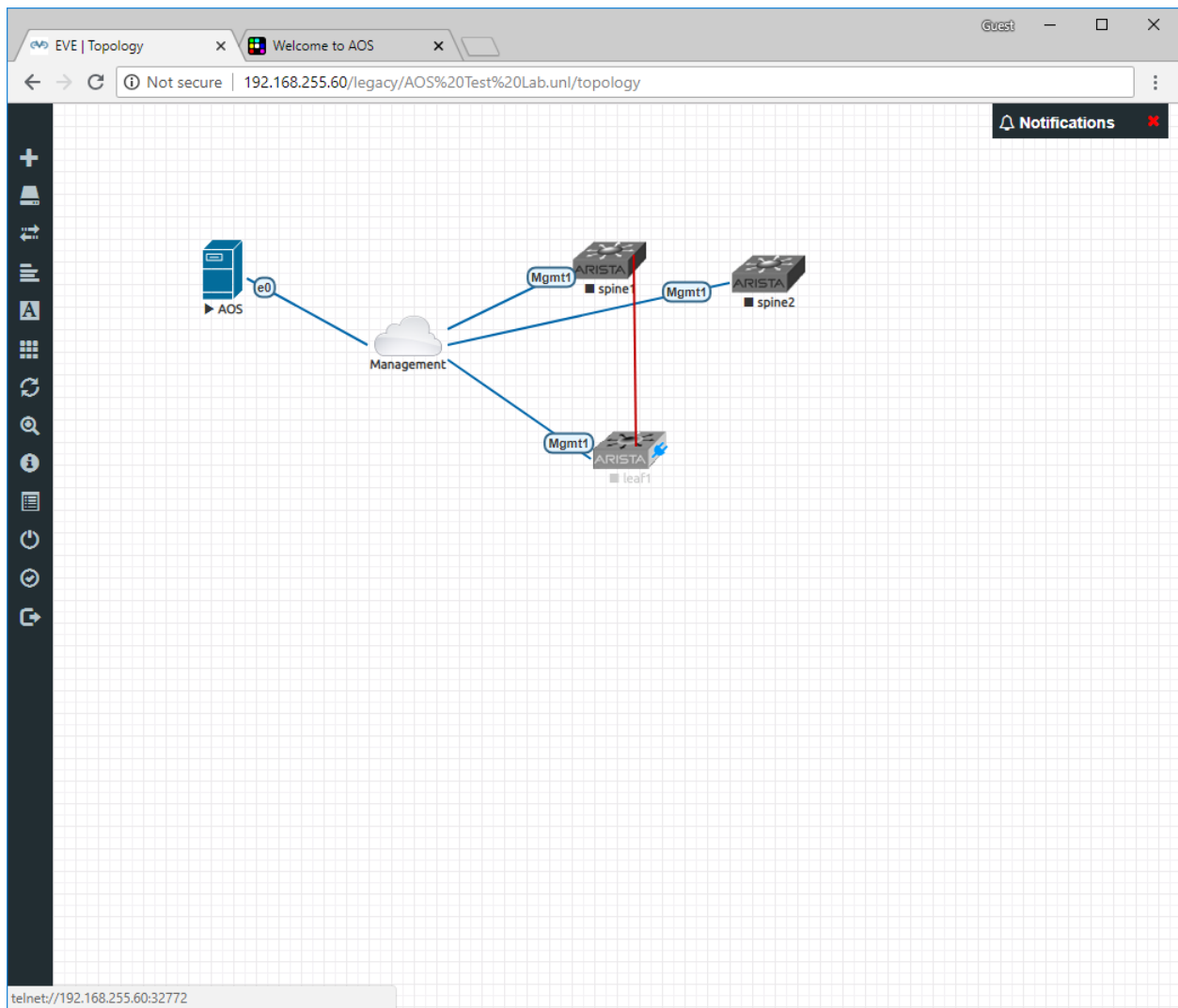




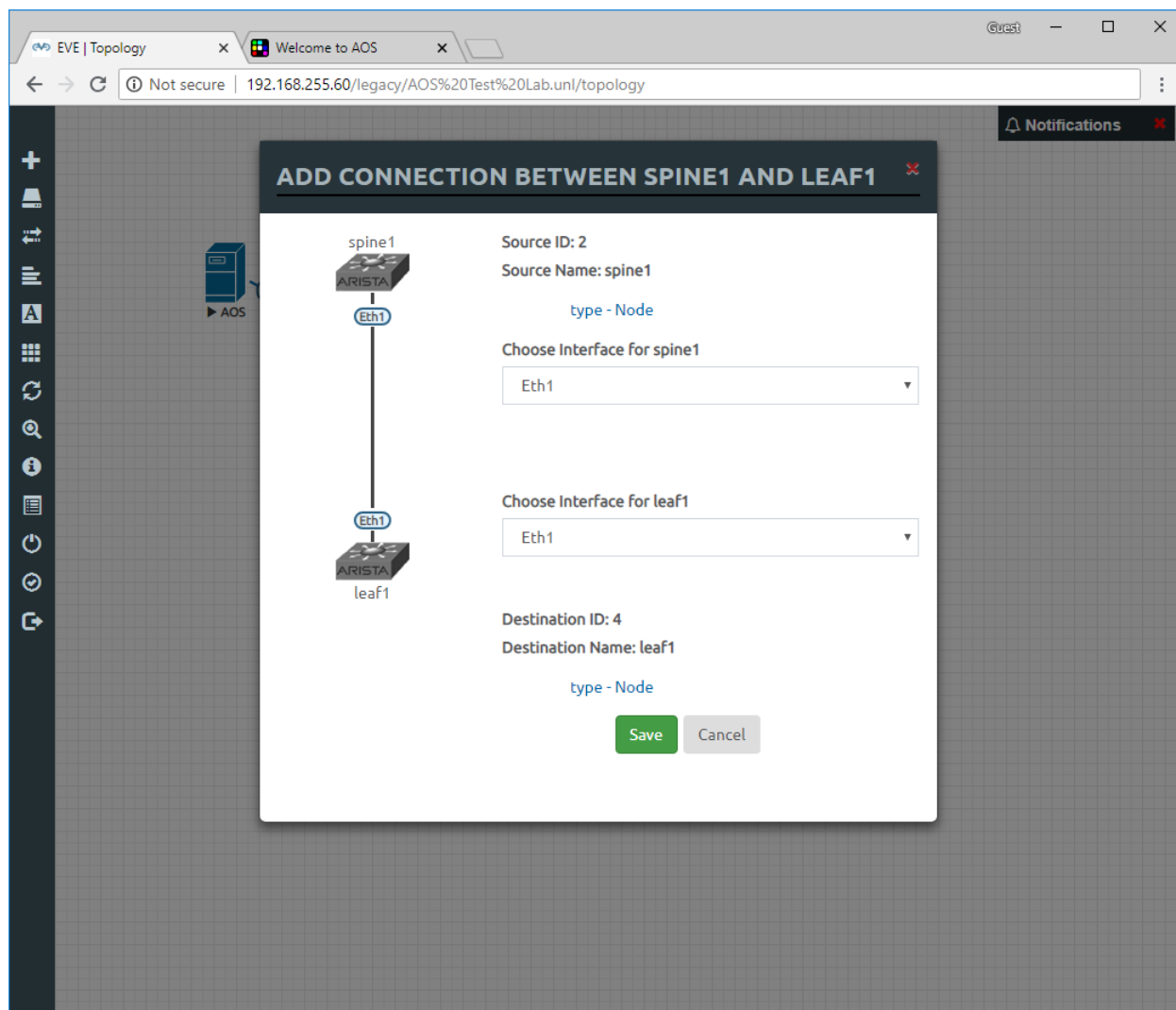
Step 3 Repeat the process for additional devices.

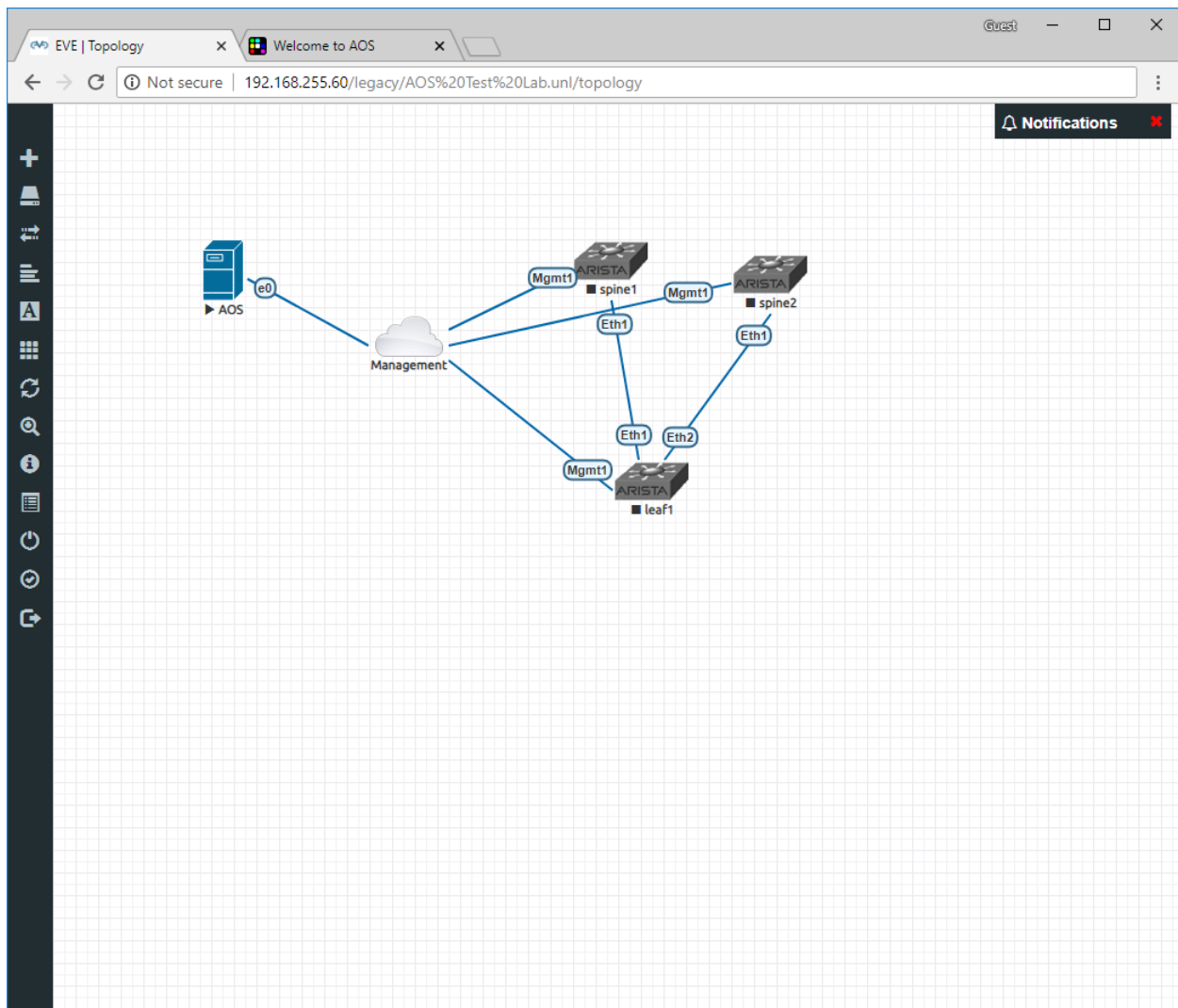


Step 4 Connect the devices together as desired.

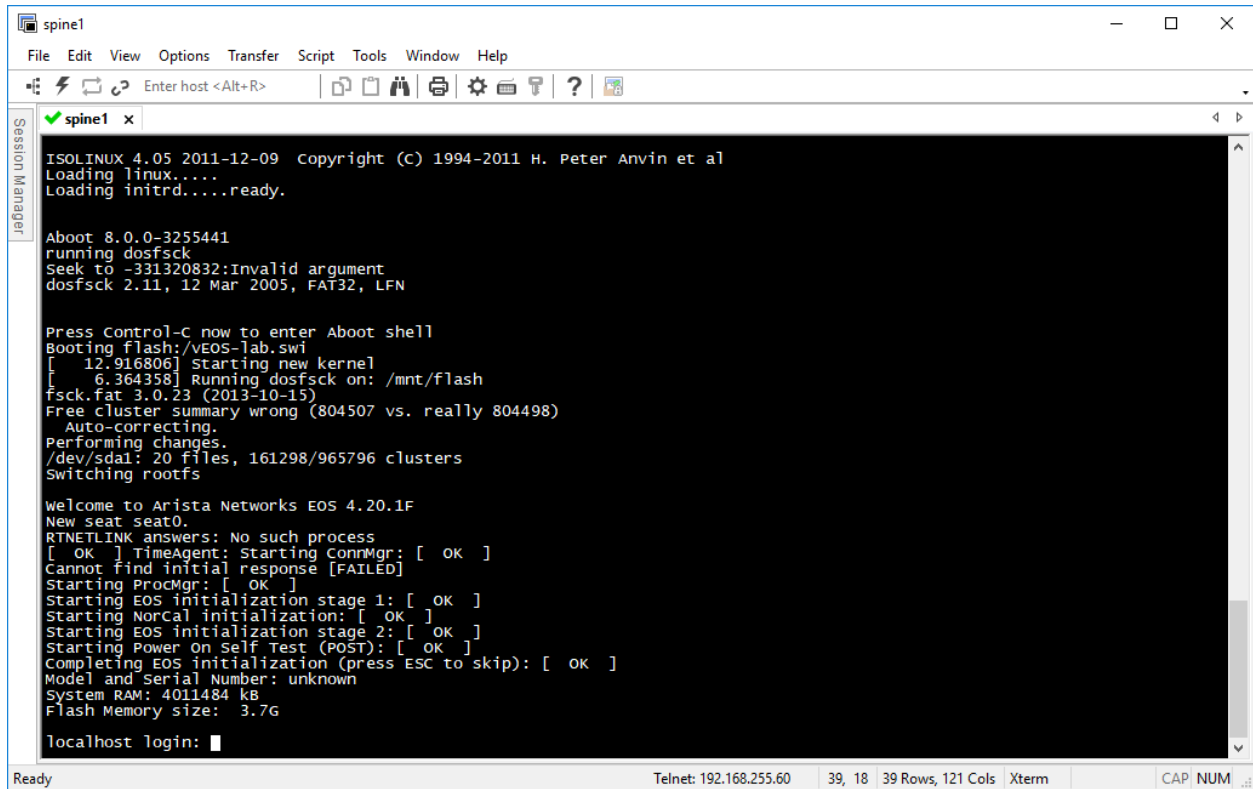


Step 5 Specify interfaces.





Step 6 Connect to the EOS device console.



```

spine1
File Edit View Options Transfer Script Tools Window Help
Enter host <Alt+R>

Session Manager
spine1 x
ISOLINUX 4.05 2011-12-09 Copyright (c) 1994-2011 H. Peter Anvin et al
Loading linux.....
Loading initrd.....ready.

About 8.0.0-3255441
running dosfsck
seek to -331320832:Invalid argument
dosfsck 2.11, 12 Mar 2005, FAT32, LFN

Press Control-C now to enter About shell
Booting flash:/VEOS-lab.swi
[ 12.916806] Starting new kernel
[ 6.364358] Running dosfsck on: /mnt/Flash
fsck.fat 3.0.23 (2013-10-15)
Free cluster summary wrong (804507 vs. really 804498)
Auto-correcting.
Performing changes.
/dev/sda1: 20 files, 161298/965796 clusters
Switching rootfs

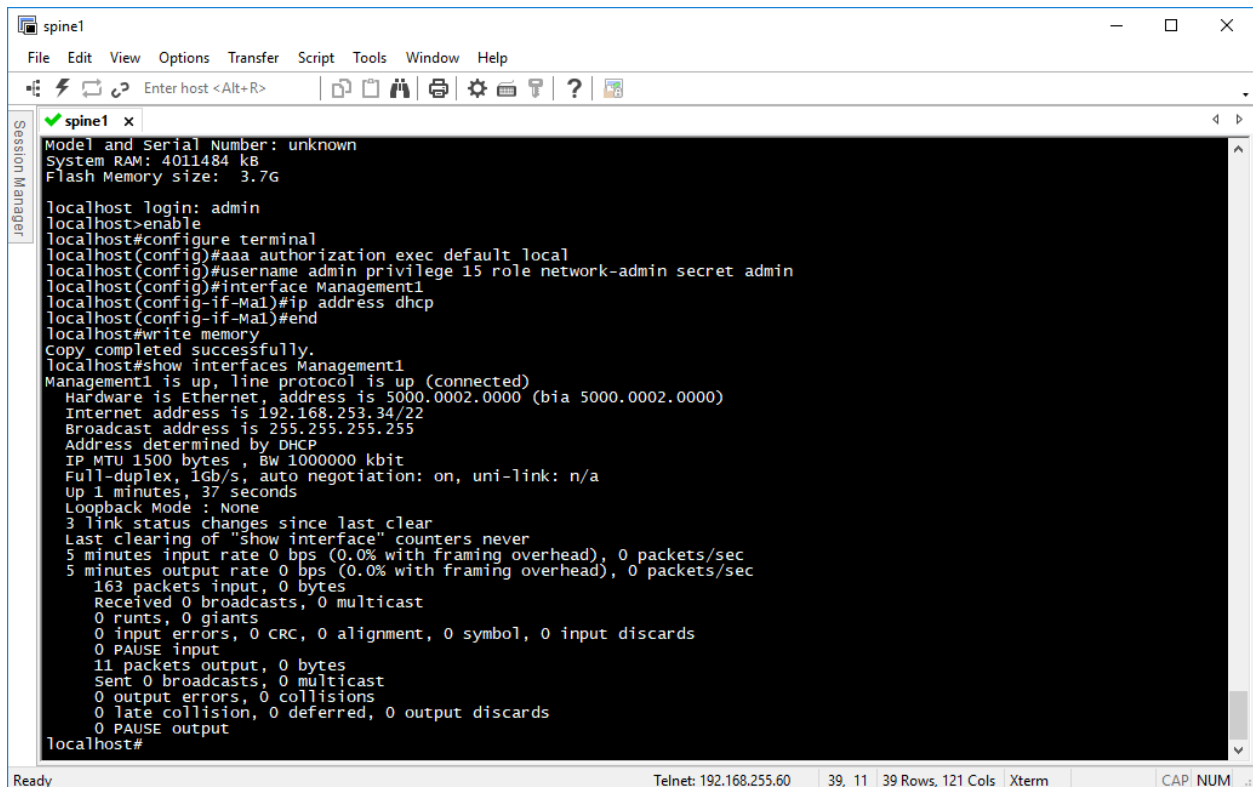
Welcome to Arista Networks EOS 4.20.1F
New seat seat0.
RTNETLINK answers: No such process
[ OK ] TimeAgent: Starting ConnMgr: [ OK ]
Cannot find initial response [FAILED]
Starting ProcMgr: [ OK ]
Starting EOS initialization stage 1: [ OK ]
Starting Norcal initialization: [ OK ]
Starting EOS initialization stage 2: [ OK ]
Starting Power on Self Test (POST): [ OK ]
Completing EOS initialization (press ESC to skip): [ OK ]
Model and Serial Number: unknown
System RAM: 4011484 kB
Flash Memory size: 3.7G

localhost login: █

```

Ready Telnet: 192.168.255.60 39, 18 39 Rows, 121 Cols Xterm CAP NUM

Step 7 Configure the necessary management configuration.



```

spine1
File Edit View Options Transfer Script Tools Window Help
Enter host <Alt+R>

Session Manager
spine1 x
Model and Serial Number: unknown
System RAM: 4011484 kB
Flash Memory size: 3.7G

localhost login: admin
localhost>enable
localhost#configure terminal
localhost(config)#aaa authorization exec default local
localhost(config)#username admin privilege 15 role network-admin secret admin
localhost(config)#interface Management1
localhost(config-if-Ma1)#ip address dhcp
localhost(config-if-Ma1)#end
localhost#write memory
Copy completed successfully.
localhost#show interfaces Management1
Management1 is up, line protocol is up (connected)
Hardware is Ethernet, address is 5000.0002.0000 (bia 5000.0002.0000)
Internet address is 192.168.253.34/22
Broadcast address is 255.255.255.255
Address determined by DHCP
IP MTU 1500 bytes, BW 1000000 kbit
Full-duplex, 1Gb/s, auto negotiation: on, uni-link: n/a
Up 1 minutes, 37 seconds
Loopback Mode : None
3 link status changes since last clear
Last clearing of "show interface" counters never
5 minutes input rate 0 bps (0.0% with framing overhead), 0 packets/sec
5 minutes output rate 0 bps (0.0% with framing overhead), 0 packets/sec
163 packets input, 0 bytes
Received 0 broadcasts, 0 multicast
0 runts, 0 giants
0 input errors, 0 CRC, 0 alignment, 0 symbol, 0 input discards
0 PAUSE input
11 packets output, 0 bytes
Sent 0 broadcasts, 0 multicast
0 output errors, 0 collisions
0 late collision, 0 deferred, 0 output discards
0 PAUSE output
localhost#

```

Ready Telnet: 192.168.255.60 39, 11 39 Rows, 121 Cols Xterm CAP NUM

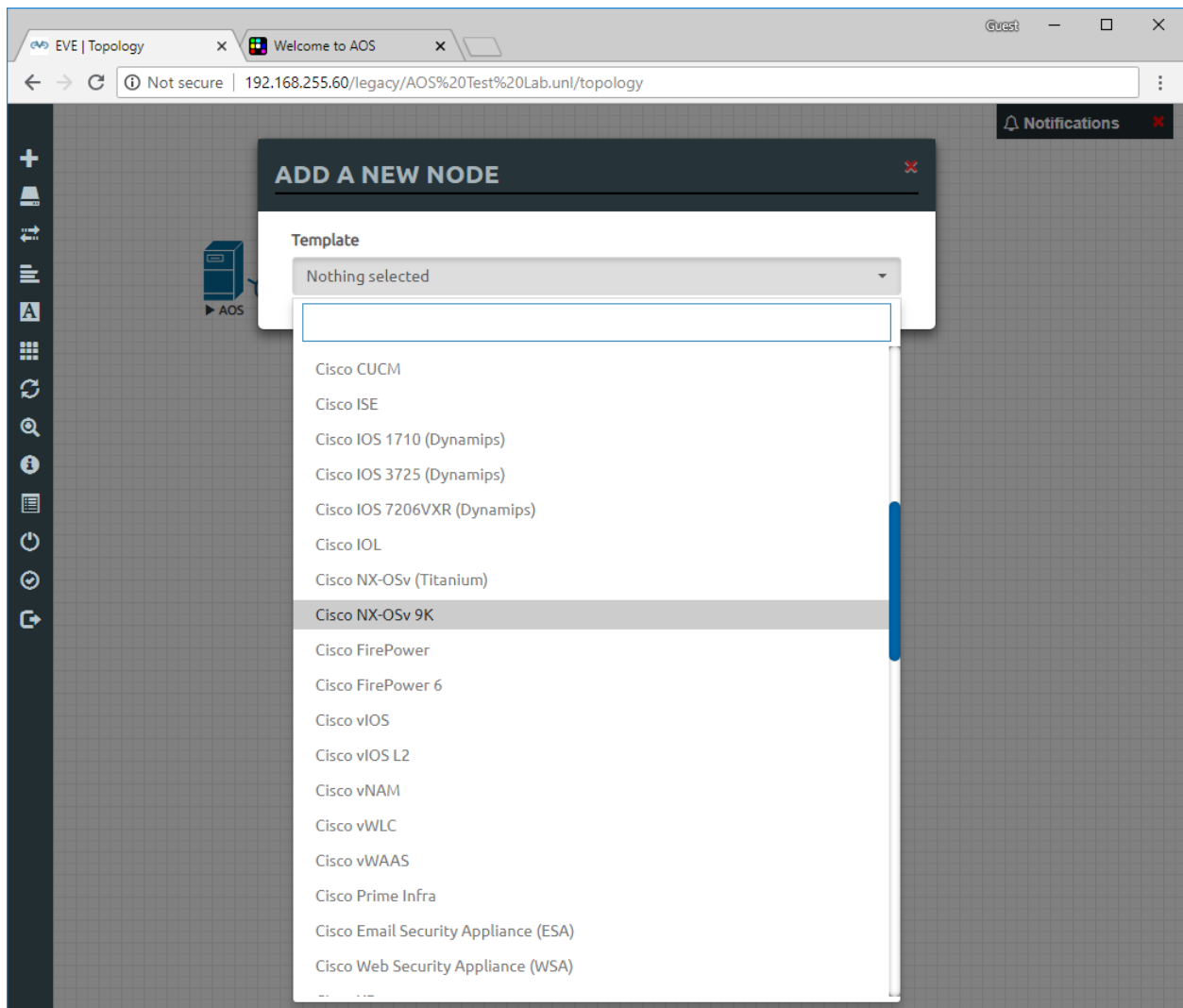
Add Cisco NX-OS Devices

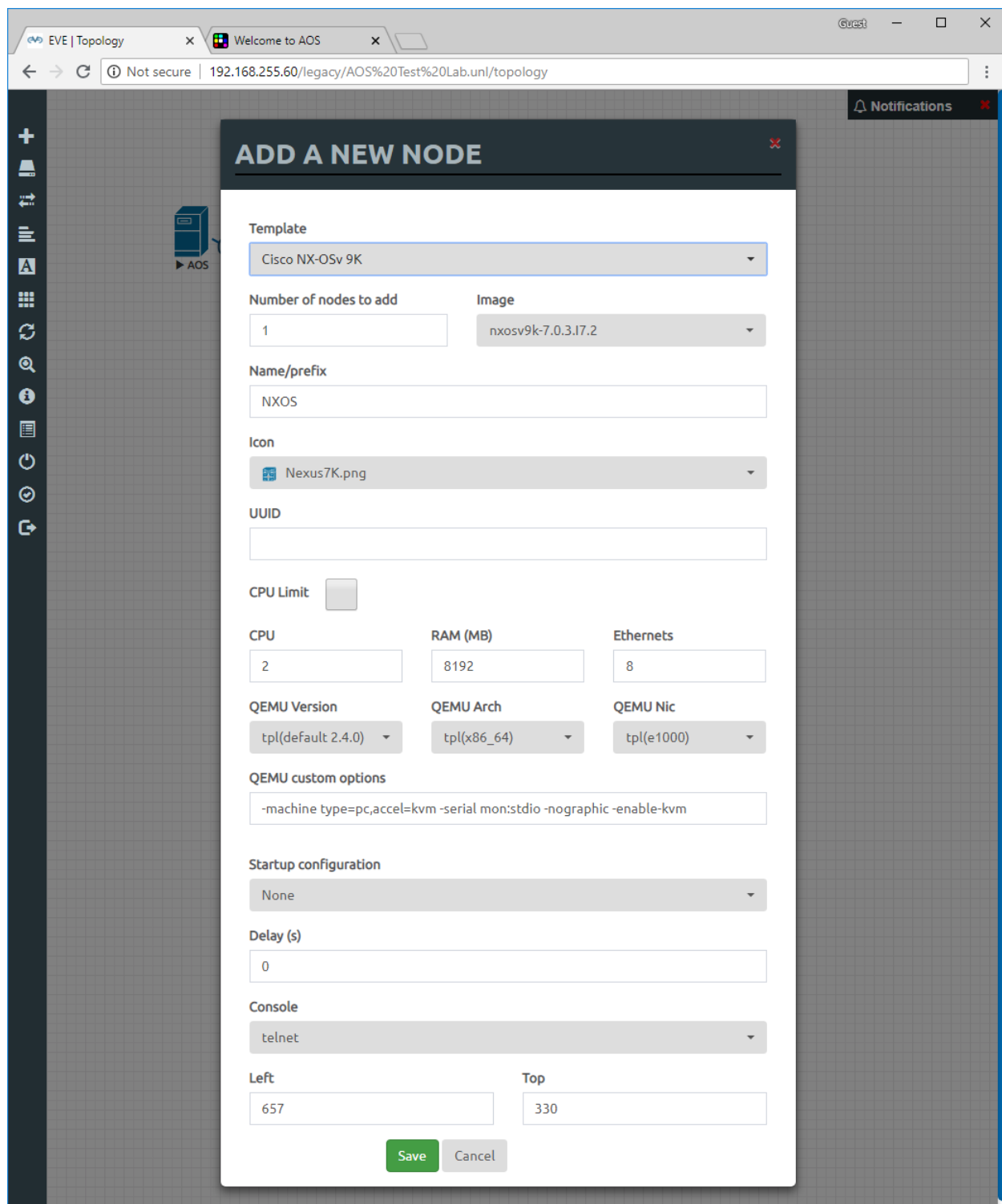
Step 1 Obtain the Cisco NX-OSv 7.0(3)I7(4) QCOW2 (nxosv-final.7.0.3.I7.4.qcow2) image from Cisco and copy to the EVE-NG Server.

Note: Please check the [AOS Feature Matrix](#) for the latest recommended version of Cisco NX-OS. Cisco NX-OS is licensed software and cannot be provided by Apstra Support.

```
root@eve-ng:~# ls -l nxosv-final.7.0.3.I7.4.qcow2
-rw-r--r-- 1 root root 1745027072 Aug 31 20:33 nxosv-final.7.0.3.I7.4.qcow2
root@eve-ng:~# mkdir -p /opt/unetlab/addons/qemu/nxosv9k-7.0.3.I7.4
root@eve-ng:~# cp nxosv-final.7.0.3.I7.4.qcow2 /opt/unetlab/addons/qemu/nxosv9k-7.0.3.
I7.4/sataa.qcow2
root@eve-ng:~# /opt/unetlab/wrappers/unl_wrapper -a fixpermissions
```

Step 2 From the EVE-NG lab, right-click and add a new Node.





ADD A NEW NODE

Template: Cisco NX-OSv 9K

Number of nodes to add: 1

Image: nxosv9k-7.0.3.17.2

Name/prefix: NXOS

Icon: Nexus7K.png

UUID:

CPU Limit: ☐

CPU: 2

RAM (MB): 8192

Ethernets: 8

QEMU Version: tpl(default 2.4.0)

QEMU Arch: tpl(x86_64)

QEMU Nic: tpl(e1000)

QEMU custom options: -machine type=pc,accel=kvm -serial mon:stdio -nographic -enable-kvm

Startup configuration: None

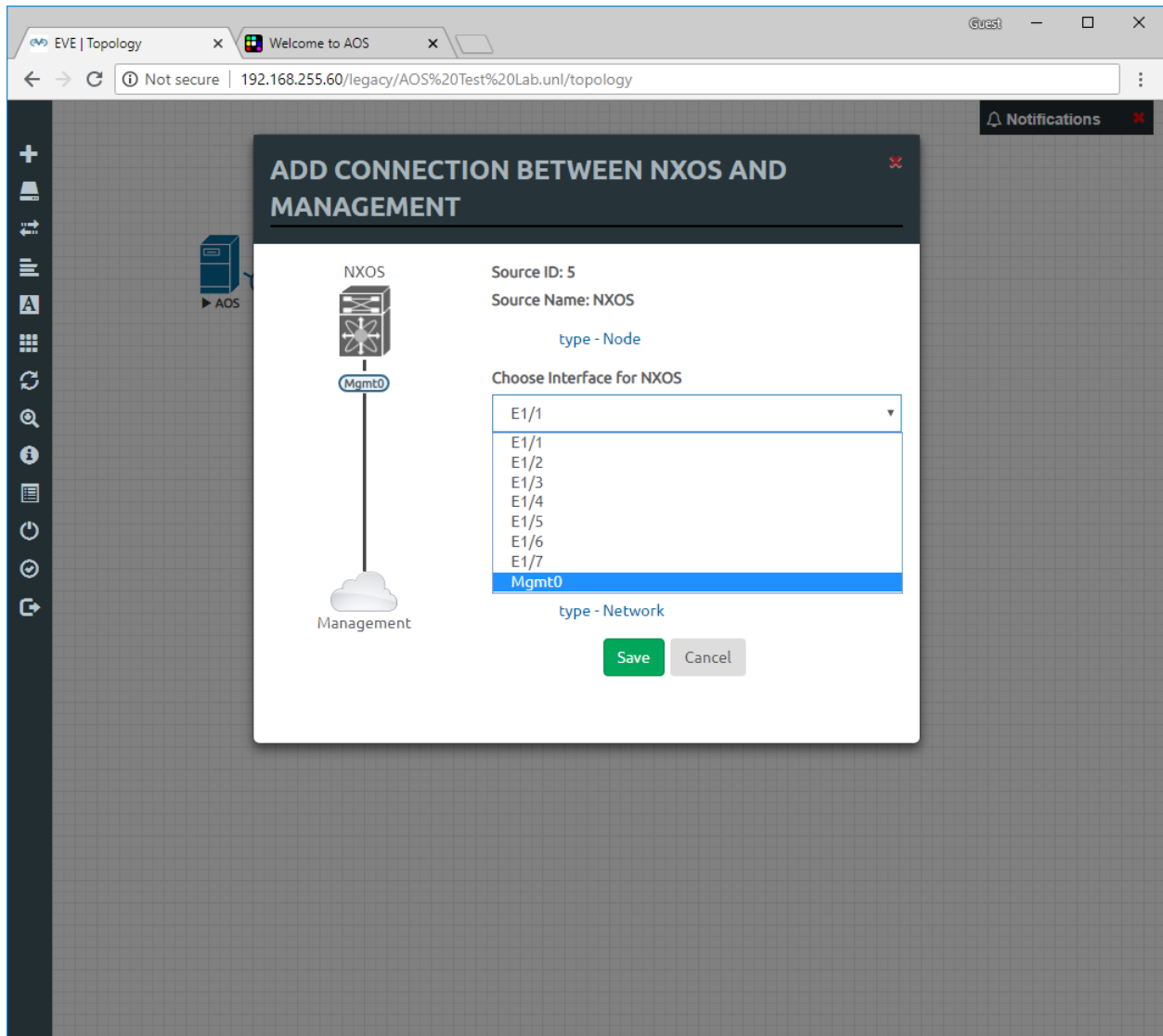
Delay (s): 0

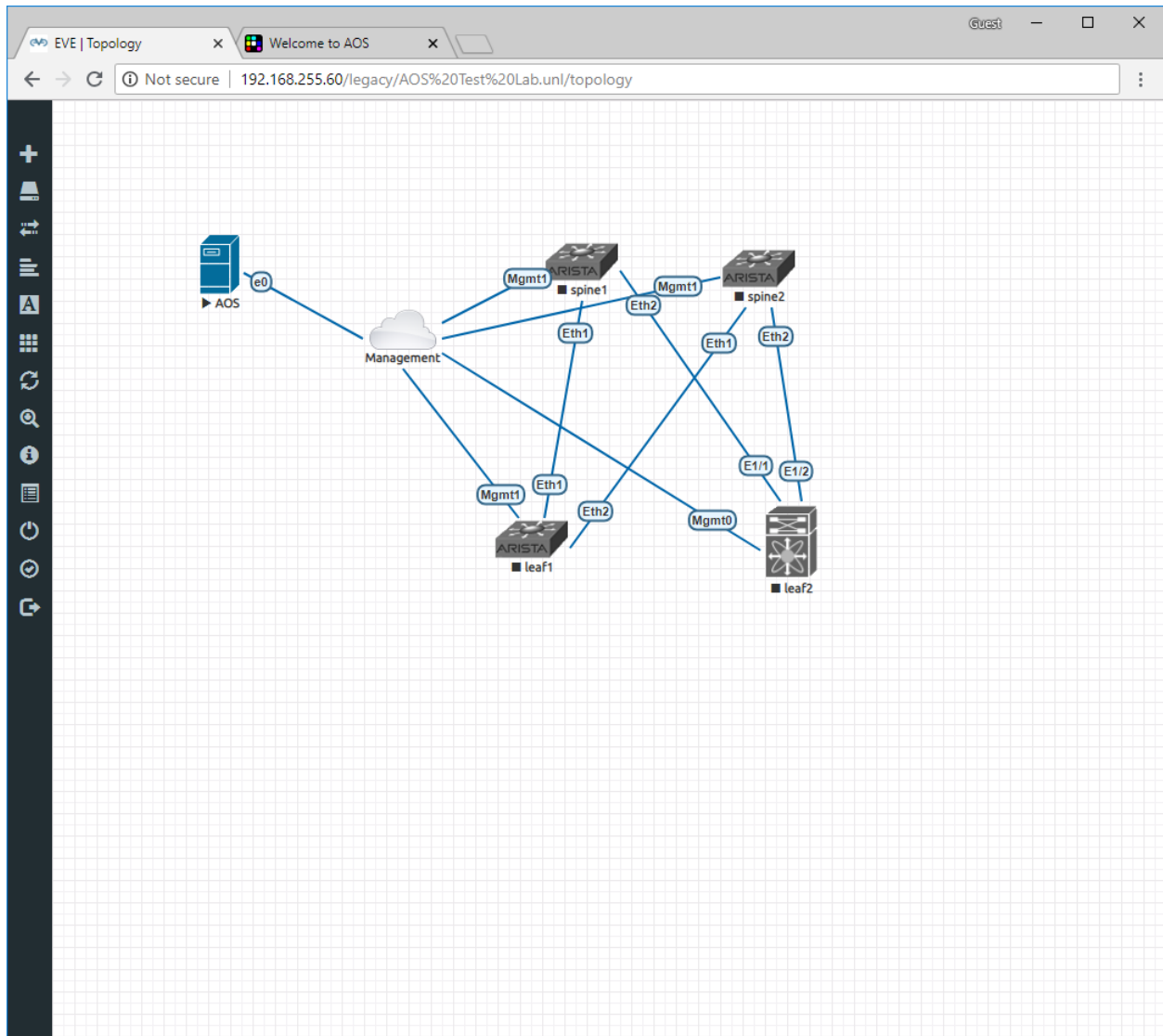
Console: telnet

Left: 657

Top: 330

Save **Cancel**





Step 3 Connect to the NX-OS device console.

Step 4 Configure the necessary management configuration.

```

leaf2
File Edit View Options Transfer Script Tools Window Help
Enter host <Alt+R>
leaf2 x
2018 Sep 11 19:50:39 switch %$ VDC-1 %$ %USER-1-SYSTEM_MSG: SWINIT failed. devid:241 inst:0 - t2usd
Done
Abort Auto Provisioning and continue with normal setup?(yes/no)[n]: 2018 Sep 11 19:50:44 switch %$ VDC-1 %$ %POAP-2-POAP_I
NITED: [9Q9RTQQZ5HZ-50:00:00:05:00:07] - POAP process initialized
y
Disabling POAP

---- System Admin Account Setup ----

Do you want to enforce secure password standard (yes/no)[y]: 2018 Sep 11 19:51:00 switch %$ VDC-1 %$ %VMAN-2-ACTIVATION_ST
ATE: Successfully activated virtual service 'guestshell+'
2018 Sep 11 19:51:00 switch %$ VDC-1 %$ %VMAN-2-GUESTSHELL_ENABLED: The guest shell has been enabled. The command 'guestshe
ll' may be used to access it, 'guestshell destroy' to remove it.
n
Enter the password for "admin":
Confirm the password for "admin": 2018 Sep 11 19:51:24 switch %$ VDC-1 %$ %ASCII-CFG-2-CONF_CONTROL: System ready

---- Basic System Configuration Dialog VDC: 1 ----

This setup utility will guide you through the basic configuration of
the system. Setup configures only enough connectivity for management
of the system.

Please register Cisco Nexus9000 Family devices promptly with your
supplier. Failure to register may affect response times for initial
service calls. Nexus9000 devices must be registered to receive
entitled support services.

Press Enter at anytime to skip a dialog. Use ctrl-c at anytime
to skip the remaining dialogs.

would you like to enter the basic configuration dialog (yes/no): no

```

Ready Telnet: 192.168.255.60 37, 69 37 Rows, 123 Cols Xterm CAP NUM

```

leaf2
File Edit View Options Transfer Script Tools Window Help
Enter host <Alt+R>
leaf2 x
TCAM region is not configured. Please configure TCAM region and retry the command

User Access Verification
login: admin
Password:

Cisco NX-OS Software
Copyright (c) 2002-2017, Cisco Systems, Inc. All rights reserved.
Nexus 9000v software ("Nexus 9000v Software") and related documentation,
files or other reference materials ("Documentation") are
the proprietary property and confidential information of Cisco
Systems, Inc. ("Cisco") and are protected, without limitation,
pursuant to United States and International copyright and trademark
laws in the applicable jurisdiction which provide civil and criminal
penalties for copying or distribution without Cisco's authorization.

Any use or disclosure, in whole or in part, of the Nexus 9000v Software
or Documentation to any third party for any purposes is expressly
prohibited except as otherwise authorized by Cisco in writing.
The copyrights to certain works contained herein are owned by other
third parties and are used and distributed under license. Some parts
of this software may be covered under the GNU Public License or the
GNU Lesser General Public License. A copy of each such license is
available at
http://www.gnu.org/licenses/gpl.html and
http://www.gnu.org/licenses/lgpl.html
*****
* Nexus 9000v is strictly limited to use for evaluation, demonstration *
* and NX-OS education. Any use or disclosure, in whole or in part of *
* the Nexus 9000v Software or documentation to any third party for any *
* purposes is expressly prohibited except as otherwise authorized by *
* Cisco in writing. *
*****
switch#

```

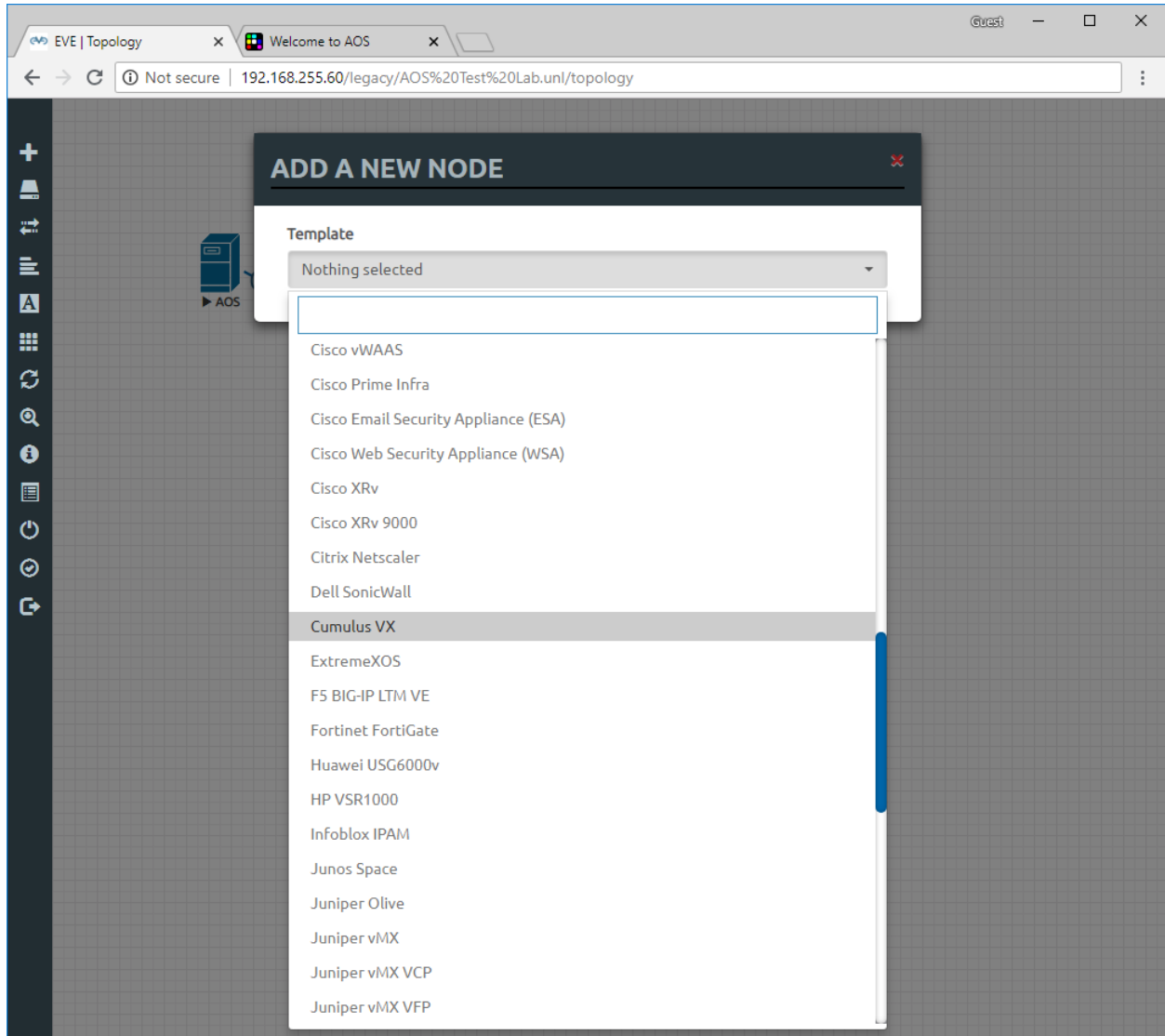
Ready Telnet: 192.168.255.60 37, 9 37 Rows, 123 Cols Xterm CAP NUM

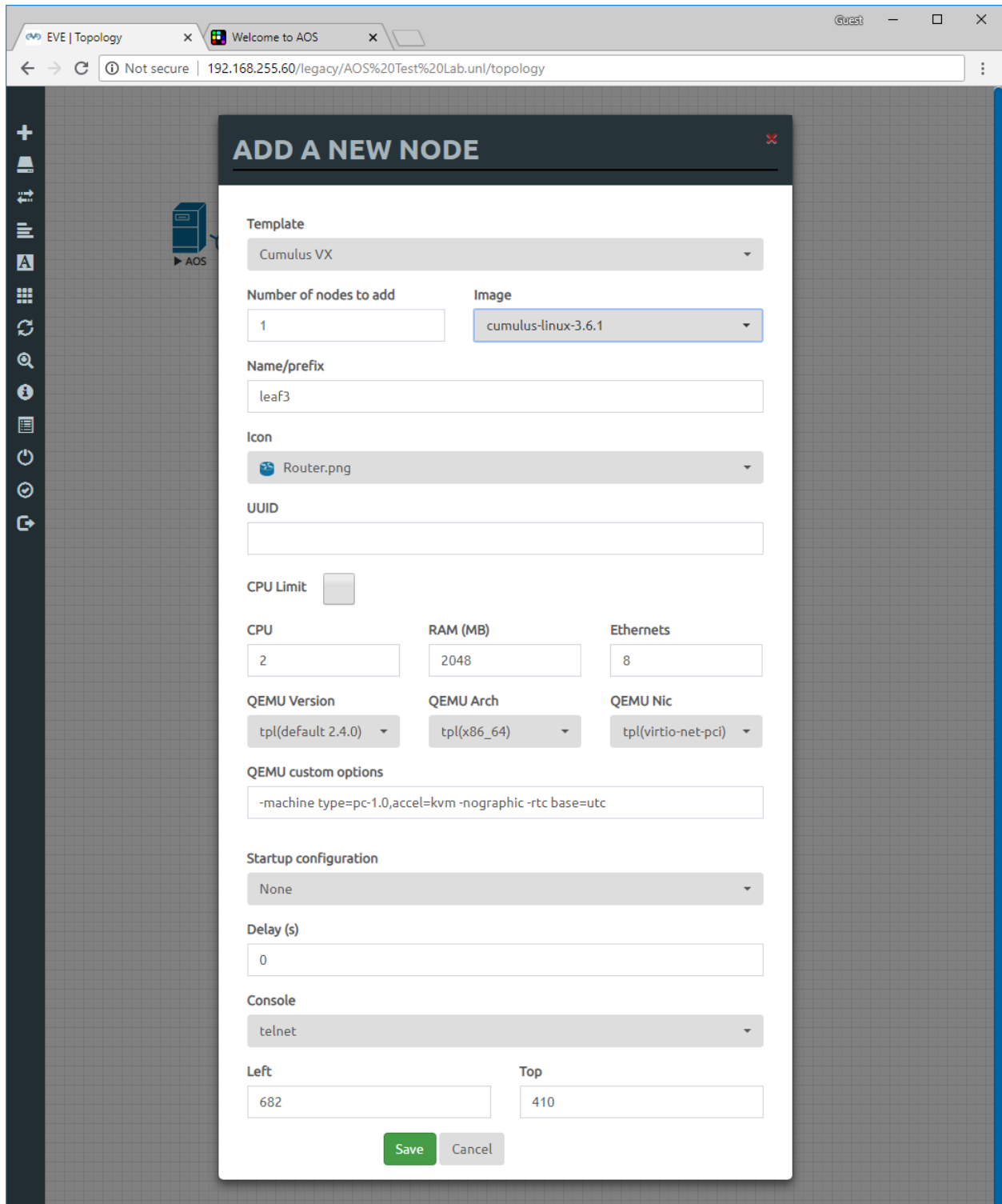
Add Cumulus Linux Devices

Step 1 Obtain the Cumulus Linux CVX QCOW2 (e.g. cumulus-linux-3.7.3-vx-amd64.qcow2) image from Cumulus and copy to the EVE-NG Server.

Note: Please check the [AOS Feature Matrix](#) for the latest recommended version of Cumulus Linux. Cumulus Linux is licensed software and cannot be provided by Apstra Support.

```
root@eve-ng:~# ls -l cumulus-linux-3.7.3-vx-amd64.qcow2
-rw-r--r-- 1 root root 1068761088 Aug 31 20:33 cumulus-linux-3.7.3-vx-amd64.qcow2
root@eve-ng:~# mkdir -p /opt/unetlab/addons/qemu/cumulus-linux-3.7.3
root@eve-ng:~# cp cumulus-linux-3.7.3-vx-amd64.qcow2 /opt/unetlab/addons/qemu/cumulus-
linux-3.7.3/hda.qcow2
root@eve-ng:~# /opt/unetlab/wrappers/unl_wrapper -a fixpermissions
```





ADD A NEW NODE

Template: Cumulus VX

Number of nodes to add: 1

Image: cumulus-linux-3.6.1

Name/prefix: leaf3

Icon: Router.png

UUID:

CPU Limit: ☐

CPU: 2

RAM (MB): 2048

Ethernets: 8

QEMU Version: tpl(default 2.4.0)

QEMU Arch: tpl(x86_64)

QEMU Nic: tpl(virtio-net-pci)

QEMU custom options: -machine type=pc-1.0,accel=kvm -nographic -rtc base=utc

Startup configuration: None

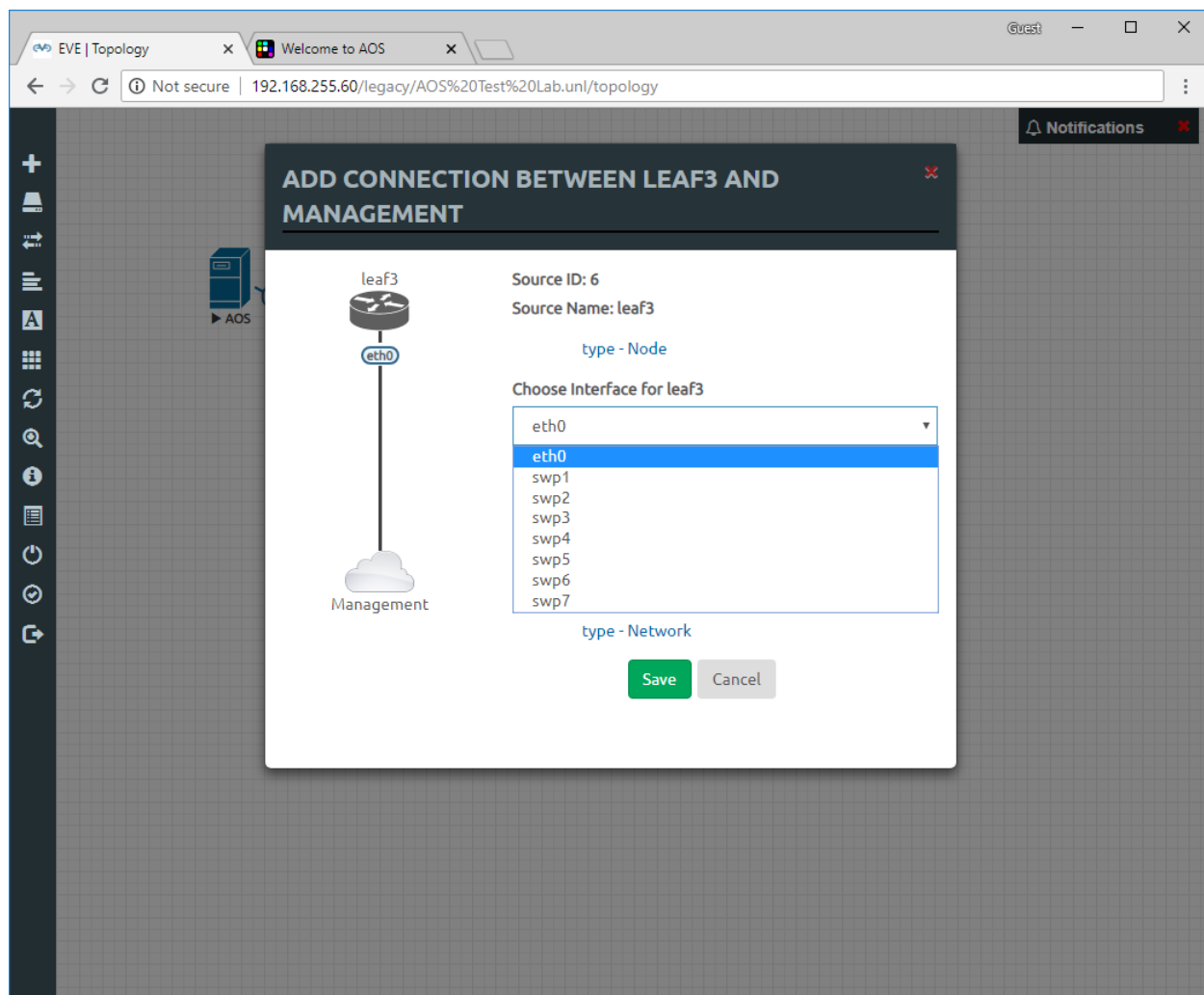
Delay (s): 0

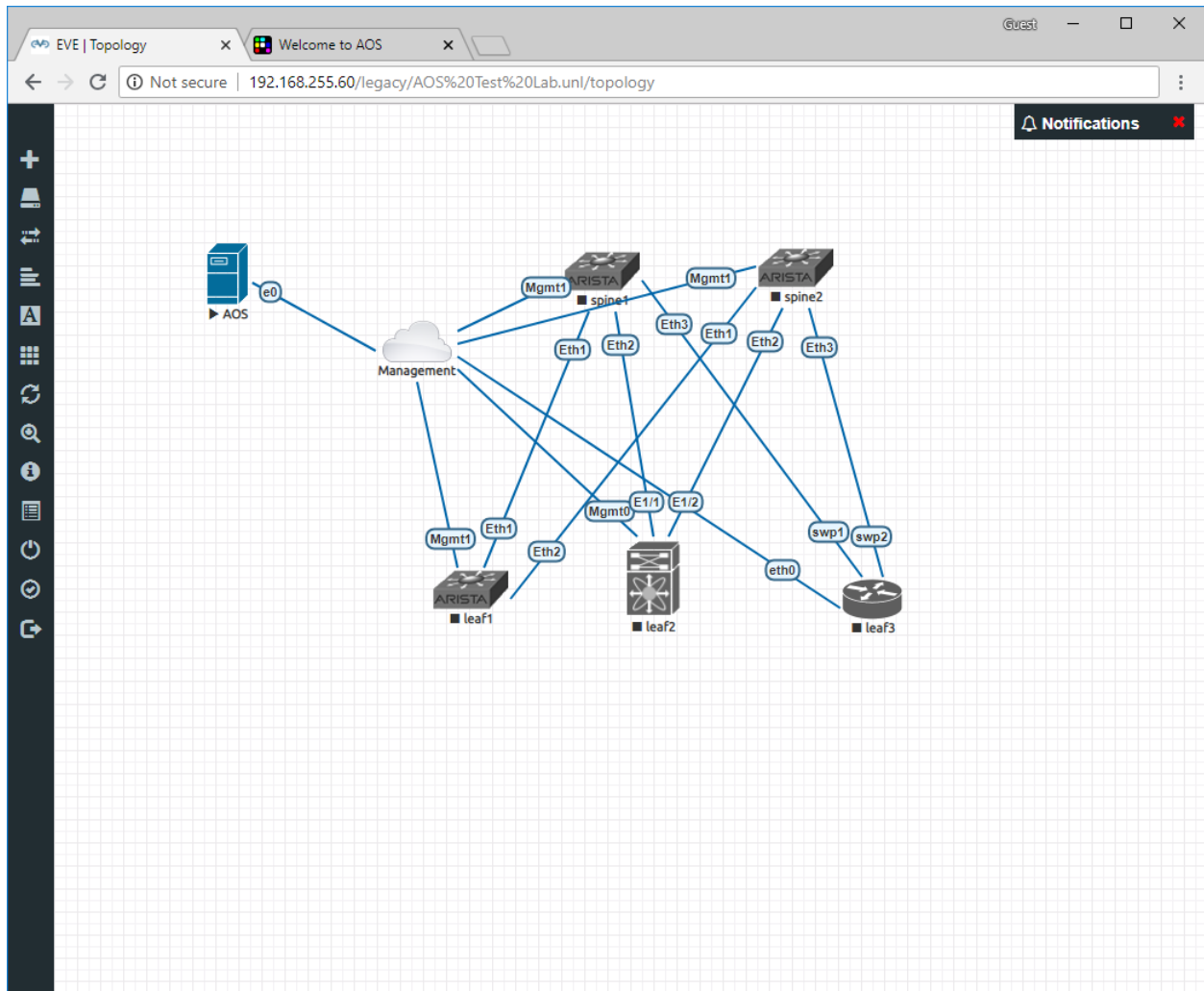
Console: telnet

Left: 682

Top: 410

Save **Cancel**

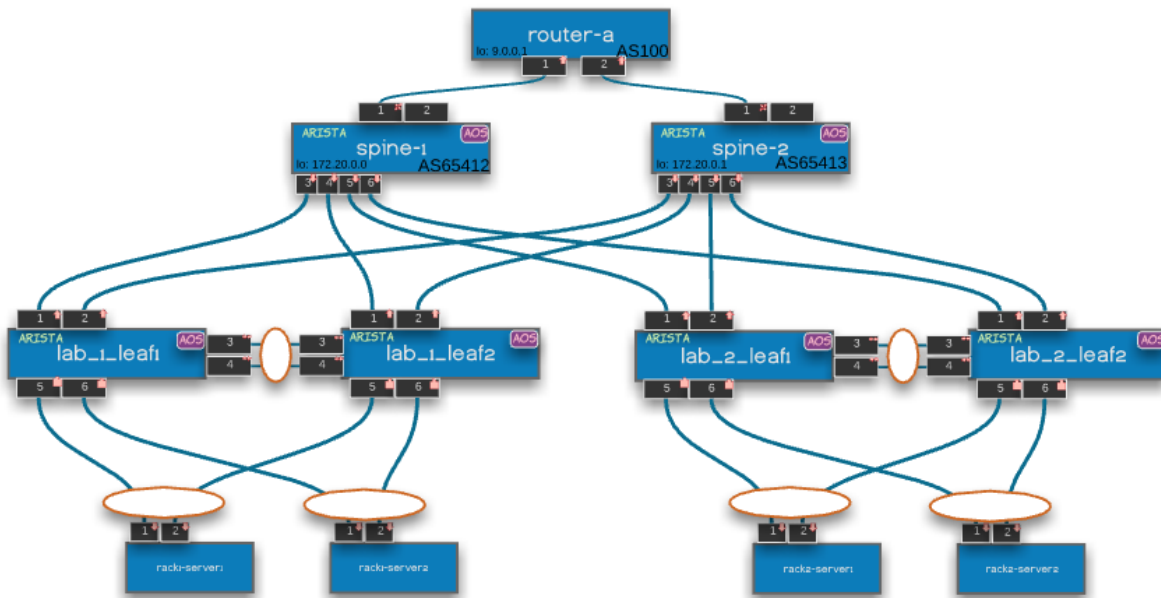




After devices have been added, proceed with installing the agents (**Devices / System Agents / Agents**).

12.4.6.2 Build an ESXi vEOS Lab

This document describes how to build a dual MLAG leaf-pair running Arista vEOS inside of VMware ESXi.



Obtain and Deploy AOS Server OVA in VMware ESXi

Install the AOS server from the “Software Downloads” section and deploy it under VMware ESXi. After the server is built, log in with username `admin` password `admin`, and assign an IP address by editing `/etc/network/interfaces`.

The AOS server is distributed as a virtual-machine image in OVA format. You will need to import this OVA into your hypervisor environment as described in the installation section.

The AOS server defaults to use DHCP to acquire its network address. You can statically configure the IP address after the server is initially installed. Please ensure that AOS server IP-address or AOS server hostname value is properly configured in the device-agent `aos.conf` file. Refer to the Vendor-specific device-agent installation guides for further details.

Start the VMware vSphere Client and connect to the ESXi host or vCenter Server managing that host on which we want to deploy the AOS Server VM.

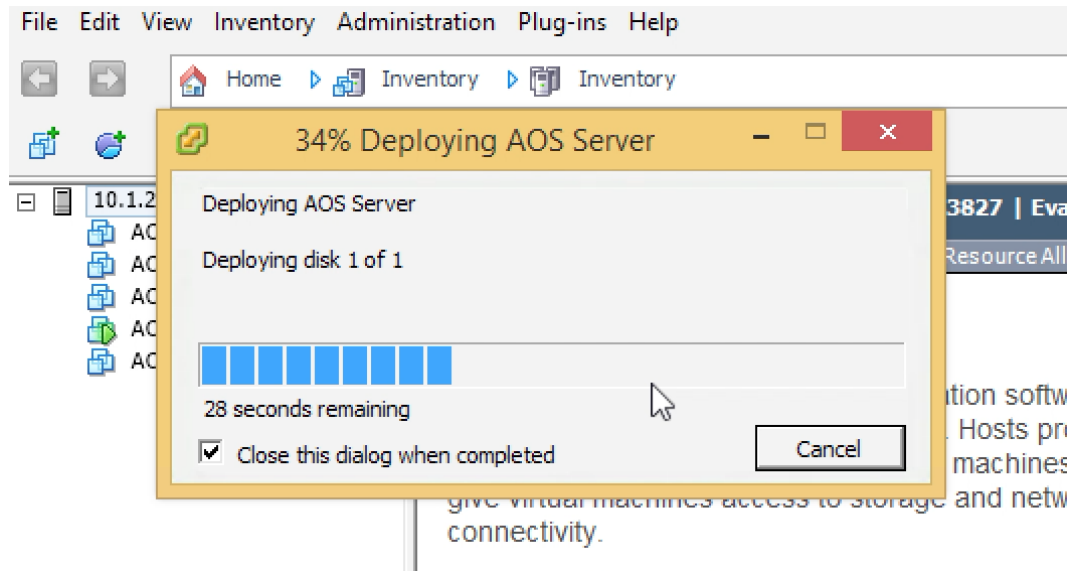
Step 1 File->Deploy OVF template

Step 2 Provide the HTTP URL or the path in the local filesystem of the AOS Server OVA file and click “Next” until a name for the Virtual Machine is asked.

Step 3 Select the Disk format “Thin Provision”

Step 4 A summary screen with the VM configuration will appear. Check “Power on after deployment” and click Finish.

Step 5 A progress bar will indicate the deployment status, at the end of which the VM will boot automatically.



Step 6 To open a console window on the Apstra server, right click on the Apstra server VM -> Open Console

Follow *Installing Apstra server* for setting up the Apstra server VM.

Build Arista vEOS Template in ESXi

Building vEOS images

This document contains instructions on how to build an Arista vEOS image for use with AOS.

In order to build a vEOS lab, we need some images

- Arista vEOS VMDK, appropriate for your hypervisor
- ABoot image (Optional)
- Arista EOS SDK (Required)

Download Arista vEOS

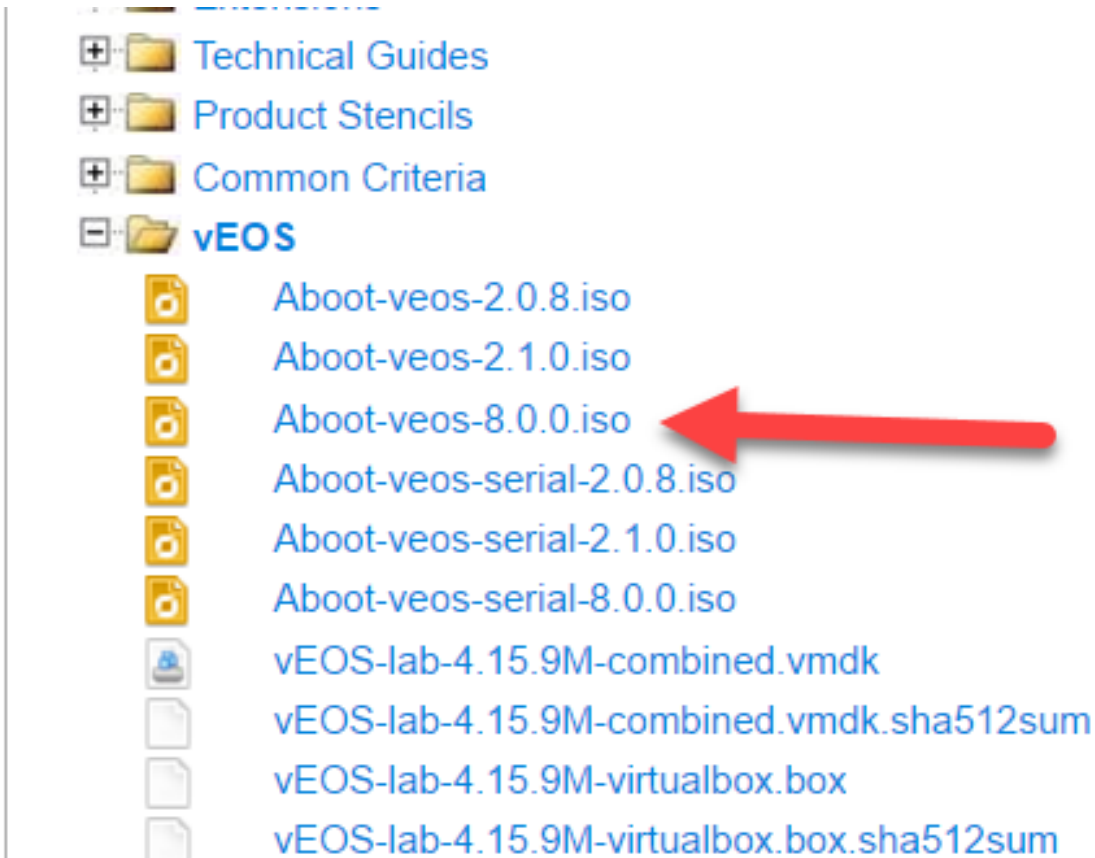
Obtain the Arista vEOS image from <https://www.arista.com/en/support/software-download>

Download Arista Aboot

The Arista ABoot ISO is used to bootstrap the VM to begin loading from the IDE:0 hard drive.

Note: The ABoot ISO is only used if the customer deploys the vEOS-lab images. vEOS-combined images already include the ABoot as part of the vmk image. We recommend -combined- images.

The Aboot image is available from the vEOS section on <https://www.arista.com/en/support/software-download>.



Arista vEOS configuration specifics

Minimum resources

| Minimum RAM | Minimum vCPU |
|-------------|--------------|
| 2.5 GB | 1 vCPU |

Arista vEOS requires 2.5GB and at least 1 vCPU for a stable installation.

While vEOS itself can run on 512mb, the AOS Agent takes about another 1.5GB of ram.

Performance note

Arista vEOS is used for **Demo Purposes Only**. Arista vEOS suffers from a lack of performance due to the way it forwards traffic. All of the 'front panel' ports - eg, Eth1, Eth2, Eth3, etc, operate in the CPU plane. There is a software emulation layer, called the *Ethernet Bridging Agent* or *Etba* for short.

Etba is responsible for all L2 and L3 traffic forwarding, spanning-tree decisions, BFD, etc. *Etba* cannot provide line-speed performance. At Apstra, we have tested that vEOS may max out at only **100Kbps** of traffic. On complicated L2 networks, this performance is even slower. These performance issues do not appear in a network based on hardware switches. We cannot improve the performance of vEOS.

When networks are deployed on top of vEOS, Apstra AOS may take a few minutes to converge. If an network administrator runs `top` on the switch, we observe that *Etba* is maxing out the CPU as it is performing calculations.

The Arista vEOS platform also exhibits the behavior of consuming 100% of a vCPU.

Warning: When a vEOS instance co-habitates another hypervisor, other applications will suffer.

To reiterate, these performance problems do not appear on physical hardware switches.

vEOS VXLAN Limitation with MLAG

The Arista implementation of VXLAN and MLAG on vEOS does not have functionality to share MAC addresses across a peer-link. This means that when applications are forwarding L2 traffic on an arista VXLAN network deployed with vEOS, traffic that is routed to leaf2 will not be forwarded across peer-link to leaf1. This will give the behavior of very long ping times and extremely high drop rates to remote vxlan endpoints. These are vEOS limitations. Packets may suddenly start dropping as the fabric changes an ECMP hash somewhere.

When configuring vxlan on vEOS, **do not deploy VXLAN on MLAG Pairs with vEOS** otherwise traffic will be dropped. Non-MLAG pairs operate normally with VXLAN and AOS 2.0+.

This is only an issue for VXLAN deployed on vEOS.

Promiscuous mode

On some hypervisors, Arista vEOS also requires the *management1* interface to be marked as *promiscuous*. This allows the ma1 interface to respond to packets on the automatically generated MAC address used for the ma1 interface when it's configured in a VRF.

Warning: If the management interface is not marked as promiscuous on a virtual platform, vEOS will not be able to send or receive any traffic. Ensure the hypervisor has the Management interface configured as promiscuous.

We can make use of a script that runs with the startup of the interface to force the nic to be promiscuous.

Listing 21: Automatically set Management1 interface as promiscuous when booting

```
event-handler ma-promisc
action bash /sbin/ifconfig ma1 promisc
delay 10
trigger on-intf Management1 operstatus
```

Change vEOS VM NIC type

For VMware ESXi, you may need to modify the .vmx file to ensure that the NICs are set to e1000 as the NIC type to make sure all nics are Intel.

Important: Failure to modify NIC types may cause the agent to fail in unexpected ways - LLDP/LACP may fail, L2 may not flow traffic, or interfaces may never come up, or a management NIC may never get a DHCP lease during ZTP.

Listing 22: ESXi NIC types

```

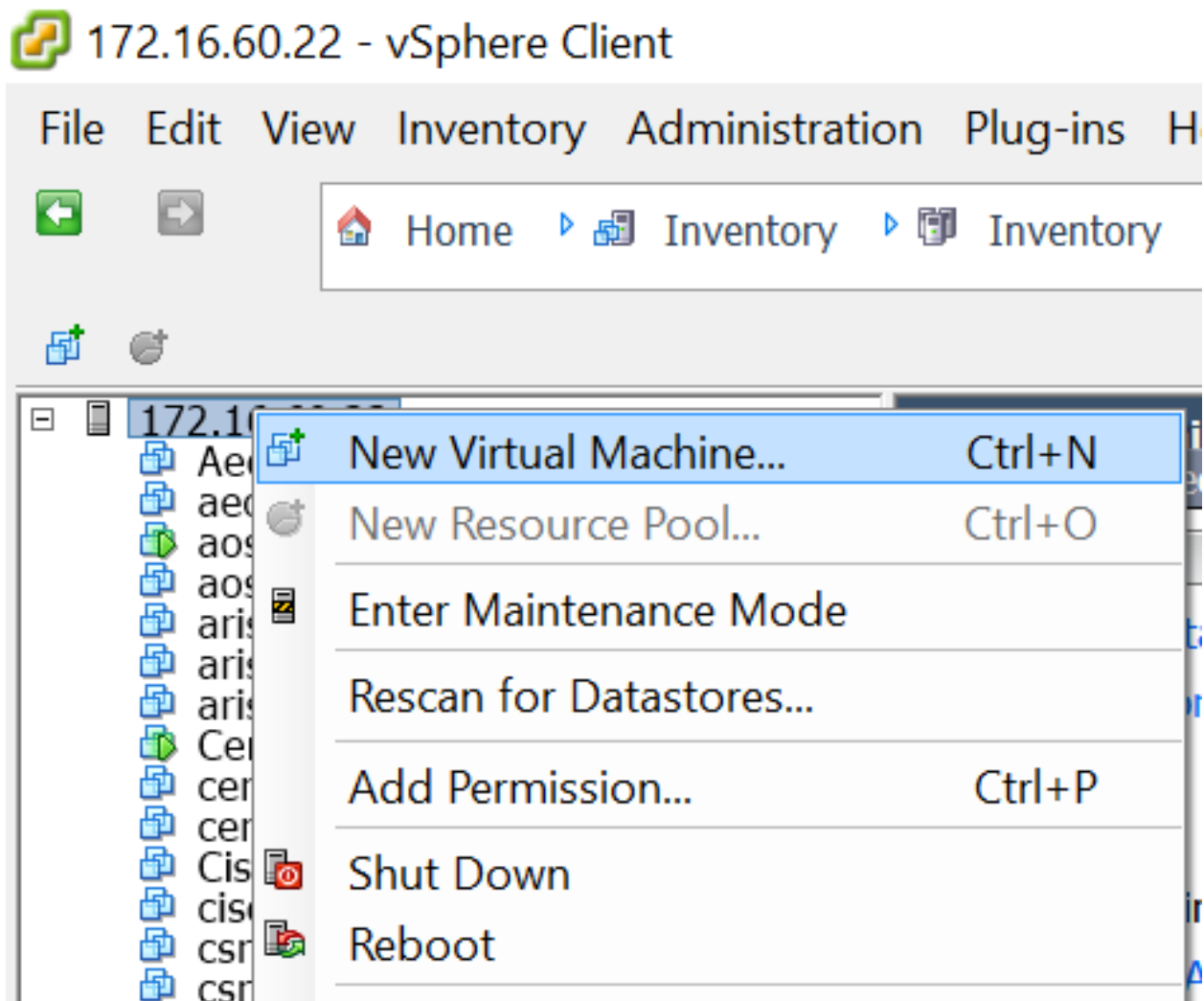
ethernet0.present = "TRUE"
ethernet0.virtualDev = "e1000"
ethernet1.present = "TRUE"
ethernet1.virtualDev = "e1000"
ethernet2.present = "TRUE"
ethernet2.virtualDev = "e1000"

```

For Virtualbox, the NIC type can be set in the GUI.

Creating the VM template

Step 1 In vSphere, create a new virtual machine.



Step 2 Choose a storage location and a name for the VM.




The details of your storage location will be specific to your environment.

Step 3 Set VM HW Version 8

Setting VM HW version 8 allows you to perform all of the operations on a VM using the legacy ESX console application without any errors.

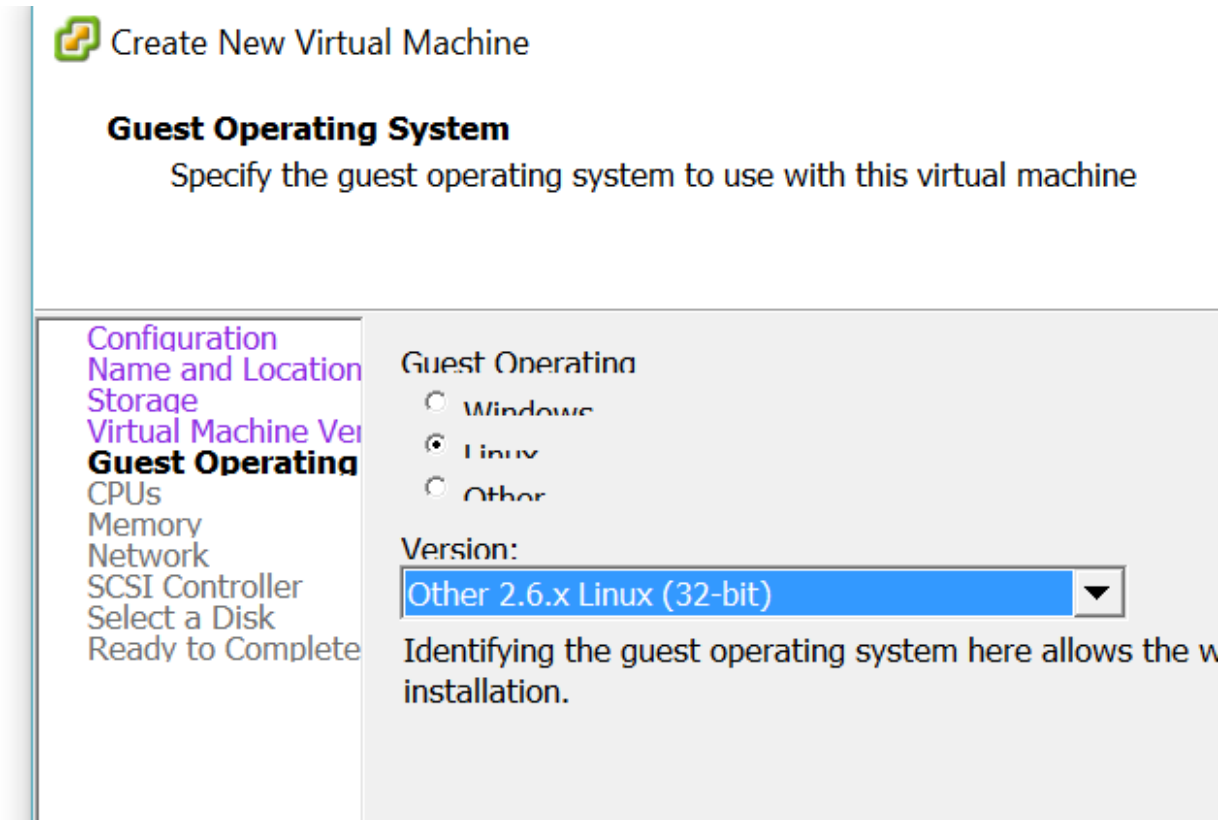
Create New Virtual Machine

Virtual Machine Version

| | |
|---|--|
| <p>Configuration</p> <p>Name and Location</p> <p>Storage</p> <p>Virtual Machine Version</p> <p>Guest Operating System</p> <p>CPU</p> <p>Memory</p> <p>Network</p> <p>SCSI Controller</p> <p>Select a Disk</p> <p>Ready to Complete</p> | <p>Virtual Machine Version</p> <p>This host or cluster supports more than one VMware virtual machine version.</p> <ul style="list-style-type: none"> <input type="radio"/> Virtual Machine Version: 4 <p>This version will run on VMware ESX 3.0 and later, and VM sharing storage or virtual machines with ESX up to 3.5.</p> <input type="radio"/> Virtual Machine Version: 7 <p>This version will run on VMware ESX/ESXi 4.0 and later. This version is not supported on hosts with ESX/ESXi up to 4.1.</p> <input checked="" type="radio"/> Virtual Machine Version: 8 <p>This version will run on VMware ESXi 5.0 and later. Choose this version if you do not need to migrate to ESX/ESXi 4.</p> <input type="radio"/> Virtual Machine Version: 9  <p>This version will run on VMware ESXi 5.1 and later.</p> <input type="radio"/> Virtual Machine Version: 10  <p>This version will run on VMware ESXi 5.5 and later.</p> <input type="radio"/> Virtual Machine Version: 11  <p>This version will run on VMware ESXi 6 and later.</p> |
|---|--|

Step 4 For Guest OS, choose Other 2.6 Linux (32-Bit)

This setting is recommended by Arista.



Create New Virtual Machine

Guest Operating System
Specify the guest operating system to use with this virtual machine

Configuration
Name and Location
Storage
Virtual Machine Version
Guest Operating System
CPUs
Memory
Network
SCSI Controller
Select a Disk
Ready to Complete

Guest Operating System

☐ Windows
☒ Linux
☐ Other

Version:
Other 2.6.x Linux (32-bit) ▼

Identifying the guest operating system here allows the w installation.

Step 5 Set a management NIC. We will add the other 7 front-panel nics in a future step.



Create New Virtual Machine

Network

Which network connections will be used by the virtual machine?

Configuration

Name and Location

Storage

Virtual Machine Ver

Guest Operating Sy

CPU

Memory

Network

SCSI Controller


Select a Disk

Ready to Complete

Create Network Connections

How many NICs do you want to

| | Network | Adapter | Conne |
|-----|---------------------------------------|------------------------------------|-------------------------------------|
| NIC | <input type="text" value="AOS_DHCP"/> | <input type="text" value="E1000"/> | <input checked="" type="checkbox"/> |

 If supported by this virtual machine version, more than

Adapter choice can affect both networking performance and migration compatibility. Consult the [VMware KnowledgeBase](#) for more information on choosing among the network adapter

Step 6 Create a LSI Logic Parallel Apstra server (default).

This is required to ensure the first disk attaches in IDE Mode.



Create New Virtual Machine

SCSI Controller

Which SCSI controller type would you like to use?

Configuration
Name and Location
Storage
Virtual Machine Ver
Guest Operating Sy
CPUs
Memory
Network
SCSI Controller
Select a Disk
Ready to Complete

SCSI controller

- ☐ BusLogic Parallel
- ☒ LSI Logic Parallel
- ☐ LSI Logic SAS
- ☐ VMware Paravirtual

Step 7 When prompted, Do NOT create a disk.



Create New Virtual Machine

Select a Disk

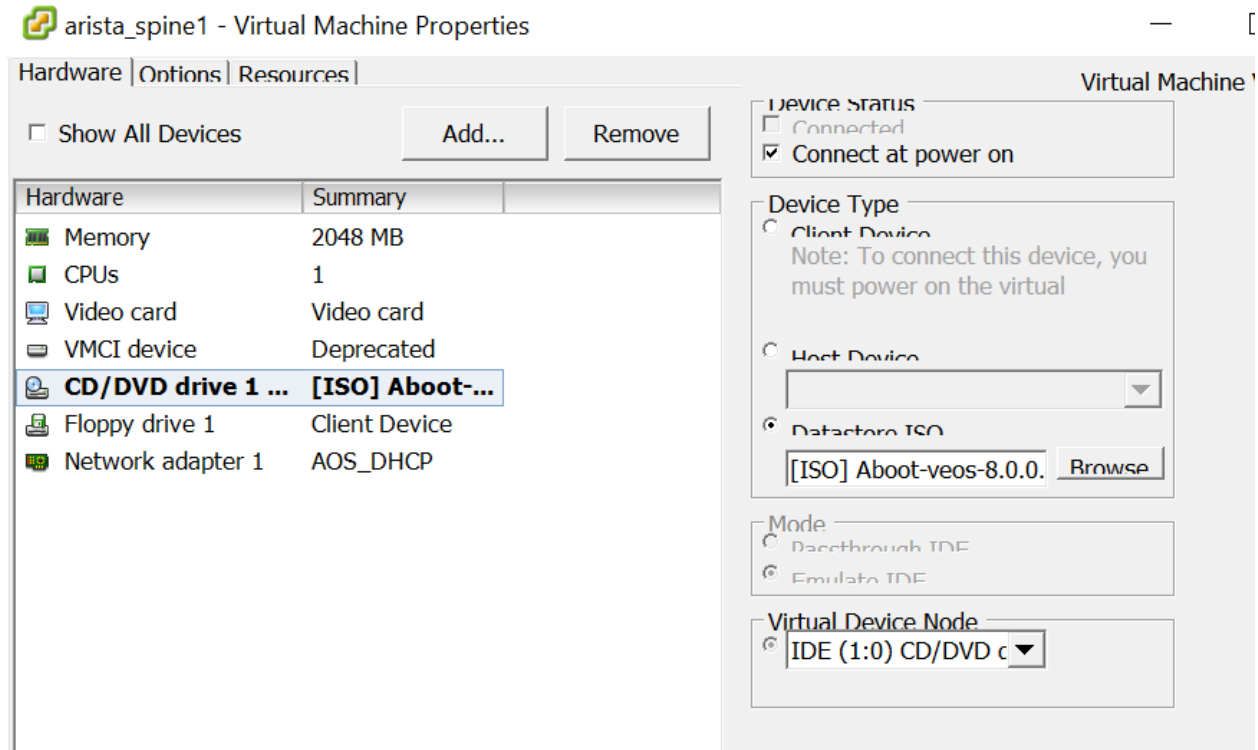
| | |
|--|--|
| <ul style="list-style-type: none"> Configuration Name and Location Storage Virtual Machine Version Guest Operating System CPU Memory Network SCSI Controller Select a Disk Ready to Complete | <p>A virtual disk is composed of one or more files.</p> <p>Select the type of disk to use.</p> <p>Disk</p> <ul style="list-style-type: none"> <input type="radio"/> Create a new virtual disk <input type="radio"/> Use an existing virtual disk <p>Reuse a previously configured virtual disk</p> <input type="radio"/> Raw Device <p>Give your virtual machine direct access to a physical disk. This option allows you to use existing disks.</p> <input checked="" type="radio"/> Do not create disk |
|--|--|

Step 8 Create the VM and edit VM settings.

Step 9 Mount the ABoot ISO you downloaded and make sure *Connect at power-on* is checked.

Note: This step is optional if using the `vEOS-combined vmdk` image.

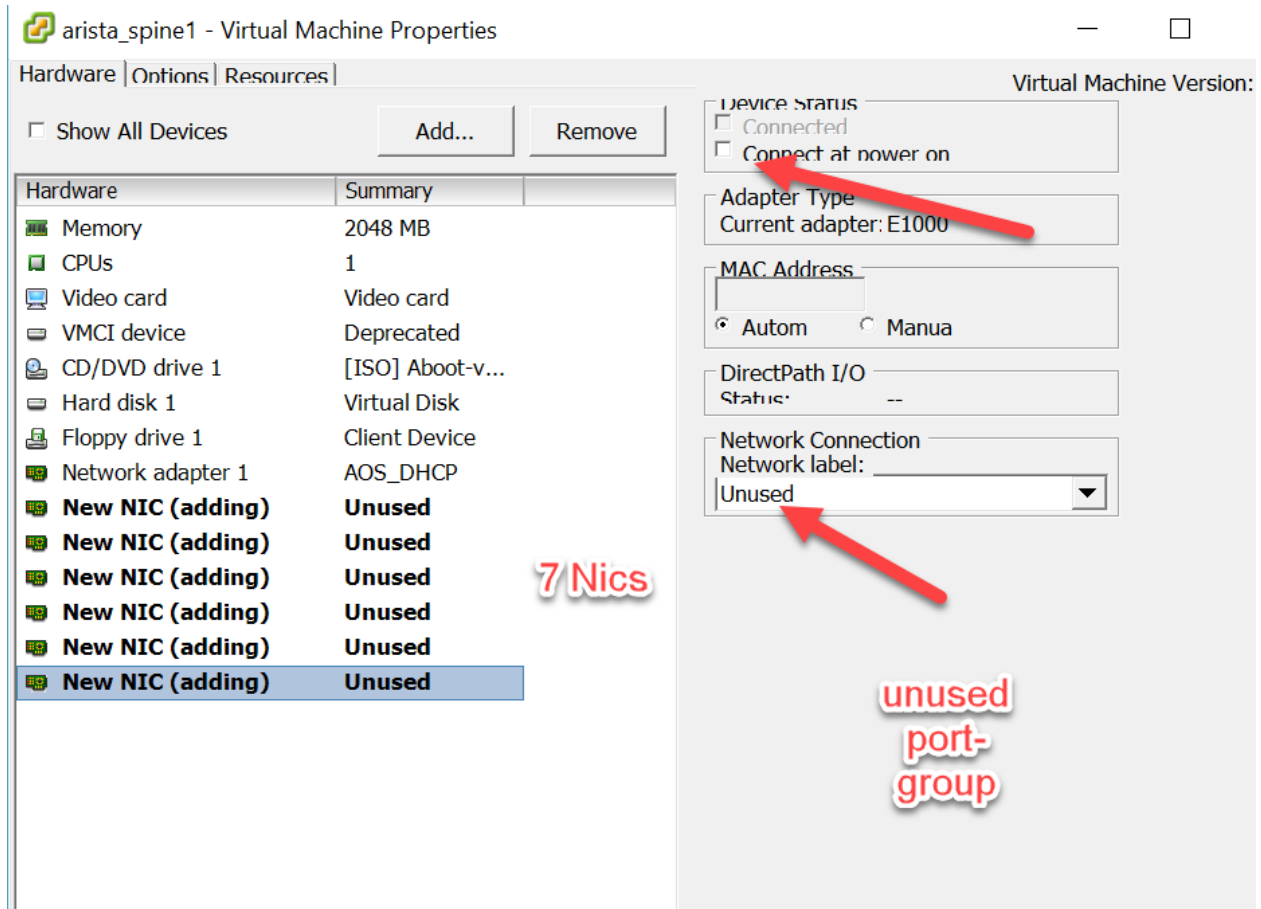
Warning: Mounting the ABoot ISO from a shared location can cause deadlocking issues on your storage infrastructure. Copy a unique ABoot ISO to each VM guest to avoid issues when booting up your VM.



Step 10 Add 7 more NICs to an “Unused” VMware Port-group.

Note: Make sure Adapter type remains E1000, connect at power-on is UNSELECTED, and network label is unused. If the ‘connect at power-on’ setting is set to yes, this may bridge all of the VMs together when they boot up with L2 interfaces.

A further step will set the NIC bindings properly.



Step 11 Copy the Arista VMDK to the VM folder. You may have to browse for this.

Warning: Do not share the VMDK with multiple vEOS images - this will cause the vEOS installation to become corrupted

```
[root@host2:/vmfs/volumes/a25f43f6-f2c3bc72/arista_lab_2_leaf_2] ls -lah
total 507788
drwxr-xr-x    2 root    root        4.0K Mar  9 00:29 .
drwxrwxrwx   59 root    root        4.0K Mar  9 00:10 ..
-rw-r--r--    1 root    root          0 Mar  9 00:11 arista_lab_2_leaf_2.vmsd
-rwxr-xr-x    1 root    root        2.4K Mar  9 00:12 arista_lab_2_leaf_2.vmx
-rwxr-xr-x    1 root    root       495.9M Mar  9 00:29 vEOS-lab-4.16.6M.vmdk
[root@host2:/vmfs/volumes/a25f43f6-f2c3bc72/arista_lab_2_leaf_2]
```

Step 12 On ESXi 6.0, we have to convert the disk to thick format.

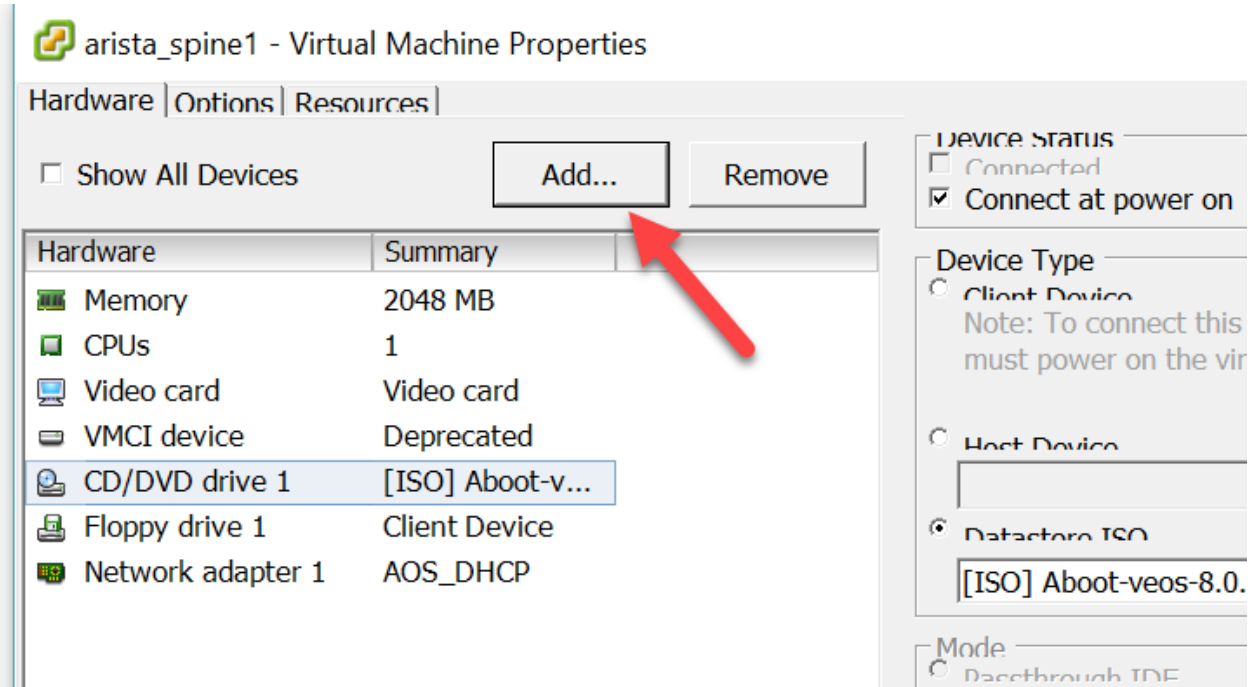
Note: The disk format that Arista ships the VMDK with may not boot properly when mounting. See <https://eos.arista.com/tip-for-arista-veos-on-vmware-esx-6/>

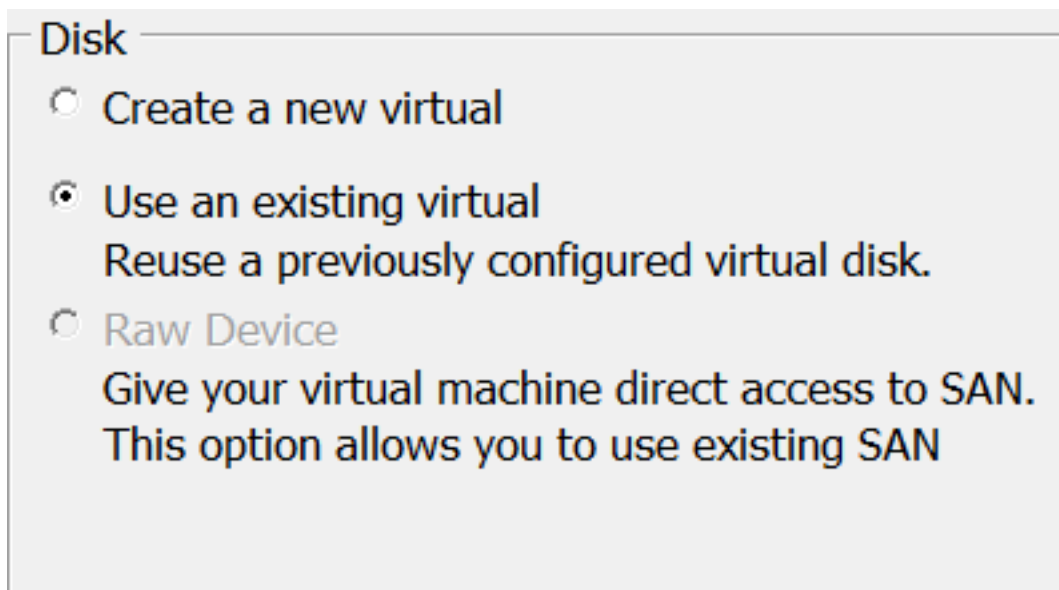
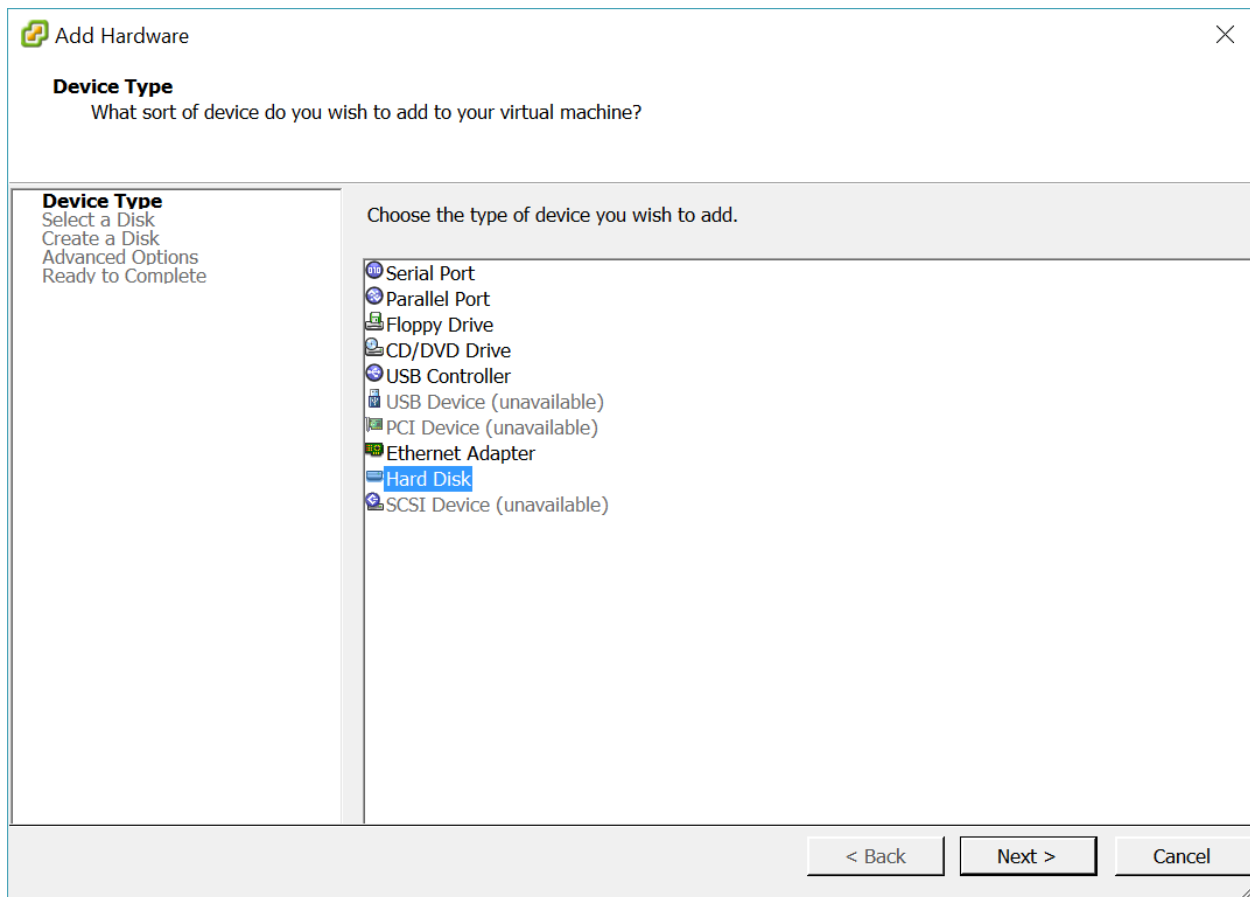
```
Run          vmkfstools -i vEOS-lab-4.16.6M.vmdk -d eagerzeroedthick vEOS-lab-4.16.6M-thick.vmdk
```



```
[root@host2:/vmfs/volumes/a25f43f6-f2c3bc72/arista_spine1] vmkfstools -i vEOS-lab-4.
↳16.6M.vmdk -d eagerzeroedthick vEOS-lab-4.16.6M-thick.vmdk
Destination disk format: VMFS eagerzeroedthick
Cloning disk 'vEOS-lab-4.16.6M.vmdk'...
Clone: 99% done.
...
Clone: 100% done.
```

Step 13 Add the VMDK as a previously mounted hard drive in ESX





Step 14 Ensure the VMDK and HDD are attached to IDE: (0:0).

Warning: Connecting the HDD to the wrong slot will cause vEOS to fail to boot

Specify the advanced options for this virtual disk.

Virtual Device Node
IDE (0:0) ▼

Mode
☐ Independent
Independent disks are not affected by snapshots.
☐ Persistent
Changes are immediately and permanently
☐ Nonpersistent
Changes to this disk are discarded when you

Step 15 Boot the switch.

```

arista_spine1 on host2.local.mistrust.ca
File View VM
Data in /mnt/flash/vEOS-lab.swi differs from previous boot image on /mnt/flash.
Saving new boot image to /mnt/flash...t (C) 1994-2011 H. Peter Anvin et al
Switching rootfs..
Loading initrd.....ready.
Welcome to Arista Networks EOS 4.16.6M
Mounting filesystems: [ OK ]
Starting udev: 5441 [ OK ]
Setting hostname localhost: [ OK ]
Entering non-interactive startup
Starting ConnMgr: ar 2005, FAT32, LFN [ OK ]
Starting TimeAgent: [ OK ]
Starting ProcMgr: [ OK ]
Starting EOS initialization stage 1: [ 26.737867] NMI watchdog: failed to be e
nabled on some cpus-lab.swi
[ 6.987560] Starting new kernel [ OK ]
ip6tables: Applying firewall rules: [ OK ]
iptables: Applying firewall rules: [ OK ]
iptables: Loading additional modules: nf_conntrack_tftp [ OK ]
Starting system logger: [ OK ]
Starting NorCal initialization: _

```

Step 16 Log in to the switch.

Note: Default credentials on vEOS are `admin` and `admin` for username and password.

Step 17 Confirm the right number of NICs are present. We expect one Management nic and 7 front-panel nics, the maximum Arista vEOS can support.

```

localhost#show interface status
Port      Name      Status      Vlan      Duplex  Speed  Type      Flags
Et1       connected  1           full      unconf  EbraTestPhyPort
Et2       connected  1           full      unconf  EbraTestPhyPort
Et3       connected  1           full      unconf  EbraTestPhyPort
Et4       connected  1           full      unconf  EbraTestPhyPort
Et5       connected  1           full      unconf  EbraTestPhyPort
Et6       connected  1           full      unconf  EbraTestPhyPort
Et7       connected  1           full      unconf  EbraTestPhyPort
Ma1       connected  routed      a-full    a-1G    10/100/1000

localhost#
localhost#
localhost#
localhost#
localhost#
localhost#
localhost#
localhost#
localhost#
localhost#
localhost#
localhost#
localhost#
localhost#
localhost#
localhost#_

```

Install Arista Device System Agents via the Web interface (**Devices / System Agents / Agent**).

Create VMware Port-groups for AOS Virtual Labs

In order for virtual switches to be able to properly communicate with each other, each interface pair needs to share a dedicated port-group in VMware that carries all VLANs and allows MAC changes, Forged Transmits, and Promiscuous mode.

Download the AOS Cable map from the AOS Blueprint. An example CSV file is below.

cabling_diagram (2).csv - LibreOffice Calc

File Edit View Insert Format Sheet Data Tools Window Help

Liberation Sans 10

B45

| | A | B | C | D | E | F | G | H |
|----|---------------------------------|---------------------|-----------------|----------------------|-----------------|-----------------|----------------------|-----------------|
| 1 | Link ID | Role | Endpoint 1 name | Endpoint 1 interface | Endpoint 1 IP | Endpoint 2 name | Endpoint 2 interface | Endpoint 2 IP |
| 2 | spine2<->lab_2_leaf1 | Spine to leaf | spine2 | Ethernet5 | 10.255.0.12/31 | lab_2_leaf1 | Ethernet2 | 10.255.0.13/31 |
| 3 | spine2<->lab_2_leaf2 | Spine to leaf | spine2 | Ethernet6 | 10.255.0.14/31 | lab_2_leaf2 | Ethernet2 | 10.255.0.15/31 |
| 4 | spine1<->lab_2_leaf2 | Spine to leaf | spine1 | Ethernet6 | 10.255.0.6/31 | lab_2_leaf2 | Ethernet1 | 10.255.0.7/31 |
| 5 | spine2<->lab_1_leaf2 | Spine to leaf | spine2 | Ethernet4 | 10.255.0.10/31 | lab_1_leaf2 | Ethernet2 | 10.255.0.11/31 |
| 6 | spine1<->lab_1_leaf1 | Spine to leaf | spine1 | Ethernet3 | 10.255.0.0/31 | lab_1_leaf1 | Ethernet1 | 10.255.0.1/31 |
| 7 | spine2<->lab_1_leaf1 | Spine to leaf | spine2 | Ethernet3 | 10.255.0.8/31 | lab_1_leaf1 | Ethernet2 | 10.255.0.9/31 |
| 8 | spine1<->lab_2_leaf1 | Spine to leaf | spine1 | Ethernet5 | 10.255.0.4/31 | lab_2_leaf1 | Ethernet1 | 10.255.0.5/31 |
| 9 | spine1<->lab_1_leaf2 | Spine to leaf | spine1 | Ethernet4 | 10.255.0.2/31 | lab_1_leaf2 | Ethernet1 | 10.255.0.3/31 |
| 10 | lab_1_leaf_pair<->lab_1_server2 | leaf_pair l2_server | lab_1_leaf_pair | | | lab_1_server2 | | |
| 11 | lab_2_leaf_pair<->lab_2_server1 | leaf_pair l2_server | lab_2_leaf_pair | | | lab_2_server1 | | |
| 12 | lab_1_leaf_pair<->lab_1_server1 | leaf_pair l2_server | lab_1_leaf_pair | | | lab_1_server1 | | |
| 13 | lab_2_leaf_pair<->lab_2_server2 | leaf_pair l2_server | lab_2_leaf_pair | | | lab_2_server2 | | |
| 14 | lab_1_leaf2<->lab_1_server2 | Leaf to L2 server | lab_1_leaf2 | port-channel2 | | lab_1_server2 | | |
| 15 | lab_2_leaf2<->lab_2_server2 | Leaf to L2 server | lab_2_leaf2 | port-channel2 | | lab_2_server2 | | |
| 16 | lab_1_leaf1<->lab_1_server1[1] | Leaf to L2 server | lab_1_leaf1 | Ethernet5 | | lab_1_server1 | | |
| 17 | lab_2_leaf1<->lab_2_server2[1] | Leaf to L2 server | lab_2_leaf1 | Ethernet6 | | lab_2_server2 | | |
| 18 | lab_1_leaf2<->lab_1_server1[1] | Leaf to L2 server | lab_1_leaf2 | Ethernet5 | | lab_1_server1 | | |
| 19 | lab_1_leaf1<->lab_1_server1 | Leaf to L2 server | lab_1_leaf1 | port-channel1 | | lab_1_server1 | | |
| 20 | lab_2_leaf1<->lab_2_server1 | Leaf to L2 server | lab_2_leaf1 | port-channel1 | | lab_2_server1 | | |
| 21 | lab_1_leaf2<->lab_1_server2[1] | Leaf to L2 server | lab_1_leaf2 | Ethernet6 | | lab_1_server2 | | |
| 22 | lab_1_leaf1<->lab_1_server1 | Leaf to L2 server | lab_1_leaf1 | port-channel1 | | lab_1_server1 | | |
| 23 | lab_1_leaf1<->lab_1_server2[1] | Leaf to L2 server | lab_1_leaf1 | Ethernet6 | | lab_1_server2 | | |
| 24 | lab_2_leaf2<->lab_2_server2[1] | Leaf to L2 server | lab_2_leaf2 | Ethernet6 | | lab_2_server2 | | |
| 25 | lab_2_leaf2<->lab_2_server1[1] | Leaf to L2 server | lab_2_leaf2 | Ethernet5 | | lab_2_server1 | | |
| 26 | lab_1_leaf1<->lab_1_server2 | Leaf to L2 server | lab_1_leaf1 | port-channel2 | | lab_1_server2 | | |
| 27 | lab_2_leaf1<->lab_2_server2 | Leaf to L2 server | lab_2_leaf1 | port-channel2 | | lab_2_server2 | | |
| 28 | lab_2_leaf1<->lab_2_server1[1] | Leaf to L2 server | lab_2_leaf1 | Ethernet5 | | lab_2_server1 | | |
| 29 | lab_2_leaf2<->lab_2_server1 | Leaf to L2 server | lab_2_leaf2 | port-channel1 | | lab_2_server1 | | |
| 30 | lab_1_leaf1<->lab_1_leaf2 | Leaf to leaf | lab_1_leaf2 | port-channel3 | | lab_1_leaf1 | port-channel3 | |
| 31 | lab_2_leaf1<->lab_2_leaf2 | Leaf to leaf | lab_2_leaf2 | port-channel3 | | lab_2_leaf1 | port-channel3 | |
| 32 | lab_1_leaf1<->lab_1_leaf2[1] | Leaf to leaf | lab_1_leaf2 | Ethernet3 | | lab_1_leaf1 | Ethernet3 | |
| 33 | lab_1_leaf1<->lab_1_leaf2[2] | Leaf to leaf | lab_1_leaf2 | Ethernet4 | | lab_1_leaf1 | Ethernet4 | |
| 34 | lab_2_leaf1<->lab_2_leaf2[1] | Leaf to leaf | lab_2_leaf2 | Ethernet3 | | lab_2_leaf1 | Ethernet3 | |
| 35 | lab_2_leaf1<->lab_2_leaf2[2] | Leaf to leaf | lab_2_leaf2 | Ethernet4 | | lab_2_leaf1 | Ethernet4 | |
| 36 | spine2<->router2 | To external router | spine2 | Ethernet1 | 192.168.66.2/31 | router1 | | 192.168.66.3/31 |
| 37 | spine1<->router1 | To external router | spine1 | Ethernet1 | 192.168.66.0/31 | router1 | | 192.168.66.1/31 |
| 38 | | | | | | | | |

The left-hand column ‘A’ describes the network name that needs to be set up under ESXi attached to a promiscuous port-group.

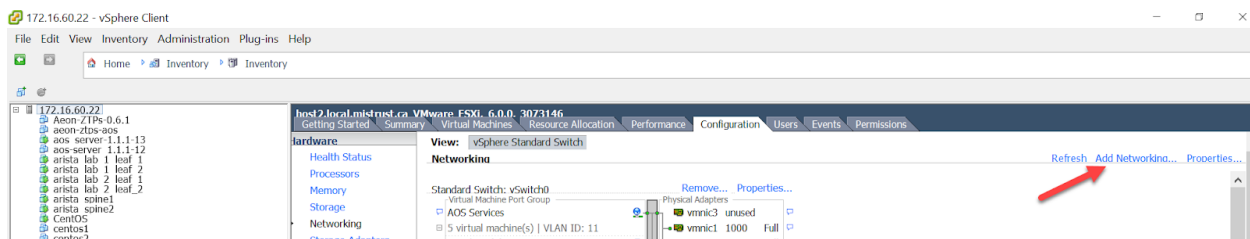
We need to connect the device on column ‘C’ to the device on column ‘F’ interface. For example: Vmware port-group: Spine1<->virtual_leaf_1_leaf2: Virtual port-group name: ‘spine1-virtual-leaf-1-leaf2’

This

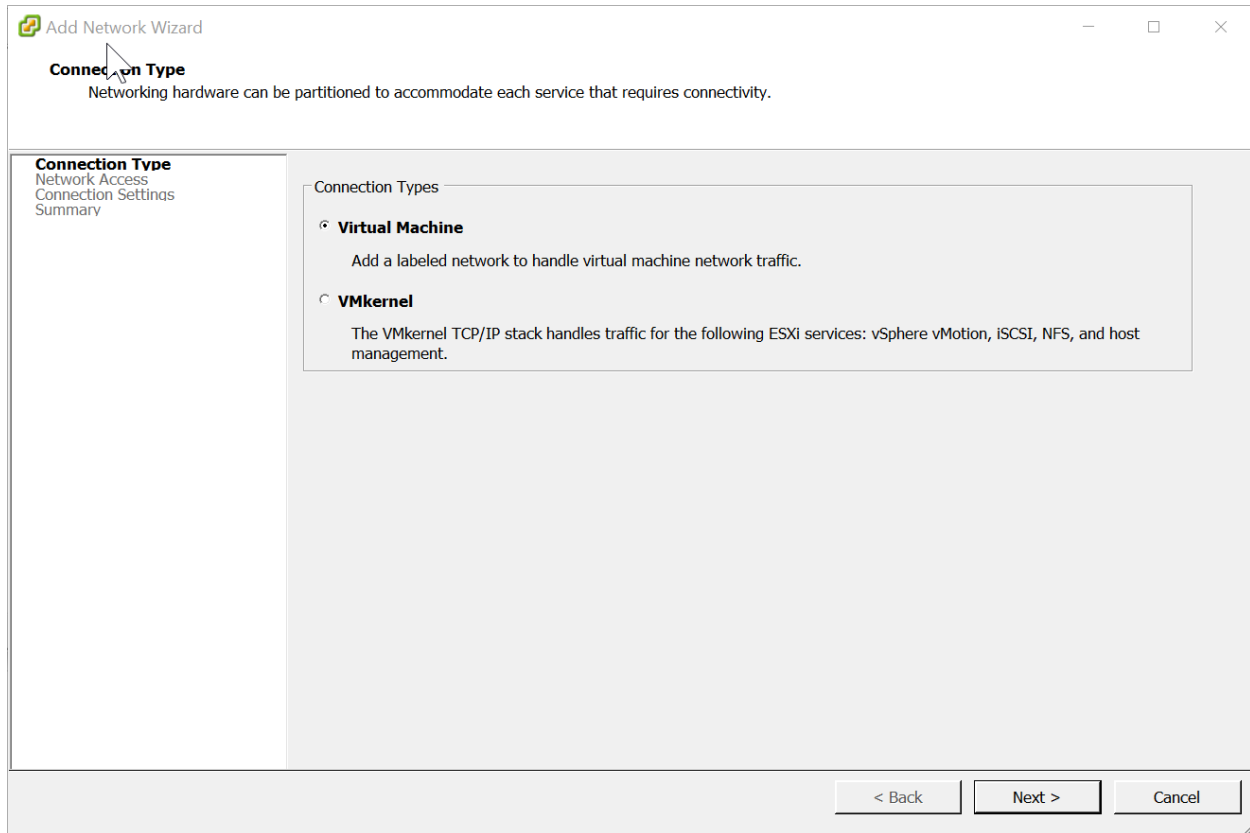
spine1 Ethernet3 (VMware Network Adapter #4) virtual_leaf_1_leaf2 Ethernet1 (VMWare network adapter #2)

The process to create a port-group is as follows.

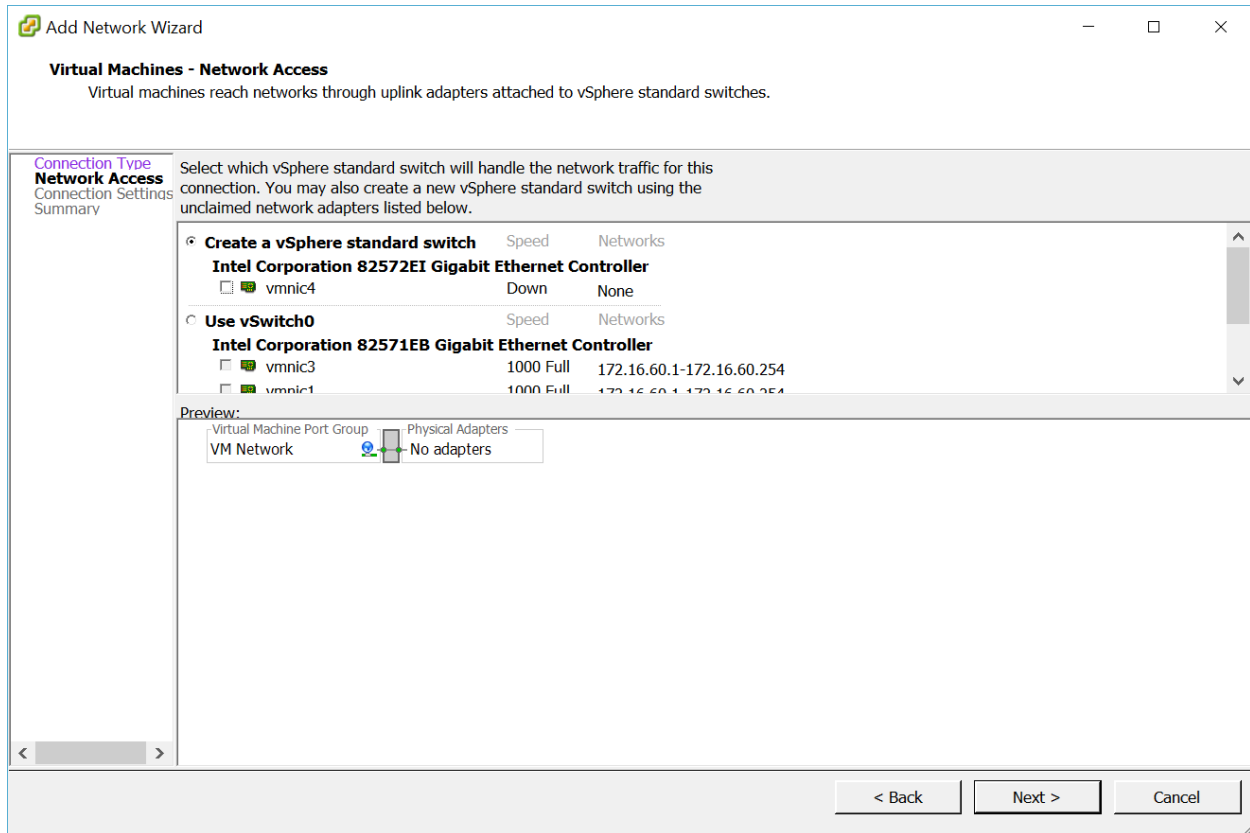
In the ESXi Console, go to the Host configuration tab, and click “Add networking”.



Choose “Virtual Machine” as the connection type.



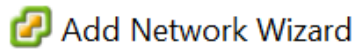
Select “Create a vSphere standard switch”.



Create a network label that helps represent the connection name. In this case in example:

‘Spine1_eth3-lab1-leaf1_eth1’

Make sure the VLAN ID is set to ‘4095’. This means “All vlans” and allows the switch interface to act promiscuously.



Add Network Wizard

Virtual Machines - Connection Settings

Use network labels to identify migration compatible connections common to two or n

Connection Type

Network Access

Connection Settings

Summary

Port Group Properties

Network Label: spine1_eth3-lab-1-leaf1_eth1

VLAN ID: 4095

Preview:

Virtual Machine Port Group

spine1_eth3-lab-1-leaf1_eth1

VLAN ID: All (4095)

Physical Adapters

No adapters

After the switch is created we have to assign the traffic policy. View the vSwitch under the UI and click 'properties'.

The screenshot displays the VMware ESXi configuration interface for host2.local.mistrust.ca, version 6.0.0, build 3073146. The 'Configuration' tab is active, showing the 'Networking' section. The left sidebar lists various configuration categories: Hardware (Health Status, Processors, Memory, Storage, Networking, Storage Adapters, Network Adapters, Advanced Settings, Power Management) and Software (Licensed Features, Time Configuration, DNS and Routing, Authentication Services, Virtual Machine Startup, Virtual Machine Swapfile, Security Profile, Host Cache Configuration). The main content area shows the 'View: vSphere Standard Switch' configuration. Under 'Networking', the 'VMkernel Port' section shows the 'Management Network' with IP address vmk0 : 172.16.60.22. Below this, two standard switches are listed: 'Standard Switch: vSwitch7' and 'Standard Switch: vSwitch1'. Each switch has a 'Virtual Machine Port Group' and a 'Physical Adapters' section. For vSwitch7, the port group is 'Unused' and the physical adapters section shows 'No adapters'. For vSwitch1, the port group is 'spine1_eth3-lab-1-leaf1_eth1' and the physical adapters section also shows 'No adapters'. A red arrow points to the 'Properties...' link for vSwitch1.

host2.local.mistrust.ca VMware ESXi 6.0.0 3073146

Getting Started Summary Virtual Machines Resource Allocation Performance Configuration

Hardware

- Health Status
- Processors
- Memory
- Storage
- Networking
- Storage Adapters
- Network Adapters
- Advanced Settings
- Power Management

Software

- Licensed Features
- Time Configuration
- DNS and Routing
- Authentication Services
- Virtual Machine Startup
- Virtual Machine Swapfile
- Security Profile
- Host Cache Configuration

View: vSphere Standard Switch

Networking

VMkernel Port

- Management Network
- vmk0 : 172.16.60.22

Standard Switch: vSwitch7 Remove... Properties...

Virtual Machine Port Group

- Unused
- 6 virtual machine(s)
- arista_spine1
- arista_spine2
- arista_lab_1_leaf_1
- arista_lab_1_leaf_2
- arista_lab_2_leaf_2
- arista_lab_2_leaf_1

Physical Adapters

No adapters

Standard Switch: vSwitch1 Remove... Properties...

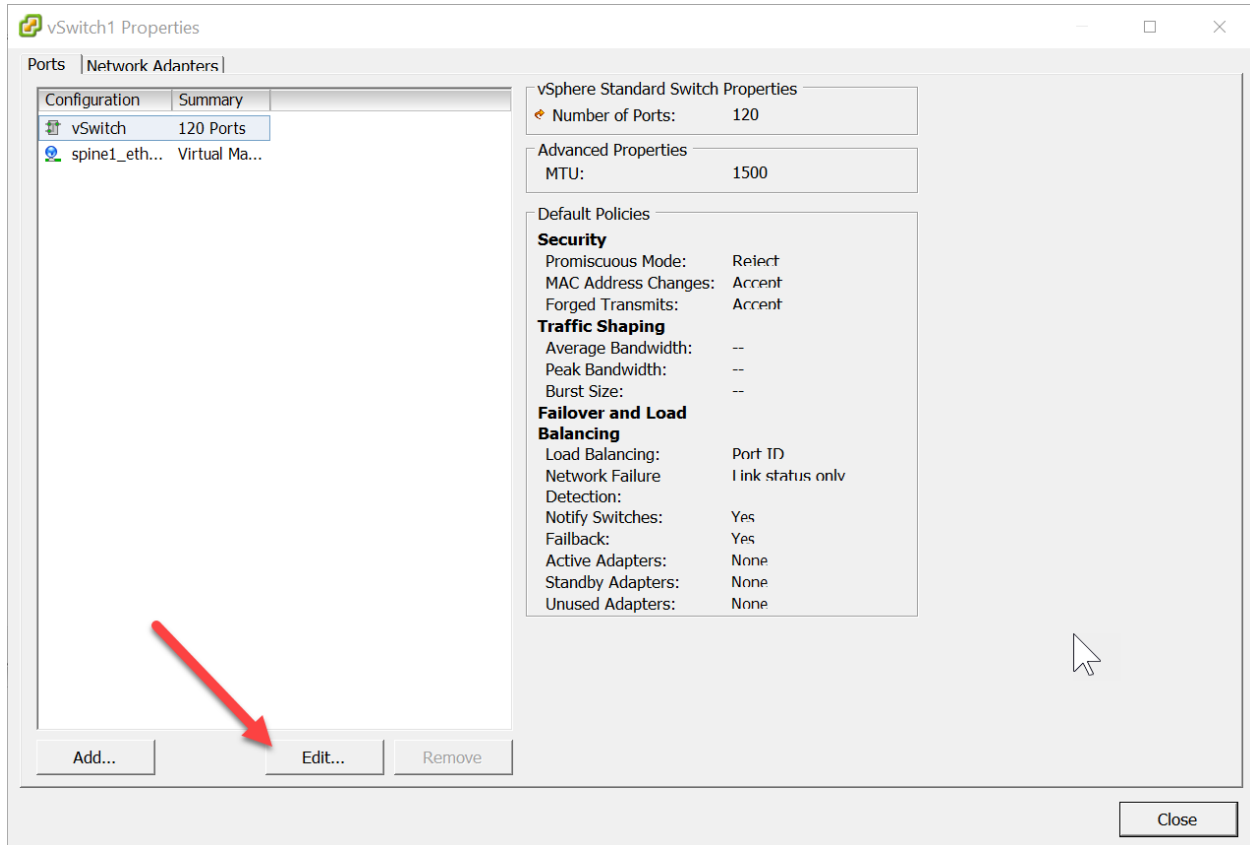
Virtual Machine Port Group

- spine1_eth3-lab-1-leaf1_eth1
- VLAN ID: All (4095)

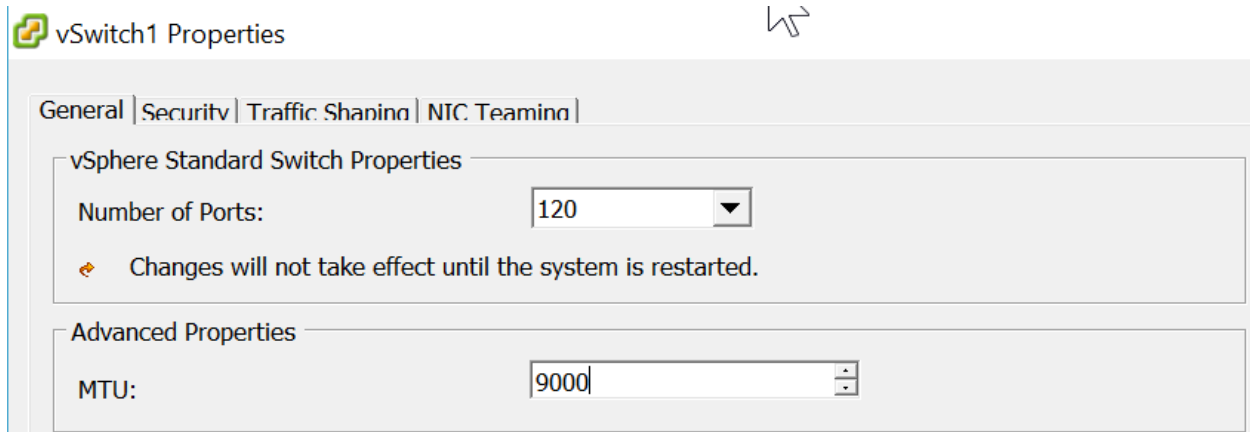
Physical Adapters

No adapters

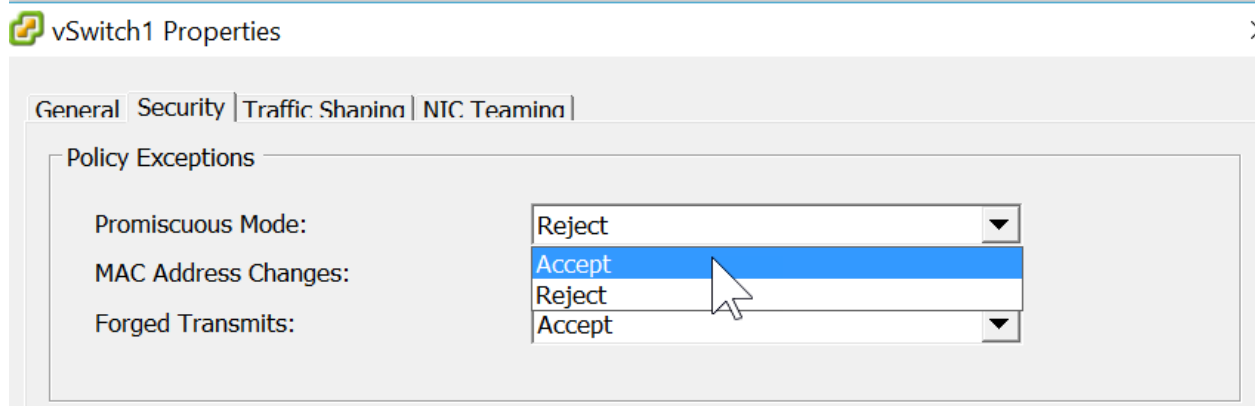
Followed by 'Edit'.



On the General tab, set the MTU to 9000 from the default 1500. We also recommend changing the number of ports from the default 120 to 8 to prevent resource exhaustion with many networks.

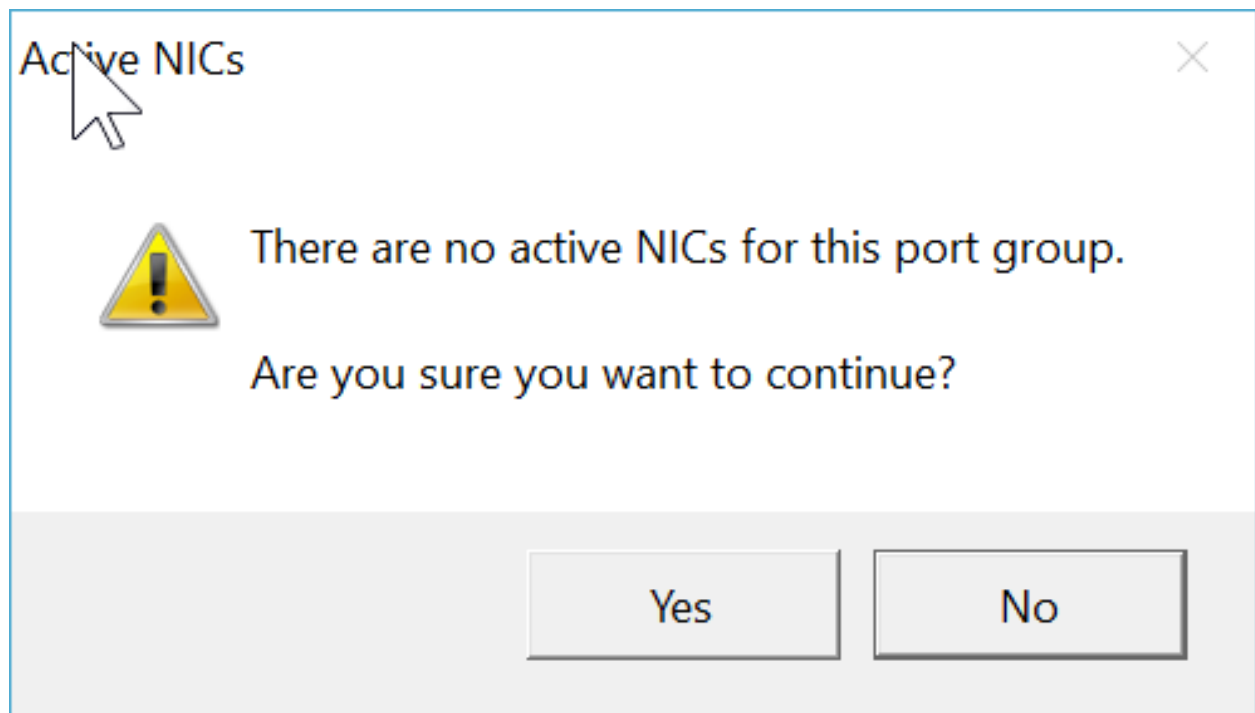


Under the Security tab, change Promiscuous Mode from Reject to Accept.



All of the values should say Accept.

If you receive this error, please ignore it - we are not connecting these virtual switches anywhere to the real world.



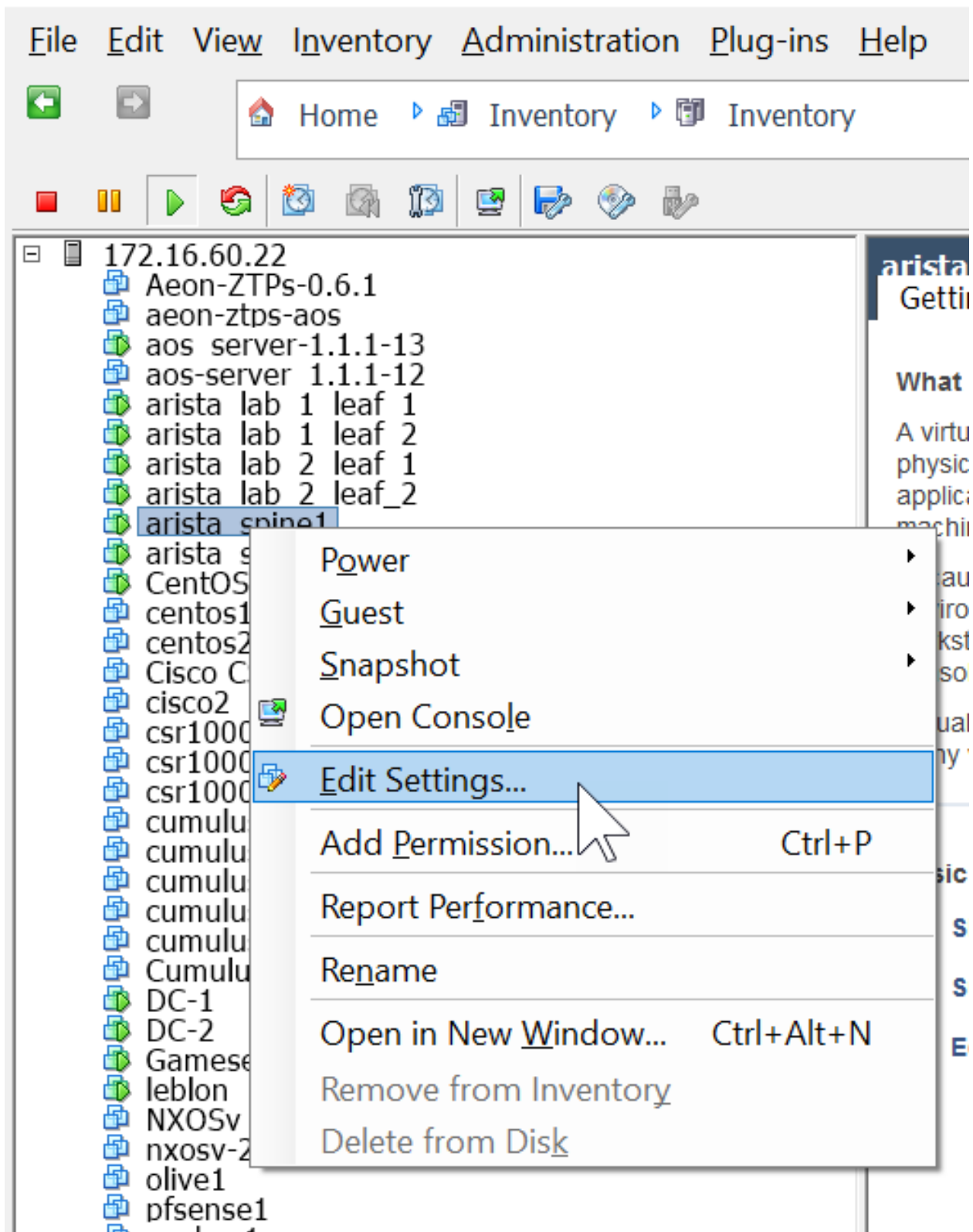
After the switch is created the properties should appear like this:

| | |
|------------------------------------|------------------|
| vSphere Standard Switch Properties | |
| Number of Ports: | 120 |
| Advanced Properties | |
| MTU: | 9000 |
| Default Policies | |
| Security | |
| Promiscuous Mode: | Accept |
| MAC Address Changes: | Accept |
| Forged Transmits: | Accept |
| Traffic Shaping | |
| Average Bandwidth: | -- |
| Peak Bandwidth: | -- |
| Burst Size: | -- |
| Failover and Load Balancing | |
| Load Balancing: | Port ID |
| Network Failure Detection: | Link status only |
| Notify Switches: | Yes |
| Failback: | Yes |
| Active Adapters: | None |
| Standby Adapters: | None |
| Unused Adapters: | None |

After the Port-group is created, we need to assign it to the VM.

Find your VM (spine1), and edit settings.

172.16.60.22 - vSphere Client


















Each network adapter is mapped out starting from Management1 - Note this allocation format.

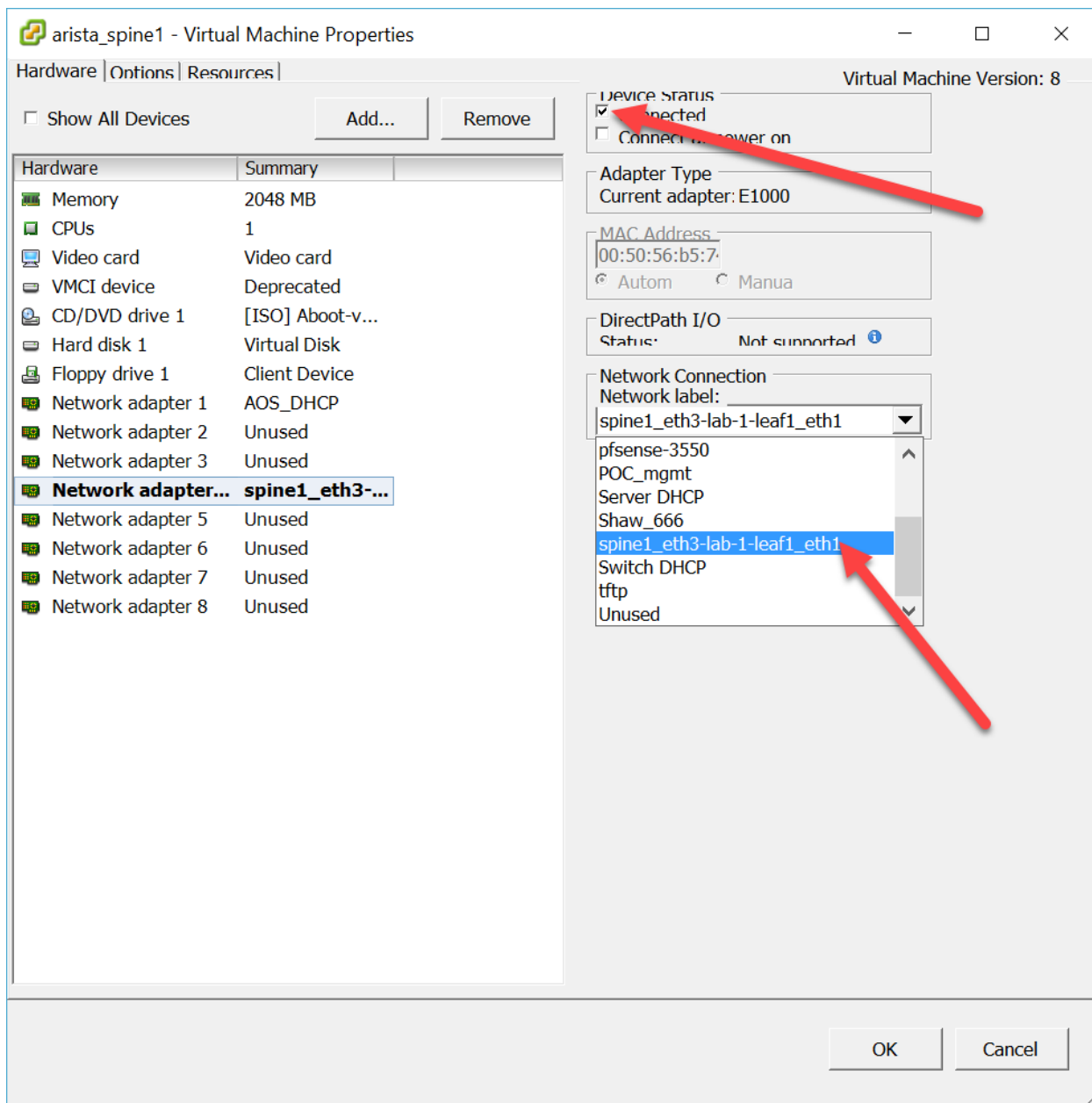
arista_spine1 - Virtual Machine Properties

Hardware | Options | Resources

☐ Show All Devices Add... Remove

| Hardware | Summary |
|---|---------------------|
|  Memory | 2048 MB |
|  CPUs | 1 |
|  Video card | Video card |
|  VMCI device | Deprecated |
|  CD/DVD drive 1 | [ISO] About-v... |
|  Hard disk 1 | Virtual Disk |
|  Floppy drive 1 | Client Device |
|  Network adapter 1 | AOS_DHCP Management |
|  Network adapter 2 | Unused Ethernet1 |
|  Network adapter 3 | Unused Ethernet2 |
|  Network adapter 4 | Unused Ethernet3 |
|  Network adapter 5 | Unused Ethernet4 |
|  Network adapter 6 | Unused Ethernet5 |
|  Network adapter 7 | Unused Ethernet6 |
|  Network adapter 8 | Unused Ethernet7 |

In this example, we want to connect Eth3 to lab-1-leaf1 Eth1. Set the NIC to connected, and change the port group.



That's a lot of steps.

A short example script can be used to create the port-groups under the ESX Console. This will create the port-groups quickly, but you will have to still assign each VM's network adapter in the GUI.

```
esxcli network vswitch standard add --vswitch-name=$NIC_NAME$ --ports=8
esxcli network vswitch standard set --vswitch-name=$NIC_NAME$ --cdp-status=down --
↳mtu=9000
esxcli network vswitch standard policy security set --vswitch-name=$NIC_NAME$ --allow-
↳forged-transmits true --allow-mac-change true --allow-promiscuous true
esxcli network vswitch standard portgroup add --vswitch-name=$NIC_NAME$ --portgroup-
↳name=$NIC_NAME$
esxcli network vswitch standard portgroup set --portgroup-name=$NIC_NAME$ -vlan-
↳id=4095
```


12.4.7 Juniper EVPN Support

12.4.7.1 Overview

New in version 3.3.0: 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**.

The Junos EVPN ESI multi-homing feature (new in version 3.3.0) enables you to directly connect end servers to leafs and provide redundant connectivity via multi-homing. This feature is supported only on LAGs that span two leafs on the fabric. EVPN ESI also removes the need for “peer-link”, and hence facilitates clean leaf-spine design.

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

12.4.7.2 Understanding EVPN multi-homing Terminology and Concepts

Please find below EVPN multi-homing terminology and concepts:

EVI - EVPN instance that spans between the leaf devices making up the EVPN. The EVPN instance (EVI) is 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 just called a “MAC table”). A unique route distinguisher and VRF target is configured per MAC-VRF.

Ethernet Segment (ES) - Ethernet links span from an end host to multiple ToR leafs and form Ethernet Segment. It constitutes a set of bundled links.

Ethernet Segment Identifier (ESI) - Represents each ES uniquely across the network. ESI is only supported on LAGs that span two leafs on the fabric.

ESI helps with end host level redundancy in an EVPN VXLAN-based blueprint. Ethernet links from each Juniper ToR leaf connected to the server are bundled as an aggregated Ethernet interface. LACP is enabled for each aggregated Ethernet interface of the Juniper devices. Multi-homed interfaces into the Ethernet segment are identified using the Ethernet Segment Identifier (ESI).

ESI has certain restrictions and requirements as listed below:

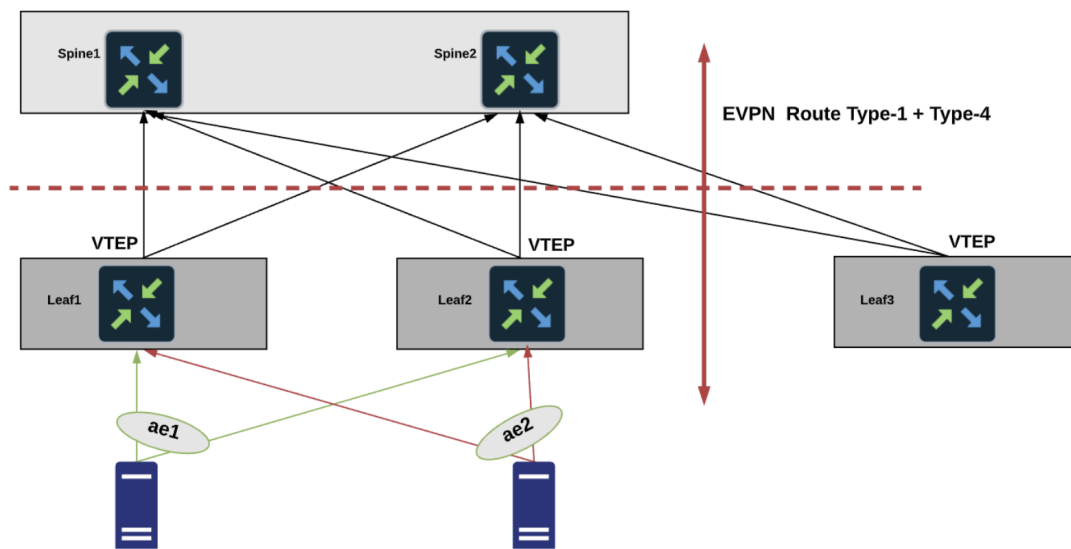
- ESI based ToR leafs cannot have any L2/L3 peer links as EVPN multi-homing eliminates peer links used by MLAG/vPC.
- A bond of two physical interfaces towards a single leaf is not supported in the ESI implementation (version 3.3.0), so make sure the server having LAG in that rack type spans two leafs.
- 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 i.e having different VLAN sets among individual leafs 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 (i.e 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.
- More than two leafs in one ESI segment using ESI based rack types is not supported.
- Switching from an ESI to MLAG rack type or vice versa is currently not supported under Flexible Fabric Expansion (FFE) operations.

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.

Topology Specification

In the example below Leaf1 and Leaf2 are part of the same Ethernet Segment (ES), and Leaf3 is the switch sending traffic towards the Ethernet Segment (ES).



Five different route types are used for Juniper EVPN multi-homing:

- Ethernet Auto-Discovery (EAD) Route (Type-1)
- MAC advertisement Route (Type-2)
- Inclusive Multicast Route (Type-3)
- Ethernet Segment Route (Type-4)
- IP Prefix Route (Type-5)

BGP EVPN running on Juniper devices use route type-2 to advertise MAC and IP (host) information, route type-3 to carry VTEP information and the EVPN route type-5 allows advertisements of IP prefixes in an Network Layer Reachability Information (NLRI).

Note: In Junos MAC/IP route-type 2 doesn't contain VNI and RT for the IP part of the route, it is derived from the accompanying route-type 5.

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 Route type 1 functionality to learn the EVPN Type 2 MAC routes from

other leaf devices. In this route type Ethernet Segment Identifier (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 (i.e Type-1) per Ethernet Segment. 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.

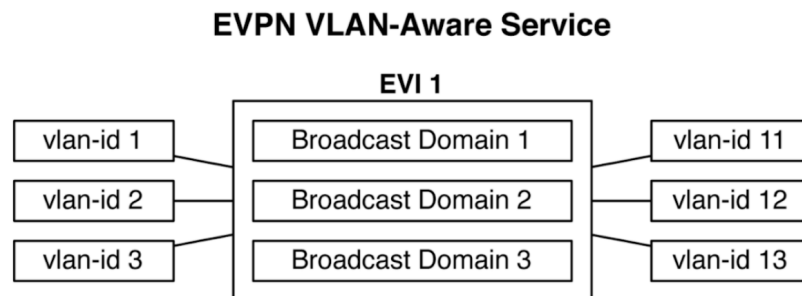
DF Election - Used to prevent forwarding of the loops and the duplicates as only a single switch is allowed to decapsulate and forward the traffic for a given Ethernet Segment. Ethernet Segment Route is exported and imported when ESI is locally configured under the LAG. Type-4 NLRI is mainly used for 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.

12.4.7.3 EVPN Services

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 Virtual Network Identifier (VNI) that is used during packet lookup to place it onto the right Bridge Domain(BD)/VLAN.

Creating an EVPN Network

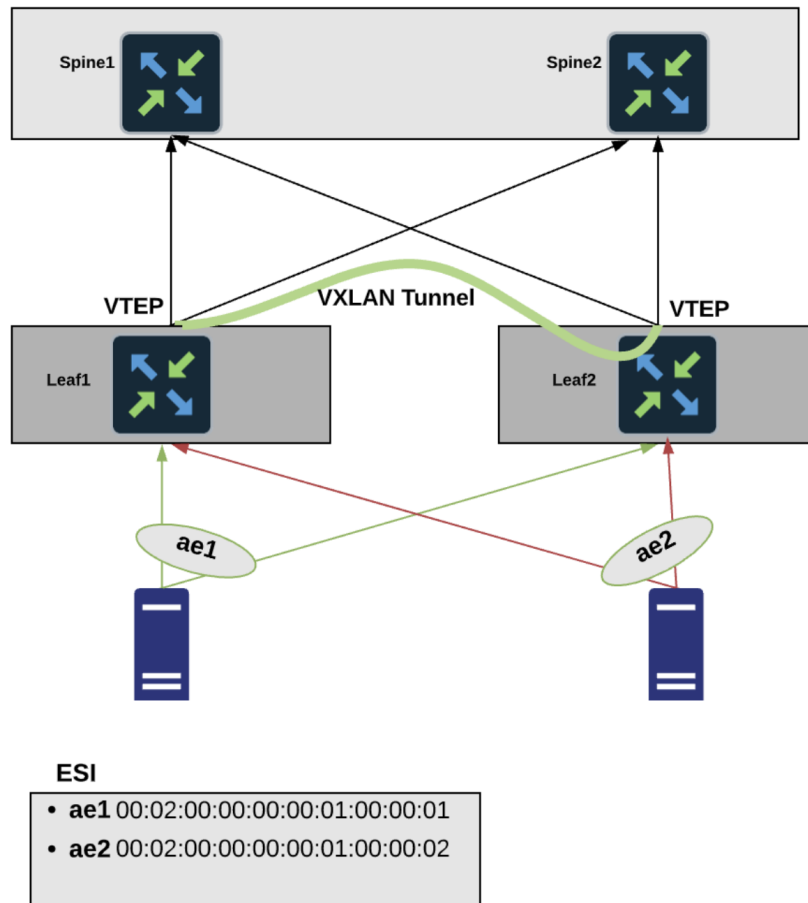
Creating an EVPN network follows the same workflow as for other networks.

1. Create/Install off-box *device agents* for all switches. Junos switches can only use off-box agents. (On-box agents are not supported.)
2. Confirm that the global catalog includes *logical devices* (Design > Logical Devices) that meet Juniper device requirements; create them if necessary:
3. Confirm that the global catalog includes *interface maps* (Design > Interface Maps) that map the logical devices to the correct *device profiles* for the Juniper devices; create them if necessary.
4. Create a *rack type*.
 - For single leaf racks, specify redundancy protocol **None** in the **Leaf** section.
 - For dual leaf racks
 - Specify redundancy protocol **ESI** in the **Leaf** section.
 - When specifying the end server in the **Server** section, specify attachment type as **Dual-Homed** towards ESI-based ToR leafs. EVPNs using ESs have a link aggregation option. Select the LAG mode **LACP (Active)**
5. Create a *rack-based template*.
6. Create *external router* and pools for *ASNs*, *IP addresses*, and *VNIs*.
7. Create a *blueprint* based on the ESI-based template, then *build* the EVPN-based network topology for the Juniper devices by assigning resources, device profiles, device IDs, and external routers.

12.4.7.4 Configuration Rendering

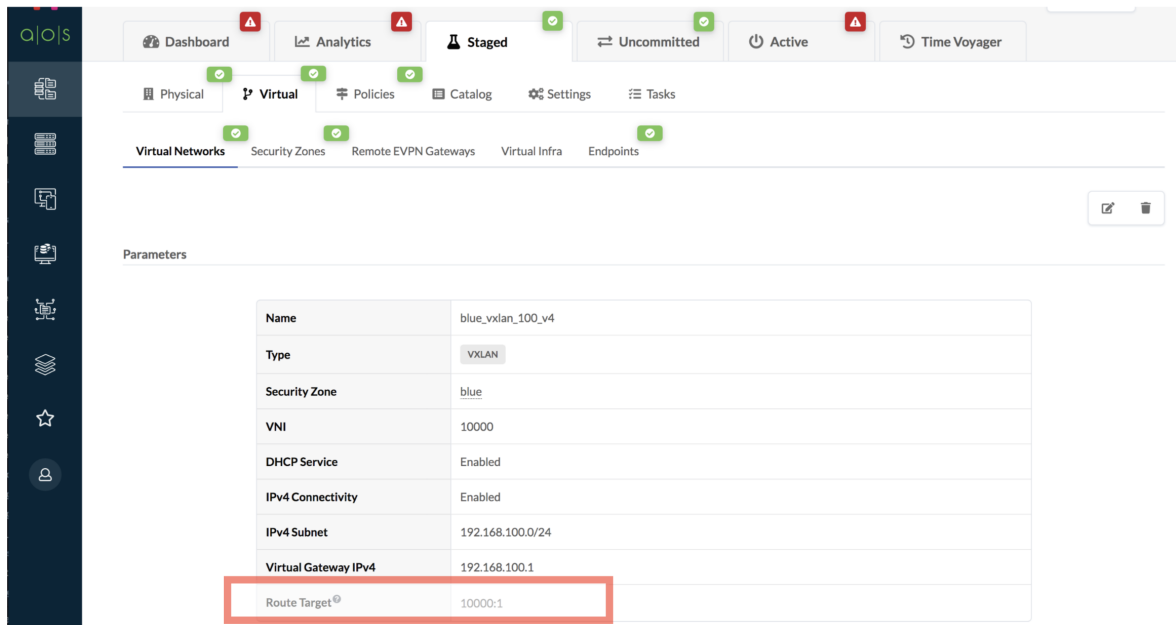
Reference Design

1. **Underlay** - The underlay in the data center fabric is layer-3 configured using standard eBGP over the physical interfaces of Juniper devices.
2. **Overlay** - Overlay will be configured eBGP over 100.0 address. EVPN VXLAN is used as an overlay protocol. All the ToR devices will be enabled with L2 VN. Each one of these L2 VNs can have its default gateway to be hosted on connected ToR Leaf. For the inter-VN traffic VXLAN routing will be done in the fabric using L3 VNIs on the Border Leaf as per standard design.
3. **VXLAN VTEPs** - On Juniper leafs one ip address on 100.0 is rendered which is used as VTEP address. The VTEP IP address is used to establish the VXLAN tunnel.
4. **EVPN multi-homing LAG** - **Unique ESI value** and **LACP system IDs** are used per EVPN LAG. The multi-homed links are configured with an ESI and a LACP system identifier is specified for each link. The ESI is used to identify LAG groups and loop prevention. To support Active/Active and multi-homing for Juniper Leafs, they are configured with the same LACP parameter for a given ESI so that they appear as a single system.



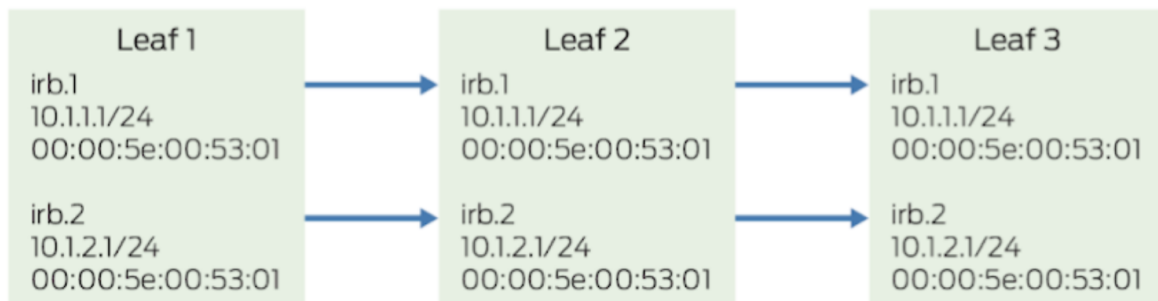
ESI MAC addresses are auto-generated internally. You can *configure the value of the most significant byte* used in the generated MAC. A new facade API is added to update the MSB value. A new node will be added to the rack based template that will contain the MAC MSB value. The default value of this 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 should be unique across multiple blueprints (IP Fabrics).

5. **L3VNIs** - L3VNI is rendered as a security zone per VRF. Multi-tenancy functionality is available to ensure that workloads remain logically separated within a virtual network (overlay) construct using security zone.
6. **Route Target(RT) for L2/L3 VNIs** - Auto-generated for L2/L3 VNIs in the format VNI:1. There is 1 RT per MAC-VRF(i.e L3VNI), this is fabric wide RT, value must be the same across all switches participating in one EVI. Please find the example as below:



7. **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 I2 VNI. Route Distinguisher(RD) exist only for Security Zone VRF in the format {primary_loopback}:vlan_id.
8. **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 (e.g. lo0.0) is provided by the underlay. The route-distinguisher(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 route-target(RT) here defines the global route target inherited by EVPN routes. It is used by Type 1 routes. A default RT value is rendered for it i.e 100:100 for global switch options across all switches.
9. **MTU** - The MTU values that are rendered for Juniper Devices:
 - L2 ports: 9100
 - L3 ports: 9216
 - Integrated Routing and Bridging(IRB) Interfaces: 9000
10. **Anycast Gateway** - The same IP on Integrated Routing and Bridging(IRB) interfaces of all the leafs is configured and no virtual gateway is set. Every IRB interface that participates in the stretched L2 service will have 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.

12.4.7.5 Limitations

The following limitations apply to EVPN multi-homing topologies for Juniper devices in version 3.3.0:

- Only two-way multi-homing is supported. More than two Juniper leafs in a multi-homed group is not supported.
- Juniper EVPN with EVPN on other network vendors in the same blueprint is not supported.
- No Static VXLAN support.
- IPv6-based fabrics do not support Junos.
- Segmentation and Group Based Policies (GBP) are not supported.
- In Juniper EVPN multi-homing, L3 External Connectivity Points (ECP) towards external routers are supported; L2 ECP is not supported.
- BGP routing from Junos leafs to AOS-managed Layer 3 servers is not supported.

12.4.8 Mixed Uplink Speeds between Leafs and Spines

New in version 3.2.1.

As of AOS 3.2.1, heterogeneous speeds between Spine and Leaf planes are allowed. This applies to the following:

- AOS Rack Based Templates
- Rack Change Operations in a Blueprint

Important:

- It is not possible to have mixed speed on parallel links between the same devices
 - As a prerequisite, the Spine Logical Device should have mixed port speeds defined, specifying the port role as Leaf for the required number of ports. This should be as per the **day 0** design requirements while keeping in mind how many rack need to be added, so that sufficient ports for Spine to Leaf connections with mixed link speed are available.
-

Warning: AOS does not support updating of Logical Device of a Spine already used in a Blueprint. For this manual patching using AOS CLI will be required. AOSCLI is an experimental tool and may not be able to provide a solution. Please contact [Apstra Global Support](#) for further assistance.

This allows for the following operations:

1. Create a Rack Based template with Rack Types having different link speeds to Spines.
 - For example, create a L2 Rack Type with Logical Device AOS-7x10-Leaf and AOS-40x10+6x40 for two Leaf switches, having respectively 10 GbE and 40 GbE as uplinks towards the Spines.

Edit Rack Type

Topology

Summary

Leaves

Servers

Name *

10gig

Logical device *

AOS-7x10-Leaf

2 x 10 Gbps
Spine

2 x 10 Gbps
Peer

2 x 10 Gbps
L2 Server • L3 Server

1 x 10 Gbps
External Router

Links per spine *

1

✕

10 Gbps

MLAG pair

External facing

Name *

40gig

Logical device *

AOS-40x10+6x40-1

40 x 10 Gbps
L2 Server • L3 Server • External Router • Peer

6 x 40 Gbps
Spine • L2 Server • L3 Server • External Router

Links per spine *

1

✕

40 Gbps

MLAG pair

External facing

- Then design a Rack Based template based on the mixed line speed Rack Type as above.

Edit Template

Common Parameters

Name *
L2 Pod_mixed

Type *
☒ RACK BASED

Policies

ASN Allocation Scheme (spine)
☐ Unique ☒ Single

Routing Policy (import)
☐ Default Only ☒ ALL

Overlay Control Protocol
☐ Static VXLAN ☒ MP-EBGP EVPN

Spine to Leaf Links Type
☒ IPv4 ☐ IPv6 ☐ IPv4-IPv6

Structure

Rack Types

mixed_rack (2x10 Gbps links to spines) X 1

Spines

Spine Logical Device *
AOS-48x10-4x40 X

Count *
1

Preview

Topology Racks Spine Logical Device

Rack Based Template using mixed line speed Rack Type

2. The user can create a “POD Based” Template with such Rack Based template as per screenshot below:

Edit Template

Name *
Pod_mixed

Type *
☒ POD BASED

Policies

Spine to Superspine Links
☒ IPv4 ☐ IPv6 ☐ IPv4-IPv6

Overlay Control Protocol (override)
☐ Static VXLAN ☒ MP-EBGP EVPN

Structure

Pods

L2 Pod_mixed X 1

Superspines

Superspine Logical Device *
AOS-7x10-Leaf X

Ports Summary

| Spine | Peer | L2 Server + L3 Server | External Router |
|-------------|-------------|-----------------------|-----------------|
| 2 x 10 Gbps | 2 x 10 Gbps | 2 x 10 Gbps | 1 x 10 Gbps |

Plane Count *
1

Per Plane Count *
1

Preview

Topology Pods Superspine Logical Device

POD template consisting of mixed speed rack-based Template.

3. AOS allows adding a rack with different uplink speed to Spines in an existing Blueprint.

- In the below example, the user can verify Logical Device AOS-48x10-4x40 for Spine switch having port role Leaf and mixed link speed (i.e 10 GbE and 40 GbE) ports being used in the Rack Type.

- The user is able to add more racks with different type link speeds between Leaf-spine as below:

12.5 AOS-CLI

Apstra provides a command line interface tool (AOS-CLI), which augments functionality provided by the Apstra web interface.

Important: AOS-CLI is considered **experimental** and use of it must be under strict supervision of *Technical Support*. Do **not** use AOS-CLI in production networks.

12.5.1 Installing AOS-CLI

AOS-CLI is distributed as a Docker container. The AOS-CLI Docker container can be used on any system running a compatible version of Docker. The steps below show how to install and run the AOS-CLI Docker container on an Apstra server.

1. Make sure you have Docker 17.12.0-ce installed on the Apstra server VM.
2. Download the [AOS-CLI Docker Container image](#).
3. Copy the AOS-CLI Docker container tar.gz file to the Apstra server.
4. Load the provided Docker image with the `docker image load` command. For example:

```
admin@aos-server:~$ ls -l
total 128680
-rw----- 1 admin admin 131766653 Mar 26 23:14 aoscli-0.1.617.tar.gz
admin@aos-server:~$ docker image load -i aoscli-0.1.617.tar.gz
beee9f30bc1f: Loading layer [=====>] 5.862MB/5.862MB
3fc750b41be7: Loading layer [=====>] 821.8kB/821.8kB
20a7b70bdf2f: Loading layer [=====>] 59.53MB/59.53MB
879c9d8666e3: Loading layer [=====>] 6.749MB/6.749MB
11d9535bbed0: Loading layer [=====>] 3.097MB/3.097MB
79608068b59b: Loading layer [=====>] 410.7MB/410.7MB
Loaded image: aoscli:0.1.617
admin@aos-server:~$
```

12.5.2 Accessing AOS-CLI

1. Start the AOS-CLI Docker container with the `docker run` command. For example:

```
admin@aos-server:~$ docker run --rm -ti -v $HOME:/mytmp aoscli:0.1.617 -s 192.168.59.3
Password [admin]:
-----
NOTE: This is a Limited Availability tool
Use it ONLY under the strict supervision of Apstra personnel
-----
Welcome to AOS CLI! Press TAB for suggestions
AOS CLI version: 0.1.617
AOS Server URL: https://192.168.59.3:443, Version 3.3.0.2-46
aos>
```

Replace “aoscli:0.1.617” with the AOS-CLI “Loaded image” version and replace “192.168.59.3” with the IP address of the Apstra server.

2. AOS-CLI comes with a built-in feature that auto-completes commands. Use the TAB key to learn about this tool and its functionality.

```
$ docker run -ti aoscli:0.1.0 aoscli -s 172.20.196.3
Password [admin]:
```

```
-----
*****
***** WARNING: This tool is experimental and should not be used in production *****
*****
-----
```

```
Welcome to AOS CLI! Press TAB for suggestions
AOS Server URL: https://172.20.196.3:443
aos> system
```

| | |
|-------------------|---|
| blueprint | Blueprint management commands |
| cabling | Cabling related commands |
| system | System management commands |
| user | User management commands |
| aaa-providers | AAA provider management commands |
| aaa-role-mappings | AAA role mapping management commands |
| vn | Virtual network management commands |
| vxlان | Vxlan management commands |
| hcl | Hardware Compatibility List management commands |
| package | Package management commands |
| proxy-agents | Proxy agent management commands |
| content | Content management commands |
| streaming | Streaming related commands |
| probe | IBA probe management commands |
| exit | Exit the AOS CLI Shell |

REFERENCE

13.1 AOS Feature Matrix

Table 1: Fabric Connectivity

| AOS Feature | EOS | NX-OS | Cumulus | Junos | SONiC |
|---------------------------------|-----|-------|---------|---------|-------|
| L3 Clos | Yes | Yes | Yes | Yes | Yes |
| 5-stage Clos | Yes | Yes | Yes | Yes | Yes |
| IPv6 Fabric RFC-5549 (non-EVPN) | Yes | Yes | Yes | Roadmap | Yes |

Table 2: Device Management

| AOS Feature | EOS | NX-OS | Cumulus | Junos | SONiC |
|-------------------------------------|-----|-----------------|-----------------|--------------|---------|
| On-box agent | Yes | Yes | Yes | Not possible | Yes |
| Off-box agent | Yes | Contact Support | Contact Support | Yes | Roadmap |
| Telemetry extensibility | Yes | Yes | Yes | Yes | Yes |
| Apstra ZTP | Yes | Yes | Yes | Yes | Yes |
| Device OS upgrade | Yes | Yes | Yes | Roadmap | Yes |
| Traffic draining (maintenance mode) | Yes | Yes | Yes | Yes | Yes |

Table 3: Access

| AOS Feature | EOS | NX-OS | Cumulus | Junos | SONiC |
|---|---------|---------|--------------|---------|--------------|
| Access layer of switches | Roadmap | Roadmap | Roadmap | Limited | Roadmap |
| LAG | Yes | Yes | Yes | Yes | Yes |
| MLAG/vPC | Yes | Yes | Yes | Roadmap | Yes |
| EVPN ESI (with LACP) | Roadmap | Roadmap | Roadmap | Yes | Not possible |
| 802.1x | Yes | Roadmap | Roadmap | Yes | Not possible |
| VLANs | Yes | Yes | Yes | Yes | Yes |
| Static VXLAN | Yes | Yes | Not possible | Roadmap | Not possible |
| EVPN VXLAN (3-stage and 5-stage) | Yes | Yes | Yes | Yes | Yes |
| BGP to AOS-managed server | Yes | Yes | Yes | Roadmap | Roadmap |
| IPv4 DHCP relay | Yes | Yes | Yes | Yes | Yes |
| IPv6 DHCP relay | Yes | Yes | Yes | Roadmap | Yes |
| EVPN DCI | Yes | Yes | Yes | Yes | Yes |
| IPv6 for applications (with EVPN and IPv4 fabric) | Yes | Yes | Yes | Roadmap | Yes |
| Group-based Policy (ACL on ToRs) | Yes | Yes | Roadmap | Roadmap | Roadmap |

Table 4: External Router Connectivity

| AOS Feature | EOS | NX-OS | Cumulus | Junos | SONiC |
|--|-----|-------|---------|---------|---------|
| BGP loopback peering (type:L2/L3) | Yes | Yes | Yes | Limited | Limited |
| BGP interface peering (type:L2/L3) | Yes | Yes | Yes | Limited | Limited |
| Address family: IPv4, IPv6 or Dual Stack | Yes | Yes | Yes | Limited | Yes |
| OSPF peering | Yes | Yes | Yes | Roadmap | Roadmap |

Table 5: Routing Policies

| AOS Feature | EOS | NX-OS | Cumulus | Junos | SONiC |
|---|-----|-------|---------|-------|-------|
| 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 |
| Custom import/export Policies | Yes | Yes | Yes | Yes | Yes |

Table 6: Miscellaneous

| AOS Feature | EOS | NX-OS | Cumulus | Junos | SONiC |
|---|-----|-------|---------|---------|-------|
| Configlets | Yes | Yes | Yes | Limited | Yes |
| FFE: add racks/add links/change speed, etc. | Yes | Yes | Yes | Yes | Yes |
| Mixed leaf/spine link speed | Yes | Yes | Yes | Yes | Yes |

Table 7: L2 Managed Server

| AOS Feature | L2 Managed Server |
|--|--|
| Supported OS Versions | Ubuntu 16.04.3, CentOS 7.5, Ubuntu 18.04 |
| Built-in Telemetry (no validation): LLDP, interface, hostname, memory, CPU | Yes |
| Extensible Telemetry | Yes |

13.2 Device and NOS Support

Supported devices and recommended Device Network Operating System (NOS) versions are listed below for different AOS versions.

13.2.1 Juniper Junos

13.2.1.1 Supported Juniper Devices

QFX10002, QFX5210, QFX5200, QFX5120, QFX5110, and QFX5100 Series

Important: Juniper 5100 models should not be used as border leaf devices due to the lack of RIOT support (Trident2 ASIC).

13.2.1.2 Recommended Junos NOS Versions

Table 8: Juniper Junos Recommended Version(s)

| AOS Version | Junos Version(s) |
|-----------------------------------|--|
| 3.3.0.2 | 20.2R2-S1.3 (QFX5110, QFX5120, QFX5200, QFX5210, QFX10002, QFX-10008, EX4300-48MP, EX4650-48Y), 18.4R2 (QFX5100, QFX5110, QFX5120-48Y, QFX10002-36Q/72Q) |
| 3.3.0 / 3.3.0.1 / 3.3.0a / 3.3.0c | 18.4R2 (QFX5100, QFX5110, QFX5120-48Y, QFX10002-36Q/72Q) |

13.2.2 Arista EOS

13.2.2.1 Supported Arista Devices

DCS-7000 Series

13.2.2.2 Recommended EOS NOS Versions

Table 9: Arista EOS Recommended Version(s)

| AOS Version | EOS Version(s) |
|-------------------------|------------------------------|
| 3.3.0.1 / 3.3.0.2 | 4.24.3M, 4.22.3M, 4.21.9M |
| 3.3.0 / 3.3.0a / 3.3.0c | 4.22.3M, 4.21.9M |
| 3.2.4 | 4.23.4.2M, 4.22.3M, 4.21.9M |
| 3.2.3 | 4.22.3M, 4.21.9M, 4.20.11M |
| 3.2.2 | 4.22.3M, 4.21.9M, 4.20.11M |
| 3.2.1 | 4.22.3M, 4.21.9M, 4.20.11M |
| 3.2.0 | 4.22.3M, 4.21.5.1F, 4.20.11M |
| 3.1.1 | 4.21.5.1F, 4.20.11M |
| 3.1.0 | 4.21.5.1F, 4.20.11M |
| 3.0.1 | 4.20.11M |
| 3.0.0 | 4.20.11M |
| 2.3.1 | 4.20.11M, 4.18.4.1F |

13.2.3 Cisco NX-OS

13.2.3.1 Supported Cisco Devices

Nexus 3000 or 9000 Platform

13.2.3.2 Recommended NX-OS Versions

Table 10: Cisco NX-OS Recommended Version(s)

| AOS Version | NX-OS Version(s) |
|---|----------------------------------|
| 3.3.0 / 3.3.0.1 / 3.3.0.2 / 3.3.0a / 3.3.0c | 9.2(2), 7.0(3)I7(8), 7.0(3)I7(7) |
| 3.2.4 | 9.3(3), 9.2(2), 7.0(3)I7(7) |
| 3.2.3 | 9.2(2), 7.0(3)I7(7) |
| 3.2.2 | 9.2(2), 7.0(3)I7(7) |
| 3.2.1 | 9.2(2), 7.0(3)I7(7) |
| 3.2.0 | 9.2(2), 7.0(3)I7(7) |
| 3.1.1 | 9.2(2), 7.0(3)I7(7) |
| 3.1.0 | 9.2(2), 7.0(3)I7(7) |
| 3.0.1 | 9.2(2), 7.0(3)I7(7) |
| 3.0.0 | 9.2(2), 7.0(3)I7(7) |
| 2.3.1 | 7.0(3)I7(4) |

13.2.4 Cumulus Linux

13.2.4.1 Supported Cumulus Devices

Cumulus Linux Supported Platforms with x86 CPU Processor

13.2.4.2 Recommended Linux Versions

Table 11: Cumulus Linux Recommended Version(s)

| AOS Version | Cumulus Linux Version(s) |
|-----------------------------------|--------------------------|
| 3.3.0.2 | 4.2.1, 3.7.13, 3.7.12 |
| 3.3.0 / 3.3.0.1 / 3.3.0a / 3.3.0c | 3.7.13, 3.7.12 |
| 3.2.4 | 3.7.13, 3.7.12 |
| 3.2.3 | 3.7.11 |
| 3.2.3 | 3.7.11 |
| 3.2.2 | 3.7.11 |
| 3.2.1 | 3.7.11 |
| 3.2.0 | 3.7.11 |
| 3.1.1 | 3.7.5 |
| 3.1.0 | 3.7.5 |
| 3.0.1 | 3.7.3 |
| 3.0.0 | 3.7.3 |
| 2.3.1 | 3.7.3 |

13.2.5 Enterprise SONiC

13.2.5.1 Supported Enterprise SONiC Devices

The table below lists the only devices that are supported by both SONiC Enterprise 3.1.0a/3.1.1 and Apstra AOS version 3.3.0a/3.3.0c.

Table 12: Enterprise SONiC Supported Devices on AOS Version 3.3.0a/3.3.0c

| Manufacturer | Model |
|-----------------|------------|
| Dell | Z9332F-ON |
| Dell | Z9264F-ON |
| Dell | Z9100-ON |
| Dell | S5296F-ON |
| Dell | S5248F-ON |
| Dell | S5232F-ON |
| Edgecore/Accton | AS7816-64X |
| Edgecore/Accton | AS7726-32X |
| Edgecore/Accton | AS7712-32X |
| Edgecore/Accton | AS7326-56X |
| Edgecore/Accton | AS5712-54X |

13.2.5.2 Recommended Enterprise SONiC Version

Starting with AOS version 3.3.0a, only SONiC Enterprise is supported. No other versions of SONiC, including SONiC community and earlier versions of Enterprise SONiC are supported by AOS.

Table 13: Enterprise SONiC Recommended Version(s)

| AOS Version | SONiC Version(s) |
|-------------|------------------------------------|
| 3.3.0a | SONiC-OS-3.1.0a-Enterprise_Base |
| 3.3.0c | SONiC-OS-3.1.1-Enterprise_Advanced |

13.2.6 Requesting NOS Support

Support for additional Network Device Network Operating Systems (NOS) versions may be available. Contact *Apstra Global Support* for more information about supported devices and NOS.

To request support for NOS not listed, please contact your Apstra Sales representative.

13.3 AOS Device Agent Configuration File

13.3.1 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 =
```

13.3.1.1 metadb

Agent Server Discovery is a client-server model. The AOS Device agent registers directly to the AOS server via the metadb connection. The AOS server can be discovered from static IP or DNS.

Dynamic DNS - By default, AOS device agents point to the DNS entry **aos-server**, relying on dhcp-provided DNS resolution and hostname resolution. On the AOS server, if the *metadb* connection entry points to a DNS entry, then the AOS agents must be able to resolve that DNS entry as well. DNS must be configured so aos-server resolves to an interface on the AOS 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.

Listing 1: Arista and Cisco static hostname

```
localhost(config)#ip host aos-server 192.168.25.250
```

Listing 2: Cumulus and Linux static hostnames /etc/hosts

```
192.168.25.250 aos-server
```

Listing 3: Obtaining the IP from the AOS 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
```

(continues on next page)

(continued from previous page)

```
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`

13.3.1.2 web

In a future release, the AOS REST API will be able to run on a separate server from the AOS server itself. This feature is for Apstra internal usage only.

13.3.1.3 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 AOS. For example, on a server, to bind the device agent to `eth1` instead of the default `eth0`, specify `interface = eth1`.

13.3.2 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.

13.3.2.1 enable_configuration_service

This field specifies the operation mode of the device agent: telemetry only or full control.

`enable_configuration_service = 0` Leaving the default value of 0 allows AOS to push telemetry (alerts) only. Configuration files are not modified. This ensures that AOS does not modify any configurations unless specified by the network administrator.

`enable_configuration_service = 1` Setting this field to 1 allows AOS to fully manage the device agent configuration, including pushing discovery and full intent-based configuration.

13.3.2.2 backup_config_restoration_timeout

By design, AOS does not *store* configuration on the device. This prevents a device from booting up and immediately participating in fabric that may not be properly configured yet. Following system startup, after the discovery phase completes, AOS configures the AOS device agent.

`backup_restoration_timeout = 00:00:00` This disabled state (default) keeps the AOS device agent from replacing the running configuration if it cannot contact the AOS server. Any previous configuration state will not be restored.

`backup_restoration_timeout = 00:15:00` Any value other than the default `00:00:00` enables the AOS 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.

13.3.3 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
```

AOS logs to the `/var/log/aos` folder under a series of files. AOS implements its own method of log rotation to prevent `/var/log/aos` from filling up. Log rotation can be enabled 1 or disabled 0. Each individual log file is rotated when it approaches the appropriate maximum size. Log rotation occurs by default every hour.

13.3.4 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
```

13.3.4.1 model

The device info section is used to modify the default device model of servers as they register to AOS. 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

13.3.5 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
```

13.4 AOS EVPN Support Addendum

This document outlines the caveats and limitations of deploying EVPN with AOS on supported hardware and NOS. Even though EVPN is a standard, vendors implement protocols in wildly different manners. On top of that, different ASICs support varying feature sets impacting EVPN BGP VXLAN implementations, for example, support for RIOT (Routing In and Out of Tunnels). This document serves to clearly state the implementations Apstra supports for an EVPN deployment.

13.4.1 Vendor Device OS Support

As of Apstra AOS version 3.3.0.2, Apstra supports EVPN on the following Vendor Device Operating Systems only (aka Network Operating Systems, NOS). Refer to *Device and NOS Support* for recommended NOS versions.

13.4.2 Hardware ASIC Support

- Cisco Cloudscale
- Mellanox Spectrum A1
- Trident Trident2 (see below)
- Trident Trident2+ (see below)
- Trident Trident3 (see below)
- Trident Tomahawk (see below)
- Juniper Q5

Table 14: AOS EVPN ASIC Support

| ASIC | Example Switches | Notes |
|-------------------------|---|--|
| Arista Trident2 | Arista DCS-7050 | Can be used as Spine, Leaf, or Border Leaf. Must set up EOS Recirculation interface(s) to be used as a Layer3 Leaf (see Arista VXLAN documentation for more information). |
| Arista Trident3 | DCS-7050CX3 | Can be used as Spine, Leaf, or Border Leaf. |
| Arista XP80 | Arista DCS-7160 | Can be used as Spine, Leaf, or Border Leaf. |
| Arista Jericho | DCS-7280R | Can be used as Spine, Leaf, or Border Leaf. |
| Cisco Cloud-scale | Cisco 93180YC-EX | Can be used as Spine, Leaf, or Border Leaf |
| Cisco Trident2 with ALE | Cisco 9396PX, 9372PX, 9332PQ, 9504 | Can be used as Spine, Leaf, or Border Leaf (see <i>TCAM Carving in NXOS</i>). |
| Cisco Trident2+ | Cisco 3132Q-V | Cannot be used as Border Leaf |
| Cumulus Trident2+ | Dell S4048T-ON, S6010-ON, EdgeCore AS5812-54X, AS6812-32X | Can be used as Spine, Leaf, or Border Leaf (see <i>Cumulus RN-766 Support</i>) in Cumulus. |
| Cumulus Maverick | Dell S4148F-ON, S4148T-ON | Can be used as Spine, Leaf, or Border Leaf (see <i>Cumulus RN-766 Support</i>) in Cumulus. |
| Cumulus Mellanox A1 | Mellanox MSN2010, MSN2100, MSN2410, MSN2700 (Spectrum A1) | Can be used as Spine, Leaf, or Border Leaf. Spectrum A0 not supported. |
| Cumulus Tomahawk | Dell Z9100-ON, EdgeCore AS7712-32X | Can be used as Spine, Leaf, or Border Leaf (see <i>Cumulus RN-766 Support</i>). Must set up Hyperloop interface(s) (see <i>Enable VXLAN Routing on Cumulus Tomahawk</i>) to be used as a Layer3 Leaf in Cumulus. |
| Juniper Q5 | Juniper QFX10002 | Can be used as Spine, Leaf, or Border Leaf |
| Juniper Trident2 | Juniper QFX5100 | Can be used as Spine or Layer2 Leaf |
| Juniper Trident2+ | Juniper QFX5110 | Can be used as Spine, Leaf, or Border Leaf |
| Juniper Trident3 | Juniper QFX5120 | Can be used as Spine, Leaf, or Border Leaf |

13.4.3 Limitations

13.4.3.1 EVPN Layer2 Limitations

- VLAN (Rack-local) Virtual networks must be in the default security zone.
- VxLAN (Inter-rack) Virtual networks cannot be part of the default security zone.

13.4.3.2 EVPN Layer3 Limitations

- External routers must connect to leaf devices
- Multi-zone security segmentations only support up to 16 security zones (VRFs) on Arista (HW Limitation)
- Inter security zone (VRF) routing must be handled on an external router (EVPN type 5 route leaking)
- All BGP sessions and loopback addresses are part of the default security zone.

13.4.4 Cumulus RN-766 Support

In all current versions of Cumulus Linux; when using Broadcom Trident II+, Trident3, and Maverick platforms in an external VXLAN routing environment; the switch does not rewrite MAC addresses and TTL, so packets are dropped by the next hop. See [Cumulus Linux Release Notes for RN-766](#) for more information.

AOS automatically implements work-arounds for Cumulus Linux RN-766 for the following criteria (as of AOS 2.3.2):

- Device profiles with “ASIC” field set to ‘T2+’, ‘T3’, or ‘maverick’ assigned to border-leaf device(s) in the AOS blueprint
- **Overlay Control Protocol** set to **MP-EBGP EVPN**
- External Connectivity Point (ECP) configured for default or new security zone
- External Connectivity Point (ECP) set to L2 mode, with Layer 2 VLAN based BGP peering with the external router
- External Connectivity Point (ECP) configured with VLAN ID, SVI Subnet, and SVI IPs for border-leaf(s) and router(s).
- User assigns VNI from a VNI pool to the AOS blueprint

AOS automatically allocates additional VNI and configuration for the Cumulus Linux RN-766 work-around.

Warning: The AOS RN-766 workaround is not supported on blueprints with MLAG L3 peer links deployed.

Warning: The AOS RN-766 workaround is not supported for AOS blueprints with **Overlay Control Protocol** set to **Static VXLAN** and any L3 External Connectivity Points (ECP).

13.4.5 TCAM Carving in NXOS

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.

See the following Knowledge Base article for details:

<https://apstra.zendesk.com/hc/en-us/articles/360049704054>

Apstra recommends that TCAM Carving be applied during the device management setup or during Cisco Power-on Auto Provisioning (POAP). TCAM Carving requires a device reboot. This all should be done prior to AOS device system agent installation.

Alternatively, TCAM Carving can be applied using AOS configlets during AOS blueprint deployment. The devices will then need to be manually rebooted.

Use the `show hardware access-list tcam region` to show and verify TCAM allocation on Cisco NX-OS.

13.4.5.1 Cisco NXOSv TCAM Carving

Listing 4: NXOSv Configlet Template Text

```
hardware access-list tcam region vacl 0
hardware access-list tcam region racl 0
hardware access-list tcam region arp-ether 256
```

Listing 5: NXOSv Configlet Negation Template Text

```
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
```

13.4.5.2 Cisco Trident2 TCAM Carving

Listing 6: Trident2 Configlet Template Text

```
hardware access-list tcam region l3qos 0
hardware access-list tcam region arp-ether 256 double-wide
```

Listing 7: Trident2 Configlet Negation Template Text

```
no hardware access-list tcam region l3qos 0
no hardware access-list tcam region arp-ether 256 double-wide
```

13.4.6 Arista EOS VxLAN Routing

13.4.6.1 Recirculation Interface for Arista Trident2 Devices

VxLAN Routing for Trident2 devices (e.g. 7050QX-32) is supported but requires that the user assign EOS recirculation interfaces to unused physical interfaces on the device. The user can use AOS configlets to deploy this to all devices requiring this configuration.

Listing 8: Template Text

```
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
```


Listing 9: Negation Template Text

```
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
```

13.4.6.2 VxLAN Routing System Profile for Arista Jericho Devices

It is recommended when using VxLAN Routing for Jericho devices (e.g. 7280SR-48C6) that the user assign EOS VxLAN Routing System Profile on the device.

Apstra recommends that Arista TCAM system profile be applied during the device management setup or during Arista Zero-Touch Provisioning (ZTP). TCAM system profile requires a device reboot. This all should be done prior to AOS device system agent installation.

Alternatively, the user can use AOS configlets to deploy this to all devices requiring this configuration and manually reboot the devices.

Listing 10: Template Text

```
hardware tcam
  system profile vxlan-routing
```

Listing 11: Negation Template Text

```
hardware tcam
  no system profile vxlan-routing
```

13.4.7 AOS Graph Node VTEP Types

13.4.7.1 Unicast VTEPs

Unicast VTEPs do not apply to Cumulus and Arista.

Cisco Unicast VTEPs - Vendor Definition: Anycast VTEP

AOS IP Allocation

Unique per leaf in MLAG pair

Not allocated to singleton switches

MLAG Configuration

Listing 12: rack1-leaf1

```
interface loopback1
  IP address 10.0.0.1/32
  IP address 10.0.0.3/32 secondary
```

(continues on next page)

(continued from previous page)

```
interface nve1
  source-interface loopback1
```

Listing 13: rack1-leaf2

```
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
```

13.4.7.2 Logical VTEPs

Arista Logical VTEPs

AOS IP Allocation

Logical VTEP configured as primary IP on loopback1 interface for both MLAG and singleton switches

All top of rack nodes share same logical VTEP IP:

- MLAG leafs share same logical VTEP IP
- Singleton leaf gets its own VTEP IP

MLAG Configuration

Listing 14: rack1-leaf1

```
interface loopback1
  IP address: 10.0.0.1/32
  IP address: 10.0.0.4/32 secondary
interface vxlan1
  vxlan source-interface loopback1
```

Listing 15: rack1-leaf2

```
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
```

(continues on next page)

(continued from previous page)

```
interface vxlan1
  vxlan source-interface loopback1
```

Cumulus Logical VTEPs

AOS IP Allocation

For MLAG (clagd) leafs, shared as clagd-vxlan-anycast-ip on lo interface, shared on leaf1 and leaf2

For singleton leafs, configured as additional IP alongside loopback ip on iface lo

MLAG Configuration

Listing 16: rack1-leaf1

```
auto lo
iface lo inet loopback
  IP address: 10.0.0.6/32
  clagd-vxlan-anycast-ip 10.0.0.4/32
```

Listing 17: rack1-leaf2

```
auto lo
iface lo inet loopback
  IP address: 10.0.0.6/32
  clagd-vxlan-anycast-ip 10.0.0.4/32
```

Single Switch Configuration

```
auto lo
iface lo inet loopback
  IP address: 10.0.0.6/32
  IP address: 10.0.0.7/32
```

13.4.7.3 Anycast VTEP

Anycast VTEPs do not apply to Cisco and Cumulus.

Arista Anycast VTEPs

AOS IP Allocation

One anycast VTEP for entire blueprint, shared between all Arista leafs

Configured as secondary IP on loopback1 interface

MLAG Configuration

Listing 18: rack1-leaf1

```
interface loopback1
  IP address 10.0.0.1/32
  IP address 10.0.0.5/32 secondary
```

(continues on next page)

(continued from previous page)

```
interface vxlan1
  vxlan source-interface loopback1
```

Listing 19: rack1-leaf2

```
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
```

13.5 Graph

AOS uses the Graph model to represent a single source of truth regarding infrastructure, policies, constraints etc. This Graph model is subject to constant change and we can query it for various reasons. It is represented as a graph. All information about the network is modeled as nodes and relationships between them.

Every object in a graph has a unique ID. Nodes have a type (a string) and a set of additional properties based on a particular type. For example, all switches in our system are represented by nodes of type *system* and can have a property *role* which determines which role in the network it is assigned (spine/leaf/server). Physical and logical switch ports are represented by an *interface* node, which also has a property called *if_type*.

Relationships between different nodes are represented as graph edges which we call *relationships*. Relationships are directed, meaning each relationship has a source node and a target node. Relationships also have a type which determines which additional properties particular relationship can have. E.g. *system* nodes have relationships of type *hosted_interfaces* towards *interface* nodes.

A set of possible node and relationship types is determined by a graph schema. The schema defines which properties nodes and relationships of particular type can have along with types of those properties (string/integer/boolean/etc) and constraints. We use and maintain an open source schema library, Lollipop, that allows flexible customization of value types.

Going back to the graph representing a single source of truth, one of the most challenging aspects was how to reason about it in the presence of change, coming from both the operator and the managed system. In order to support this we developed what we call *Live Query* mechanism which has three essential components:

- Query Specification
- Change Notification
- Notification Processing

Having modeled our domain model as a graph, you can find particular patterns (subgraphs) in a graph by running searches on the graph specified by graph queries. The language to express the query is conceptually based on Gremlin, an open source graph traversal language. We also have parsers for queries expressed in another language - Cypher, which is a query language used by popular graph database neo4j.

13.5.1 Reference Design Schema

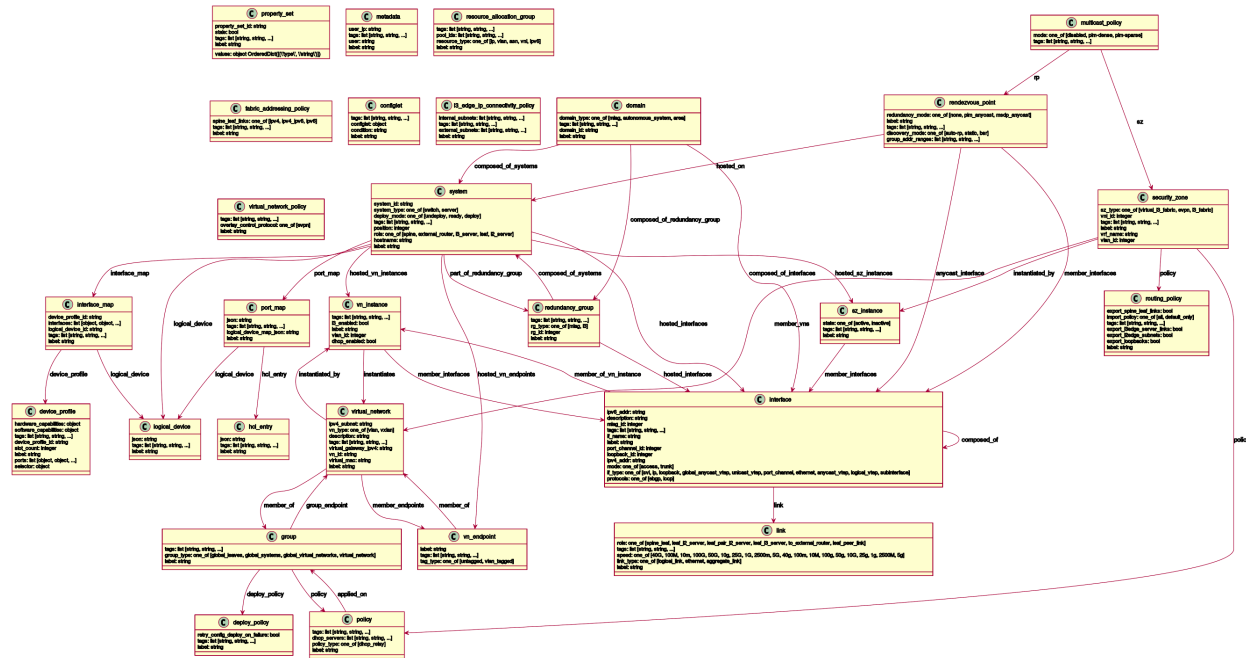


Fig. 1: Click on image to zoom

13.5.2 Query Specification

You start with a node() and then keep chaining method calls, alternating between matching relationships and nodes:

```
node('system', name='system').out().node('interface', name='interface').out().node(
    'link', name='link')
```

The query above translated in english reads something like: *starting from a node of type system, traverse any outgoing relationship that reaches node of type interface, and from that node traverse all outgoing relationship that lead to node of type 'link'.*

At any point you can add extra constraints:

```
node('system', role='spine', name='system').out().node('interface', if_type='ip',
    name='interface')
```

Notice role='spine' argument, it will select only *system* nodes that have *role* property set to *spine*.

Same with *if_type* property for *interface* nodes.

```
node('system', role=is_in(['spine', 'leaf']), name='system')
    .out()
    .node('interface', if_type=ne('ip'), name='interface')
```

That query will select all *system* nodes that have role either *spine* or *leaf* and *interface* nodes that have *if_type* anything but *ip* (*ne* means *not equal*).

You can also add cross-object conditions which can be arbitrary Python functions:

```
node('system', name='system')
.out().node('interface', name='if1')
.out().node('link')
.in().node('interface', name='if2')
.in().node('system', name='remote_system')
.where(lambda if1, if2: if1.if_type != if2.if_type)
```

You refer to objects by giving them names and using those names as argument names for your constraint function (of course you can override that but it makes a convenient default behavior). So, in example above it will take two *interface* nodes named *if1* and *if2*, pass them into given *where* function and filter out those paths, for which function returns False. Don't worry about where you place your constraint: it will be applied during search as soon as all objects referenced by constraint are available.

Now, you have a single path, you can use it to do searches. However, sometimes you might want to have a query more complex than a single path. To support that, query DSL allows you to define multiple paths in the same query, separated by comma(s):

```
match(
    node('a').out().node('b', name='b').out().node('c'),
    node(name='b').out().node('d'),
)
```

This `match()` function creates a grouping of paths. All objects that share the same name in different paths will actually be referring to the same object. Also, `match()` allows adding more constraints on objects with `where()`. You can do a distinct search on particular objects and it will ensure that each combination of values is seen only once in results:

```
match(
    node('a', name='a').out().node('b').out().node('c', name='c')
).distinct(['a', 'c'])
```

This matches a chain of *a* -> *b* -> *c* nodes. If two nodes *a* and *c* are connected through more than one node of type *b*, the result will still contain only one (*a*, *c*) pair.

There is another convenient pattern to use when writing queries: you separate your structure from your criteria:

```
match(
    node('a', name='a').out().node('b').out().node('c', name='c'),
    node('a', foo='bar'),
    node('c', bar=123),
)
```

Query engine will optimize that query into:

```
match(
    node('a', name='a', foo='bar')
    .out().node('b')
    .out().node('c', name='c', bar=123)
)
```

No cartesian product, no unnecessary steps.

13.5.3 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.

13.5.4 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 below example is an example of 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 AOS web interface.

Listing 20: Property matcher example:

```
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 `NodeIterator` 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.

13.5.5 Putting It All Together

This below query is an example of an internal rule that AOS 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.

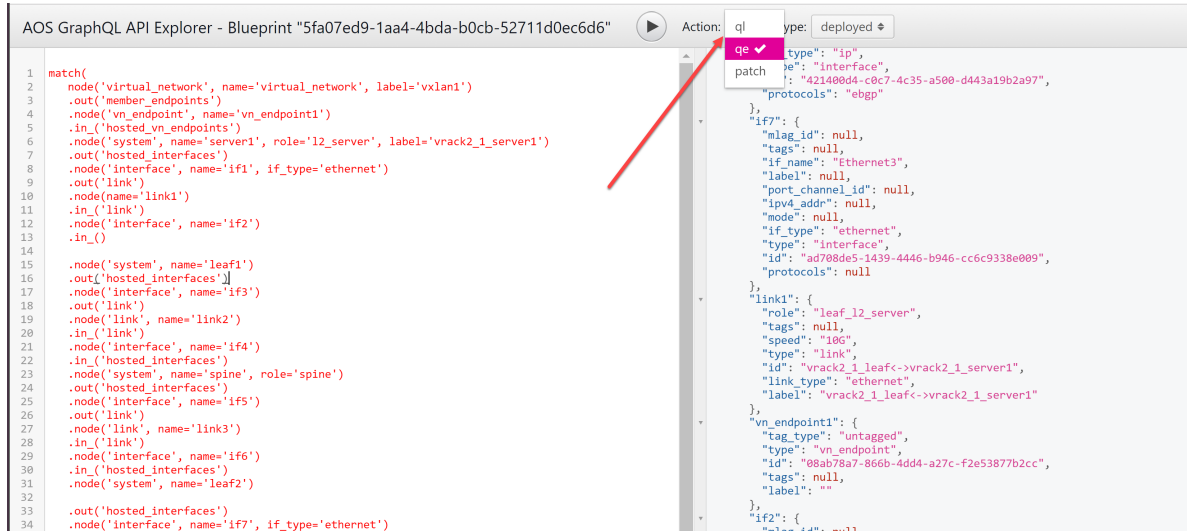
```
@rule(match(
    node('system', name='spine_device', role='spine')
    .out('hosted_interfaces')
    .node('interface', name='spine_if')
    .out('link')
    .node('link', name='link')
    .in_('link')
    .node('interface', name='leaf_if')
    .in_('hosted_interfaces')
    .node('system', name='leaf_device', 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
```


13.5.6 Convenience Functions

To avoid creating complex `where()` clauses when building a graph query, use convenience functions, available from the AOS web interface.

1. From the **Staged** or **Active** view of the Blueprint, 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.



13.5.6.1 Functions

The Query Engine describes a number of helpful functions:

match (*path_queries)

This function returns a `QueryBuilder` object containing each result of a matched query. This is generally a useful shortcut for grouping multiple match queries together.

These two queries are not a ‘path’ together (no intended relationship). Notice the comma to separate out arguments. This query will return all of the leafs and spines together.

```
match(
  node('system', name='leaf', role='leaf'),
  node('system', name='spine', role='spine'),
)
```

node (self, type=None, name=None, id=None, **properties)

Parameters

- **type** (*str* or *None*) – Type of node to search for
- **name** (*str* or *None*) – Sets the name of the property matcher in the results
- **id** (*str* or *None*) – Matches a specific node by node ID in the graph
- **properties** (*dict* or *None*) – Any additional keyword arguments or additional property matcher convenience functions to be used

Returns Query builder object for chaining queries

Return type QueryBuilder

While both a function, this is an alias for the PathQueryBuilder nodes – see below.

iterate()

Returns generator

Return type generator

Iterate gives you a generator function that you can use to iterate on individual path queries as if it were a list.

For example:

```
def find_router_facing_systems_and_intfz(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')
    ))
```

13.5.6.2 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 will return a path query object. When querying a path, you usually want to specify a node *type*: eg `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 (eg, *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 (eg, *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)
```

13.5.6.3 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",
```

(continues on next page)

(continued from previous page)

```

        "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']  
↪ '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)
```

13.5.7 AOS Graph Datastore

The AOS 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. AOS 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.

13.6 Juniper Apstra Technology Previews (Tech Previews)

Tech Previews give you the ability to test functionality and provide feedback during the development process of innovations that are not final production features. The goal of a Tech Preview is for the feature to gain wider exposure and potential full support in a future release. Customers are encouraged to provide feedback and functionality suggestions for a Technology Preview feature before it becomes fully supported.

Tech Previews may not be functionally complete, may have functional alterations in future releases, or may get dropped under changing markets or unexpected conditions, at Juniper’s sole discretion. Juniper recommends that you use Tech Preview features in non-production environments only.

Juniper considers feedback to add and improve future iterations of the general availability of the innovations. Your feedback does not assert any intellectual property claim, and Juniper may implement your feedback without violating your or any other party’s rights.

These features are “as is” and voluntary use. Juniper Support will attempt to resolve any issues that customers experience when using these features and create bug reports on behalf of support cases. However, Juniper may not provide comprehensive support services to Tech Preview features. Certain features may have reduced or modified security, accessibility, availability, and reliability standards relative to General Availability software. Tech Preview is not supported under existing service agreements, SLAs, or support service.

For additional details, please contact *Juniper Support* or your local account team.

INDEX

`ensure_different()`built-in function, 925

`eq()`built-in function, 926

`ge()`built-in function, 927

`gt()`built-in function, 927

`in_()`built-in function, 925

`is_in()`built-in function, 928

`is_none()`built-in function, 928

`iterate()`built-in function, 924

`le()`built-in function, 928

`lt()`built-in function, 928

`match()`built-in function, 923

`ne()`built-in function, 927

`node()`built-in function, 923, 924

`not_in()`built-in function, 928

`not_none()`built-in function, 928

`out()`built-in function, 924

`where()`built-in function, 925