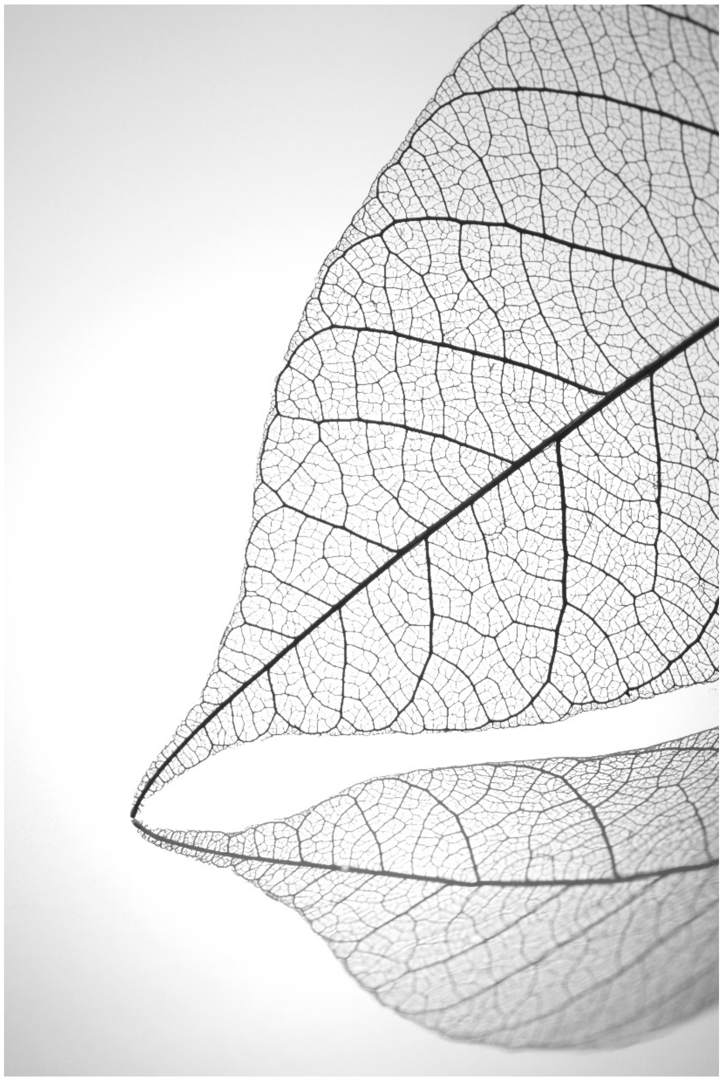


JUNOS EVO FUNDAMENTALS

VER 1.2

This document is an overview of the differences between JunOS and JunOS EVO and basic troubleshooting commands for operations

CONTENTS

Document Revision

Doc Version	Date	Revision owner	Comments
Ver 1.0	29-Mar-2024	Jimmy Jimenez Salas	Document creation
Ver 1.1	24-May-2024	Jimmy Jimenez Salas	Final revisions

QUICK FACTS

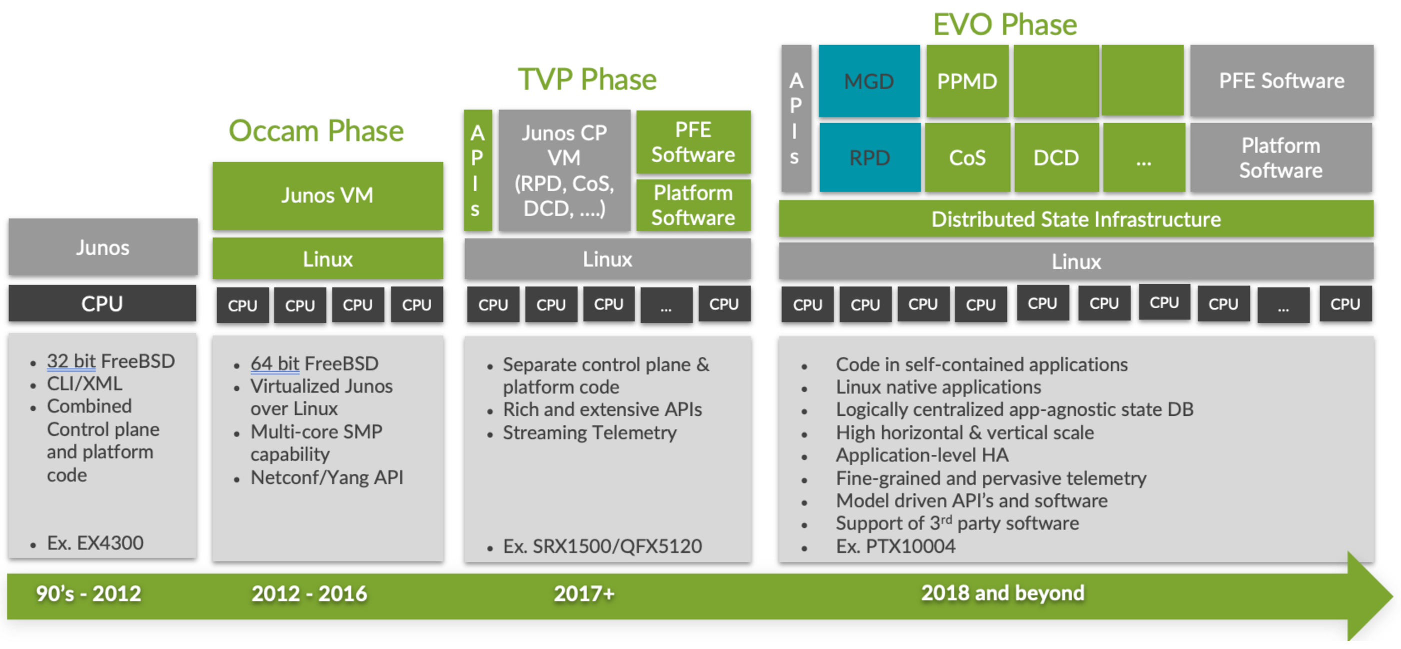
- Most CLI commands remain the same in EVO
- Base OS is Linux, not FreeBSD
- Most processes work now as a native Linux application
- PTX10004 is a modular platform:
    - Dual routing engines for control plane redundancy, one master and one backup
    - Graceful switchover is always enabled by default and can't be disabled
    - Nonstop Active Routing can be used to preserve routing state between REs
    - Multiple forwarding ASICs (PFEs) on each linecard
- Each card and larger component functions as a node part of a 'cloud' forming the entire System
- Uses Distributed Asynchronous State (DDS)
- All states are modeled as objects and collections of objects as graphs.

WHAT IS EVO?

Evo, short for Junos Evolution, is the name of the project for the next-generation network operating system that will run on Juniper Networks devices.

Characteristics:

- A distributed, Linux based, general purpose operating system
- Runs on all CPUs that run Linux
- Object oriented which supports app development
- Hardware-independent
- Application-independent


Junos
CPU
• 32 bit FreeBSD
• CLI/XML
• Combined Control plane and platform code
• Ex. EX4300
Occam Phase
Junos VM
Linux
CPU CPU CPU CPU
• 64 bit FreeBSD
• Virtualized Junos over Linux
• Multi-core SMP capability
• Netconf/Yang API
TVP Phase
APIs
Junos CP VM (RPD, CoS, DCD, ...)
PFE Software
Platform Software
Linux
CPU CPU CPU ... CPU
• Separate control plane & platform code
• Rich and extensive APIs
• Streaming Telemetry
• Ex. SRX1500/QFX5120
EVO Phase
APIs
MGD
PPMD
PFE Software
RPD
CoS
DCD
...
Platform Software
Distributed State Infrastructure
Linux
CPU CPU CPU CPU CPU CPU CPU CPU ... CPU
• Code in self-contained applications
• Linux native applications
• Logically centralized app-agnostic state DB
• High horizontal & vertical scale
• Application-level HA
• Fine-grained and pervasive telemetry
• Model driven API's and software
• Support of 3rd party software
• Ex. PTX10004
90's - 2012
2012 - 2016
2017+
2018 and beyond

Junos Evolved in a Nutshell

Consistent control, management and data plane

- RPD, MGD, L2 apps
- PFE 2.0 (AFT) for custom ASIC based cards

Openness

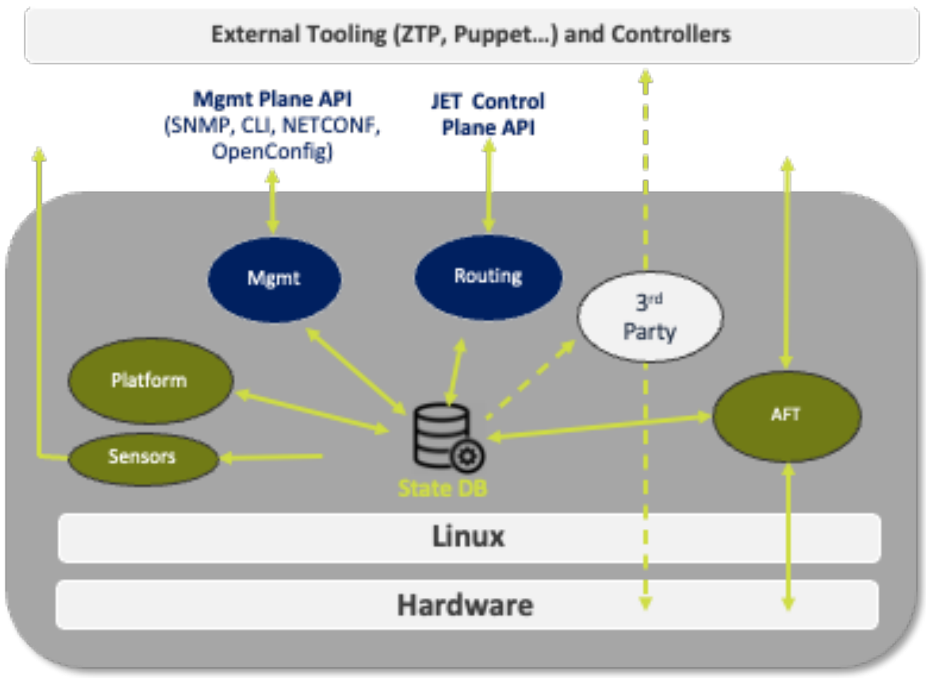
- Linux native platform and apps
- Support for 3rd party software and tools

Modularity

- Component level design with resiliency
- Support for hitless component upgrade

Logically Centralized Database

- All state modelled and API Accessible
- Pub-sub communication between components
- Strong fault isolation between components
- Improved diagnostics for rapid debugging


External Tooling (ZTP, Puppet...) and Controllers
Mgmt Plane API (SNMP, CLI, NETCONF, OpenConfig)
JET Control Plane API
Mgmt
Routing
3rd Party
Platform
Sensors
State DB
AFT
Linux
Hardware

WHAT'S CHANGED?

System vs. Node

In JUNOS, most of the time, "system" means the Routing Engine itself. In Evo, "system" is a collaboration of all the "nodes" - Routing Engines, FPCs…etc. As a result, "node" specific operation is under "show node"/"request node"

```
root@vbrackla_RE0> request node ?
Possible completions:
  halt                   Halt the node
  offline                Offline the node
  online                 Online the node
  power-off              Power-off the node
  power-on               Power-on the node
  reboot                 Reboot the node
root@vbrackla_RE0> show node ?
Possible completions:
  reboot                 Show any pending node halt/reboot/poweroff requests
  statistics             Show network statistics of a node
root@vbrackla_RE0>
```

"system" refers to all the nodes. eg. "request system software add" will add the image to all Routing Engine nodes

```
root@vbrackla_RE0> request system shutdown ?
Possible completions:
  halt                   Halt the system
  power-off              Power off the system
  reboot                 Reboot the system
root@vbrackla_RE0>
root@vbrackla_RE0> request node ?
Possible completions:
  halt                   Halt the node
  offline                Offline the node
  online                 Online the node
  power-off              Power-off the node
  power-on               Power-on the node
  reboot                 Reboot the node
root@vbrackla_RE0>
```

Each node will have its name appended to the system hostname for differentiation:

```
amazon@az1-1-co-cor-ptx10k-re0> show configuration system host-name | display inheritance | display set
set system host-name az1-1-co-cor-ptx10k

{master}
amazon@az1-1-co-cor-ptx10k-re0> request routing-engine login other-routing-engine
Password:
--- JUNOS 23.2R2-S1.4-EVO Linux (none) 5.2.60-yocto-standard-g3c005ea #1 SMP PREEMPT Sun Sep 24 00:31:15
UTC 2023 x86_64 x86_64 x86_64 GNU/Linux
{backup}
amazon@az1-1-co-cor-ptx10k-re1> exit

{master}
amazon@az1-1-co-cor-ptx10k-re0> start shell pfe network fpc0
Trying 128.0.0.16...
Connected to fpc0.
Escape character is '^]'.

root@az1-1-co-cor-ptx10k-fpc0:pfe>
```

Another example, “show system storage” displays information from all nodes (REs and FPCs):

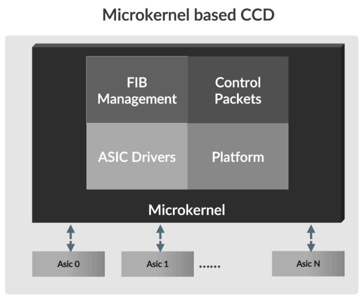
```
root@az1-1-co-cor-ptx10k-re0> show system storage
fpc0:
--------------------------------------------------------------------------
Filesystem              Size       Used      Avail  Capacity   Mounted on
/dev/root                34M        34M          0      100%  /run/initramfs
/<..>

re0:
--------------------------------------------------------------------------
Filesystem              Size       Used      Avail  Capacity   Mounted on
/dev/root                37M        37M          0      100%  /run/initramfs
/dev/sda2                32G       4.3G        26G       15%  /soft
/dev/sda5               3.0G        95M       2.7G        4%  /data
/dev/sda7               144G        35G       103G       25%  /var
<..>

re1:
--------------------------------------------------------------------------
Filesystem              Size       Used      Avail  Capacity   Mounted on
<..>
```

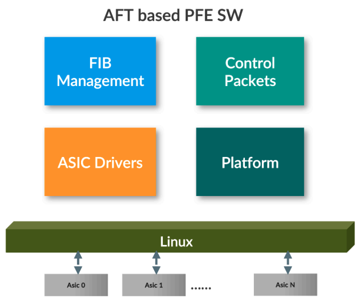
Synchronization status between routing-engines can be verified from the backup RE node with ‘show system switchover’:

```
amazon@az1-1-co-cor-ptx10k-re0> request routing-engine login other-routing-engine
Password:
Last login: Fri May 24 16:15:57 2024 from 128.0.0.4
--- JUNOS 23.2R2-S1.4-EVO Linux (none) 5.2.60-yocto-standard-g3c005ea #1 SMP PREEMPT Sun Sep 24
00:31:15 UTC 2023 x86_64 x86_64 x86_64 GNU/Linux

amazon@az1-1-co-cor-ptx10k-re1> show system switchover
Graceful switchover: On
Configuration database: Ready
Object database: Ready
Applications' ready state: Ready
Switchover Status: Ready
<..>
```

System Applications

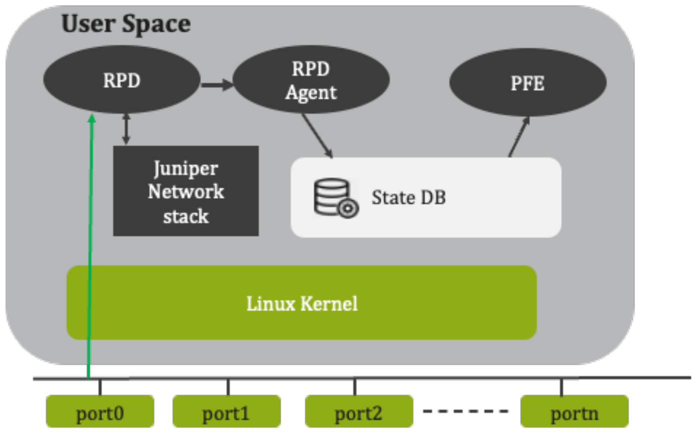
- Since we have re-designed how the "system" works, some well-known applications might not be there anymore.
- Some kernel functions are moved out to become an App to keep the kernel "clean" (eg. Icmpd, ndp, arpd).
- No more "ukern" on the FPC. It's replaced by Linux based AFT(Advanced Forwarding Toolkit):





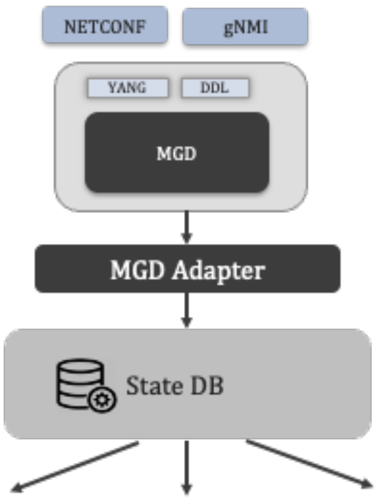
RPD on Linux

RPD runs under Linux. It's single sourced with JunOS so the behavior is similar. An RPD agent maps RPD states to EVO native modeled states



MGD Integration

- MGD is single sourced across Junos and JUNOS-EVO
- Ensures feature consistency across Junos and Evo
- Configuration and operational commands remain consistent across Junos and Evo.
- Some outputs and syntax may be different



System Logging

In Junos OS Evolved, each node has the standard `journalctl` tool, which is an interface to retrieve and filter the system journal. System log messages are parsed from the system journal. The `relay-eventd` process runs on all nodes and retrieves events (based on the syslog configuration) from the system journal as well as error messages from the different applications and forwards them to the `master-eventd` process. The `master-eventd` process runs on the primary Routing Engine and writes the log messages and errors to disk.
In Junos OS Evolved there is no `messages` file on the backup Routing Engine. All backup Routing Engine logs are in the `messages` file on the primary Routing Engine node.

```
root@vbrackla_RE0:~# journalctl --list-boot
-3 920f2d4d48e24bd48d34695067f7e1e7 Sun 2018-09-16 22:35:15 PDT<E2><80><94>Mon 2018-09-17 08:07:17 PDT
-2 089ca2a2cb134313a85b22f97bb81154 Mon 2018-09-17 08:08:29 PDT<E2><80><94>Tue 2018-09-18 08:13:48 PDT
-1 6480d1e2e85d4f3480d93fa186451f83 Tue 2018-09-18 08:14:45 PDT<E2><80><94>Tue 2018-09-18 17:45:49 PDT
 0 0ba8fb82c6174790b1f81cdf155c643d Tue 2018-09-18 17:46:43 PDT<E2><80><94>Tue 2018-09-18 22:27:59 PDT

root@vbrackla_RE0:~# journalctl -b0
-- Logs begin at Sun 2018-09-16 22:35:15 PDT, end at Tue 2018-09-18 22:28:29 PDT. --
Sep 18 17:46:43 re0 systemd-journald[2169]: Runtime journal (/run/log/journal/) is 8.0M, max 64.0M, 56.0M
free.
Sep 18 17:46:43 re0 systemd-journald[2169]: System journal (/var/log/journal/) is 1.5G, max 1.5G, 0B free.
Sep 18 17:46:43 re0 systemd-journald[2169]: Time spent on flushing to /var is 92.914ms for 2 entries.
Sep 18 17:46:43 re0 kernel: Linux version 4.8.24-WR2.2.1_standard (jenkins@jbm-ec-plt03) (gcc version 6.2.0
(GCC) ) #61 SMP PREEMPT Fri Sep 7 14:14:35 PDT 2018
```

Traceoptions

Junos OS Evolved uses a new tracing architecture. All running applications create trace information, with multiple instances of the same application having their own trace information. As a result, Junos OS Evolved does not support the traceoptions statement at many of the hierarchy levels that Junos OS supports. However, some hierarchy levels, such as those under [edit protocols], still require configuring the traceoptions statement to enable trace messages.

In Junos OS Evolved, you do not view trace files directly, and you should never add, edit, or remove trace files under the /var/log/traces directory because this can corrupt the traces. Instead, you use the `show trace` `application application-name node node-name` command to read and decode trace messages stored in the trace files.

```
show trace application application-name node node-name – Read and decode trace files.

clear trace – Manually clean up trace files.

set system trace application – Modify trace message configurations at the application level.
```

Routing Engine Firewall Filters

In Junos OS, to control the flow of local packets between the physical interfaces and the Routing Engine, you can apply stateless firewall filters to the input or output of the loopback interface. The loopback interface (lo0) is the interface to the Routing Engine and carries no data packets. In Junos OS, filters applied to the loopback interface apply to both network control traffic and management traffic

In Junos OS Evolved, firewall filters applied to the loopback interface apply only to network control traffic. You must explicitly apply firewall filters to the management interface to filter management traffic.

Management interface naming also changes, <node>:mgmt-0 for example on PTX10004.

XINETD SERVICE

xinetd and an extension to inetd and performs all the network TCP connections to the system. Controlling which service is active, redirections, resource utilization control and access control. Each time a connection is attempted to the system, validation is performed in xinetd rules defined. Access problems to the system either via WAN interfaces or mgmt interface could be related to the xinetd TCP service.

CLI Command Comparison with EVO

Routing Protocols

- MGD & RPD process shares a single source for JUNOS & JUNOS EVO
- There are no differences in CLI operational commands for most RPD protocols.

System OS

- Since the base kernel for EVO platform is Linux, few OS level commands under show system hierarchy have differences in output
  - e.g. show system memory, show system directory, show system statistics, show system storage

Platform/PFE

- Largely driven by differences in the platform/chipset (fixed vs chassis etc.)

Port speed and channelization

Configuration for channelization in JunOS (4x10):

```
set chassis fpc 0 pic 0 port x channel-speed 10g
set chassis fpc 0 pic 0 port x channel-speed 10g
```

Configuration in EVO:

```
root@scapa-04-re0# set interfaces et-4/0/0 speed ?
Possible completions:
100g
10g
25g
400g
40g
50g
```

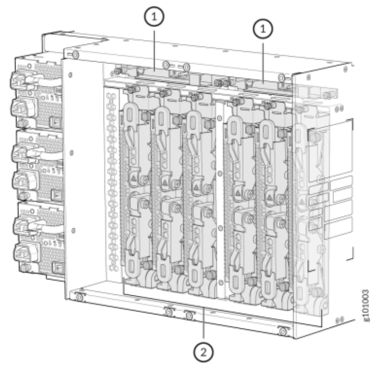
To channelize a port, use `number-of-sub-ports` along with `speed`. For example, to set 4x10g mode for a port:

```
Set interfaces et-4/0/0 speed 10g number-of-sub-ports 4
```

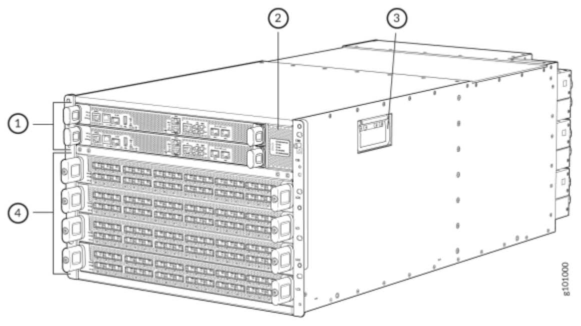
To get supported port speeds per FPC/PIC:

```
show chassis pic fpc-slot 0 pic-slot 0
FPC slot 0, PIC slot 0 information:
  Type                               JNP10K-4Q56DD-32Q28-LZ-LC-PIC
  State                              Online
  PIC version                     255.255
  Uptime                      30 days, 21 hours, 40 minutes, 42 seconds
<..>
Port speed information:

  Port  PFE      Capable Port Speeds
  0      0       1x10G 4x10G 1x40G 2x50G 4x25G 1x100G
  1      0       1x100G
  2      0       1x10G 4x10G 1x40G 4x25G 1x100G 2x50G
  3      0       1x100G
  4      0       1x10G 4x10G 1x40G 4x25G 2x50G 1x100G 8x25G 2x100G 3x100G 4x100G 1x400G
  5      0       1x10G 4x10G 1x40G 4x25G 1x100G 2x50G
  6      0       1x10G 4x10G 1x40G 4x25G 1x100G 2x50G
<..>
```

PTX10004 ARCHITECTURE



1– Fan tray controllers	2– Switch fabric



1– RCBs	3– Handles
2– Status panel	4– Line cards

Control Plane (RCB)

- The control board and routing engine are combined in a single FRU (Field replaceable unit). Up to two are supported.
- CBs communicate with all components via individual I2C bus segments. These are used for identification and power ON/OFF of the FRUs.
- PCI is used for connectivity with the SIBs to configure and control the fabric ASICs (ZF)
- 10G internal ethernet connectivity between CBs and Linecards, used to download software and IPC communication
- RS232 connectivity between CB and FPCs for internal CTY connectivity.
- EVO running on both REs as active/Standby, connectivity between both uses the internal 10G ethernet connection
- Supports Graceful switchover for Master RE changes without rebooting any component and nonstop routing to maintain protocol synchronization between REs.
- GRES/NSR is enabled by default
- The primary role switches to the backup Routing Engine if:
    - The primary Routing Engine kernel stops operating.
    - The primary Routing Engine experiences a hardware failure.
    - The administrator initiates a manual switchover.

Verify RCB mastership:

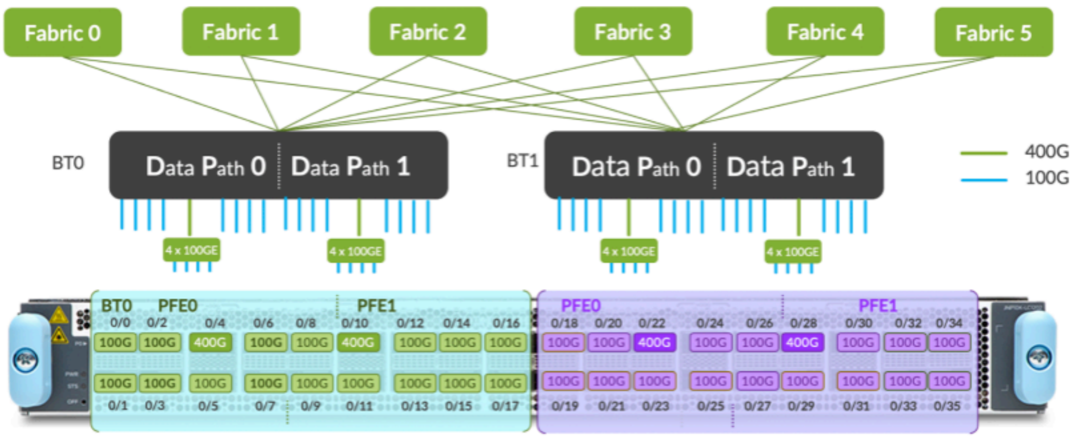
```
amazon@az1-1-co-cor-ptx10k-re0> show chassis routing-engine
Routing Engine status:
  Slot 0:
    Current state                  Master
    Election priority              Master
    Temperature                 38 degrees C / 100 degrees F
    CPU temperature             38 degrees C / 100 degrees F
    DRAM                      120595 MB (131072 MB installed)
<..>
Routing Engine status:
  Slot 1:
    Current state                  Backup
    Election priority              Backup
    Temperature                 34 degrees C / 93 degrees F
    CPU temperature             34 degrees C / 93 degrees F
    DRAM                      121754 MB (131072 MB installed)
<..>
```

Verify GRES/NSR readiness:

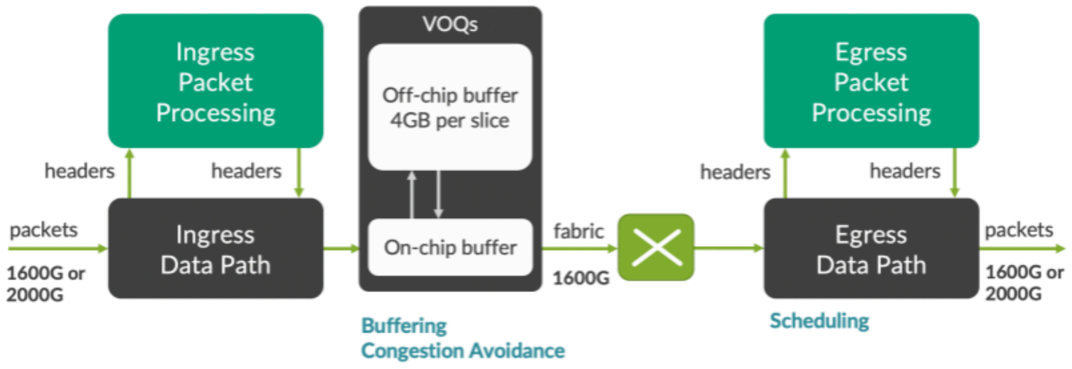
```
{master}
amazon@az1-1-co-cor-ptx10k-re0> request chassis routing-engine master switch check
Switchover Ready
{master}
amazon@az1-1-co-cor-ptx10k-re0> request routing-engine login other-routing-engine
{backup}
amazon@az1-1-co-cor-ptx10k-re1> show system switchover
Graceful switchover: On
Configuration database: Ready
Object database: Ready
Applications' ready state: Ready
Switchover Status: Ready
```

Data Path

- Up to 4 FPCs on PTX10004
- The Packet Forwarding Engine is based on Triton (BT). 2 ASICs located on each line card
- Each BT ASIC uses HBM memory devices for data and table storage.
- Each BT ASIC in the system connects to the Switching Fabric either directly or through retimers
- BT ASICs on a line card are initialized and controlled by the local CPU of the card



1LC 1202 Port mapping and Fabric conenctivity



2 BT Architecture

Verify FPC/PIC/PFE status:

```
amazon@az1-1-co-cor-ptx10k-re0> show chassis fpc
                     Temp  CPU Utilization (%)   CPU Utilization (%)  Memory    Utilization (%)
Slot State            (C)  Total  Interrupt      1min   5min   15min  DRAM (MB) Heap     Buffer
  0  Online           61      4          0        6      6      6     32768      25          0
  1  Online           52      1          0        1      1      1     32768      25          0

{master}
amazon@az1-1-co-cor-ptx10k-re0> show chassis fpc pic-status
Slot 0   Online       JNP10K-LC1202
  PIC 0  Online       JNP10K-4Q56DD-32Q28-LZ-LC-PIC
Slot 1   Online       JNP10K-LC1202
  PIC 0  Online       JNP10K-4Q56DD-32Q28-LZ-LC-PIC

amazon@az1-1-co-cor-ptx10k-re0> show chassis fpc pfe-instance all

FPC 0
PFE-Instance      PFE-State
   0              ONLINE
   1              ONLINE

FPC 1
PFE-Instance      PFE-State
   0              ONLINE
   1              ONLINE
```

Switching Fabric

- Switching Fabric consists of up to six SIB boards, each with two ZF ASICs.
- Every BT ASIC in the system is connected to every ZF ASIC, resulting in single-hop connectivity from any BT to any other BT
- Switch fabric is configured and controlled by Master CB in the system

Verify SIB status:

```
{master}
amazon@az1-1-co-cor-ptx10k-re0> show chassis sibs detail
Slot 0 information:
  State                               Online
  Uptime                              38 days, 22 hours, 41 minutes, 17 seconds
  Fabric links                        Active
  Errors                              None
Slot 1 information:
  State                               Online
  Uptime                              38 days, 22 hours, 41 minutes, 16 seconds
  Fabric links                        Active
  Errors                              None
Slot 2 information:
  State                               Online
  Uptime                              38 days, 22 hours, 41 minutes, 13 seconds
  Fabric links                        Active
  Errors                              None
Slot 3 information:
  State                               Online
  Uptime                              38 days, 22 hours, 41 minutes, 11 seconds
  Fabric links                        Active
  Errors                              None
Slot 4 information:
  State                               Online
  Uptime                              38 days, 22 hours, 41 minutes, 12 seconds
  Fabric links                        Active
  Errors                              None
Slot 5 information:
  State                               Online
  Uptime                            38 days, 22 hours, 41 minutes, 13 seconds
  Fabric links                      Active
  Errors                            None
```

SOFTWARE INSTALLATION AND MAINTENANCE

Multiple Software versions

Junos EVO stores multiple versions of software on the drive, for rollback and recovery, show system software list displays all the installed software. Request system software rollback allows to roll back to a previous version stored in the system:

```
amazon@az1-1-co-cor-ptx10k-re0> show system software list
<..>
-------------------------------
node: re0
-------------------------------
Active boot device is primary: /dev/sda
List of installed version(s) :

    '-' running version
    '>' next boot version after upgrade/downgrade
    '<' rollback boot version
    '*' deleted JSU version

 -   junos-evo-install-ptx-x86-64-23.2R1-S2.6-EVO - [2024-02-19 02:38:10]
 <   junos-evo-install-ptx-x86-64-22.4R2-S2.4-EVO - [2024-02-14 17:08:53]
-------------------------------
node: re1
-------------------------------
Active boot device is primary: /dev/sda
List of installed version(s) :

    '-' running version
    '>' next boot version after upgrade/downgrade
    '<' rollback boot version
    '*' deleted JSU version

 -   junos-evo-install-ptx-x86-64-23.2R1-S2.6-EVO - [2024-02-19 02:39:08]
 <   junos-evo-install-ptx-x86-64-22.4R2-S2.4-EVO - [2024-02-14 17:35:21]
```

Each software image stores the configuration that was running when the image was active

Previous images consume disk space, to delete all but the current and the rollback versions of the software, use the 'request system software delete archived' command.

Software synchronization with dual RE systems

Junos OS Evolved ensures that all nodes in a system are running the same software version. If a new RE is inserted and is running the same code as the master RE, the configuration and other software versions are synchronized automatically. If the new RE has a different version, it is kept from joining the system, and an alarm is generated until a manual sync is done:

```
user@host-re0> show system alarms
2 alarms currently active
Alarm time Class Description
2021-04-19 16:02:26 PDT Major Re1 Node unreachable
2021-04-19 16:04:46 PDT Major Software Version Mismatch on re1:junos-evo-install-ptx-x86-64-20.4R2.6-EVO
```

Automatic synchronization can be done if the ‘`set system auto-sw-sync enable`’ knob is used. With this configuration, the system detects the RE and synchronizes the images. A reboot is performed on the new RE. To synchronize manually, use the ‘`request system software sync all-versions`’ command.

BASIC HEALTH CHECK

These are a collection of very basic commands to validate the health of a system which should not take more than a couple of minutes.

Command overview

- show system alarms and errors
- show platform commands
- show chassis commands
- show system core-dumps
- DDOS violations
- show pfe statistics traffic
- message and journal logs
- xinetd app verification

`show system alarms and errors`

Ensure there are no Minor or Major alarms which could potentially impact operations. Also, the system errors inactive details are key since it will list historical events since the system reboot. Check if there are any major events in the past or if events are increasing upon second check. Below the Minor Alarms have no operational impact. If Disk Usage is exceeded you need to clear up files.

```
labroot@strawberry-re0> show system alarms
6 alarms currently active
Alarm time               Class  Description
2021-11-11 18:56:31 CET  Minor  FPC 0 Secure boot disabled or not enforced
2021-11-11 18:57:22 CET  Minor  port-1/0/4: Optics does not support configured speed
2021-11-11 18:52:48 CET  Minor  RE 0 Secure boot disabled or not enforced
2021-11-11 18:52:48 CET  Minor  Host 0 Active Disk Usage Exceeded
2021-11-11 18:52:45 CET  Minor  RE 1 Secure boot disabled or not enforced
2021-11-11 18:52:45 CET  Minor  Host 1 Active Disk Usage Exceeded

> show system errors active
System Active Errors Information
CB 0
----------------------------------
Active Minor Errors           : 0
Active Major Errors           : 0
Active Fatal Errors           : 0
CB 1
----------------------------------
Active Minor Errors           : 0
Active Major Errors           : 0
Active Fatal Errors           : 0
<..>

> show system errors inactive
System Inactive Errors Information
CB 0
----------------------------------
Inactive Minor Errors           : 0
Inactive Major Errors           : 0
Inactive Fatal Errors           : 0
CB 1
----------------------------------
Inactive Minor Errors           : 0
Inactive Major Errors           : 0
Inactive Fatal Errors           : 0
```

```
<..>

```

show platform commands

Make sure all Applications and dependency-states are in OK Status.

```
abroot@strawberry-re0> show platform application
Application                       Status
accountd                          ok
aft-sysinfo                       ok
aft-sysinfo                       ok
agentd                            ok
aggd                              ok
alarm-mgmtd                       ok
alarmd                            ok
arpd                              ok
bios-manager                      ok
charonctl                         ok
charonctl                         ok
clksyncd                          ok
clksynced                         ok
clockd                            ok
<..>
```

Here you see the dependency state with errors, this should be investigated.

```
{master}
labroot@strawberry-re0> show platform dependency-state summary

Dependency state summary :
  Application           Node    Context     Status
  aft-sysinfo           fpc0    all         OK
  clockd                fpc0    all         OK
  evo-aftmand-bt        fpc0    all         ERROR
  evo-cda-bt            fpc0    all         OK
  evoaft-jvisiond-bt    fpc0    all         OK
  fabspoked-pfe         fpc0    all         OK
  fpa                   fpc0    all         OK
  hwdfpc                fpc0    all         OK
  msvcsd                fpc0    all         OK
  pci-agent             fpc0    all         OK
  picd                  fpc0    all         OK
  relay-eventd          fpc0    all         OK
  resiliencyd           fpc0    all         OK
  securityd             fpc0    all         OK
  timingd-lc            fpc0    all         OK
  aft-sysinfo           fpc1    all         OK
  clockd                fpc1    all         OK
  evo-aftmand-bt        fpc1    all         ERROR
  evo-cda-bt            fpc1    all         OK
  evoaft-jvisiond-bt    fpc1    all         OK
  fabspoked-pfe         fpc1    all         OK
  fpa                   fpc1    all         OK
  hwdfpc                fpc1    all         OK
```

show system core-dumps

There should be no coredumps reported and if there are stale historical data clean them out. IF there are coredumps a case should be opened for investigation.

```
{master}
labroot@strawberry-re0> show system core-dumps
re0:
--------------------------------------------------------------------------
```

```
-rw-r--r--  1 root  root   642958435 Nov 11 16:29 /var/core/fpc1/evo-aftmand-
bt.fpc_x86_64.fpc1.17447.2021_11_11.16_25_43.tar.gz
-rw-r--r--  1 root  root   987481432 Nov 8  17:06 /var/core/re0/rpd.re.re0.17917.2021_11_08.17_03_55.tar.gz
total files: 5

re1:
--------------------------------------------------------------------------
-rw-r--r--  1 root  root   349872804 Nov 5  15:48 /var/core/re1/rpd.re.re1.24364.2021_11_05.15_47_10.tar.gz
-rw-r--r--  1 root  root   1023936779 Nov 8  15:15 /var/core/re1/rpd.re.re1.30090.2021_11_08.15_12_22.tar.gz
total files: 2
```

DDOS violations

Review all the ddos protocol violations with `show ddos-protection protocols violations`. Depending on network conditions and events, some may be considered expected.

```
labroot@strawberry-re0> show ddos-protection protocols violations
Packet types: 101, Currently violated: 1

Protocol    Packet      Bandwidth  Arrival   Peak      Policer bandwidth
group       type        (pps)      rate(pps) rate(pps) violation detected at
ldp         ldp-hello   5000       4.0       57493.0   2023-10-29 08:55:22 CET
```

show pfe statistics traffic

This is a very high-level overview and can be collected system wide and fpc specific and the elements to watch out

- Data error which is a summary of all trap stats that are considered not expected.
- Info cell drops which is a sign of lookup oversubscription.
- Hardware input drops can be a sign of control path congestion
- Fabric drops can indicate fabric oversubscription or errors.

Verify that xinetd is running for SSH/connection attempts failing

From shell (start shell):

```
systemctl status xinetd-external.service
systemctl status xinetd.service
```

```
$ systemctl status xinetd-external.service

* xinetd-external.service - Xinetd Server to Launch WAN Management Services
   Loaded: loaded (/etc/systemd/system/xinetd-external.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2024-02-19 02:44:11 EST; 1 months 0 days ago
 Main PID: 14780 (xinetd-external)
    Tasks: 2
   Memory: 21.9M
   CGroup: /system.slice/xinetd-external.service
           |-14780 /bin/bash /usr/evo/xinetd-external.sh -dontfork
           `-14831 /usr/sbin/xinetd -f /etc/xinetd-external.conf -stayalive -pidfile /var/run/xinetd-external.pid
-dontfork

$ systemctl status xinetd.service

* xinetd.service - Xinetd A Powerful Replacement For Inetd
   Loaded: loaded (/etc/systemd/system/xinetd.service; static; vendor preset: enabled)
   Active: active (running) since Fri 2024-03-01 03:33:23 EST; 2 weeks 5 days ago
 Main PID: 3169 (xinetd)
    Tasks: 1
   Memory: 864.0K
   CGroup: /system.slice/xinetd.service
           > 3169 /usr/sbin/xinetd -stayalive -dontfork -pidfile /var/run/xinetd.pid
```

How to monitor connection attempts:

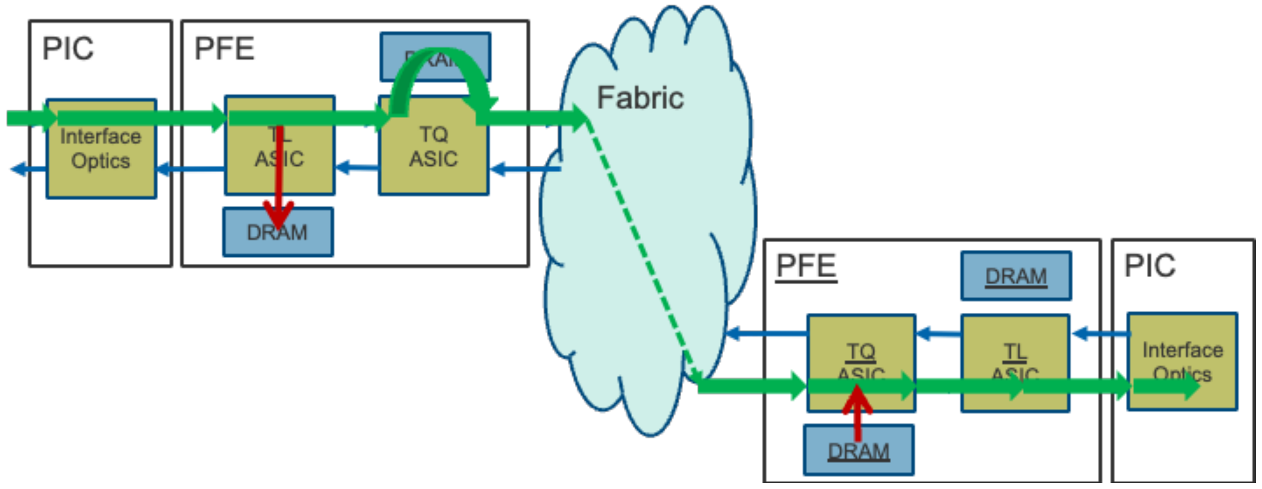
From shell (start shell):

```
journalctl -f -u xinetd-external
```

```
$ journalctl -f -u xinetd-external
-- Logs begin at Wed 2024-02-14 17:16:07 EST. --
Mar 20 21:12:44 az1-1-co-cor-ptx10k-re0 sshd[10102]: Accepted keyboard-interactive/pam for ccl from 10.10.192.16
port 40032 ssh2
Mar 20 21:12:44 az1-1-co-cor-ptx10k-re0 sshd[10102]: pam_unix(sshd:session): session opened for user ccl by
(uid=0)
Mar 20 22:12:44 az1-1-co-cor-ptx10k-re0 sshd[4803]: Accepted keyboard-interactive/pam for ccl from 10.10.192.16
port 44630 ssh2
Mar 20 22:12:44 az1-1-co-cor-ptx10k-re0 sshd[4803]: pam_unix(sshd:session): session opened for user ccl by (uid=0)
Mar 20 23:12:45 az1-1-co-cor-ptx10k-re0 sshd[32091]: Accepted keyboard-interactive/pam for ccl from 10.10.192.16
port 48936 ssh2
```

TRAFFIC FLOW TROUBLESHOOTING

In a chassis based platform such as the PTX10004, traffic has to follow a more complex path from ingress to egress. The diagram below shows an overview of this path which consists of Ingress PFE – Fabric(Sibs) – Egress PFE (which can reside in the same linecard). Troubleshooting packet loss requires additional steps.



Chassis health:

A bad FPC, a PFE (ASIC) with errors, or a Sib with link errors for example could affect traffic for multiple destinations. A basic health check can discard those problems.

- FPC is online:
  ```
  show chassis fpc
  ```
- SIB is online:
  ```
  show chassis sibs
  ```
- PIC is online:
  ```
  show chassis pic-status fpc <>
  ```
- Interfaces up and no L1 errors:
  `show interfaces et-x/y/z extensive` - Carrier transitions, CRC/Align errors, FEC uncorrected errors, etc.
- No Chassis/Fabric Alarms seen in:
  ```
  show chassis alarms
  show chassis fabric topology
  show chassis fabric link all
  show chassis fabric fpcs
  ```
- No errors reported by:
  ```
  show system errors {active | inactive} detail fpc <slot>
  ```
- `Check logs for any strange error messages:`
  ```
  show log messages
  ```

COS related drops

Indicate that there is a congested interface.

- Identify ingress and egress interface for the traffic

```
show route <IP>
show arp no-resolve
show ethernet-switching table
show route forwarding-table destination <ip/mac>
```

- Verify if CoS drop counters are non-zero:

  `show interfaces extensive <interface>` - Look for ‘Queue counters’:

```
Queue counters:       Queued packets  Transmitted packets      Dropped packets
0                                  0                    0                    0
1                                  0                    0                    0
2                                  0                    0                    0
3                              95582                95582                    0
```

```
show interfaces queue <interface>
```

All ports in a PIC/group of ports are down or dropping traffic?

Check the status of the PFE by logging to the corresponding FPC:

```
root@az1-1-co-cor-ptx10k-fpc0:pfe> show pfe id info
First Active PFE            : 0
Effective First Active PFE  : 0
Number of Active PFEs       : 2
Pfe Mask                    : 0x0000000000000003

PFEInst   Down-Flags   PPFE-ID    PFE-State   PFE-Name
------   ----------   -------------   ---------   --------
   0        0            0           ONLINE    /Chassis[0]/Fpc[0]/Pfe[0]
   0        0            1           ONLINE    /Chassis[0]/Fpc[0]/Pfe[1]
   1        0            2           ONLINE    /Chassis[0]/Fpc[0]/Pfe[2]
   1        0            3           ONLINE    /Chassis[0]/Fpc[0]/Pfe[3]
```

Not sure if traffic is reaching the device?

Transit traffic (passing through the box) can’t be captured with TCPDump/monitor traffic. The only way is via firewall filters with ‘log’ and ‘count’ actions, port mirroring, or sampling.
One example of a firewall filter to count DHCP IPv4 traffic:

```
set firewall family inet filter LOG_DHCP interface-specific ##Creates one firewall instance per
interface/direction
set firewall family inet filter LOG_DHCP term 1 from protocol udp
set firewall family inet filter LOG_DHCP term 1 from source-port 67
set firewall family inet filter LOG_DHCP term 1 from source-port 68
set firewall family inet filter LOG_DHCP term 1 from destination-port 67
set firewall family inet filter LOG_DHCP term 1 from destination-port 68
set firewall family inet filter LOG_DHCP term 1 then count DHCP
set firewall family inet filter LOG_DHCP term 1 then log
set firewall family inet filter LOG_DHCP term 1 then accept
set firewall family inet filter LOG_DHCP term ACCEPT-ALL then accept
```

Then apply to ingress/egress interface accordingly

```
Set interfaces et-x/y/z unit A family inet filter input|output LOG_DHCP
```

To see the results after commit:

```
show firewall log detail
show firewall
```

CHECK IF A ROUTE IS PROGRAMMED IN AFT

Initially, a mismatch in platform dependency in aft or cda related processes would indicate a problem of this sort and an alarm would be raised, verify with:

```
amazon@az1-1-co-cor-ptx10k-re0> show system alarms
<..>
amazon@az1-1-co-cor-ptx10k-re0> show platform dependency-state summary
<..>
amazon@az1-1-co-cor-ptx10k-re0> show platform dependency-state summary | match "aft|cda"
  aft-sysinfo          fpc0    all        OK
  evo-aftmand-bt       fpc0    all        OK
  evo-cda-bt           fpc0    all        OK
  evoaft-jvisiond-bt   fpc0    all        OK
  aft-sysinfo          fpc1    all        OK
  evo-aftmand-bt       fpc1    all        OK
  evo-cda-bt           fpc1    all        OK
  evoaft-jvisiond-bt   fpc1    all        OK
```

If something like this is seen a case should be raised for investigation. To do some manual verification, we need to find if the route points to the correct next hop index. Then if the token associated with that route contains the correct list of outgoing interfaces (OIFs):

For example, we use this route:

```
amazon@az1-1-co-cor-ptx10k-re0> show route

inet.0: 119 destinations, 120 routes (116 active, 0 holddown, 3 hidden)
+ = Active Route, - = Last Active, * = Both

0.0.0.0/0          *[BGP/170] 4d 20:59:41, localpref 100
                      AS path: 64733 I, validation-state: unverified
                    >  to 10.114.120.175 via ae8.0
```

Log in to PFE shell (use the FPC reporting the alarm, if no alarm check on all FPCs):

```
amazon@az1-1-co-cor-ptx10k-re0> start shell pfe network fpc0
Trying 128.0.0.16...
Connected to fpc0.
Escape character is '^]'.

root@az1-1-co-cor-ptx10k-fpc0:pfe>
```

Find the table index (0 for inet.0, or different for routing-instances) and dump all IPv4 routes. You can get the next hop ID and Token from each route from this output, however if the table is large the output will also be large:

```
root@az1-1-co-cor-ptx10k-fpc0:pfe> show route table proto ip

IPv4 Route Tables:
```

```
Table Name                       Index   Routes   Size(b)   Token
-------------------------------- ------- -------- --------- -------
default.0                          0         353     42360    1234
__juniper_private1__.1           1             0         0    1266
__master.anon__.50               50            5       600    1328
:vxlan.54                        54          114     13680    2185
mgmt_junos.36738                 36738         5       600    1293

Find route entries

root@az1-1-co-cor-ptx10k-fpc0:pfe> show route proto ip

Index Destination                     NH Id     NH Type   NH Token  GUID
----- ------------------------------- --------- --------- --------- --------
0      default                          53435     software  11542     1013612636366
0      0.0.0.0                         34        discard   1229      906238099527
0      1.1.10.0/31                     9041      resolve   6072      867583396901
0      1.1.10.1                        36        reject    1231      1013612636133
0      1.1.11.0/31                     9043      resolve   6080      867583396907
0      1.1.11.1                        36        reject    1231      1013612636132
0      1.1.12.0/31                     9045      resolve   6087      867583396913
0      1.1.12.1                        36        reject    1231      10136126361310      10.85.158.128/25
1005      resolve   1215      670014899091
<..>
```

Another option is to find the next hop ID and Token for a specific prefix (index 0 corresponds to the routing-instance) and verify the next-hop topology (output may change depending on the type of next hop, i.e single port, ae, multipath, etc.):

```
root@az1-1-co-cor-ptx10k-fpc0:pfe> show route proto ip index 0 prefix 0/0

Index Destination                     NH Id     NH Type   NH Token  GUID
----- ------------------------------- --------- --------- --------- --------
0      default                         53435     software  11542     1013612636366


root@az1-1-co-cor-ptx10k-fpc0:pfe> show nh recursive index 53435
53435(software, Protocol:ipv4, Ifl:13185 ae8.0, Token:11542)
  1208(unicast, Protocol:ipv4, Ifl:13185 ae8.0, Token:10954)

Next hop details:

root@az1-1-co-cor-ptx10k-fpc0:pfe> show nh recursive index 53435
53435(software, Protocol:ipv4, Ifl:13185 ae8.0, Token:11542)
  1208(unicast, Protocol:ipv4, Ifl:13185 ae8.0, Token:10954)
```

Get token details for the next hop (nh token from above):

```
root@az1-1-co-cor-ptx10k-fpc0:pfe> show sandbox token 11542
 AftNode details:
 AftIndirect token:11542 group:0 tag:None nodeMask:Default indirect:10954 hwInstall:0 proto: index:2

 JexprHandle details: Handle Type : JexprHandleNh, pfeMask: 0x3, progMode: global
NH type: Indirect, proto: 0, hwInstall: 0, rootPfe: 0
NHAttr:
   nhType: default
Internal handle type: indirect

Nexthop Entries:

Instance 255:
handle 0x7fc5a9d6d240, flags 0xc00, refcount 2
NH installed at addr NH_ALIGN 0x88ac, INT_SEQ 0x10044562 segment 2, index 2220
Raw dump of the nh words of size 2 words
        0x100000a2 0x040214b8
SEQ [100000a2]  Interm, SIZE 2, NO_ACT 0, NEXT_ADDR: 00014, SZ: 2

    ACT: [1][040214b8]  EgNHId: EG_NHID: 0214b8
    NH installed at addr NH 0x14/2
    segment 0, index 20
    SEQ [40000081] Eq_List, SIZE 2, FIN 0, BASE ADDR/SZ 0x10/1, HASH 3  segment 0, index 16
MASK: SZ 1, SEL 1, PTR/MASK 0x81 OFST 0
     EQ-List Mask Words via mask pointer [mask/size]: 0xf0000000/4

        NH 0 installed at addr NH 0x10/1
        segment 0, index 16
        SEQ [20010220]   Final, SIZE 1, TYPE 0002, EGPRT_VAL 1, EGPORT 0220, VPFE 011, GRPID 00,

        NH 1 installed at addr NH 0x11/1
        segment 0, index 17
        SEQ [20010221]   Final, SIZE 1, TYPE 0002, EGPRT_VAL 1, EGPORT 0221, VPFE 011, GRPID 01,

        NH 2 installed at addr NH 0x12/1
        segment 0, index 18
        SEQ [20010222]   Final, SIZE 1, TYPE 0002, EGPRT_VAL 1, EGPORT 0222, VPFE 011, GRPID 02,

        NH 3 installed at addr NH 0x13/1
        segment 0, index 19
        SEQ [20010223]   Final, SIZE 1, TYPE 0002, EGPRT_VAL 1, EGPORT 0223, VPFE 011, GRPID 03,
```

If the port is an ae, get the list of child interfaces:

```
root@az1-1-co-cor-ptx10k-fpc0:pfe> show interfaces ae8
 Name: ae8           Index: 1237   GUID: 867583397764 IflCount: 1 Type: 29        Weight: 0

 CfgState:     Up    OverallState: Up               GlobalSlot: 255  PfeInst: 255
                     IsAggregate:  Yes              MTU:  1514       PfeId:   255
 LinkState:    Up                                   Pic:  255        PicPort: 0
                                                    StatsMgr Map: Present
 ChannelCount: 0                                    IfdSpeed:     400000000000

 Flags:           0x0000000000000000
 SpecificFlags:   0x0000000000000000
 LinkProtection:  Disabled
 MacAddress:      40:7f:5f:09:24:b1
 InitTime:        Thu Mar 21 18:05:42 2024
 StateChangeTime: Thu Mar 21 18:06:02 2024
 GE Flags:
 IfdToken: 9647, stream: 4294967295, dp Inst: 4294967295, pp slice: 4294967295, voq: 0, OIF token:
18446744073709551615, active token: 18446744073709551615, dmac reject token: 18446744073709551615, active child
counts: 4

 802.1BR Extended Port:   EC-ID: 43624    Satellite-Id: 38847

 Aggregate member list
 IfdIndex     Name                  Weight    State    ActiveToken    LinkRole
 1238         et-0/0/0              1         Up       10957         Active
 1239         et-0/0/1              1         Up       10958         Active
 1240         et-0/0/2              1         Up       10959         Active
 1241         et-0/0/3              1         Up       10960         Active
```

Check each individual link and get OIF token:

```
root@az1-1-co-cor-ptx10k-fpc0:pfe> show interfaces et-0/0/0
 Name: et-0/0/0      Index: 1238   GUID: 867583397766 IflCount: 1 Type: 223       Weight: 1

 CfgState:     Up    OverallState: Up               GlobalSlot: 0    PfeInst: 0
                     IsAggregate:  No               MTU:  1514       PfeId:   1
 LinkState:    Up                                   Pic:  0          PicPort: 0
                                                    StatsMgr Map: Present
 ChannelCount: 0                                    IfdSpeed:     100000000000

 Flags:           0x0000000000000000
 SpecificFlags:   0x0000000000000000
 LinkProtection:  Disabled
 MacAddress:      40:7f:5f:09:24:b1
 InitTime:        Thu Mar 21 18:05:42 2024
 StateChangeTime: Thu Mar 21 18:06:02 2024
 GE Flags:
 IfdToken: 10902, stream: 48, dp Inst: 1, pp slice: 0, voq: 544, OIF token: 10901, active token: 10957, dmac
reject token: 10903, active child counts: 0

 802.1BR Extended Port:   EC-ID: 43624    Satellite-Id: 38847
```

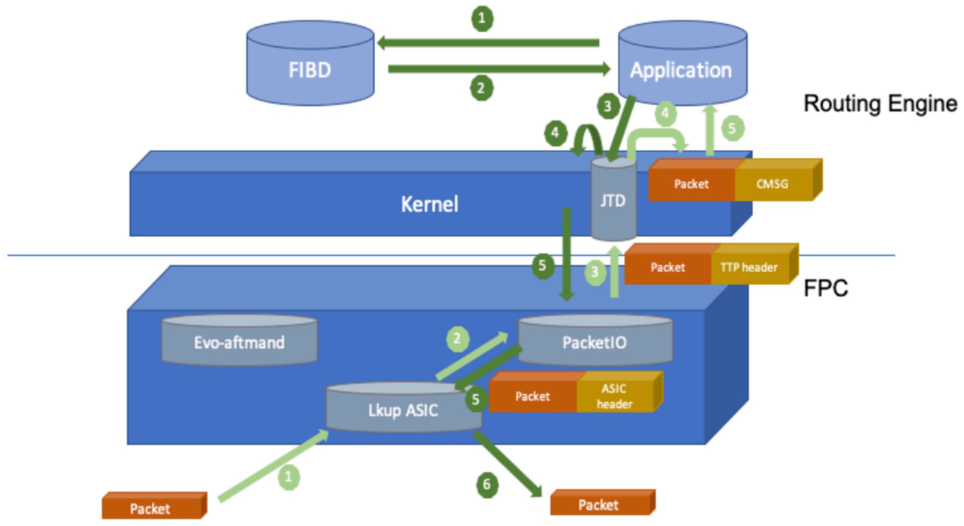
Get the token information for each link, EGPORT and VPFE should match the list from above (next-hop token):

```
root@az1-1-co-cor-ptx10k-fpc0:pfe> show sandbox token 10901
 AftNode details:
 AftExprNhOIF token:10901 group:0 tag:guid(if) TagIndex:867583397766
 nodeMask:Default vpfe:17 oqGroup:0

 JexprHandle details: Handle Type : JexprHandleNh, pfeMask: 0x3, progMode: global
NH type: OIF, proto: 0
NHAttr:
   nhType: default

Nexthop Entries:

Instance 255:
handle 0x7fc5a678b1c0, flags 0xc00, refcount 1
NH not installed
Raw dump of the nh words of size 1 words
        0x20010220
SEQ [20010220]   Final, SIZE 1, TYPE 0002, EGPRT_VAL 1, EGPORT 0220, VPFE 011, GRPID 00,



From previous output:

root@az1-1-co-cor-ptx10k-fpc0:pfe> show sandbox token 11542
 AftNode details:
 AftIndirect token:11542 group:0 tag:None nodeMask:Default indirect:10954 hwInstall:0 proto: index:2
<..>
        NH 0 installed at addr NH 0x10/1
        segment 0, index 16
        SEQ [20010220]   Final, SIZE 1, TYPE 0002, EGPRT_VAL 1, EGPORT 0220, VPFE 011, GRPID 00,

        NH 1 installed at addr NH 0x11/1
        segment 0, index 17
        SEQ [20010221]   Final, SIZE 1, TYPE 0002, EGPRT_VAL 1, EGPORT 0221, VPFE 011, GRPID 01,

        NH 2 installed at addr NH 0x12/1
        segment 0, index 18
        SEQ [20010222]   Final, SIZE 1, TYPE 0002, EGPRT_VAL 1, EGPORT 0222, VPFE 011, GRPID 02,

        NH 3 installed at addr NH 0x13/1
        segment 0, index 19
        SEQ [20010223]   Final, SIZE 1, TYPE 0002, EGPRT_VAL 1, EGPORT 0223, VPFE 011, GRPID 03,
```

CONTROL TRAFFIC AND PROTOCOL TROUBLESHOOTING

This applies to protocols flapping, pings not completing, arp not being resolved, etc.

Path of a control packet in EVO(RX):



Normal CLI and protocol-related commands still apply (i.e show bgp .*, show arp no-resolve, etc.)

Commands to monitor for internal packet drops:

- show system statistics ttp
- show system statistics jtd
- show ddos-protection protocols statistics terse

Monitor traffic from CLI helps capture control packets, examples for ARP and BGP:

```
> monitor traffic interface ae2.0 matching arp no-resolve layer2-headers

NOTE: MAC Addresses 00:00:00:00:00:00 are used when L2 header information in not available. For such packets, L2
headers are added by PFE when transmit and removed before being punted to RE
Local vib interface has IP 128.0.0.4.
reading from file -, link-type EN10MB (Ethernet)
16:02:09.083979 40:7f:5f:09:24:ad > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 120: Request who-has
10.1.202.0 tell 10.1.202.1, length 106
16:02:09.084807 64:c3:d6:62:2c:a4 > 40:7f:5f:09:24:ad, ethertype ARP (0x0806), length 156: Reply 10.1.202.0 is-at
64:c3:d6:62:2c:a4, length 142

> monitor traffic interface ae3.0 no-resolve layer2-headers matching "tcp port 179" detail

NOTE: MAC Addresses 00:00:00:00:00:00 are used when L2 header information in not available. For such packets, L2
headers are added by PFE when transmit and removed before being punted to RE
Local vib interface has IP 128.0.0.4.
reading from file -, link-type EN10MB (Ethernet)
16:03:38.627056 00:00:00:00:00:00 > 00:00:00:00:00:00, ethertype IPv4 (0x0800), length 165: (tos 0xc0, ttl 1, id
3189, offset 0, flags [none], proto TCP (6), length 75)
    10.1.201.21.179 > 10.1.201.20.64757: Flags [P.], cksum 0xcb1f (correct), seq 2723773171:2723773194, ack
696020397, win 32620, options [nop,nop,TS val 534799263 ecr 876614896], length 23: BGP
        Route Refresh Message (5), length: 23
          AFI IPv4 (1), SAFI Unicast (1)
16:03:38.732669 00:00:00:00:00:00 > 00:00:00:00:00:00, ethertype IPv4 (0x0800), length 148: (tos 0xc0, ttl 1, id
7916, offset 0, flags [none], proto TCP (6), length 52)
    10.1.201.20.64757 > 10.1.201.21.179: Flags [.], cksum 0xb176 (correct), ack 23, win 16384, options [nop,nop,TS
val 876639524 ecr 534799263], length 0
```

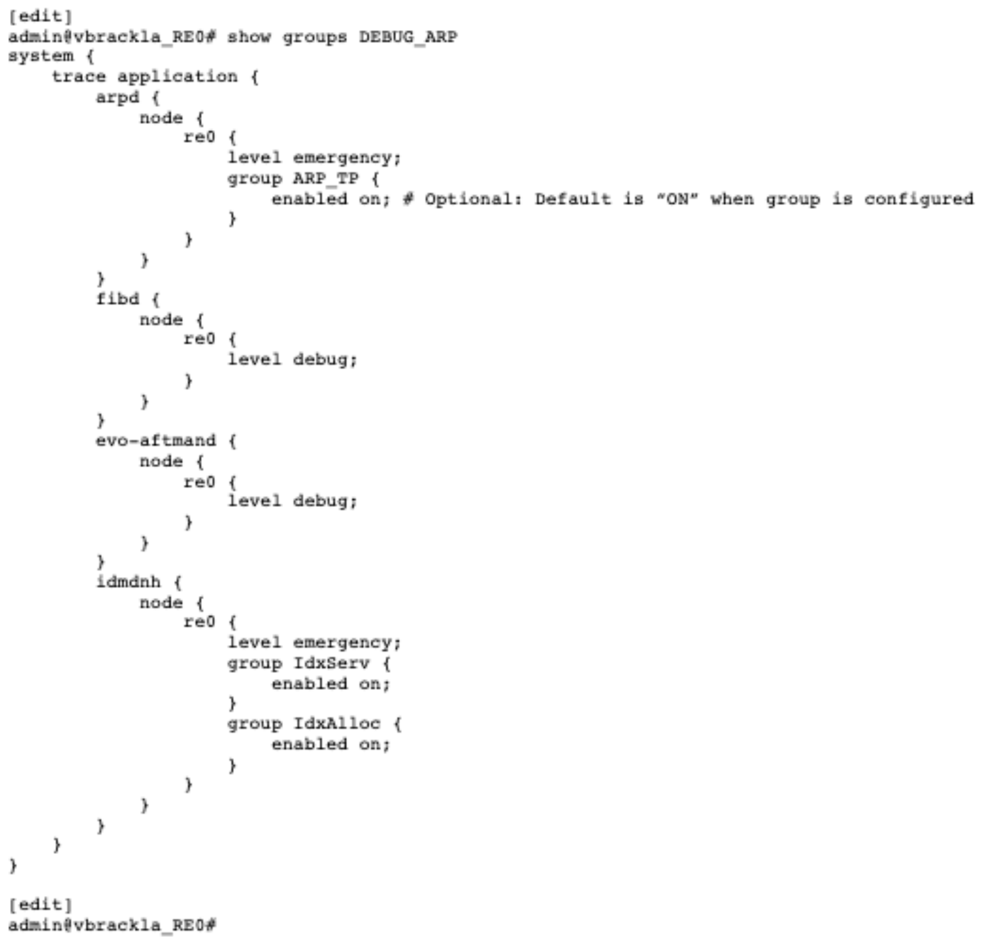
```
16:03:38.830437 00:00:00:00:00:00 > 00:00:00:00:00:00, ethertype IPv4 (0x0800), length 459: (tos 0xc0, ttl 1, id
7928, offset 0, flags [none], proto TCP (6), length 363)
    10.1.201.20.64757 > 10.1.201.21.179: Flags [P.], cksum 0x8ce8 (correct), seq 1:312, ack 23, win 16384, options
[nop,nop,TS val 876639626 ecr 534799263], length 311: BGP
        Update Message (2), length: 82
          Origin (1), length: 1, Flags [T]: IGP
          AS Path (2), length: 10, Flags [T]: 65201 65101
          Next Hop (3), length: 4, Flags [T]: 10.1.201.20
          Updated routes:
            10.1.1.102/32
            10.1.1.101/32
            10.1.101.20/31
            10.1.102.0/31
            10.100.12.0/31
            10.1.101.10/31
            10.100.11.0/31
```

‘matching’ filter can be used with similar syntax as tcpdump. Using tcpdump is also an option from shell.

TRACE CONFIGURATION EXAMPLE

ARP Processing:

- Arpd is the App handling ARP processing.
- Fibd learns the host route after the Arp resolution.
- Evo-aftmand consumes the Arp related objects and install the host route into the forwarding table under the PFE.
- Idmdnh handles the nexthop index allocation



```
[edit]
admin@vbrackla_RE0# show groups DEBUG_ARP
system {
    trace application {
        arpd {
            node {
                re0 {
                    level emergency;
                    group ARP_TP {
                        enabled on; # Optional: Default is "ON" when group is configured
                    }
                }
            }
        }
        fibd {
            node {
                re0 {
                    level debug;
                }
            }
        }
        evo-aftmand {
            node {
                re0 {
                    level debug;
                }
            }
        }
        idmdnh {
            node {
                re0 {
                    level emergency;
                    group IdxServ {
                        enabled on;
                    }
                    group IdxAlloc {
                        enabled on;
                    }
                }
            }
        }
    }
}

[edit]
admin@vbrackla_RE0#
```