CARRIER GRADE NAT IMPLEMENTATION GUIDE

Table of Contents

Table of Figures

Introduction

Today's Internet service providers (ISPs) have to address the problem of IPv4 depletion (IPv4-Exaustion)(IANA-Allocation) and plan for IPv6 migration and IPv6-based services to continue unrestricted growth. There are a number of technologies that can be used to solve these problems such as NAT44(4), DS-Lite, 6rd, and many others. Carrier Grade Network Address Translation (CGN) is a common building block for virtually all of the main technologies available.

Scope

This implementation guide provides an overview of NAT44(4) features, as well as configuration and design guidelines for implementing NAT44(4) with MS-PIC/MS-DPC blades in your network. The guide will also cover configuration steps, tips, and design guidelines related to operations, administration, and management for CGN when deploying NAT44(4). An in-depth look at technologies for an IPv4 exhausted world can be found at: **www.juniper.net/ipv6**.

Design Considerations

One approach to addressing the impending threat from IPv4 address space depletion is to share the remaining or available IPv4 addresses among larger numbers of customers. This is done by using CGN, which primarily pulls the address allocation to a more centralized NAT in the service provider network. CGN (sometimes known as Large Scale NAT or LSN) is a highly scalable NAT placed between the customer premises equipment (CPE) and the core of the network that implements [RFC 4787][RFC 5382][RFC 5508] and a few additional requirements discussed in [NAT444][CGN][INC-CGN][1] (see bibliography section below for links). NAT44(4) is one technology that uses CGN and helps manage the IPv4 address space depletion issue.

NAT44(4) uses three layers of IPv4 addressing:

- A private IPv4 block within the user network (behind the CPE NAT)
- A different private IPv4 block for the user to provider links (between the CPE NAT and the CGN)
- A public IPv4 address on the outside of the CGN

Figure 1 illustrates NAT44(4) architecture.

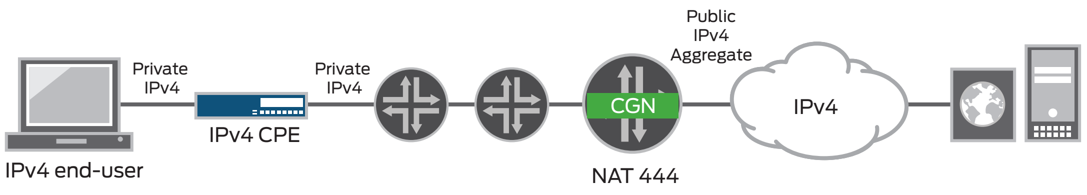

Figure 1: NAT44(4) CGN IPv4 architecture

NAT44(4) has several significant advantages from an ISP's point of view:

- There are no changes to end user hosts.
- NAT44 application-layer gateways (ALGs) can be leveraged.
- There are no changes to CPE for IPv4 services.
- Both code and deployment are mature.

Terminology

Following is the list of keywords that will be used throughout this document:

- "basic-nat44" (Reference [RFC 2663])
- "napt44" (Reference [RFC 2663])
- "stateful-nat64" (Reference [NAT64])
- "stateless-nat64" (Reference [XLATE])
- "basic-nat66" (same as basic-nat44 but for IPv6 family)
- "basic-nat-pt" (Reference [NAPT])
- "napt-pt" (Reference [NAPT])
- "twice-nat44" (Reference [RFC2663])
- "dynamic-nat44" (Dynamic SRC Address-only)
- "stateless-nat66" (Reference [NAT66])
- "napt66" (same as napt44 but for IPv6 family)

Implementation

Juniper Networks CGN Solution Overview

Juniper has an incredibly feature rich CGN implementation. Below is a summary of all NAT translation types supported in the Layer 3 package and their configurations on MS-DPC and MS-PIC on MX Series and M Series routers.

Table 1. NAT Translation Types

NAT TYPE	CONFIGURATION SUMMARY
basic-nat44	• Source NAT pool with address range/prefix. • Translation type is source static.
napt44	• Source NAT pool with address-range/prefix and port range. • Translation type is source dynamic.
basic-nat66	• NAT pool with IPv6 address-range/prefix. • Translation type is source static.
basic-nat-pt	• Source NAT pool with IPv4 address range/prefix. • Destination NAT pool with /96 prefix. • NAT match condition has IPv6 address/address range. • Translation type is source dynamic, destination static.
napt-pt	• Source NAT pool with Ipv4 address range/prefix and port range. • Destination NAT pool with /96 prefix. • NAT match condition has IPv6 address/address range. • Translation type is source dynamic, destination static.
twice-nat44	• Source NAT pool with address range/prefix. • Destination pool with address range/prefix. • Translation type is source dynamic, destination static.
dynamic-nat44	• Source NAT pool with address range/prefix. • Translation type is source dynamic.
napt66	• Source NAT pool with IPv6 address range/prefix and port range. • Translation type is source dynamic.

CGN Configuration Example

The topology for the annotated configuration is as follows.

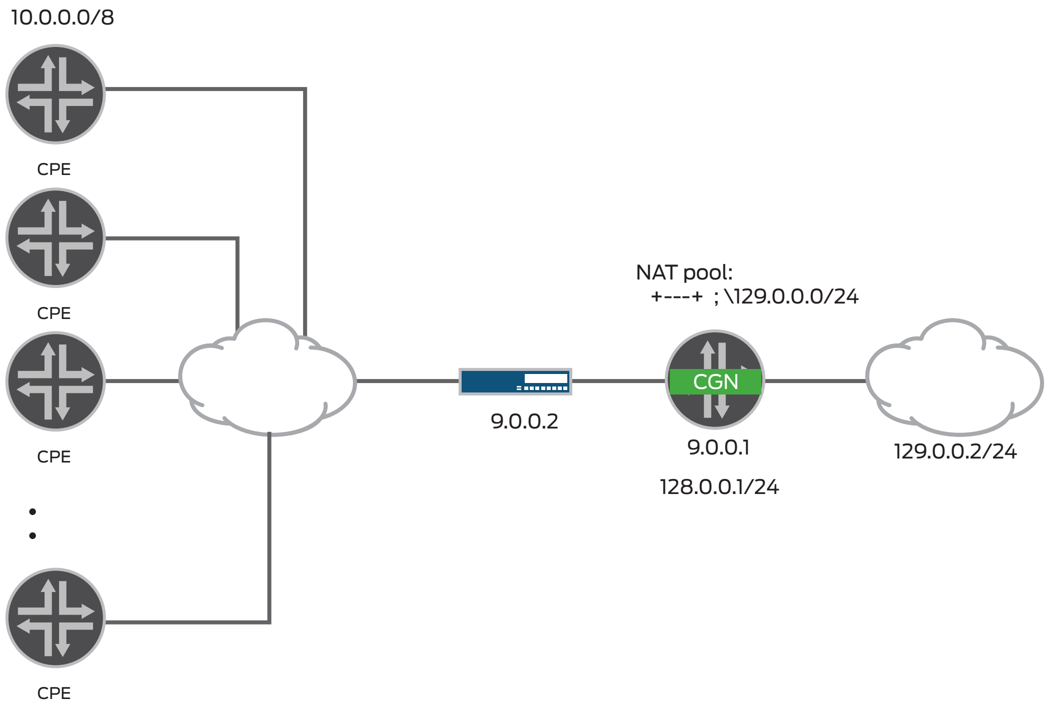

Figure 2: NAT configuration example

Chassis Configuration

Assuming the PIC is in Flexible PIC Concentrator (FPC) 5 slot 0.

```
[edit chassis]
user@router# show
fpc 5 {
    pic 0 {
        adaptive-services {
            service-package layer-3;
        }
    }
 }
```

Interfaces Configuration

```
ge-1/3/5 {
    description "Private";
    unit 0 {
        family inet {
            service {
                input {
                    service-set sset2;
                }
                output {
                    service-set sset2;
                }
            }
            address 9.0.0.1/24;
        }
    }
}
ge-1/3/6 {
    description "Public";
    unit 0 {
        family inet {
            address 128.0.0.1/24;
        }
    }
}

sp-5/0/0 {
    unit 0 {
        family inet;
    }
}
```

Network Address and Port Translation Configuration

NAPT44

The sample configuration below illustrates using the port attribute within the NAT pool and setting translation type to source dynamic enabled NAPT44:

```
user@router> show configuration services nat
pool p1 {
    address 129.0.0.0/24;
    port automatic random-allocation;
}
rule r1 {
    match-direction input;
    term t1 {
        from {
            source-address {
                10.0.0.0/16;
                10.1.0.0/16;
            }
        }
        then {
            translated {
                source-pool p1;
                translation-type {
                    source dynamic;
                }
            }
        }
    }
}
```

Port Random Allocation

Random allocation of NAT ports (used in configuration NAPT44) means that the starting point for free port search is random, and from this starting point, we search in a sequential fashion. If you don't configure random port allocation, the starting point is remembered (based on previous successful allocation) and search for free NAT port is sequential from that point onwards. In a real-world scenario, as the flows are created and closed in a random fashion (i.e., the first flow created doesn't have to be the first flow to age out), after the first round of port allocation in a NAT pool, further port allocations look random. With port automatic, we start allocating from 1024, 1025, etc. in that order.

Address Random Allocation

This address allocation method is designed to make optimum use of multiple address ranges in a NAT pool. To enable this configure "address-allocation round-robin" under the NAT pool hierarchy as described here:

```
edit services nat]
   pool napt {
        address-range low 9.9.99.1 high 9.9.99.3;
        address-range low 9.9.99.4 high 9.9.99.6;
        address-range low 9.9.99.8 high 9.9.99.10;
        address-range low 9.9.99.12 high 9.9.99.13;
        port {
            range low 3333 high 3334 random-allocation;
        }
        address-allocation round-robin;
    }
```

For every different source address, a different NAT address is allocated in a round robin fashion.

Address allocation starts with the first address in the NAT pool. In the above example it starts with address 9.9.99.1 continues through 9.9.99.3, and the next address picked would be 9.9.99.4 and this continues through 9.9.99.13 before it wraps around back to 9.9.99.1.

Address-Pooling behavior is unchanged. This means that new sessions that match an existing mapping will continue to use the same public IPv4 address.

Address-Only Dynamic NAT

If the pool does not have a port attribute configured, and the translation type is "source dynamic," then it is 1:1 address-only dynamic NAT. This means that NAT is performed on addresses only, but the 1:1 relationship between private and public addresses is established dynamically (as opposed to static NAT discussed in Static NAT below). In this sample configuration, note that the "from clause" is not mandatory for address-only source dynamic NAT. If all NAT addresses are used up, then new sessions will be dropped unless an overload pool is configured. The overload pool (if configured) is then used to perform NAPT for the new sessions:

```
pool p1 {
    address 129.0.0.0/24;
}

rule r1 {
    match-direction input;
    term t1 {
        from {
            source-address {
                10.0.0.0/16;
                10.1.0.0/16;
            }
        }
        then {
            translated {
                source-pool p1;
                translation-type {
                    source dynamic;
                }
            }
        }
    }
}
```

Static NAT

If the pool does not have a port attribute configured and the translation type is "source static," then the NAT type is 1:1 static NAT and the "from clause" is mandatory. Below is an example configuration for source static NAT. Note that the "from clause" is mandatory for static NAT, and the number of addresses in the "from clause" must be less than or equal to the number of addresses in the NAT pool:

```
pool p1 {
    address 129.0.0.0/24;
}

rule r1 {
    match-direction input;
    term t1 {
        from {
            source-address {
                10.10.10.0/24;
            }
        }
        then {
            translated {
                source-pool p1;
                translation-type {
                    source static;
                }
            }
        }
    }
}
```

Service Set Configuration

Once the NAT rules (along with other rules, if needed) are configured, a service set with all relevant rules must be configured. In case of interface style service set, the service set must be applied on the media interface. A simple service set with an NAT rule looks like this:

```
service-set sset2 {
    nat-rules r1;
    interface-service {
        service-interface sp-5/0/0;
    }
}
```

Port Block Allocation (PBA)

PBA is port allocation method that is designed to reduce the logs generated with NAT. It is configured under "nat pool" hierarchy as described here:

```
      services {
               nat {
         pool pool1 {
             address-range low 32.32.32.1 high 32.32.32.32;
             port {
                 automatic {
                     random-allocation;
                 }
                 secured-port-block-allocation {
                    block-size 256; /* Min 8, Max 64512, default 128 */
                    max-blocks-per-user 8; /* Max 2048, default 8 */
                    active-block-timeout 300; /* 0(default), Min 120secs, Max MAX_UINT */
                 }
             }
             address-allocation round-robin;
             mapping-timeout 120;
         }
     }
 }
```

With PBA, instead of allocating one port from NAT pool, a block of ports is allocated for a subscriber with private IP address and from that block, ports are assigned as and when a connection is initiated from the subscriber. Port allocation is done till the block is exhausted or till "active-block-timeout" is exceeded. Mulptiple port blocks can be assigned to a subscriber, but a port is allocated only from an active port block, and there can be only one active port block per subscriber for a configured nat pool. The limit on number of port blocks that a subscriber may be assigned to is determined by "max-blocks-per-user" and ensures that one subscriber does not exhaust all the port blocks.

Since a port block is associated to a private IP address, one log entry is generated per block of ports at the time of allocation, and another log entry is generated when all the ports in the block are freed. This method results in reduced log entries.

Since ports are being allocated from a block till the block is exhausted, port prediction can happen. So to randomize port allocation, even if ports are available in a port block, if the block activity is beyond certain configured time governed by active-block-timeout, it would result in allocating a new port block, and port would be allocated from that new port block.

A subscriber can choose to disable active-block-timeout functionality by configuring it to zero.

If the number of blocks allocated to a subscriber is 1 per natpool, then that block is tied to the subscriber and released only when the subscriber times out. This timeout is governed by mapping-timeout in the natpool. This is to avoid multiple logs, for example, when subscriber activity is limited and the webpages they browse are refreshing content without user intervention. Due to subscriber inactivity, if we release the port block, a log would be generated. While the subscriber is going through the webpage,contents can get refreshed by the webserver at some intervals, and for that a port block would need to be allocated again, and so a log will be generated. This can continue, resulting in multiple port allocation and release and thus multiple logs for that subscriber. To handles such scenarios, if the number of blocks per nat pool for that subscriber is one, then that block is released only when the subscriber timeout. This way the intermediate connections initiated by the web server, without multiple release and allocation of ports. This is achieved by configuring an appropriate "mapping-timeout"

Address Pooling and Endpoint Independent Mapping (EIM)

There is a lot of confusion around these two features and how they relate to one another. This section attempts to clarify each one in a succinct way.

Address Pooling

Address pooling means assigning the same external address for all sessions originating from the same internal host:

- It applies when you have a pool of addresses.
- It does not say anything about ports.
- It does not say which connections are going to be accepted from the outside.

Address pooling solves the problems of an application opening multiple connections. For example:

- If a Session Initiation Protocol (SIP) client is sending Real-Time Transport Protocol (RTP) and Real-Time Control Protocol (RTCP) packets, it is expected that they come from the same IP address, even after they go through NAT. Otherwise, an alternate scheme should have been negotiated beforehand. If RTP and RTCP IP addresses are different, the receiving endpoint might drop packets.
- Any point-to-point (P2P) protocol that negotiates ports (assuming address stability) will benefit from address pooling paired.

Use Cases

Instant Messaging

The chat and control sessions of some IM clients need to come from the same public source address. If they don't, the server will reject them. When the AOL Instant Messenger (AIM) client is first started, as an example, it authenticates with the AIM server to identify the user. Then when the user starts a chat window, a different session is established. If the chat session originates from a source address that is different from the authentication session, the AIM server will reject the chat session, as it is not perceived as an authenticated session.

SSL

Certain websites such as online banking require that all connections from a given host (SSL or not) come from the same IP address.

Configuration Example

Below is a sample configuration with address pooling enabled:

```
rule r1-address-pooling {
    match-direction input;
    term t1 {
        from {
            applications [junos-sip junos-rtsp];
        }
        then {
            translated {
                source-pool p1;
                translation-type {
                    source dynamic;
                }
                address-pooling paired;
            }
        }
    }
}
```

EIM

EIM means assigning the same external address and port for all connections from a given host if they use the same internal port. This means if they come from a different source port, you are free to assign a different external address.

Address polling is what requires assigning the same external IP address.

Enabling EIM means that you have a stable external IP address + port (for a period of time) that external hosts can use to connect. Note: The determination of who can connect to an internal host is done by End Point Independent Filtering (EIF), not EIM.

Configuration Example

Below is a sample configuration with EIM enabled. Note that you can have both EIM and address pooling enabled in a single rule:

```
rule r1-eim {
    match-direction input;
    term t1 {
        from {
            applications [junos-sip junos-rtsp];
        }
        then {
            translated {
                source-pool p1;
                translation-type {
                    source dynamic;
                }
                mapping-type endpoint-independent;
            }
        }
    }
}
```

In essence, address pooling solves the outbound stability problem and EIM solves the inbound stability problem.

Preserve Range and Preserve Parity

When a CGN allocates a source port for an outbound connection, it might be configured to preserve the range or parity of the packet source port. (See discussions on this topic in RFC 4787)

Although these features are not heavily used, they are discussed here because they are needed in some specific deployments or applications. You can configure preserve parity and preserve range under the NAT pool definition as follows:

```
[edit services nat pool test-pool]
user@router# show
address 203.0.113.0/24;
port {
    automatic;
    preserve-parity;
    preserve-range;
}
```

Preserve Range

RFC 4787 defines two ranges: 0 to 1,023, and 1,024 to 65,535. When the “preserve-range” knob is configured and the incoming port falls into one of these ranges, we try to allocate port from that range only. If there is no available port, we fail the port allocation request and that session is not created. The failure is reflected on counters and system logging but no Internet Control Message Protocol (ICMP) message is generated.

If this knob is not configured, we allocate from configured port range irrespective of which port range the incoming port falls in. The exception is some ALGs (like shell) that have special zones.

Preserve Parity

When the “preserve-parity” knob is configured, if incoming port number is odd, we try to allocate odd port (and same for even). If the port with desired parity is not found, the port allocation request fails, the session is not created, and the packet is dropped.

VRF Based CGN Deployments

Another powerful feature is support for VPN routing and forwarding (VRF)-aware CGN, the ability to support multiple service instances on the CGN device and associate these with multiple routing topology and/or Layer 3 VPN services. This feature provides a lot of deployment flexibility and ways to integrate with traffic engineering policies[1].

The following configuration tasks are needed for VRF based CGN deployment:

- Configure next-hop style service sets
- Configure a VRF routing instance
- Bind subscriber and MS-DPC/MS-PIC interfaces to VRF
- Configure forwarding and policy options

For this example we will consider there is 1 MS-PIC in the system and subscriber originated traffic comes through interface ge-5/2/0. The core-facing interface is ge-5/2/1.

```
user@router> show configuration services
service-set sset0 {
    nat-rules r0;
    next-hop-service {
        inside-service-interface sp-4/0/0.10;
        outside-service-interface sp-4/0/0.20;
    }
}
nat {
    pool p0 {
        address 100.0.0.0/24;
        port {
            automatic;
        }
    }
    rule r0 {
        match-direction input;
        term t0 {
            then {
                translated {
                    source-pool p0;
                    translation-type {
                        source dynamic;
                    }
                }
            }
        }
    }
```

[1] Refer to “Design Consideration for Next Generation Network Addressing Solutions” : www.juniper.net/us/en/local/pdf/whitepapers/2000434-en.pdf and IETF draft “NAT444 Deployment Options and Experiences”: http://tools.ietf.org/html/draft-kuarsingh-lsn-deployment-01”.

A VRF is created with the inside service interface and subscriber media interface.

```
user@router> show configuration routing-instances
vrf-private {
    instance-type vrf;
    interface sp-4/0/0.10;
    interface ge-5/2/0.0;
    route-distinguisher 100:1;
    vrf-import lb-policy;
    vrf-export lb-policy;
    routing-options {
        static {
            route 0.0.0.0/0 {
                next-hop [ sp-4/0/0.10 ];
                preference 0;
            }
        }
    }
}
```

The next step is to configure a policy based on ISP routing requirements. Example illustrated here:

```
policy-options {
    policy-statement lb-policy {
        term t0 {
            then {
                load-balance per-packet;
            }
        }
        }
}
```

Load-Balancing Across all MS-DPC/MS- PICs

When multiple service blades are installed in an M Series or MX Series system, traffic can be distributed or load balanced across the MS-DPC/MS-PICs in multiple ways. Some of the load balancing techniques supported are source IP address based flow distribution, Filter Based Forwarding, L3VPN based or class based forwarding.

In this section we provide configuration steps to enable source based hashing. The distribution of packets is based on source IP address to ensure traffic from a particular subscriber is handled by the same MS-PIC. Furthermore, APP, EIM and EIF functionality is maintained. Automatic redistribution is supported to ensure load-balancing is optimized if any MS-DPC/MS-PIC failure occurs.

To enable source-based hashing with next-hop style service sets, the following configuration tasks are needed:

- Configure next-hop style service sets
- Binding new SPICs interfaces to VRF
- Create a VRF route with multiple next-hops
- Configure forwarding and policy options

For this example we will consider there are 2 additional MS-PICs in the system: sp-4/2/0 and sp-4/3/0. The final services configuration is the following:

```
user@router> show configuration services
service-set sset0 {
    nat-rules r0;
    next-hop-service {
        inside-service-interface sp-4/0/0.10;
        outside-service-interface sp-4/0/0.20;
    }
}
service-set sset2 {
    nat-rules r2;
    next-hop-service {
        inside-service-interface sp-4/2/0.10;
        outside-service-interface sp-4/2/0.20;
    }
}
service-set sset3 {
    nat-rules r3;
    next-hop-service {
        inside-service-interface sp-4/3/0.10;
        outside-service-interface sp-4/3/0.20;
    }
}
nat {
    pool p0 {
        address 100.0.0.0/24;
        port {
            automatic;
        }
    }
    pool p2 {
        address 112.0.0.0/24;
        port {
            automatic;
        }
    }
    pool p3 {
        address 113.0.0.0/24;
        port {
            automatic;
        }
    }
    rule r0 {
        match-direction input;
        term t0 {
            then {
                translated {
                    source-pool p0;
                    translation-type {
                        source dynamic;
                    }
                }
            }
        }
    }
    rule r2 {
        match-direction input;
        term t2 {
            then {
                translated {
                    source-pool p2;
                    translation-type {
                        source dynamic;
                    }
                }
            }
        }
    }
    rule r3 {
        match-direction input;
        term t3 {
            then {
                translated {
                    source-pool p3;
                    translation-type {
                        source dynamic;
                    }
                }
            }
        }
    }
}
```

The final VRF configuration is as described below. Note the route with multiple next-hops points to all MS-PICs in the system to load-balance across them.

```
user@router> show configuration routing-instances
vrf-private {
    instance-type vrf;
    interface sp-4/0/0.10;
    interface sp-4/2/0.10;
    interface sp-4/3/0.10;
    interface ge-5/2/0.0;
    route-distinguisher 100:1;
    vrf-import lb-policy;
    vrf-export lb-policy;
    routing-options {
        static {
            route 0.0.0.0/0 {
                next-hop [ sp-4/0/0.10 sp-4/2/0.10 sp-4/3/0.10 ];
                preference 0;
            }
        }
    }
}
```

Next, the router needs to be configured to distribute packets based on source address. For I-chip-based DPC cards, the "hash-key" configuration hierarchy is used as shown here.

```
user@router> show configuration forwarding-options
hash-key {
    family inet {
        layer-3 {
            source-address;
        }
    }
}
```

For Trio-based MPC cards, the "enhanced-hash-key" configuration hierarchy is used:

```
user@router> show configuration forwarding-options
enhanced-hash-key {
    services-loadbalancing {
        family inet {
            layer-3-services {
                source-address;
            }
        }
    }
}
```

The last step is to enable load-balancing with the policy configured (lb-policy) in the prior steps.

```
user@router> show configuration routing-options
forwarding-table {
    export lb-new;
}
```

Logging

Logging is an important part of any CGN deployment—specifically log generation, performance, and format.

Log Generation

Currently, MS-PIC uses the system logging protocol to generate session logging. System log messages can be sent directly from the MS-PIC to an external system logging server. For this to succeed, the services PIC interface must have an IP address and appropriate system logging options configured. See example below:

```
[edit interfaces sp-5/0/0]
user@router# show
services-options {
    syslog {
        host 130.0.0.1 {
            services any;
        }
    }
}
unit 0 {
    family inet {
        address 150.0.0.1/32;
    }
}
```

Log Format

For each session, currently three logs are generated. The three logs allow correlation of start and end times for each session. Note that in case of “dynamic-source-address only NAT, the NAT pool release log is not generated.

```
Jun 28 15:29:20  cypher (FPC Slot 5, PIC Slot 0) {sset2}[FWNAT]: ASP_SFW_CREATE_ACCEPT_
FLOW: proto 6 (TCP) application: any, ge-1/3/5.0:10.0.0.1:8856 -> 128.0.0.2:80, creating
forward or watch flow ; source address and port translate to 129.0.0.1:1028

Jun 28 15:29:23  cypher (FPC Slot 5, PIC Slot 0) {sset2}[FWNAT]:ASP_NAT_POOL_RELEASE:
natpool release 129.0.0.1:1028[1]

Jun 28 15:29:23  cypher (FPC Slot 5, PIC Slot 0) {sset2}[FWNAT]: ASP_SFW_DELETE_FLOW: proto
6 (TCP) application: any, (null)(null)10.0.0.1:8856 -> 128.0.0.2:80, deleting forward or
watch flow ; source address and port translate to 129.0.0.1:1028
```

System Log Throttling

Today, logging can be controlled from the services PIC command-line interface (CLI). The following command shows the current logging configuration:

```
interfaces {
  sp-1/2/0 {
    services-options {
      syslog {
+       message-rate-limit <>    <<<
        host <host> {
          services error;
                        facility-override kernel;
                        log-prefix PREFIX;
        }
      }
    }
  }
}


This knob can be setup to rate limit the syslogs
generated from the PIC. The default is
different for local versus remote syslogs.

Local: 10000 (Existing limit today)
Remote: 200000 (Set to high value and administrator can tune down
                based on the capacity of external server).
```

System Logging Performance

This section illustrates sample logging performance numbers for an NAPT44 test using stateful HTTP traffThe total number of packets per second is about 90,000:

```
Interface    Link  Input packets        (pps)     Output packets        (pps)
 ge-3/3/6      Up      143063372      (36050)          201807391      (50520)
```

The total number of flows is about 75,000, and the total flow churn is > 40,000 per second:

```
user@router> show services stateful-firewall flow-analysis
  Services PIC Name:    sp-5/0/0
Flow Analysis Statistics:
  Total Flows Active               :75468
  Total TCP Flows Active           :75468
  Total UDP Flows Active           :0
  Total Other Flows Active         :0
  Created Flows per Second         :20887
  Deleted Flows per Second         :21013
  Peak Total Flows Active          :76420
  Peak Total TCP Flows Active      :76420
  Peak Total UDP Flows Active      :0
  Peak Total Other Flows Active    :0
  Peak Created Flows per Second    :21990
  Peak Deleted Flows per Second    :23234
  Average TCP Flow Lifetime(ms)    :8
  Average UDP Flow Lifetime(ms)    :0
  Average HTTP Flow Lifetime(ms)   :10
```

With logging turned off, we get the following numbers on CPU usage:

```
user@router> show services service-sets cpu-usage
                                                                       CPU
Interface   Service Set(or system category)                        Utilization
sp-5/0/0    sset2                                                      7.67 %
```

After activating system logging, we get:

```
user@router> show services service-sets cpu-usage
                                                                       CPU
Interface   Service Set(or system category)                        Utilization
sp-5/0/0    sset2                                                     10.55 %
```

In order words, logging every session when the total flow churn is > 40,000/sec only has an impact of approximately 3% on CPU performance.

Configuration Tips

EIM

Although EIM is a supported feature and requested by many customers, it is not widely used at this point because applications have grown to be able to traverse NAT and receive inbound connections over the same outbound connection. Also, applications that need ALGs are still prevalent.

If EIM is needed, it should be on a per application basis. In other words, only enable EIM for the applications that need it (see example below):

```
rule sip-eim {
    match-direction input;
    term t1 {
        from {
            applications junos-sip;
        }
        then {
            translated {
                source-pool p1;
                translation-type {
                    source dynamic;
                }
                mapping-type endpoint-independent;
            }
        }
    }
}
```

Juniper has several CGN deployments where EIM is not used and applications work fine, but some applications such as Apple's FaceTime and Google Talk might need EIM. Address pooling, on the other hand, has practical implications as discussed earlier and in general should be enabled.

Flow Limits

In order to avoid high CPU usage and degradation of performance when the number of active flows reaches the maximum number supported by a certain service set, we recommend making use of the max-flow knob. This knob limits the number of flows a service set can consume. For example:

```
[edit services service-set nat-cgn]
user@router# show
max-flows 3m;
nat-rule-sets nat-cgn-rs;
interface-service {
    service-interface sp-5/0/0;
}
```

New flows are checked against the limit before CPU intensive operations such as port allocation are attempted. This efficiently eliminates any time or resource to create (and fail) new flows. Packets that might require new flows will get dropped, but already established flows will be handled effectively.

Subscriber Session Limits

In certain deployments, it may be desirable to restrict the number of concurrent sessions each subscriber (source IPv4 address) can establish. This is easily done by configuring the intrusion detection services as below and including this in the service set:

```
[edit services]
user@router# show ids
rule r1 {
    match-direction input;
    term t1 {
        then {
            session-limit {
                by-source {
                    maximum 100;
                }
            }
        }
    }
}

[edit services service-set sset2]
nat-rules r1;
ids-rules r1;
interface-service {
service-interface sp-5/0/0;
}
```

Sessions and Bandwidth Recommendations

The number of sessions and the bandwidth per subscriber (source IPv4) differs greatly across different geographic regions. However, there is a common trend of increase in bandwith usage across the globe and HTTP has become the workhorse of the Internet.Although the number of sessions per subscribers has grown, it is far from the apocalyptic scenario some predicted. The average user consumes very few concurrent sessions.

Juniper can provide fine-tuned recommendations on how to size a CGN in terms of subscribers, sessions, and bandwidth depending on your region. Please contact Juniper TAC for further information.

Application Inactivity Timeouts

It is possible to configure inactivity timeouts per application or services PIC.

Per Application Timeout

As the name suggests, this is specific to a single application:

```
[edit applications]
user@router# show
application new_http {
    protocol tcp;
    destination-port 80;
    inactivity-timeout 600;
}
```

Per MS-PIC timeout

This applies to all applications on the services PIC:

```
sp-3/0/0 {
    services-options {
        inactivity-timeout 300;
    }
    unit 0 {
        family inet;
    }
}
```

ALG Support

ALGs for ICMP, traceroute, Trivial File Transfer Protocol (TFTP), Remote Shell (RSH), Distributed Computing Environment/ Remote Procedure Calls (DCE RPC) (, FTP, SIP, H.323, and Real-Time Streaming Protocol (RTSP) are all supported by Junos OS. The complete list and details of ALG support is documented at **www.juniper.net/techpubs/en_US/junos11.3/topics/concept/alg-descriptions.html**.

Operations and Management

NAT CLI

The administrator has a large number of commands available to perform operations and management. In the next sections, we discuss a few of them. The user is referred to the appropriate Juniper Networks® Junos® operating system manual for a more complete set.

Flows and Conversation

The administrator can use the command “show services stateful-firewall flows” to check the creation of softwires, as well as pre-NAT and post NAT flows within them. An example output for the configuration discussed above is shown below:

```
user@router> show services stateful-firewall flows
Interface: sp-5/0/0, Service set: sset2
Flow                                                State    Dir       Frm count
TCP        10.1.167.93:36795 ->      128.0.0.3:80    Forward  I               5
    NAT source     10.1.167.93:36795   ->       129.0.0.9:22779
TCP          128.0.0.3:80    ->      129.0.0.9:20049 Forward  O               3
    NAT dest         129.0.0.9:20049   ->    10.1.175.134:35665
TCP          128.0.0.3:80    ->      129.0.0.9:12216 Forward  O               3
    NAT dest         129.0.0.9:12216   ->     10.1.107.55:32466
TCP       10.1.154.134:35952 ->      128.0.0.3:80    Forward  I               5
    NAT source    10.1.154.134:35952   ->       129.0.0.9:20746
TCP          10.1.1.99:29573 ->      128.0.0.3:80    Forward  I               5
    NAT source       10.1.1.99:29573   ->       129.0.0.9:5201
TCP       10.0.166.148:33620 ->      128.0.0.2:80    Forward  I               5
    NAT source    10.0.166.148:33620   ->       129.0.0.9:15080
TCP         10.1.84.46:30407 ->      128.0.0.3:80    Forward  I               5
    NAT source      10.1.84.46:30407   ->       129.0.0.9:7249
```

To see the total number of flows:

```
user@router> show services stateful-firewall flows count
Interface   Service set                                                   Flow count
sp-5/0/0    sset2                                                              67851
```

Another important command that shows conversations (collection of related flows):

```
user@router> show services stateful-firewall conversations
Interface: sp-5/0/0, Service set: sset2

Conversation: ALG protocol: tcp
  Number of initiators: 1, Number of responders: 1
Flow                                                State    Dir       Frm count
TCP         10.0.14.25:62722 ->      128.0.0.2:80    Forward  I               5
    NAT source      10.0.14.25:62722   ->      129.0.0.22:48546
TCP          128.0.0.2:80    ->     129.0.0.22:48546 Forward  O               3
    NAT dest        129.0.0.22:48546   ->      10.0.14.25:62722

Conversation: ALG protocol: tcp
  Number of initiators: 1, Number of responders: 1
Flow                                                State    Dir       Frm count
TCP        10.0.175.93:63225 ->      128.0.0.2:80    Forward  I               5
    NAT source     10.0.175.93:63225   ->      129.0.0.22:49837
TCP          128.0.0.2:80    ->     129.0.0.22:49837 Forward  O               3
    NAT dest        129.0.0.22:49837   ->     10.0.175.93:63225

Conversation: ALG protocol: tcp
  Number of initiators: 1, Number of responders: 1
Flow                                                State    Dir       Frm count
TCP        10.1.42.166:52344 ->      128.0.0.3:80    Forward  I               5
    NAT source     10.1.42.166:52344   ->      129.0.0.22:22916
TCP          128.0.0.3:80    ->     129.0.0.22:22916 Forward  O               3
    NAT dest        129.0.0.22:22916   ->     10.1.42.166:52344
```

Global statistics help debug many common problems:

```
user@router> show services stateful-firewall statistics extensive
Interface: sp-5/0/0
  Service set: sset2
    New flows:
      Accepts: 2597666, Discards: 0, Rejects: 0
    Existing flows:
      Accepts: 18186040, Discards: 3, Rejects: 0
    Drops:
      IP option: 0, TCP SYN defense: 120
      NAT ports exhausted: 0
    Errors:
      IP: 0, TCP: 3
      UDP: 0, ICMP: 0
      Non-IP packets: 0, ALG: 0
    IP errors:
      IP packet length inconsistencies: 0
      Minimum IP header length check failures: 0
      Reassembled packet exceeds maximum IP length: 0
      Illegal source address: 0
      Illegal destination address: 0
      TTL zero errors: 0, Illegal IP protocol number (0 or 255: 0
      Land attack: 0
      Non-IPv4 packets: 0, Bad checksum: 0
      Illegal IP fragment length: 0
      IP fragment overlap: 0
      IP fragment reassembly timeout: 0
      Unknown: 0
    TCP errors:
      TCP header length inconsistencies: 0
      Source or destination port number is zero: 0
      Illegal sequence number and flags combinations: 0
      SYN attack (multiple SYN messages seen for the same flow: 0
      First packet not a SYN message: 3
      TCP port scan (TCP handshake, RST seen from server for SYN: 0
      Bad SYN cookie response: 0
    UDP errors:
      IP data length less than minimum UDP header length (8 bytes): 0
      Source or destination port number is zero: 0
      UDP port scan (ICMP error seen for UDP flow: 0
    ICMP errors:
      IP data length less than minimum ICMP header length (8 bytes: 0
      ICMP error length inconsistencies: 0
      Duplicate ping sequence number: 0
      Mismatched ping sequence number: 0
    ALG errors:
      BOOTP: 0, DCE-RPC: 0, DCE-RPC portmap: 0
      DNS: 0, Exec: 0, FTP: 0
      H323: 0, ICMP: 0, IIOP: 0
      Login: 0, NetBIOS: 0, NetShow: 0
      Real Audio: 0, RPC: 0, RPC portmap: 0
      RTSP: 0, Shell: 0, SIP: 0
      SNMP: 0, SQLNet: 0, TFTP: 0
      Traceroute: 0
```

NAT Information

In order to find global NAT statistics related to pool usage, the following command could be used: "show services nat pool detail." This command is normally used in conjunction with "show service stateful-firewall." See the example shown below:

```
user@router> show services nat pool detail
Interface: sp-5/0/0, Service set: sset2
  NAT pool: p1, Translation type: dynamic
    Address range: 129.0.0.1-129.0.0.254
    Port range: 512-65535, Ports in use: 35591, Out of port errors: 0, Max ports used:
40178
```

SNMP Support and Configuration

The snmpwalk/gets/getnexts and SNMP NAT traps for address usage per pool are supported. A trap is generated when the address usage percent goes above 90%. This trap is cleared when the usage falls below 80%. To view the traps set the location of the trap manager under [edit snmp] hierarchy.

```
user@router> show configuration snmp
trap-options;
trap-group services {
    version v2;
    destination-port 9002;
    targets {
        10.212.162.107;
    }
}
traceoptions {
    flag all;
}
```

The above configuration indicates the trap manager is located at IP 10.212.162.107 and it is expecting to receive traps on port 9002. Also a snmp version v2 is used in the example above. (SNMP version v1 can be configured if desired).

On the router that generates the traps, the traps will be seen in the file /var/log/snmpd.

Sample output (from snmpd log):

```
Oct  8 21:33:57 snmpd[0]  <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
Oct  8 21:33:57 snmpd[0]  <<< V2 Trap
Oct  8 21:33:57 snmpd[0]  <<<  Source:      10.209.15.166
Oct  8 21:33:57 snmpd[0]  <<<  Destination: 10.212.162.107
Oct  8 21:33:57 snmpd[0]  <<<  Version:     SNMPv2
Oct  8 21:33:57 snmpd[0]  <<<  Community:   services
Oct  8 21:33:57 snmpd[0]  <<<
Oct  8 21:33:57 snmpd[0]  <<<   OID  : sysUpTime.0
Oct  8 21:33:57 snmpd[0]  <<<   type : TimeTicks
Oct  8 21:33:57 snmpd[0]  <<<   value: (425038)  1:10:50.38
Oct  8 21:33:57 snmpd[0]  <<<
Oct  8 21:33:57 snmpd[0]  <<<   OID  : snmpTrapOID.0
Oct  8 21:33:57 snmpd[0]  <<<   type : Object
Oct  8 21:33:57 snmpd[0]  <<<   value: jnxNatAddrPoolThresholdStatus
Oct  8 21:33:57 snmpd[0]  <<<
Oct  8 21:33:57 snmpd[0]  <<<   OID  : jnxNatTrapSrcPoolName.0
Oct  8 21:33:57 snmpd[0]  <<<   type : OctetString
Oct  8 21:33:57 snmpd[0]  <<<   value: "p2"
Oct  8 21:33:57 snmpd[0]  <<<   HEX  : 70 32
Oct  8 21:33:57 snmpd[0]  <<<
Oct  8 21:33:57 snmpd[0]  <<<   OID  : jnxNatAddrPoolUtil.0
Oct  8 21:33:57 snmpd[0]  <<<   type : Number
Oct  8 21:33:57 snmpd[0]  <<<   value: 90
Oct  8 21:33:57 snmpd[0]  <<<
Oct  8 21:33:57 snmpd[0]  <<<   OID  : snmpTrapEnterprise.0
Oct  8 21:33:57 snmpd[0]  <<<   type : Object
Oct  8 21:33:57 snmpd[0]  <<<   value: jnxProductNameMX480
Oct  8 21:33:57 snmpd[0]  <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
```

The pool name (jnxNatTrapSrcPoolName.0) is ‘p2’ and the address usage (jnxNatAddrPoolUtil.0) indicates 90%.

CLI

We need to know the table name to perform the SNMP walk. To check all pools configured:

```
user@router> show snmp mib walk ascii jnxNatPoolTable
jnxNatPoolType.”pool1” = 1
jnxNatPoolType.”pool2” = 4
jnxNatPoolType.”pool3” = 3
jnxNatPoolType.”pool4” = 1
jnxNatPoolType.”pool5” = 1
jnxNatPoolType.”pool6” = 1
jnxNatPoolType.”pool7” = 3
jnxNatPoolType.”destpool” = 2
jnxNatPoolType.”new_pool” = 4
jnxNatPoolType.”destination_ip” = 2
jnxNatPoolTransHits.”pool1” = 0
jnxNatPoolTransHits.”pool2” = 0
jnxNatPoolTransHits.”pool3” = 0
jnxNatPoolTransHits.”pool4” = 0
jnxNatPoolTransHits.”pool5” = 0
jnxNatPoolTransHits.”pool6” = 0
jnxNatPoolTransHits.”pool7” = 0
jnxNatPoolTransHits.”destpool” = 0
jnxNatPoolTransHits.”new_pool” = 0
jnxNatPoolTransHits.”destination_ip” = 0
```

NOTE: The SNMP operation will show the pools that are currently configured under a term and the rule is attached to an active service set.

To check the Stats of all pools used with source translation:

```
user@router> show snmp mib walk ascii jnxSrcNatStatsTable
jnxNatSrcXlatedAddrType.”pool1” = 1
jnxNatSrcXlatedAddrType.”pool2” = 1
jnxNatSrcXlatedAddrType.”pool3” = 1
jnxNatSrcXlatedAddrType.”pool4” = 1
jnxNatSrcXlatedAddrType.”pool5” = 1
jnxNatSrcXlatedAddrType.”pool6” = 1
jnxNatSrcXlatedAddrType.”pool7” = 1
jnxNatSrcXlatedAddrType.”new_pool” = 1
jnxNatSrcPoolType.”pool1” = 1
jnxNatSrcPoolType.”pool2” = 2
jnxNatSrcPoolType.”pool3” = 3
jnxNatSrcPoolType.”pool4” = 1
jnxNatSrcPoolType.”pool5” = 1
jnxNatSrcPoolType.”pool6” = 1
jnxNatSrcPoolType.”pool7” = 3
jnxNatSrcPoolType.”new_pool” = 2
jnxNatSrcNumPortAvail.”pool1” = 0
jnxNatSrcNumPortAvail.”pool2” = 1001
jnxNatSrcNumPortAvail.”pool3” = 0
jnxNatSrcNumPortAvail.”pool4” = 0
jnxNatSrcNumPortAvail.”pool5” = 0
jnxNatSrcNumPortAvail.”pool6” = 0
jnxNatSrcNumPortAvail.”pool7” = 0
jnxNatSrcNumPortAvail.”new_pool” = 501
jnxNatSrcNumPortInuse.”pool1” = 0
jnxNatSrcNumPortInuse.”pool2” = 0
jnxNatSrcNumPortInuse.”pool3” = 0
jnxNatSrcNumPortInuse.”pool4” = 0
jnxNatSrcNumPortInuse.”pool5” = 0
jnxNatSrcNumPortInuse.”pool6” = 0
jnxNatSrcNumPortInuse.”pool7” = 0
jnxNatSrcNumPortInuse.”new_pool” = 0
jnxNatSrcNumAddressAvail.”pool1” = 254
jnxNatSrcNumAddressAvail.”pool2” = 1
jnxNatSrcNumAddressAvail.”pool3” = 254
jnxNatSrcNumAddressAvail.”pool4” = 255
jnxNatSrcNumAddressAvail.”pool5” = 256
jnxNatSrcNumAddressAvail.”pool6” = 256
jnxNatSrcNumAddressAvail.”pool7” = 255
jnxNatSrcNumAddressAvail.”new_pool” = 1
jnxNatSrcNumAddressInUse.”pool1” = 0
jnxNatSrcNumAddressInUse.”pool2” = 0
jnxNatSrcNumAddressInUse.”pool3” = 0
jnxNatSrcNumAddressInUse.”pool4” = 0
jnxNatSrcNumAddressInUse.”pool5” = 0
jnxNatSrcNumAddressInUse.”pool6” = 0
jnxNatSrcNumAddressInUse.”pool7” = 0
jnxNatSrcNumAddressInUse.”new_pool” = 0
jnxNatSrcNumSessions.”pool1” = 0
jnxNatSrcNumSessions.”pool2” = 0
jnxNatSrcNumSessions.”pool3” = 0
jnxNatSrcNumSessions.”pool4” = 0
jnxNatSrcNumSessions.”pool5” = 0
jnxNatSrcNumSessions.”pool6” = 0
jnxNatSrcNumSessions.”pool7” = 0
jnxNatSrcNumSessions.”new_pool” = 0
```

NOTE: The SNMP operation will show the source pools that are currently configured under a term and the rule is attached to an active service set. To check the Stats of all pools used with source translation:

```
user@router> show snmp mib walk ascii jnxSrcNatStatsTable
jnxNatSrcXlatedAddrType.”pool1” = 1
jnxNatSrcXlatedAddrType.”pool2” = 1
jnxNatSrcXlatedAddrType.”pool3” = 1
jnxNatSrcXlatedAddrType.”pool4” = 1
jnxNatSrcXlatedAddrType.”pool5” = 1
jnxNatSrcXlatedAddrType.”pool6” = 1
jnxNatSrcXlatedAddrType.”pool7” = 1
jnxNatSrcXlatedAddrType.”new_pool” = 1
jnxNatSrcPoolType.”pool1” = 1
jnxNatSrcPoolType.”pool2” = 2
jnxNatSrcPoolType.”pool3” = 3
jnxNatSrcPoolType.”pool4” = 1
jnxNatSrcPoolType.”pool5” = 1
jnxNatSrcPoolType.”pool6” = 1
jnxNatSrcPoolType.”pool7” = 3
jnxNatSrcPoolType.”new_pool” = 2
jnxNatSrcNumPortAvail.”pool1” = 0
jnxNatSrcNumPortAvail.”pool2” = 1001
jnxNatSrcNumPortAvail.”pool3” = 0
jnxNatSrcNumPortAvail.”pool4” = 0
jnxNatSrcNumPortAvail.”pool5” = 0
jnxNatSrcNumPortAvail.”pool6” = 0
jnxNatSrcNumPortAvail.”pool7” = 0
jnxNatSrcNumPortAvail.”new_pool” = 501
jnxNatSrcNumPortInuse.”pool1” = 0
jnxNatSrcNumPortInuse.”pool2” = 0
jnxNatSrcNumPortInuse.”pool3” = 0
jnxNatSrcNumPortInuse.”pool4” = 0
jnxNatSrcNumPortInuse.”pool5” = 0
jnxNatSrcNumPortInuse.”pool6” = 0
jnxNatSrcNumPortInuse.”pool7” = 0
jnxNatSrcNumPortInuse.”new_pool” = 0
jnxNatSrcNumAddressAvail.”pool1” = 254
jnxNatSrcNumAddressAvail.”pool2” = 1
jnxNatSrcNumAddressAvail.”pool3” = 254
jnxNatSrcNumAddressAvail.”pool4” = 255
jnxNatSrcNumAddressAvail.”pool5” = 256
jnxNatSrcNumAddressAvail.”pool6” = 256
jnxNatSrcNumAddressAvail.”pool7” = 255
jnxNatSrcNumAddressAvail.”new_pool” = 1
jnxNatSrcNumAddressInUse.”pool1” = 0
jnxNatSrcNumAddressInUse.”pool2” = 0
jnxNatSrcNumAddressInUse.”pool3” = 0
jnxNatSrcNumAddressInUse.”pool4” = 0
jnxNatSrcNumAddressInUse.”pool5” = 0
jnxNatSrcNumAddressInUse.”pool6” = 0
jnxNatSrcNumAddressInUse.”pool7” = 0
jnxNatSrcNumAddressInUse.”new_pool” = 0
jnxNatSrcNumSessions.”pool1” = 0
jnxNatSrcNumSessions.”pool2” = 0
jnxNatSrcNumSessions.”pool3” = 0
jnxNatSrcNumSessions.”pool4” = 0
jnxNatSrcNumSessions.”pool5” = 0
jnxNatSrcNumSessions.”pool6” = 0
jnxNatSrcNumSessions.”pool7” = 0
jnxNatSrcNumSessions.”new_pool” = 0
```

NOTE: The SNMP operation will show the source pools that are currently configured under a term and the rule is attached to an active service set.

To check all rules configure: Since each term in a rule will have different hits, statistics for each term are displayed:

```
user@router> show snmp mib walk ascii jnxNatRuleTable
jnxNatRuleType.”rule1:term1” = 1
jnxNatRuleType.”rule2:term1” = 3
jnxNatRuleType.”rule3:term1” = 4
jnxNatRuleType.”rule4:term1” = 2
jnxNatRuleType.”rule5:term1” = 1
jnxNatRuleType.”rule5:term2” = 1
jnxNatRuleType.”rule5:term3” = 1
jnxNatRuleType.”new_rule:term1” = 3
jnxNatRuleType.”napt_rule:term1” = 4
jnxNatRuleType.”destination_rule:term1” = 2
jnxNatRuleTransHits.”rule1:term1” = 0
jnxNatRuleTransHits.”rule2:term1” = 0
jnxNatRuleTransHits.”rule3:term1” = 0
jnxNatRuleTransHits.”rule4:term1” = 0
jnxNatRuleTransHits.”rule5:term1” = 0
jnxNatRuleTransHits.”rule5:term2” = 0
jnxNatRuleTransHits.”rule5:term3” = 0
jnxNatRuleTransHits.”new_rule:term1” = 0
jnxNatRuleTransHits.”napt_rule:term1” = 0
jnxNatRuleTransHits.”destination_rule:term1” = 0
```

NOTE: The SNMP operation will show the rules configured and the attached active service set.

NOTE: keyword ‘ascii’ will show the key in ascii. The name is the key here.

Without the keyword ascii:

```
user@router> show snmp mib walk jnxNatRuleTable
jnxNatRuleType.11.114.117.108.101.49.58.116.101.114.109.49 = 1
jnxNatRuleType.11.114.117.108.101.50.58.116.101.114.109.49 = 3
jnxNatRuleType.11.114.117.108.101.51.58.116.101.114.109.49 = 4
jnxNatRuleType.11.114.117.108.101.52.58.116.101.114.109.49 = 2
jnxNatRuleType.11.114.117.108.101.53.58.116.101.114.109.49 = 1
jnxNatRuleType.11.114.117.108.101.53.58.116.101.114.109.50 = 1
jnxNatRuleType.11.114.117.108.101.53.58.116.101.114.109.51 = 1
jnxNatRuleType.14.110.101.119.95.114.117.108.101.58.116.101.114.109.49 = 3
jnxNatRuleType.15.110.97.112.116.95.114.117.108.101.58.116.101.114.109.49 = 4
jnxNatRuleType.22.100.101.115.116.105.110.97.116.105.111.110.95.114.117.108.101.58.
116.101.114.109.49 = 2
jnxNatRuleTransHits.11.114.117.108.101.49.58.116.101.114.109.49 = 0
jnxNatRuleTransHits.11.114.117.108.101.50.58.116.101.114.109.49 = 0
jnxNatRuleTransHits.11.114.117.108.101.51.58.116.101.114.109.49 = 0
jnxNatRuleTransHits.11.114.117.108.101.52.58.116.101.114.109.49 = 0
jnxNatRuleTransHits.11.114.117.108.101.53.58.116.101.114.109.49 = 0
jnxNatRuleTransHits.11.114.117.108.101.53.58.116.101.114.109.50 = 0
jnxNatRuleTransHits.11.114.117.108.101.53.58.116.101.114.109.51 = 0
jnxNatRuleTransHits.14.110.101.119.95.114.117.108.101.58.116.101.114.109.49 = 0
jnxNatRuleTransHits.15.110.97.112.116.95.114.117.108.101.58.116.101.114.109.49 = 0
jnxNatRuleTransHits.22.100.101.115.116.105.110.97.116.105.111.110.95.114.117.108.
101.58.116.101.114.109.49 = 0
```

Note the key is shown in numerical format with the starting number denoting the length of the key.

To perform the “snmpget” and “snmpgetnext” we require the object in the above form.

SNMP get:

```
user@router> show snmp mib get ascii
jnxNatRuleType.14.110.101.119.95.114.117.108.101.58.116.101.114.109.49
jnxNatRuleType.”new_rule:term1” = 3
```

SNMP getnext:

```
user@router> show snmp mib get-next ascii
jnxNatRuleType.14.110.101.119.95.114.117.108.101.58.116.101.114.109.49
jnxNatRuleType.”napt_rule:term1” = 4
```

External Management Station

The query from an external box requires the format shown here:

```
snmpwalk -v 2c -OsaT -M <MIB directory> -m <MIB file in that directory> -c public <Router
name> <Table name in the MIB>
```

-v denotes version and -OsaT is for formatting in ascii , -Obs will format the output in decimal.

To check all pools configured:

```
[user@host ~]$ snmpwalk -v 2c -OsaT -M /b/sidrc/nat_commit/src/junos/shared/mibs/ -m /b/
sidrc/nat_commit/src/junos/shared/mibs/jnx-sp-nat.mib -c public sledding jnxNatPoolTable
jnxNatPoolType.”pool1” = INTEGER: static-source(1)
jnxNatPoolType.”pool2” = INTEGER: napt(4)
jnxNatPoolType.”pool3” = INTEGER: dynamic-source(3)
jnxNatPoolType.”pool4” = INTEGER: static-source(1)
jnxNatPoolType.”pool5” = INTEGER: static-source(1)
jnxNatPoolType.”pool6” = INTEGER: static-source(1)
jnxNatPoolType.”pool7” = INTEGER: dynamic-source(3)
jnxNatPoolType.”destpool” = INTEGER: static-destination(2)
jnxNatPoolType.”new_pool” = INTEGER: napt(4)
jnxNatPoolType.”destination_ip” = INTEGER: static-destination(2)
jnxNatPoolTransHits.”pool1” = Gauge32: 0
jnxNatPoolTransHits.”pool2” = Gauge32: 0
jnxNatPoolTransHits.”pool3” = Gauge32: 0
jnxNatPoolTransHits.”pool4” = Gauge32: 0
jnxNatPoolTransHits.”pool5” = Gauge32: 0
jnxNatPoolTransHits.”pool6” = Gauge32: 0
jnxNatPoolTransHits.”pool7” = Gauge32: 0
jnxNatPoolTransHits.”destpool” = Gauge32: 0
jnxNatPoolTransHits.”new_pool” = Gauge32: 0
jnxNatPoolTransHits.”destination_ip” = Gauge32: 0
```

To check all source pool stats:

```
[user@host ~]$ snmpwalk -v 2c -OsaT -M /b/sidrc/nat_commit/src/junos/shared/mibs/ -m /b/
sidrc/nat_commit/src/junos/shared/mibs/jnx-sp-nat.mib -c public sledding jnxSrcNatStatsTable
jnxNatSrcXlatedAddrType.”pool1” = INTEGER: ipv4(1)
jnxNatSrcXlatedAddrType.”pool2” = INTEGER: ipv4(1)
jnxNatSrcXlatedAddrType.”pool3” = INTEGER: ipv4(1)
jnxNatSrcXlatedAddrType.”pool4” = INTEGER: ipv4(1)
jnxNatSrcXlatedAddrType.”pool5” = INTEGER: ipv4(1)
jnxNatSrcXlatedAddrType.”pool6” = INTEGER: ipv4(1)
jnxNatSrcXlatedAddrType.”pool7” = INTEGER: ipv4(1)
jnxNatSrcXlatedAddrType.”new_pool” = INTEGER: ipv4(1)
```

```
jnxNatSrcPoolType.”pool1” = INTEGER: static(1)
jnxNatSrcPoolType.”pool2” = INTEGER: dynamic-napt(2)
jnxNatSrcPoolType.”pool3” = INTEGER: dynamic-nat(3)
jnxNatSrcPoolType.”pool4” = INTEGER: static(1)
jnxNatSrcPoolType.”pool5” = INTEGER: static(1)
jnxNatSrcPoolType.”pool6” = INTEGER: static(1)
jnxNatSrcPoolType.”pool7” = INTEGER: dynamic-nat(3)
jnxNatSrcPoolType.”new_pool” = INTEGER: dynamic-napt(2)
jnxNatSrcNumPortAvail.”pool1” = Gauge32: 0
jnxNatSrcNumPortAvail.”pool2” = Gauge32: 1001
jnxNatSrcNumPortAvail.”pool3” = Gauge32: 0
jnxNatSrcNumPortAvail.”pool4” = Gauge32: 0
jnxNatSrcNumPortAvail.”pool5” = Gauge32: 0
jnxNatSrcNumPortAvail.”pool6” = Gauge32: 0
jnxNatSrcNumPortAvail.”pool7” = Gauge32: 0
jnxNatSrcNumPortAvail.”new_pool” = Gauge32: 501
jnxNatSrcNumPortInuse.”pool1” = Gauge32: 0
jnxNatSrcNumPortInuse.”pool2” = Gauge32: 0
jnxNatSrcNumPortInuse.”pool3” = Gauge32: 0
jnxNatSrcNumPortInuse.”pool4” = Gauge32: 0
jnxNatSrcNumPortInuse.”pool5” = Gauge32: 0
jnxNatSrcNumPortInuse.”pool6” = Gauge32: 0
jnxNatSrcNumPortInuse.”pool7” = Gauge32: 0
jnxNatSrcNumPortInuse.”new_pool” = Gauge32: 0
jnxNatSrcNumAddressAvail.”pool1” = Gauge32: 254
jnxNatSrcNumAddressAvail.”pool2” = Gauge32: 1
jnxNatSrcNumAddressAvail.”pool3” = Gauge32: 254
jnxNatSrcNumAddressAvail.”pool4” = Gauge32: 255
jnxNatSrcNumAddressAvail.”pool5” = Gauge32: 256
jnxNatSrcNumAddressAvail.”pool6” = Gauge32: 256
jnxNatSrcNumAddressAvail.”pool7” = Gauge32: 255
jnxNatSrcNumAddressAvail.”new_pool” = Gauge32: 1
jnxNatSrcNumAddressInUse.”pool1” = Gauge32: 0
jnxNatSrcNumAddressInUse.”pool2” = Gauge32: 0
jnxNatSrcNumAddressInUse.”pool3” = Gauge32: 0
jnxNatSrcNumAddressInUse.”pool4” = Gauge32: 0
jnxNatSrcNumAddressInUse.”pool5” = Gauge32: 0
jnxNatSrcNumAddressInUse.”pool6” = Gauge32: 0
jnxNatSrcNumAddressInUse.”pool7” = Gauge32: 0
jnxNatSrcNumAddressInUse.”new_pool” = Gauge32: 0
jnxNatSrcNumSessions.”pool1” = Gauge32: 0
jnxNatSrcNumSessions.”pool2” = Gauge32: 0
jnxNatSrcNumSessions.”pool3” = Gauge32: 0
jnxNatSrcNumSessions.”pool4” = Gauge32: 0
jnxNatSrcNumSessions.”pool5” = Gauge32: 0
jnxNatSrcNumSessions.”pool6” = Gauge32: 0
jnxNatSrcNumSessions.”pool7” = Gauge32: 0
jnxNatSrcNumSessions.”new_pool” = Gauge32: 0
```

To check all rules/terms configured:

```
[user@host ~]$ snmpwalk -v 2c -OsaT -M /b/sidrc/nat_commit/src/junos/shared/mibs/ -m /b/
sidrc/nat_commit/src/junos/shared/mibs/jnx-sp-nat.mib -c public sledding jnxNatRuleTable
jnxNatRuleType.”rule1:term1” = INTEGER: static-source(1)
jnxNatRuleType.”rule2:term1” = INTEGER: dynamic-source(3)
jnxNatRuleType.”rule3:term1” = INTEGER: napt(4)
jnxNatRuleType.”rule4:term1” = INTEGER: static-destination(2)
jnxNatRuleType.”rule5:term1” = INTEGER: static-source(1)
jnxNatRuleType.”rule5:term2” = INTEGER: static-source(1)
jnxNatRuleType.”rule5:term3” = INTEGER: static-source(1)
jnxNatRuleType.”new_rule:term1” = INTEGER: dynamic-source(3)
jnxNatRuleType.”napt_rule:term1” = INTEGER: napt(4)
jnxNatRuleType.”destination_rule:term1” = INTEGER: static-destination(2)
jnxNatRuleTransHits.”rule1:term1” = Gauge32: 0
jnxNatRuleTransHits.”rule2:term1” = Gauge32: 0
jnxNatRuleTransHits.”rule3:term1” = Gauge32: 0
jnxNatRuleTransHits.”rule4:term1” = Gauge32: 0
jnxNatRuleTransHits.”rule5:term1” = Gauge32: 0
jnxNatRuleTransHits.”rule5:term2” = Gauge32: 0
jnxNatRuleTransHits.”rule5:term3” = Gauge32: 0
jnxNatRuleTransHits.”new_rule:term1” = Gauge32: 0
jnxNatRuleTransHits.”napt_rule:term1” = Gauge32: 0
jnxNatRuleTransHits.”destination_rule:term1” = Gauge32: 0
```

To perform a get/getnext operation,denote the object in decimal notation:

```
[user@host ~]$ snmpwalk -v 2c -Obs -M /b/sidrc/nat_commit/src/junos/shared/mibs/ -m /b/
sidrc/nat_commit/src/junos/shared/mibs/jnx-sp-nat.mib -c public sledding jnxNatRuleTable
jnxNatRuleType.11.114.117.108.101.49.58.116.101.114.109.49 = INTEGER: static-source(1)
jnxNatRuleType.11.114.117.108.101.50.58.116.101.114.109.49 = INTEGER: dynamic-source(3)
jnxNatRuleType.11.114.117.108.101.51.58.116.101.114.109.49 = INTEGER: napt(4)
jnxNatRuleType.11.114.117.108.101.52.58.116.101.114.109.49 = INTEGER: static-destination(2)
jnxNatRuleType.11.114.117.108.101.53.58.116.101.114.109.49 = INTEGER: static-source(1)
jnxNatRuleType.11.114.117.108.101.53.58.116.101.114.109.50 = INTEGER: static-source(1)
jnxNatRuleType.11.114.117.108.101.53.58.116.101.114.109.51 = INTEGER: static-source(1)
jnxNatRuleType.14.110.101.119.95.114.117.108.101.58.116.101.114.109.49 = INTEGER: dynamic-
source(3)
jnxNatRuleType.15.110.97.112.116.95.114.117.108.101.58.116.101.114.109.49 = INTEGER:
napt(4)
jnxNatRuleType.22.100.101.115.116.105.110.97.116.105.111.110.95.114.117.108.101.58.116.101.
114.109.49 = INTEGER: static-destination(2)
jnxNatRuleTransHits.11.114.117.108.101.49.58.116.101.114.109.49 = Gauge32: 0
jnxNatRuleTransHits.11.114.117.108.101.50.58.116.101.114.109.49 = Gauge32: 0
jnxNatRuleTransHits.11.114.117.108.101.51.58.116.101.114.109.49 = Gauge32: 0
jnxNatRuleTransHits.11.114.117.108.101.52.58.116.101.114.109.49 = Gauge32: 0
jnxNatRuleTransHits.11.114.117.108.101.53.58.116.101.114.109.49 = Gauge32: 0
jnxNatRuleTransHits.11.114.117.108.101.53.58.116.101.114.109.50 = Gauge32: 0
jnxNatRuleTransHits.11.114.117.108.101.53.58.116.101.114.109.51 = Gauge32: 0
jnxNatRuleTransHits.14.110.101.119.95.114.117.108.101.58.116.101.114.109.49 = Gauge32: 0
jnxNatRuleTransHits.15.110.97.112.116.95.114.117.108.101.58.116.101.114.109.49 = Gauge32: 0
jnxNatRuleTransHits.22.100.101.115.116.105.110.97.116.105.111.110.95.114.
117.108.101.58.116.
101.114.109.49 = Gauge32: 0
[user@host ~]$ snmpget -v 2c -Obs -M /b/sidrc/nat_commit/src/junos/shared/mibs/
-m /b/sidrc/nat_commit/src/junos/shared/mibs/jnx-sp-nat.mib -c public sledding
jnxNatRuleType.14.110.101.119.95.114.117.108.101.58.116.101.114.109.49
jnxNatRuleType.14.110.101.119.95.114.117.108.101.58.116.101.114.109.49 = INTEGER: dynamic-
source(3)
[user@host ~]$ snmpget -v 2c -OsaT -M /b/sidrc/nat_commit/src/junos/shared/mibs/
-m /b/sidrc/nat_commit/src/junos/shared/mibs/jnx-sp-nat.mib -c public sledding
jnxNatRuleType.14.110.101.119.95.114.117.108.101.58.116.101.114.109.49
jnxNatRuleType."new_rule:term1" = INTEGER: dynamic-source(3)
```

The output can be formatted as required.

```
[user@host ~]$ snmpgetnext -v 2c -OsaT -M /b/sidrc/nat_commit/src/junos/shared/
mibs/ -m /b/sidrc/nat_commit/src/junos/shared/mibs/jnx-sp-nat.mib -c public sledding
jnxNatRuleType.14.110.101.119.95.114.117.108.101.58.116.101.114.109.49
jnxNatRuleType."napt_rule:term1" = INTEGER: napt(4)
```

Summary

In this implementation guide, we have provided an overview of NAT444 features as part of CGN function supported on MS-PIC and MS-DPC blade. The guide covers all the basic configuration steps to implement NAT444 and provides key configuration tips and guidelines on optional features and their applications. For more information related to CGN please visit **www.juniper.net/ipv6**.

Bibliography

[NAPT] **http://www.ietf.org/rfc/rfc2766.txt**

[NAT64] **http://www.ietf.org/id/draft-ietf-behave-v6v4-xlate-stateful-12.txt**

[XLATE] **http://www.ietf.org/id/draft-ietf-behave-v6v4-xlate-23.txt**

[NAT66] **http://tools.ietf.org/html/draft-mrw-behave-nat66-02**

[DS-Lite] A. Durand, ed., "*Dual-Stack Lite Broadband Deployments Post IPv4 Exhaustion*," draft-ietf-softwire-dual-stack-lite-03, October 2009.

[RFC2473] A. Conta and S. Deering, "Generic Packet Tunneling in IPv6 Specification," RFC 2473, December 1998.

[RFC2663] P. Srisuresh and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations," RFC 2663, August 1999.

[RFC4787] F. Audet and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP," BCP 127, RFC 4787, January 2007.

[RFC4925] X. Li, S. Dawkins, D. Ward, and A. Durand, "Softwire Problem Statement," RFC 4925, July 2007.

[RFC5382] S. Guha, K. Biswas, B. Ford, S. Sivakumar, and P. Srisuresh," NAT Behavioral Requirements for TCP," BCP 142, RFC 5382, October 2008.

[RFC5508] P. Srisuresh, B. Ford, S. Sivakumar, and S. Guha, "NAT Behavioral Requirements for ICMP," BCP 148, RFC 5508, April 2009.

[IPv4-Exaustion] **http://www.potaroo.net/tools/ipv4/index.html**

[IANA-Allocation] **http://www.iana.org/assignments/ipv4-address-space/ipv4-address-space.xhtml**

[NAT444] **http://tools.ietf.org/html/draft-shirasaki-nat444-isp-shared-addr-03**

[INC-CGN] **http://tools.ietf.org/html/draft-ietf-v6ops-incremental-cgn-01**

[CGN] **https://wiki.tools.ietf.org/html/draft-nishitani-cgn-04**

About Juniper Networks

Juniper Networks is in the business of network innovation. From devices to data centers, from consumers to cloud providers, Juniper Networks delivers the software, silicon and systems that transform the experience and economics of networking. The company serves customers and partners worldwide. Additional information can be found at **www.juniper.net**.

Corporate and Sales Headquarters
Juniper Networks, Inc.
1194 North Mathilda Avenue
Sunnyvale, CA 94089 USA
Phone: 888.JUNIPER (888.586.4737)
or 408.745.2000
Fax: 408.745.2100
www.juniper.net

APAC Headquarters
Juniper Networks (Hong Kong)
26/F, Cityplaza One
1111 King's Road
Taikoo Shing, Hong Kong
Phone: 852.2332.3636
Fax: 852.2574.7803

EMEA Headquarters
Juniper Networks Ireland
Airside Business Park
Swords, County Dublin, Ireland
Phone: 35.31.8903.600
EMEA Sales: 00800.4586.4737
Fax: 35.31.8903.601

To purchase Juniper Networks solutions, please contact your Juniper Networks representative at 1-866-298-6428 or authorized reseller.

8010076-002-EN Nov 2011

Printed on recycled paper