Cisco CRS Router Carrier Grade NAT Configuration Guide, Release 6.1.x

First Published: 2016-11-15

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS

Preface

The Preface contains these sections:

Changes to This Document

The following table lists the technical changes made to this document since it was first published.

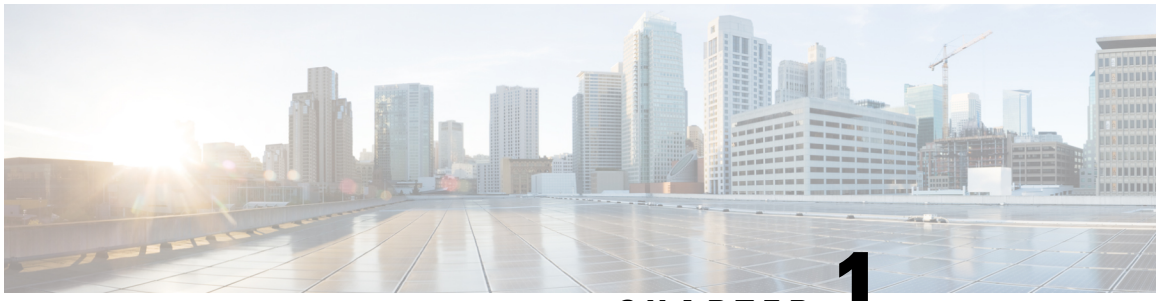
Date	Change Summary
November 2016	Initial release of this document.

Communications, Services, and Additional Information

- To receive timely, relevant information from Cisco, sign up at Cisco Profile Manager.
- To get the business impact you're looking for with the technologies that matter, visit Cisco Services.
- To submit a service request, visit Cisco Support.
- To discover and browse secure, validated enterprise-class apps, products, solutions and services, visit Cisco Marketplace.
- To obtain general networking, training, and certification titles, visit Cisco Press.
- To find warranty information for a specific product or product family, access Cisco Warranty Finder.

Cisco Bug Search Tool

Cisco Bug Search Tool (BST) is a web-based tool that acts as a gateway to the Cisco bug tracking system that maintains a comprehensive list of defects and vulnerabilities in Cisco products and software. BST provides you with detailed defect information about your products and software.



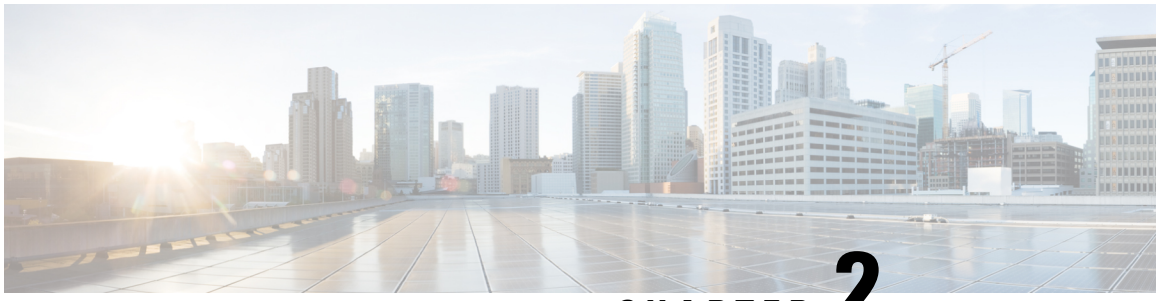
CHAPTER 1

New and Changed Carrier Grade NAT Feature

This table summarizes the new and changed information for the *Cisco IOS XR Carrier Grade NAT Configuration Guide for the Cisco CRS Router*, and tells you where the features are documented.

New and Changed Carrier Grade NAT Features

Feature	Description	Introduced/Changed in Release	Where Documented
None	No new features introduced	Release 6.1.2	Not applicable



CHAPTER 2

Implementing Carrier Grade NAT on Cisco IOS XR Software

This chapter provides an overview of the implementation of Carrier Grade NAT on Cisco IOS XR Software.

Carrier Grade NAT Overview and Benefits

To implement the Carrier Grade NAT, you should understand the following concepts:

Carrier Grade NAT Overview

Carrier Grade Network Address Translation (CGN) is a large scale NAT that is capable of providing private IPv4 to public IPv4 address translation in the order of millions of translations to support a large number of subscribers, and at least 10 Gbps full-duplex bandwidth throughput.

CGN is a workable solution to the IPv4 address completion problem, and offers a way for service provider subscribers and content providers to implement a seamless transition to IPv6. CGN employs network address and port translation (NAPT) methods to aggregate many private IP addresses into fewer public IPv4 addresses. For example, a single public IPv4 address with a pool of 32 K port numbers supports 320 individual private IP subscribers assuming each subscriber requires 100 ports. For example, each TCP connection needs one port number.

A CGN requires IPv6 to assist with the transition from IPv4 to IPv6.

Benefits of Carrier Grade NAT

CGN offers these benefits:

- Enables service providers to execute orderly transitions to IPv6 through mixed IPv4 and IPv6 networks.

- Provides address family translation but not limited to just translation within one address family.
- Delivers a comprehensive solution suite for IP address management and IPv6 transition.

IPv4 Address Shortage

A fixed-size resource such as the 32-bit public IPv4 address space will run out in a few years. Therefore, the IPv4 address shortage presents a significant and major challenge to all service providers who depend on large blocks of public or private IPv4 addresses for provisioning and managing their customers.

Service providers cannot easily allocate sufficient public IPv4 address space to support new customers that need to access the public IPv4 Internet.

NAT and NAPT Overview

A Network Address Translation (NAT) box is positioned between private and public IP networks that are addressed with non-global private addresses and a public IP addresses respectively. A NAT performs the task of mapping one or many private (or internal) IP addresses into one public IP address by employing both network address and port translation (NAPT) techniques. The mappings, otherwise referred to as bindings, are typically created when a private IPv4 host located behind the NAT initiates a connection (for example, TCP SYN) with a public IPv4 host. The NAT intercepts the packet to perform these functions:

- Rewrites the private IP host source address and port values with its own IP source address and port values
- Stores the private-to-public binding information in a table and sends the packet. When the public IP host returns a packet, it is addressed to the NAT. The stored binding information is used to replace the IP destination address and port values with the private IP host address and port values.

Traditionally, NAT boxes are deployed in the residential home gateway (HGW) to translate multiple private IP addresses. The NAT boxes are configured on multiple devices inside the home to a single public IP address, which are configured and provisioned on the HGW by the service provider. In enterprise scenarios, you can use the NAT functions combined with the firewall to offer security protection for corporate resources and allow for provider-independent IPv4 addresses. NATs have made it easier for private IP home networks to flourish independently from service provider IP address provisioning. Enterprises can permanently employ private IP addressing for Intranet connectivity while relying on a few NAT boxes, and public IPv4 addresses for external public Internet connectivity. NAT boxes in conjunction with classic methods such as Classless Inter-Domain Routing (CIDR) have slowed public IPv4 address consumption.

Network Address and Port Mapping

Network address and port mapping can be reused to map new sessions to external endpoints after establishing a first mapping between an internal address and port to an external address. These NAT mapping definitions are defined from RFC 4787:

- Endpoint-independent mapping—Reuses the port mapping for subsequent packets that are sent from the same internal IP address and port to any external IP address and port.
- Address-dependent mapping—Reuses the port mapping for subsequent packets that are sent from the same internal IP address and port to the same external IP address, regardless of the external port.

Note CGN on ISM implements Endpoint-Independent Mapping.

Translation Filtering

RFC 4787 provides translation filtering behaviors for NATs. These options are used by NAT to filter packets originating from specific external endpoints:

- **Endpoint-independent filtering**—Filters out only packets that are not destined to the internal address and port regardless of the external IP address and port source.
- **Address-dependent filtering**—Filters out packets that are not destined to the internal address. In addition, NAT filters out packets that are destined for the internal endpoint.
- **Address and port-dependent filtering**—Filters out packets that are not destined to the internal address. In addition, NAT filets out packets that are destined for the internal endpoint if the packets were not sent previously.

Prerequisites for Implementing the Carrier Grade NAT

The following prerequisites are required to implement Carrier Grade NAT:

- You must be running Cisco IOS XR software Release 3.9.1 or above.
- You must have installed the CGN service package or the **pie hfr-services-p.pie-x.x.x** or **hfr-services-px.pie-x.x.x** (where x.x.x specifies the release number of Cisco IOS XR software)

  Note The CGN service package was termed as **hfr-cgn-p.pie** or **hfr-cgn-px.pie** for releases prior to Cisco IOS XR Software Release 4.2.0. The CGN service package is referred as **hfr-services-p.pie** or **hfr-services-px.pie** in Cisco IOS XR Software Release 4.2.0 and later.

- You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command.
- In case of Intra chassis redundancy, enable CGSE data and control path monitoring in configuration mode, where R/S/CPU0 is the CGSE Location -
  - service-plim-ha location is R/S/CPU0 datapath-test
  - service-plim-ha location is R/S/CPU0 core-to-core-test
  - service-plim-ha location is R/S/CPU0 pci-test
  - service-plim-ha location is R/S/CPU0 coredump-extraction
  - service-plim-ha location R/S/CPU0 linux-timeout 500
  - service-plim-ha location R/S/CPU0 msc-timeout 500

Note All the error conditions result in card reload that triggers switchover to standby CGSE. The option of revertive switchover (that is disabled by default) and forced switchover is also available and can be used if required. Contact Cisco Technical Support with **show tech-support cgn** information.

- In case of standalone CGSE (without intra chassis redundancy), enable CGSE data and control path monitoring in configuration mode, where R/S/CPU0 is the CGSE Location with auto reload disabled and
  - service-plim-ha location R/S/CPU0 datapath-test
  - service-plim-ha location R/S/CPU0 core-to-core-test
  - service-plim-ha location R/S/CPU0 pci-test
  - service-plim-ha location R/S/CPU0 coredump-extraction
  - service-plim-ha location R/S/CPU0 linux-timeout 500
  - service-plim-ha location R/S/CPU0 msc-timeout 500
  - (admin-config) hw-module reset auto disable location R/S/CPU0

Note All the error conditions result in a syslog message. On observation of Heartbeat failures or any HA test failure messages, contact Cisco Technical Support with **show tech-support cgn** information.

Note If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.

CGSE PLIM

A Carrier-Grade Services Engine (CGSE) is a physical line interface module (PLIM) for the Cisco CRS-1 Router. When the CGSE is attached to a single CRS modular service card (forwarding engine), it provides the hardware system running applications such as NAT44, XLAT, Stateful NAT64 and DS-Lite. An individual application module consumes one CRS linecard slot. Multiple modules can be placed inside a single CRS chassis to add capacity, scale, and redundancy.

There can be only one ServiceInfra SVI per CGSE Slot. This is used for the Management Plane and is required to bring up CGSE. This is of local significance within the chassis.

ServiceApp SVI is used to forward the data traffic to the CGSE applications. You can scale up to 256 ServiceApp interfaces for each CGSE. These interfaces can be advertised in IGP/EGP.

CGSE Multi-Chassis Support

The CGSE line card is supported in a multi-chassis configuration. 16 CGSE line cards are supported on each Cisco CRS Router chassis. A maximum of 32 CGN instances are supported.

For CGN applications, such as NAT44, NAT64, XLAT, DS-Lite and 6RD, a maximum of 20 million sessions are supported by each CGSE line card.

CGSE Plus PLIM

CGSE Plus is a mutli-service PLIM for the Cisco CRS-3 Router. The module has a maximum packet processing speed of 40 Gbps full-duplex with a reduced boot time and latency.

Note The actual throughput of the application depends on the software logic and the CPU cycles consumed by the software.

It also supports services redundancy and QoS for service applications.

Note On a Cisco CRS router, you cannot enable MPLS or LDP along with a CGSE Plus card. Disable MPLS and LDP before bringing up a CGSE Plus card. Ensure that an MPLS or LDP label is not assigned on a NAT-associated IP address.

CGSE Plus is brought up in two modes:

- CGN mode — The Cisco IOS XR and Linux software are tuned to host CGN applications such as NAT44 and 6RD.
- SESH mode — The Cisco IOS XR and Linux software are tuned to host future applications such as Arbor DDoS services.

For more information on CGSE Plus PLIM, see *Cisco CRS Carrier Grade Services Engine Physical Layer Interface Module Installation Note.*

Note The heartbeat messages must always be in the keep alive state. In case of heartbeat messages being dropped, you must specify an ACL rule to let the heartbeat messages transmit from ServiceInfra mask to ServiceInfra IP.

ServiceInfra Configuration:

```
P/0/RP0/CPU0:CRS-B#show running-config interface serviceinfra *
Sat Dec 16 07:28:41.088 EDT
interface ServiceInfra3
ipv4 address 3.3.3.1 255.255.255.252
service-location 0/3/CPU0
flow ipv4 monitor snfv4 sampler samp ingress
ipv4 access-group ABF-CGNAT ingress
!
interface ServiceInfra6
ipv4 address 6.6.6.1 255.255.255.252
```

```
service-location 0/6/CPU0
ipv4 access-group ABF-CGNAT ingress
```

ACL Configuration:

```
RP/0/RP0/CPU0:CRS-B#sh ipv4 access-list ABF-CGNAT
Sat Dec 16 07:29:02.433 EDT
ipv4 access-list ABF-CGNAT
10 permit ipv4 172.18.12.0 0.0.0.3 host 172.23.6.1 nexthop1 vrf CGNAT-COLLECTOR
11 permit ipv4 172.18.13.0 0.0.0.3 host 172.23.6.1 nexthop1 vrf CGNAT-COLLECTOR
20 permit ipv4 172.18.12.0 0.0.0.3 host 172.23.6.2 nexthop1 vrf CGNAT-COLLECTOR
21 permit ipv4 172.18.13.0 0.0.0.3 host 172.23.6.2 nexthop1 vrf CGNAT-COLLECTOR
30 permit udp host 3.3.3.2 host 3.3.3.1
40 permit udp host 6.6.6.2 host 6.6.6.1
RP/0/RP0/CPU0:CRS-B#
```

In the above example, hearbeat messages are destined from 3.3.3.2 to 3.3.3.1 and 6.6.6.2 to 6.6.6.1.

Information About Carrier Grade NAT

These sections provide the information about implementation of NAT using ICMP and TCP:

Implementing NAT with ICMP

This section explains how the Network Address Translation (NAT) devices work in conjunction with Internet Control Message Protocol (ICMP).

The implementations of NAT varies in terms of how they handle different traffic.

ICMP Query Session Timeout

RFC 5508 provides ICMP Query Session timeouts. A mapping timeout is maintained by NATs for ICMP queries that traverse them. The ICMP Query Session timeout is the period during which a mapping will stay active without packets traversing the NATs. The timeouts can be set as either Maximum Round Trip Time (Maximum RTT) or Maximum Segment Lifetime (MSL). For the purpose of constraining the maximum RTT, the Maximum Segment Lifetime (MSL) is considered a guideline to set packet lifetime.

If the ICMP NAT session timeout is set to a very large duration (240 seconds) it can tie up precious NAT resources such as Query mappings and NAT Sessions for the whole duration. Also, if the timeout is set to very low it can result in premature freeing of NAT resources and applications failing to complete gracefully. The ICMP Query session timeout needs to be a balance between the two extremes. A 60-second timeout is a balance between the two extremes.

Implementing NAT with TCP

This section explains the various NAT behaviors that are applicable to TCP connection initiation. The detailed NAT with TCP functionality is defined in RFC 5382.

Address and Port Mapping Behavior

A NAT translates packets for each TCP connection using the mapping. A mapping is dynamically allocated for connections initiated from the internal side, and potentially reused for certain connections later.

Internally Initiated Connections

A TCP connection is initiated by internal endpoints through a NAT by sending SYN packet. All the external IP address and port used for translation for that connection are defined in the mapping.

Generally for the client-server applications where an internal client initiates the connection to an external server, to translate the outbound SYN, the resulting inbound SYN-ACK response mapping is used, the subsequent outbound ACK, and other packets for the connection.

The 3-way handshake corresponds to method of connection initiation.

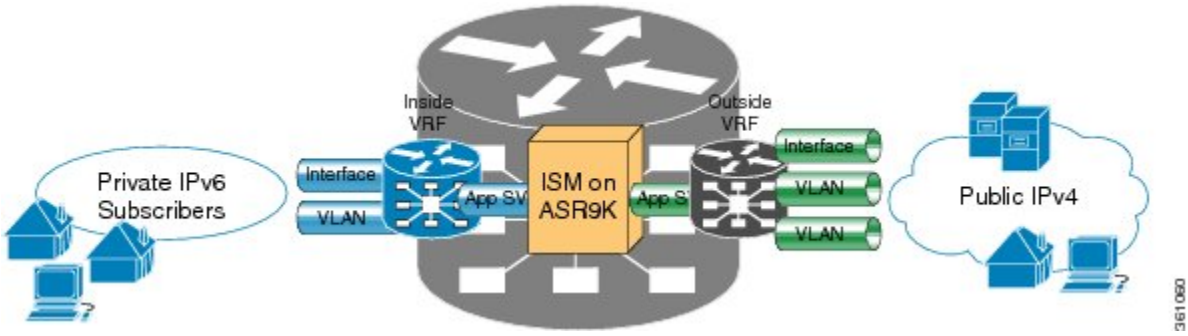
Externally Initiated Connections

For the first connection that is initiated by an internal endpoint NAT allocates the mapping. For some situations, the NAT policy may allow reusing of this mapping for connection initiated from the external side to the internal endpoint.

Implementing NAT 44 over ISM

These sections provide the information about implementation of NAT.

The following figure illustrates the implementation of NAT 44 over ISM

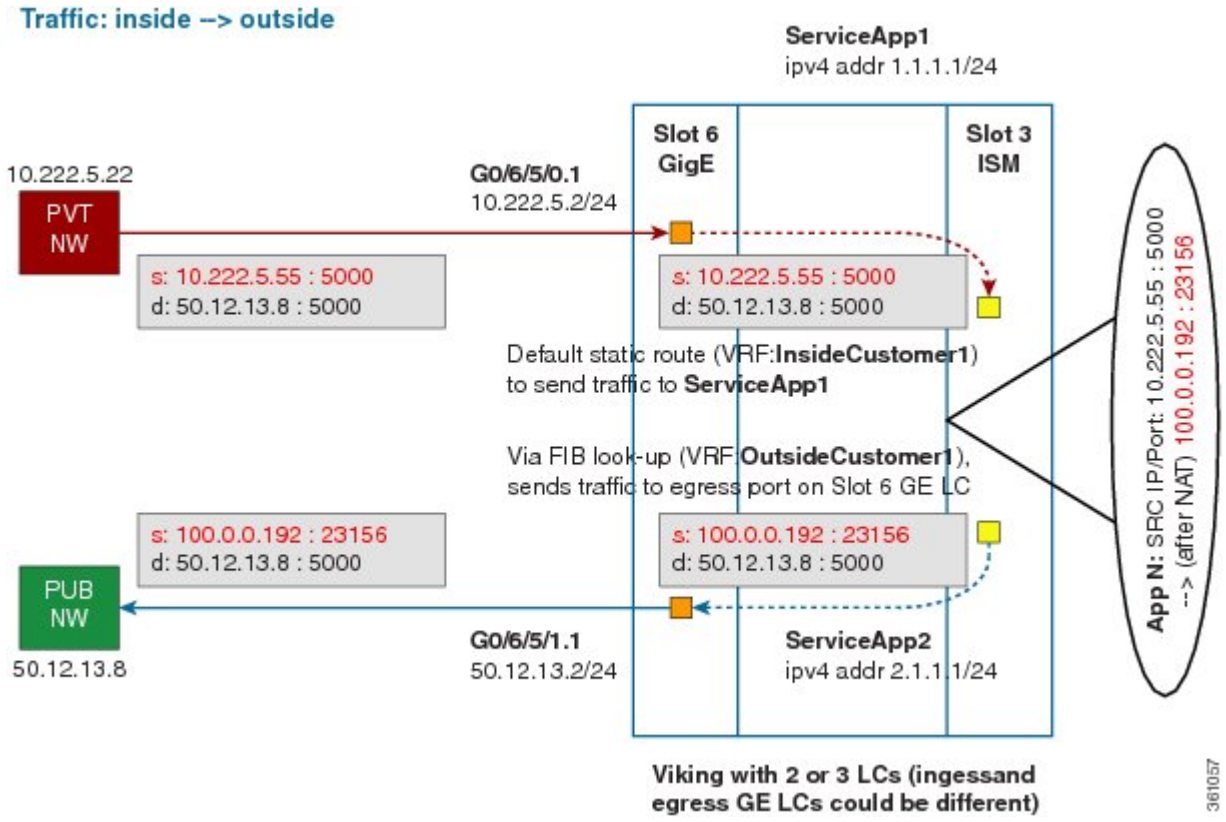


The components of this illustration are as follows:

- Private IP4 subscribers: It denotes a private network.
- Interface/VLAN: It denotes a designated interface or VLAN which is associated with the VRF.
- Inside VRF: It denotes the VRF that handles packets coming from the subscriber network. It is known as inside VRF as it forwards packets from the private network.
- App SVI: It denotes an application interface that forwards the data packet to and from the ISM. The data packet may be sent from another line card through a backplane. Because the ISM card does not have a physical interface, the APP SVI acts as a logical entry into it.

  The inside VRF is bound to an App SVI. There are 2 App SVIs required; one for the inside VRF and the other one for the outside VRF. Each App SVI pair will be associated with a unique "inside VRF" and a unique public IP address pool. The VRF consists of a static route for forwarding packets to App SVI1.
- Outside VRF: It denotes the VRF that handles packets going out to the public network. It is known as outside VRF as it forwards packets from the public network.
- Public IPV4: It denotes a public network.

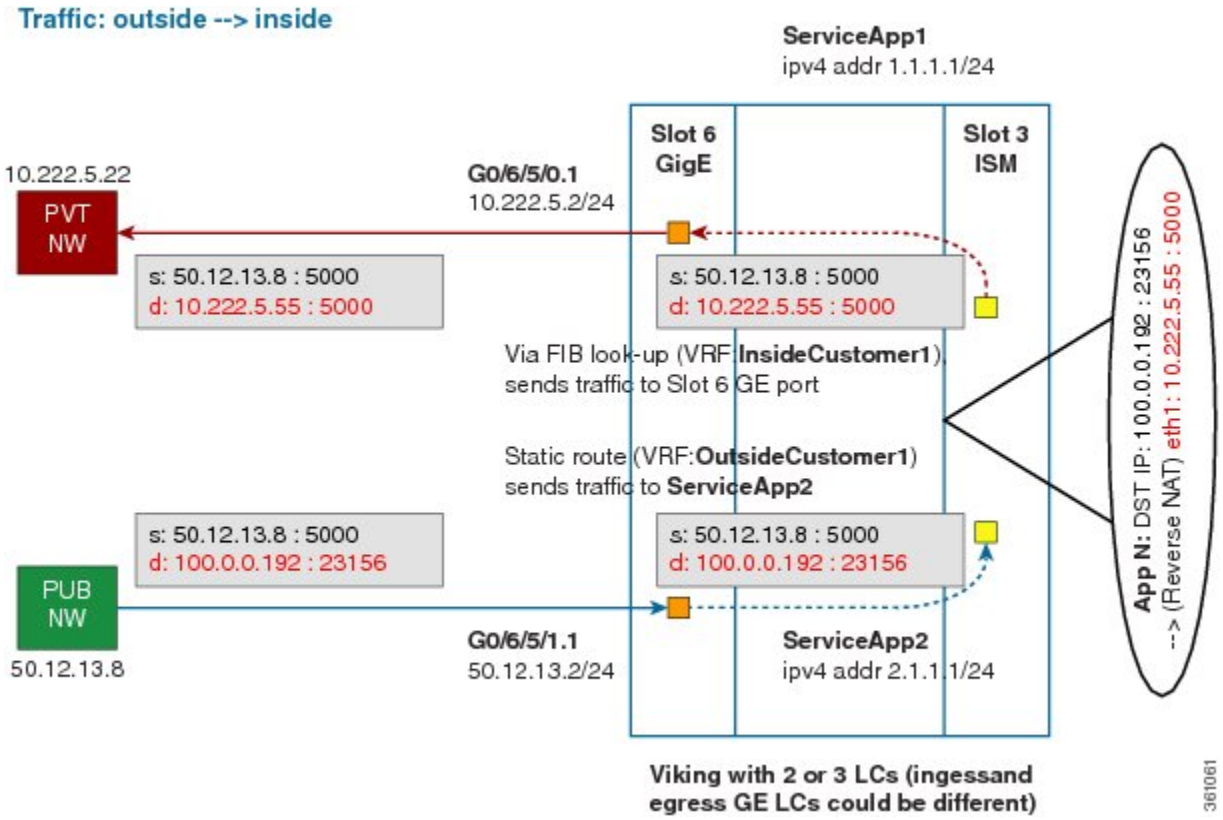
The following figure illustrates the path of the data packet from a private network to a public network in a NAT implementation.



The packet goes through the following steps when it travels from the private network to the public network:

1. In the network shown in this figure, the packet travels from the host A (having the IP address 10.222.5.55) in the private network to host B (having the IP address 5.5.5.2) in the public network. The private address has to be mapped to the public address by NAT44 that is implemented in ISM.
2. The packet enters through the ingress port on the Gigabit Ethernet (GigE) interface at Slot 0. While using NAT44, it is mandatory that the packet enters through VRF.
3. Once the packet reaches the designated interface or VLAN on ASR9K, it is forwarded to the inside VRF either through static routing or ACL-based forwarding (ABL). After the inside VRF determines that the packet needs address translation, it is forwarded to the App SVI that is bound to the VRF.
4. The packet is forwarded by AppSVI1 through a default static route (ivrf1). The destination address and the port get translated because of the CGN configuration applied on ISM.
5. The ISM applies NAT44 to the packet and a translation entry is created. The CGN determines the destination address from the FIB Look Up. It pushes the packet to the egress port.
6. The packet is then forwarded to the egress port on the interface through App SVI2. An inside VRF is mapped to an outside VRF. The outside VRF is associated with this interface. The packet is forwarded by App SVI2 through the default static route (ovrf1). Then the packet is sent to the public network.
7. The packets that do not need the address translation can bypass the App SVI and can be forwarded to the destination through a different static route and a different egress port.

The following figure illustrates the path of the packet coming from the public network to the private network.

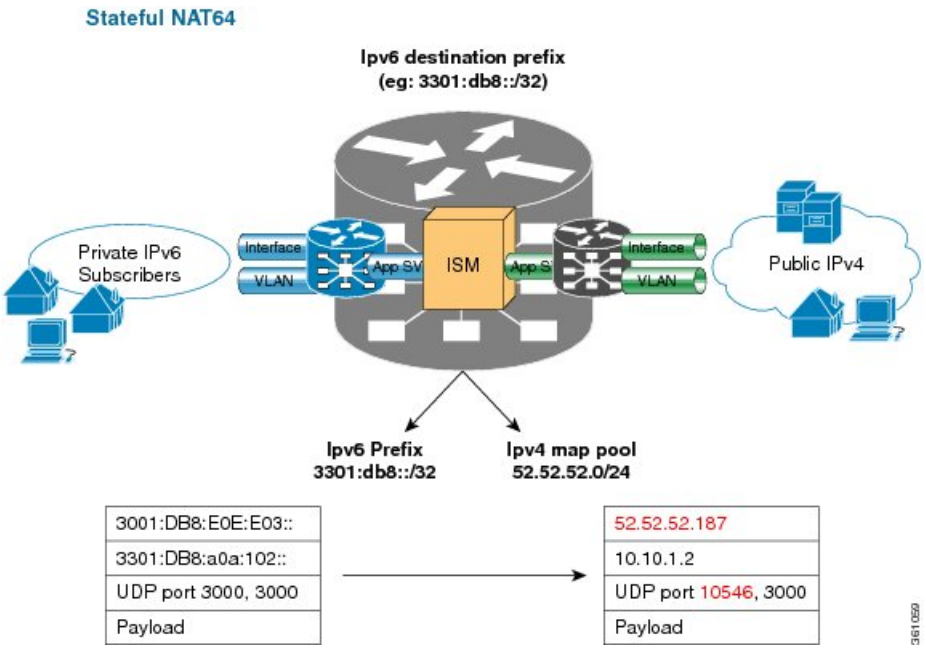


The packet goes through the following steps when it travels from the public network to the private network:

1. In the network shown in this figure, the packet travels from the host A (having the IP address 10.222.5.55) in the public network to host B (having the IP address 5.5.5.2) in the private network. The public address has to be mapped to the private address by NAT44 that is implemented in ISM.
2. The packet enters through the ingress port on the Gigabit Ethernet (GigE) interface at Slot 0.
3. Once the packet reaches the designated interface or VLAN on ASR9K, it is forwarded to the outside VRF either through static routing or ACL-based forwarding (ABL).
4. The packet is forwarded by App SVI2 through a default static route. The destination address and the port are mapped to the translated address.
5. The ISM applies NAT44 to the packet. The CGN determines the destination address from the FIB Look Up. It pushes the packet to the egress port.
6. The packet is then forwarded to the egress port on the interface through App SVI2. Then the packet is sent to the private network through the inside VRF.
7. The packets that do not need the address translation can bypass the App SVI and can be forwarded to the destination through a different static route and a different egress port.

Implementing NAT 64 over ISM

This section explains how NAT64 is implemented over ISM. The figure illustrates the implementation of NAT64 over ISM.

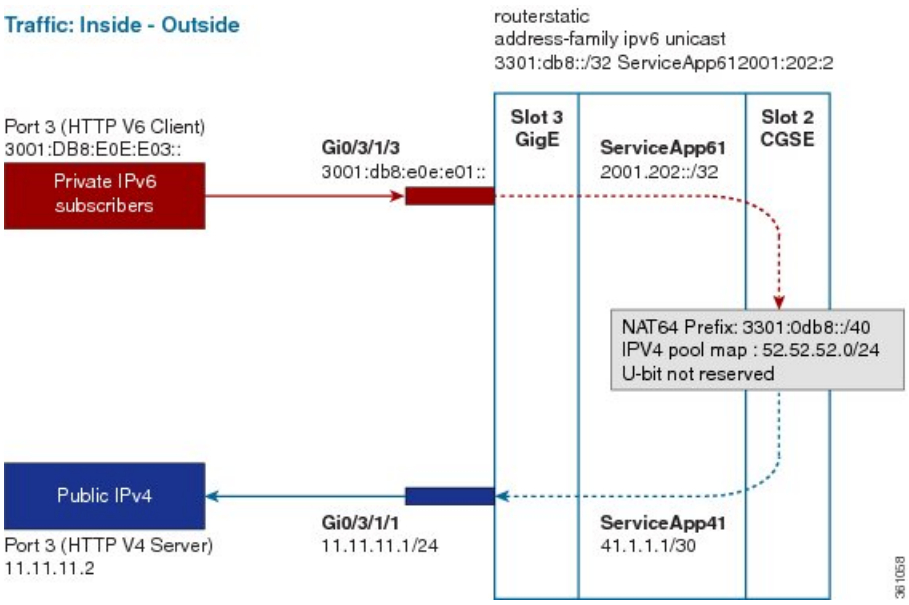


The components of this implementation are as follows:

- Private IP6 subscribers – It denotes a private network.
- Interface/VLAN- It denotes a designated interface or VLAN which is associated with the VRF.
- Inside VRF – It denotes the VRF that handles packets coming from the subscriber network. It is known as inside VRF as it forwards packets from the private network.
- App SVI- It denotes an application interface that forwards the data packet to and from the ISM. The data packet may be sent from another line card through a backplane. Because the ISM card does not have a physical interface, the APP SVI acts as a logical entry into it.

  The inside VRF is bound to an App SVI. There are 2 App SVIs required; one for the inside VRF and the other one for the outside VRF. Each App SVI pair will be associated with a unique "inside VRF" and a unique public IP address pool. The VRF consists of a static route for forwarding packets to App SVI1.
- Outside VRF- It denotes the VRF that handles packets going out to the public network. It is known as outside VRF as it forwards packets from the public network.
- Public IPV4- It denotes a public network.

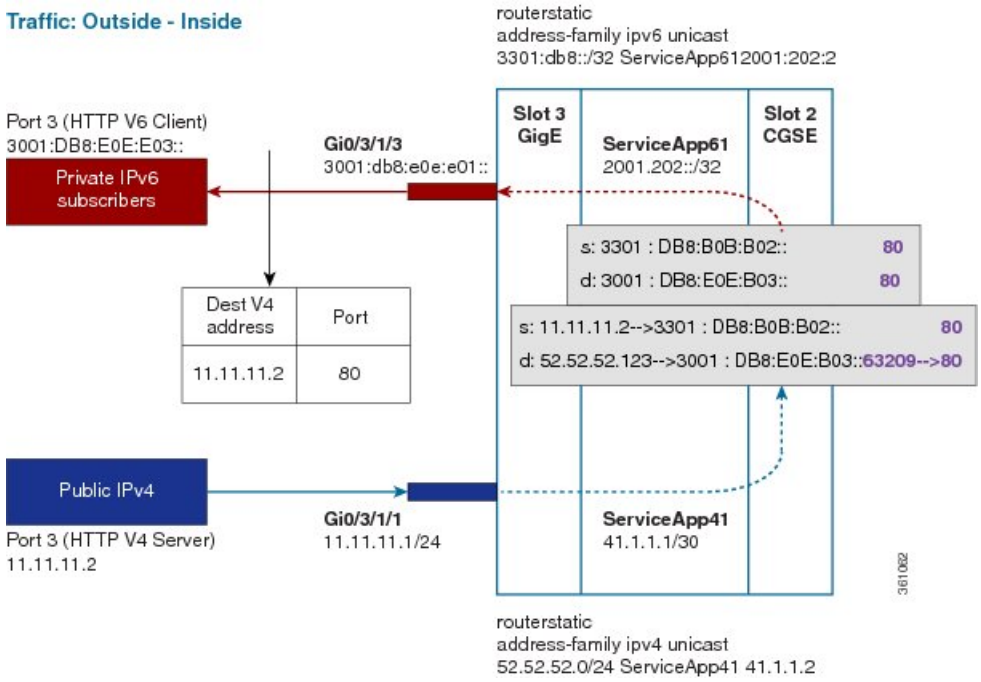
The following figure illustrates the path of the data packet from a private network to a public network in a NAT64 implementation.



The packet goes through the following steps when it travels from the private network to the public network:

1. In the network shown in this figure, the packet travels from the host A (having the IP address 3001:DB8:E0E:E03::/40) in the private network to host B (having the IP address 11.11.11.2) in the public network. The private address has to be mapped to the public address by NAT64 that is implemented in ISM.
2. The packet enters through the ingress port on the Gigabit Ethernet (GigE) interface at Slot 3.
3. Once the packet reaches the designated interface or VLAN on ASR9K, it is forwarded to the inside VRF either through static routing or ACL-based forwarding (ABL). Based on this routing decision, the packet that needs address translation is determined and is forwarded to the App SVI that is bound to the VRF.
4. The packet is forwarded by AppSVI1 through a default static route. The destination address and the port get translated because of the CGN configuration applied on ISM.
5. The ISM applies NAT64 to the packet and a translation entry is created. The CGN determines the destination address from the FIB Look Up. It pushes the packet to the egress port.
6. The packet is then forwarded to the egress port on the interface through App SVI2. The packet is forwarded by App SVI2 through the default static route. Then the packet is sent to the public network.
7. The packets that do not need the address translation can bypass the App SVI and can be forwarded to the destination through a different static route and a different egress port.

The following figure illustrates the path of the packet coming from the public network to the private network.



The packet goes through the following steps when it travels from the public network to the private network:

1. In the network shown in this figure, the packet travels from the host A (having the IP address 11.11.11.2) in the public network to host B (having the IP address 3001:DB8:E0E:E03::) in the private network. The public address has to be mapped to the private address by NAT64 that is implemented in ISM.
2. The packet enters through the ingress port on the Gigabit Ethernet (GigE) interface at Slot 3.
3. Once the packet reaches the designated interface or VLAN on ASR9K, it is forwarded to the outside VRF either through static routing or ACL-based forwarding (ABL). Based on this routing decision, the packet is forwarded to the App SVI that is bound to the VRF.
4. The packet is forwarded by App SVI2 through a default static route. The destination address and the port are mapped to the translated address.
5. The ISM applies NAT64 to the packet. The CGN determines the destination address from the FIB Look Up. It pushes the packet to the egress port.
6. The packet is then forwarded to the egress port on the interface through App SVI2. Then the packet is sent to the private network through the inside VRF.
7. The packets that do not need the address translation can bypass the App SVI and can be forwarded to the destination through a different static route and a different egress port.

Table 1: Supported Interfaces and Forwarding Features on CGv6

	4.3.x	**5.1.x**	**5.2.x**	**5.3.x**
Egress Interfaces				
Physical Interface	Yes	Yes	Yes	Yes

	4.3.x	5.1.x	5.2.x	5.3.x
VLAN Subinterface	Yes	Yes	Yes	Yes
Bundle Interface	Yes	Yes	Yes	Yes
Bundle Subinterface	Yes	Yes	Yes	Yes
BVI Interface	No	No	No	No
BNG IP-Subinterface/PPPoE	No	Yes	Yes	Yes
Ethernet Attachment Circuit or Pseudowire	No	No	No	No
GRE Tunnel	No	No	No	No
L3 Unicast Forwarding Features				
Basic IPv4 IGP Forwarding	Yes	Yes	Yes	Yes
BGP Traffic	Yes	Yes	Yes	Yes
Forwarding in VRF	Yes	Yes	Yes	Yes
Recursive Routes	Yes	Yes	Yes	Yes
uRPF	No	No	No	No
BGP-PA	No	No	No	No
MPLS and Fast Reroute (FRR) Support Note: The ISM card does not generate label for packets. It only processes unlabeled packets.				
MPLS-TE Paths	No	Yes	Yes	Yes
Basic Labeled Path	Yes	Yes	Yes	Yes
MPLS-TE Tunnel	No	Yes	Yes	Yes
MPLS-TP Tunnel	No	No	No	No
TE-FRR	No	Yes	Yes	Yes
IP-FRR	No	No	No	No
LDP-FRR or LFA-FRR	No	No	No	No
Multicast				
IP Multicast	No	No	No	No

	4.3.x	5.1.x	5.2.x	5.3.x
MVPN	No	No	No	No
Label Switched Multicast	No	No	No	No
ServiceApp Interfaces				
ABF to ServiceApp Interface	Yes	Yes	Yes	Yes
ABF from ServiceApp Interface	No	No	No	No
ACL on ServiceApp Interface	No	No	No	No
QoS on ServiceApp Interface	No	No	No	No
Lawful Intercept (LI) on Service App Interface	No	No	No	No
IPv4 Enable/Disable (Per Interface)	No	No	No	No
MPLS Enable/Disable (Per Interface)	No	No	No	No
MTU Setting (Per Interface)	No	No	No	No
Statistics on ServiceApp Interface	Partial Per-interface per-protocol packet/byte count is supported.	Partial Per-interface per-protocol packet/byte count is supported.	Yes	Yes
Per-Label/Tunnel Statistics	No	No	No	No

Note

- The table refers to packet handling after CGv6 processing (from ingress to egress).
- The CGv6 application processes only L3 unicast traffic. Other traffic types such as L2 and L3 multicast are not supported.
- The forwarding features that are supported are only those where traffic is injected from the CGv6 application as an IPv4 or IPv6 packet.

Double NAT 444

The Double NAT 444 solution offers the fastest and simplest way to address the IPv4 depletion problem without requiring an upgrade to IPv6 anywhere in the network. Service providers can continue offering new IPv4 customers access to the public IPv4 Internet by using private IPv4 address blocks, if the service provider is large enough; However, they need to have an overlapping RFC 1918 address space, which forces the service provider to partition their network management systems and creates complexity with access control lists (ACL).

Double NAT 444 uses the edge NAT and CGN to hold the translation state for each session. For example, both NATs must hold 100 entries in their respective translation tables if all the hosts in the residence of a subscriber have 100 connections to hosts on the Internet). There is no easy way for a private IPv4 host to communicate with the CGN to learn its public IP address and port information or to configure a static incoming port forwarding.

Address Family Translation

The IPv6-only to IPv4-only protocol is referred to as address family translation (AFT). The AFT translates the IP address from one address family into another address family. For example, IPv6 to IPv4 translation is called NAT 64 or IPv4 to IPv6 translation is called NAT 46.

Predefined NAT

In classic NAT, the process of mapping a private IP to a public IP or a private port to an outside port is random. Therefore, it becomes difficult to track the subscribers using an IP and a port at a given time. Predefined NAT avoids this random process by mapping a private IP address to a range of ports associated with the corresponding public IP address. This is done through an algorithm that helps the user to recognize a private IP address without having to refer to the massive CGN logs. The address and port translation is done in accordance with the algorithm.

In a predefined NAT configuration, if you want to trace a subscriber's private IP address from a public IP address and the associated port, perform the following steps:

- Whenever NAT is configured on a router or when there is a change in the existing configuration, use the following command to get the complete mapping information of private to public users:

  show cgn nat44 *instance-name* **mapping** {**inside-address | outside-address**} **inside-vrf***vrf-instance* **start-addr** *start address* [**end-addr***end address*]

  In the above command, specify the lowest address of the configured public IP pool as start address and the highest address of the pool as end address. This command dumps all the mapping for each private IP, the translated public IP, and port range. It is recommended that you divert this output in to a file and save it for future reference. Save this output to separate files each time you change the NAT44 configuration parameters and note down the time at which the changes were made and the corresponding file name.

- Whenever there is a request to trace back the subscriber's private IP address, access the right file based on the timestamp provided. The file will have the public IP and port range to which the specified port belongs. The private IP address in that row will help identify the subscriber.

Considerations and Limitations of Predefined NAT

The considerations and the limitations of the predefined mode for NAT 44 are as follows:

- You can configure the predefined mode for each of the inside VRF instance.
- A new parameter, private address range, has been added to the NAT 44 configuration for the predefined mode. You can specify a minimum of one private address range to a maximum of eight private address ranges. Ensure that you specify atleast one private address range because the available public addresses and the associated ports are mapped to the private addreses specified in this range. If the incoming packet has an address that is outside the private address range, then the packet is discarded. Ensure that the sum of all addresses should not exceed one million across all predefined mode-enabled VRFs.
- The Bulk Port Allocation configuration is not available in the predefined mode. If you try to configure Bulk Port Allocation on an inside VRF that has the predefined mode enabled, the configuration is rejected during verification.
- The port-preservation option is not available in the predefined mode.
- The global port limit parameter is not available for the predefined mode. Even though you will be allowed to configure the global port limit, the inside VRF, which has predefined mode enabled, ignores that port limit and uses the port limit configured by the algorithm.
- If you turn the predefined mode on or off for an inside VRF during the active translations, all the translations on that VRF are deleted.
- If a request for configuring static port on a private address that is not in the address range is made, the request is rejected.
- Ensure that you configure NetFlow or syslog only if it is very much required.
- Any configuration change that results in changes in mapping deletes the existing translations. Therefore, ensure that you record such configuration changes. You might need this information to trace the port usage by a subscriber.
- Ensure uniform port allocation uniform for all subscribers.

Cisco Carrier NAT Applications

These applications are deployed on the CGSE line card.

IPv4/IPv6 Stateless Translator

IPv4/IPv6 Stateless Translator (XLAT), which runs on the CRS Carrier Grade Services Engine (CGSE), enables an IPv4-only endpoint that is situated in an IPv4-only network, to communicate with an IPv6-only end-point that is situated in an IPv6-only network. This like-to-unlike address family connectivity paradigm provides backwards compatibility between IPv6 and IPv4.

A Stateless XLAT (SL-XLAT) does not create or maintain any per-session or per-flow data structures. It is an algorithmic operation performed on the IP packet headers that results in the translation of an IPv4 packet to an IPv6 packet, and vice-versa. SL-XLAT requires Cisco IOS XR Software Release 3.9.3 or 4.0.1 or 4.1.0 or later.

IPv6 Rapid Deployment

IPv6 Rapid Deployment (6RD) is a mechanism that allows service providers to provide a unicast IPv6 service to customers over their IPv4 network.

Stateful NAT64

The Stateful NAT64 (Network Address Translation 64) feature provides a translation mechanism that translates IPv6 packets into IPv4 packets and vice versa. NAT64 allows IPv6-only clients to contact IPv4 servers using unicast UDP, TCP, or ICMP. The public IPv4 address can be shared with several IPv6-only clients. NAT64 supports communication between:

- IPv6 Network and Public IPv4 Internet
- Public IPv6 Internet and IPv4 Network

NAT64 is implemented on the Cisco CRS router CGSE and CGSE plus platform. CGSE (Carrier Grade Service Engine) has four octeons and supports 20 Gbps full duplex traffic. CGSE plus has two XLP processors and supports 80 Gbps full duplex traffic. It works on Linux operating system and traffic into CGSE or CGSE plus is forwarded using serviceApp interfaces. SVIs (Service Virtual Interfaces) are configured to enable traffic to flow in and out of CGSE or CGSE plus.

Each NAT64 instance configured is associated with two serviceApps for the following purposes:

- One serviceApp is used to carry traffic from IPv6 side
- Another serviceApp is used to carry traffic from IPv4 side of the NAT64.

NAT64 instance parameters are configured using the CGN CLI. The NAT64 application in the octeons or XLP processors updates its NAT64 instance and serviceApp databases, which are used to perform the translation between IPv6 and IPv4 and vice versa.

Active CGN instance configuration is replicated in the standby CGN instance through the XR control plane. Translations that are established on the Active CGN instance are exported to the Standby CGN instance as the failure of the Active CGN affects the service until translations are re-established through normal packet flow. Service interruption is moderate for the given fault detection time and translation learning rate in terms of seconds or tens of seconds for a large translation database.

Dual Stack Lite

The Dual Stack Lite (DS-Lite) feature enables legacy IPv4 hosts and server communication over both IPv4 and IPv6 networks. Also, IPv4 hosts may need to access IPv4 internet over an IPv6 access network. The IPv4 hosts will have private addresses which need to have network address translation (NAT) completed before reaching the IPv4 internet. The Dual Stack Lite application has these components:

- **Basic Bridging BroadBand Element (B4)**: This is a Customer Premises Equipment (CPE) router that is attached to the end hosts. The IPv4 packets entering B4 are encapsulated using a IPv6 tunnel and sent to the Address Family Transition Router (AFTR).
- **Address Family Transition Router(AFTR)**: This is the router that terminates the tunnel from the B4. It decapsulates the tunneled IPv4 packet, translates the network address and routes to the IPv4 network. In the reverse direction, IPv4 packets coming from the internet are reverse network address translated and the resultant IPv4 packets are sent the B4 using a IPv6 tunnel.

The Dual Stack Lite feature helps in these functions:

1. Tunnelling IPv4 packets from CE devices over IPv6 tunnels to the CGSE blade.
2. Decapsulating the IPv4 packet and sending the decapsulated content to the IPv4 internet after completing network address translation.
3. In the reverse direction completing reverse-network address translation and then tunnelling them over IPv6 tunnels to the CPE device.

IPv6 traffic from the CPE device is natively forwarded.

VSM scale numbers supported in Dual Stack Lite

Dual Stack Lite supports the following VSM scale number:

Parameter Name	Value per VSM	Value per ASR9K Chassis with VSM
DS-Lite Sessions	80 Millions	

Port Control Protocol

Port Control Protocol (PCP) allows an IPv6 or IPv4 host to control how incoming IPv6 or IPv4 packets are translated and forwarded by a network address translator.

PCP version 1 as documented in http://tools.ietf.org/html/draft-ietf-pcp-base-19 is supported.

It also allows packets to be received from the Internet to a host and allows a host to reduce keepalive traffic of connections to a server.

Policy Functions

Application Level Gateway

The Application Level Gateway (ALG) deals with the applications that are embedded in the IP address payload. Active File Transfer Protocol (FTP), Point-to-Point Tunneling Protocol (PPTP), and Real Time Streaming Protocol (RTSP) are supported.

FTP-ALG

CGN supports both passive and active FTP. FTP clients are supported with inside (private) address and servers with outside (public) addresses. Passive FTP is provided by the basic NAT function. Active FTP is used with the ALG.

RTSP-ALG

CGN supports the Real Time Streaming Protocol (RTSP), an application-level protocol for control over the delivery of data with real-time properties. RTSP provides an extensible framework to enable controlled, on-demand delivery of real-time data, such as audio and video. Sources of data can include both live data feeds and stored clips.

PPTP-ALG

PPTP is a network protocol that enables secure transfer of data from a remote client to a private enterprise server by creating a Virtual Private Network (VPN). It is used to provide IP security at the network layer. PPTP uses a control channel over TCP and a GRE tunnel operating to encapsulate PPP packets.

PPTP-ALG is a CGN solution that allows traffic from all clients through a single PPTP tunnel.

A PPTP tunnel is instantiated on the TCP port. This TCP connection is then used to initiate and manage a second GRE tunnel to the same peer.

PPTP uses an access controller and network server to establish a connection.

PPTP Access Controller (PAC)

A device attached to one or more PSTN or ISDN lines capable of PPP operation and handling the PPTP protocol. It terminates the PPTP tunnel and provides VPN connectivity to a remote client.

PPTP Network Server (PNS)

A device which provides the interface between the Point-to-Point Protocol (encapsulated in the PPTP protocol) and a LAN or WAN. The PNS uses the PPTP protocol to support tunneling between a PPTP PAC and the PNS. It requests to establish a VPN connectivity using PPTP tunnel.

Control Connection

A control connection is established between a PAC and a PNS for TCP.

Tunnel

A tunnel carries GRE encapsulated PPP datagrams between a PAC and a PNS

Note Active FTP, PPTP ALG, and RTSP ALG are supported on NAT44 applications. Active FTP and RTSP ALG are supported on DS-Lite applications.

TCP Maximum Segment Size Adjustment

When a host initiates a TCP session with a server, the host negotiates the IP segment size by using the maximum segment size (MSS) option. The value of the MSS option is determined by the maximum transmission unit (MTU) that is configured on the host.

Static Port Forwarding

Static port forwarding helps in associating a private IP address and port with a statically allocated public IP and port. After you have configured static port forwarding, this association remains intact and does not get removed due to timeouts until the CGSE is rebooted. In case of redundant CGSE cards, it remains intact until both of the CGSEs are reloaded together or the router is reloaded. There are remote chances that after a reboot, this association might change. This feature helps in cases where server applications running on the private network needs access from public internet.

1:1 Redundancy

CGSE and CGSE Plus support 1:1 redundancy. Two CGSE or CGSE Plus cards can be placed in the active-standby configuration. The card that comes up first gets into the active mode first. If the first card that is in the active mode fails, the second card that is in the standby mode becomes active and processes the traffic. When the failure occurs, the switchover occurs within a second. This redundancy model is in the warm standby mode as the second card is already booted and preconfigured. Once it becomes active, it only has to re-establish the sessions.

Note The 1:1 redundancy feature does not support the mixing of CGSE and CGSE Plus cards. So ensure that you use either two CGSE cards or two CGSE Plus cards.

You can check the status of the redundancy of the CGSE or CGSE Plus cards by using the **show services redundancy** command.

The failover and failback operations can be forced by using the **service redundancy** command.

Back-to-Back Deployment

The CGSE and CGSE-PLUS cards can be used in Back-to-Back CRS chasis configuration. In this configuration, the active card and the standby card are in different chasis, thereby supporting inter-chasis redundancy. The performance of the cards on different chasis would be the same as it would be if they were co-allocated on the same chasis.

Note The two CGSE or CGSE-PLUS cards that are used in redundancy configuration can be in the same chassis or different chassis.

Intelligent Port Management

Intelligent Port Management is an efficient and flexible way of managing the public ports. This management process consists of the following features:

- Configuration of multiple public address pools
- Reduction of the minimum size of bulk allocation
- Configuration of port limit per VRF
- Configuration of same public pool across different NAT instances

Configuration of Multiple Public Address Pools

From this release onwards, you can create multiple pools of address for each inside VRF. This configuration currently supports 8 address pools that do not overlap with each other.

Note Ensure that you do not add more than 8 address pools as it might result in verification errors, thereby leading to the rejection of the configuration.

Some of the considerations regarding the configuration of multiple public address pools are as follows:

- The outside VRF and outside ServiceApp remain the same for different pools of addresses for a given inside VRF.

  Note If the outside VRF and the outside ServiceApp are changed, then there are chances that a subscriber packet is routed onto different outside VRFs and different ServiceApps at various times. Hence if you try to configure different address pools with different outside VRF and different ServiceApps, the configuration is rejected.

- The maximum size of the public address pool is 65536 addresses per CGN instance.
- The minimum size of the public address pool is 64 addresses.
- When a particular address pool is deleted, the associated translations are also deleted.

Reduction of the Minimum Size of Bulk Allocation

The minimum size of bulk allocation has been reduced to 8. This size can be specified by using the **bulk-port-alloc size** command.

Configuration of Port Limit per VRF

For NAT44, you can configure different port limits for different inside-VRF instances. The port limit specified per VRF overrides the port limit value specified globally. But if the port limit per VRF is not specified, then the global port limit is applied. This configuration is supported for CGSE as well as CGSE-PLUS.

If the port limit is reduced, CGSE or CGSE-PLUS card will not terminate any translation. But no new translations are created until the usage by the subscribers for that VRF falls below the port limit.

Configuration of Same Public Address Pool across Different NAT instances

A public address pool can be reused by different instances of NAT. But the address pool can be reused only by different CGN instances on different service cards.

A syslog message on the route processor (RP) appears when reused address pool is configured on the system. The message alerts the user to verify whether the reuse address pool is configured inadvertently. Any inadvertent reuse of address pool in independent and active CGN instances may result in unpredictable routing.

Two or more different instances of CGN can act as active-standby in an N:1 redundancy. In such configurations, two CGSE cards can be in active mode with different address pools. A third CGSE card can act as a common standby for both of them. In this case, it makes easy if the third CGSE card is allowed to reuse the address pools used on the other 2 CGSE cards.

Port Preservation on CGSE Plus for NAT44

During the network address and port translation (NAPT) operation, the private source port of the subscriber packets changes. (This port is also the destination port for packets in the reverse direction). The change in source port number is not acceptable in certain cases. In such cases, it is important to preserve the source port numbers during the NAPT operation; this preservation is achieved by the port-preservation feature. The port-preservation feature provides the ability to preserve ports on a best-effort basis for pre-configured ports that uses UDP or TCP transport protocol.

Restrictions

- The port-preservation feature is supported only on CGSE plus NAT44 applications.
- The port-preservation and the predefined NAT features are mutually exclusive; that is, if the user has configured the port-preservation feature, then the predefined NAT feature must not be configured, and vice-versa.
- The port-preservation feature is not supported for ICMP protocol.

Enabling the Port Preservation Feature

1. Create a Port-Set for a CGN instance, on page 25:
   - A port-set contains a set of ports for a specific transport protocol.
   - Each port-set can contain up to 20 ports per UDP or TCP transport protocol. A unique name is assigned to this port set.
   - Users can configure up to 8 port-sets in a CGN instance.
2. Attach the Port-Set to the NAT Inside-VRF Instance, on page 26:
   - A port-set is attached to the VRF instance that handles packets from the subscriber network (inside-VRF).
   - Users can attach only one port-set to the NAT inside-vrf instance. If multiple port-sets are attached to the inside-vrf instance, then only the last attached port-set is considered for the NAPT operation. However, a port-set can be attached to multiple inside-vrf instances. If a port-set is in use by one or more NAT inside-vrf instances, users cannot delete that port-set until the associations with all NAT inside-vrf instances are removed. However, the user can modify the contents of the port-set while they are in use and have the modifications take effect immediately.

Example

The following example explains the creation and the attaching of a port-set to the NAT inside-vrf instance:

```
! Defines the CGN Instance
service cgn cgn1
! Defines the port-set
port-set set1
! Defines the UDP Protocol
  protocol udp
! Defines the ports to be preserved
   preserve-port 1021 1031 1041 1101 1202 1303 1404 15015 1606
```

```
  !
! Defines the TCP Protocol
  protocol tcp
! Defines the ports to be preserved
   preserve-port 1020 1050 1100 1200 1300 1400 1500 1600
  !
!
! Defines the port-set
port-set set2

! Defines the UDP Protocol
  protocol udp
! Defines the ports to be preserved
     preserve-port 21 22 23 24 25 26 27 33 34 35 36 37 38 39 40
  !
! Defines the TCP Protocol
  protocol tcp
! Defines the ports to be preserved
   preserve-port 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
  !

!
! Defines the service type keyword definition
service-type nat44 nat1
! Configures an inside VRF instance
inside-vrf red
! Specifies the address pool for port-preservation
   map address-pool 100.1.1.0/24
! Attaches the port-set to be in use by the inside VRF instance
attach port-set set1
```

Create a Port-Set for a CGN instance

Perform this task to create a port-set for a CGN instance.

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **port-set** *name*
4. **protocol {udp | tcp}**
5. preserve-port *port1 port 2 port3*
6. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters the global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGN application and enters the CGN configuration mode.

	Command or Action	Purpose
Step 3	**port-set** *name* **Example:** `RP/0/RP0/CPU0:router(config-cgn)# port-set set1`	Configures the port-set with a name.
Step 4	**protocol {udp | tcp}** **Example:** `RP/0/RP0/CPU0:router(config-cgn-portset)# protocol udp`	Defines the UDP or TCP protocol.
Step 5	preserve-port *port1 port 2 port3* **Example:** `RP/0/RP0/CPU0:router(config-cgn-proto)# preserve-port 1021 1031 1041 1101 1202 1303 1404 15015 1606`	Defines the ports to be preserved.
Step 6	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn-proto)# end` `or` `RP/0/RP0/CPU0:router(config-cgn-proto)# commit`	commit—Saves the configuration changes and remains within the configuration session. end—Prompts user to take one of these actions: • Yes— Saves configuration changes and exits the configuration session. • No—Exits the configuration session without committing the configuration changes. • Cancel—Remains in the configuration session, without committing the configuration changes.

Attach the Port-Set to the NAT Inside-VRF Instance

Perform this task to attach the port-set to the NAT inside-vrf instance.

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-type nat44***instance-name*
4. **inside-vrf***instance-name*
5. **map address-pool** *address/prefix*
6. **attach port-set** *port-set name*
7. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters the global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGN application and enters the CGN configuration mode.
Step 3	**service-type nat44***instance-name* **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type nat44 nat1`	Configures the service type keyword definition for CGv6 NAT44 application.
Step 4	**inside-vrf***instance-name* **Example:** `RP/0/RP0/CPU0:router(config-cgn-nat44)# inside-vrf red`	Enters the inside VRF configuration mode.
Step 5	**map address-pool** *address/prefix* **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf)# map address-pool 100.1.1.0/24`	Specifies the address pool for port-preservation.
Step 6	**attach port-set** *port-set name* **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf)# attach port-set set1`	Attaches the port-set to be in use by the inside VRF instance.
Step 7	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf)# end` `or` `RP/0/RP0/CPU0:router(config-cgn-invrf)# commit`	commit—Saves the configuration changes and remains within the configuration session. end—Prompts user to take one of these actions: • Yes— Saves configuration changes and exits the configuration session. • No—Exits the configuration session without committing the configuration changes. • Cancel—Remains in the configuration session, without committing the configuration changes.

Throughput Measurement

The service card, like CGSE , has smaller throughput compared to the other cards in the platform. Therefore it drops packets at the service application interface if the traffic diverted to it is more than it can handle. Hence it becomes very important to measure the throughput for a service card. The throughput of the CGSE card for the last 1 second and the last 5 minutes can be seen by using the **show cgn** *instance-name* **utilization throughput** command.

Considerations

Some of the considerations of the throughput measurement feature are as follows:

- The traffic processed by CGSE is measured in terms of kilo bits per second (kbps) and packets per second (pps) for the last 1 second as well as the last 5 minutes.
- The throughput measurement is done for the traffic coming into CGSE , either from Inside-to-Outside direction or from Outside-to-Inside direction.

High Availability on the Data Path Service Virtual Interface (SVI)

The CGSE card already supports high availability tests to detect failures within specific CPU cores (the service-plim-ha core-to-core test) so that an alert is generated if one or more CPU cores of a CGSE card should fail. If configured, the CGSE card goes for a reload and traffic is diverted to standby (or other active cards depending on the configuration) upon detecting any core failures. The CGSE card already supports similar test to confirm the integrity of the packet path by sending test packets via ServiceInfra interface (the service-plim-ha data-path test). The card can configured to reload upon failure of this test as well.

However, till now, there were no test mechanism to confirm the integrity of path via ServiceApp interfaces (which bring in and send out subscriber traffic). With this release, a test mechanism has been added for ServiceApp interfaces configured for 6RD application (for both V4 and V6 ServiceApps). The test can be enabled via configuration. The test packets are generated from CGSE card and made to traverse through the fabric and come back in to CGSE or card via ServiceApp interfaces. Should there be a failure in receiving the packets, a syslog message is generated to alert the administrator. Optionally, the ServiceApp interfaces can be configured to be shut down upon detecting failure of this test. Shutting down the failed ServiceApp interfaces is useful in case of active-active configuration where traffic is automatically diverted to other CGSE blades and hence traffic loss can be prevented without manual intervention.

After the high availability feature is configured, the CGSE card can detect the conditions where the data path SVI (ServiceApp interface) is not able to forward traffic. If such a condition is detected, then the following actions are taken:

- A syslog message is logged by default.
- The SVI can be shut down. But you need to ensure that the packets are diverted to the SVIs of the other available CGSEs .

Considerations

Some of the considerations regarding the high availability on the data path SVIs are as follows:

- In the current release, the high availability configuration is supported only for V4 and V6 ServiceApps of 6rd application.
- In case of a failure, the syslog message is generated irrespective of the shutdown of the SVI instance.

External Logging

External logging configures the export and logging of the NAT table entries, private bindings that are associated with a particular global IP port address, and to use Netflow to export the NAT table entries.

Netflow v9 Support

The NAT44 and DS Lite features support Netflow for logging of the translation records. Logging of the translation records can be mandated by for Lawful Intercept. The Netflow uses binary format and hence requires software to parse and present the translation records.

Syslog Support

The NAT44, Stateful NAT64, and DS Lite features support Netflow for logging of the translation records. Logging of the translation records can be mandated by for Lawful Intercept. The Netflow uses binary format and hence requires software to parse and present the translation records.

In Cisco IOS XR Software Release 4.2.1 and later, the DS Lite and NAT44 features support Syslog as an alternative to Netflow. Syslog uses ASCII format and hence can be read by users. However, the log data volume is higher in Syslog than Netflow.

Bulk Port Allocation

The creation and deletion of NAT sessions need to be logged and these create huge amount of data. These are stored on Syslog collector which is supported over UDP. In order to reduce the volume of data generated by the NAT device, bulk port allocation can be enabled. When bulk port allocation is enabled and when a subscriber creates the first session, a number of contiguous outside ports are pre-allocated. A bulk allocation message is logged indicating this allocation. Subsequent session creations will use one of the pre-allocated port and hence does not require logging.

Note Session-logging and bulk port allocation are mutually exclusive.

Destination-Based Logging

Destination-Based Logging (DBL) includes destination IPv4 address and port number in the Netflow create and delete records used by NAT44, Stateful NAT64, and DS-Lite applications. It is also known as Session-Logging.

Note Session-Logging and Bulk Port Allocation are mutually exclusive.

Implementing Carrier Grade NAT on Cisco IOS XR Software

This chapter provides an overview of the implementation of Carrier Grade NAT on Cisco IOS XR Software.

Getting Started with the Carrier Grade NAT

Perform these tasks to get started with the CGN configuration tasks.

Configuring the Service Role

Perform this task to configure the service role on the specified location to start the CGN service.

Note Removal of service role is strictly not recommended while the card is active. This puts the card into FAILED state, which is service impacting.

SUMMARY STEPS

1. **configure**
2. **hw-module service cgn location node-id**
3. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**hw-module service cgn location node-id** **Example:** `RP/0/RP0/CPU0:router(config)# hw-module service cgn location 0/1/CPU0`	Configures a CGN service role on location 0/1/CPU0.
Step 3	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config)# end` `or` `RP/0/RP0/CPU0:router(config)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.

	Command or Action	Purpose
		• Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the Service Instance and Location for the Carrier Grade NAT

Perform this task to configure the service instance and location for the CGN application.

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-location preferred-active** *node-id* **[preferred-standby** *node-id***]**
4. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGN application and enters CGN configuration mode.
Step 3	**service-location preferred-active** *node-id* **[preferred-standby** *node-id***]** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-location preferred-active 0/1/CPU0 preferred-standby 0/4/CPU0`	Configures the active and standby locations for the CGN application.
Step 4	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# end` `or` `RP/0/RP0/CPU0:router(config-cgn)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?*

	Command or Action	Purpose
		[cancel]: • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the Service Virtual Interfaces

Configuring the Infrastructure Service Virtual Interface

Perform this task to configure the infrastructure service virtual interface (SVI) to forward the control traffic. The subnet mask length must be at least 30 (denoted as /30). CGSE uses SVI and it is therefore recommended that access control list (ACL) be configured to protect it from any form of denial of service attacks. For a sample ACL configuration, see Configuring ACL for a Infrastructure Service Virtual Interface: Example.

Note
- Do not remove or modify service infra interface configuration when the card is in Active state. The configuration is service affecting and the line card must be reloaded for the changes to take effect.
- When configuring CGNAT in CGSE+ cards, the IP address configured for the ServiceInfra interface must be in the x.x.x.1 format.

SUMMARY STEPS

1. **configure**
2. **interface ServiceInfra** *value*
3. **service-location** *node-id*
4. **ipv4 address** *address/mask*
5. **end** or **commit**
6. **reload**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**interface ServiceInfra** *value* **Example:** `RP/0/RP0/CPU0:router(config)# interface ServiceInfra 1` `RP/0/RP0/CPU0:router(config-if)#`	Configures the infrastructure service virtual interface (SVI) as 1 and enters CGN configuration mode.
Step 3	**service-location** *node-id* **Example:** `RP/0/RP0/CPU0:router(config-if)# service-location 0/1/CPU0`	Configures the location of the CGN service for the infrastructure SVI.
Step 4	**ipv4 address** *address/mask* **Example:** `RP/0/RP0/CPU0:router(config-if)# ipv4 address 1.1.1.1/30`	Sets the primary IPv4 address for an interface.
Step 5	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-if)# end` `or` `RP/0/RP0/CPU0:router(config-if)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.
Step 6	**reload** **Example:**	Once the configuration is complete, the card must be reloaded for changes to take effect. *WARNING: This will take the requested node out of service.*

	Command or Action	Purpose
	`RP/0/RP0/CPU0:Router#hw-mod location 0/3/cpu0 reload`	*Do you wish to continue?[confirm(y/n)] y*

Configuring the Application Service Virtual Interface (NAT44)

Perform this task to configure the application service virtual interface (SVI) to forward data traffic.

SUMMARY STEPS

1. **configure**
2. **interface ServiceApp** *value*
3. **service cgn** *instance-name* **service-type** *nat44*
4. **vrf** *vrf-name*
5. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**interface ServiceApp** *value* **Example:** `RP/0/RP0/CPU0:router(config)# interface ServiceApp 1` `RP/0/RP0/CPU0:router(config-if)#`	Configures the application SVI as 1 and enters interface configuration mode.
Step 3	**service cgn** *instance-name* **service-type** *nat44* **Example:** `RP/0/RP0/CPU0:router(config-if)# service cgn cgn1`	Configures the instance named cgn1 for the CGN application and enters CGN configuration mode.
Step 4	**vrf** *vrf-name* **Example:** `RP/0/RP0/CPU0:router(config-if)# vrf insidevrf1`	Configures the VPN routing and forwarding (VRF) for the Service Application interface
Step 5	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-if)# end` `or` `RP/0/RP0/CPU0:router(config-if)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.

	Command or Action	Purpose
		• Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the Service Type Keyword Definition

Perform this task to configure the service type key definition.

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-type nat44 nat1**
4. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGN application and enters CGN configuration mode.
Step 3	**service-type nat44 nat1** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type nat44 nat1`	Configures the service type keyword definition for CGN NAT44 application.
Step 4	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# end` `or` `RP/0/RP0/CPU0:router(config-cgn)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:*

	Command or Action	Purpose
		• Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring an Inside and Outside Address Pool Map (NAT44)

Perform this task to configure an inside and outside address pool map with the following scenarios:

Note Do not configure multiple outside address-pools to be mapped to a single inside-vrf. If you have multiple outside address-pools to be mapped, then create multiple inside-vrfs and map each outside address-pool to a single inside-vrf inside the NAT44 configuration.

- The designated address pool is used for CNAT.
- One inside VRF is mapped to only one outside VRF.
- Multiple non-overlapping address pools can be used in a specified outside VRF mapped to different inside VRF.
- Max Outside public pool per CGSE/CGN instance is 64 K or 65536 addresses. That is, if a /16 address pool is mapped, then we cannot map any other pool to that particular CGSE.
- Multiple inside vrf cannot be mapped to same outside address pool.
- While Mapping Outside Pool Minimum value for prefix is 16 and maximum value is 26.

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-type nat44 nat1**
4. **inside-vrf** *vrf-name*
5. **map** [**outside-vrf** *outside-vrf-name*] **address-pool** *address/prefix*
6. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGN application and enters CGN configuration mode.
Step 3	**service-type nat44 nat1** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type nat44 nat1`	Configures the service type keyword definition for CGN NAT44 application.
Step 4	**inside-vrf** *vrf-name* **Example:** `RP/0/RP0/CPU0:router(config-cgn-nat44)# inside-vrf insidevrf1` `RP/0/RP0/CPU0:router(config-cgn-invrf)#`	Configures an inside VRF named insidevrf1 and enters CGN inside VRF configuration mode.
Step 5	**map** [**outside-vrf** *outside-vrf-name*] **address-pool** *address/prefix* **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf)# map outside-vrf outside vrf1 address-pool 10.10.0.0/16` `or` `RP/0/RP0/CPU0:router(config-cgn-invrf)# map address-pool 100.1.0.0/16`	Configures an inside VRF to an outside VRF and address pool mapping.
Step 6	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-afi)# end` `or` `RP/0/RP0/CPU0:router(config-cgn-invrf-afi)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.

	Command or Action	Purpose
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the Policy Functions for the Carrier Grade NAT

Perform these tasks to configure the policy functions.

Configuring Port Limit per Subscriber

Perform this task to restrict the number of ports used by an IPv6 address.

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-type nat64 stateful** *instance-name*
4. **portlimit** *value*
5. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGv6 application and enters CGv6 configuration mode.
Step 3	**service-type nat64 stateful** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type nat64 stateful nat64-inst` `RP/0/RP0/CPU0:router(config-cgn-nat64-stateful)#`	Configures the service type keyword definition for CGv6 Stateful NAT64 application.
Step 4	**portlimit** *value* **Example:** `RP/0/RP0/CPU0:router(config-cgn-nat64-stateful)#portlimit 66` `RP/0/RP0/CPU0:router(config-cgn-nat64-stateful)`	Configures a value to restrict the number of ports used by an IPv6 address.

	Command or Action	Purpose
Step 5	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn-nat64-stateful)# end` `or` `RP/0/RP0/CPU0:router(config-cgn-nat64-stateful)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the Timeout Value for the Protocol

Configuring the Timeout Value for the ICMP Protocol

Perform this task to configure the timeout value for the ICMP type for the CGN instance.

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-type nat44 nat1**
4. **protocol icmp**
5. **timeout***seconds*
6. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.

	Command or Action	Purpose
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGN application and enters CGN configuration mode.
Step 3	**service-type nat44 nat1** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type nat44 nat1`	Configures the service type keyword definition for CGN NAT44 application.
Step 4	**protocol icmp** **Example:** `RP/0/RP0/CPU0:router(config-cgn-nat44)# protocol icmp` `RP/0/RP0/CPU0:router(config-cgn-proto)#`	Configures the ICMP protocol session. The example shows how to configure the ICMP protocol for the CGN instance named cgn1.
Step 5	**timeout***seconds* **Example:** `RP/0/RP0/CPU0:router(config-cgn-proto)# timeout 908`	Configures the timeout value as 908 for the ICMP session for the CGN instance named cgn1.
Step 6	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn-proto)# end` `or` `RP/0/RP0/CPU0:router(config-cgn-proto)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the Timeout Value for the TCP Session

Perform this task to configure the timeout value for either the active or initial sessions for TCP.

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-type nat44 nat1**
4. protocol tcp
5. **session** {**active** | **initial**} **timeout** *seconds*
6. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGN application and enters CGN configuration mode.
Step 3	**service-type nat44 nat1** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type nat44 nat1`	Configures the service type keyword definition for CGN NAT44 application.
Step 4	protocol tcp **Example:** `RP/0/RP0/CPU0:router(config-cgn-nat44)# protocol tcp` `RP/0/RP0/CPU0:router(config-cgn-proto)#`	Configures the TCP protocol session. The example shows how to configure the TCP protocol for the CGN instance named cgn1.
Step 5	**session** {**active**	**initial**} **timeout** *seconds* **Example:** `RP/0/RP0/CPU0:router(config-cgn-proto)# session initial timeout 90`	Configures the timeout value as 90 for the TCP session. The example shows how to configure the initial session timeout.
Step 6	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn-proto)# end` `or` `RP/0/RP0/CPU0:router(config-cgn-proto)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.

	Command or Action	Purpose
		• Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the Timeout Value for the UDP Session

Perform this task to configure the timeout value for either the active or initial sessions for UDP.

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-type nat44 nat1**
4. **protocol udp**
5. **session** {**active** | **initial**} **timeout** *seconds*
6. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGN application and enters CGN configuration mode.
Step 3	**service-type nat44 nat1** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type nat44 nat1`	Configures the service type keyword definition for CGN NAT44 application.
Step 4	**protocol udp** **Example:** `RP/0/RP0/CPU0:router(config-cgn-nat44)# protocol udp` `RP/0/RP0/CPU0:router(config-cgn-proto)#`	Configures the UDP protocol session. The example shows how to configure the UDP protocol for the CGN instance named cgn1.

	Command or Action	Purpose
Step 5	**session** {**active** | **initial**} **timeout** *seconds* **Example:** `RP/0/RP0/CPU0:router(config-cgn-proto)# session initial timeout 90`	Configures the timeout value as 90 for the UDP session. The example shows how to configure the initial session timeout.
Step 6	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn-proto)# end` `or` `RP/0/RP0/CPU0:router(config-cgn-proto)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the FTP ALG for NAT44 Instance

Perform this task to configure the FTP ALG for the specified NAT44 instance.

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-type nat44 nat1**
4. **alg activeFTP**
5. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.

	Command or Action	Purpose
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGN application and enters CGN configuration mode.
Step 3	**service-type nat44 nat1** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type nat44 nat1`	Configures the service type keyword definition for NAT44 application.
Step 4	**alg activeFTP** **Example:** `RP/0/RP0/CPU0:router(config-cgn-nat44)# alg activeFTP`	Configures the FTP ALG on the NAT44 instance.
Step 5	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# end` `or` `RP/0/RP0/CPU0:router(config-cgn)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the RTSP ALG for NAT44 Instance

Perform this task to configure the ALG for the rtsp for the specified NAT44 instance. RTSP packets are usually destined to port 554. But this is not always true because RTSP port value is configurable.

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*

3. **service-type nat44 nat1**
4. **alg rtsp [server-port]** *value*
5. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGN application and enters CGN configuration mode.
Step 3	**service-type nat44 nat1** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type nat44 nat1`	Configures the service type keyword definition for NAT44application.
Step 4	**alg rtsp [server-port]** *value* **Example:** `RP/0/RP0/CPU0:router(config-cgn-nat44)# alg rtsp server-port 5000`	Configures the rtsp ALG on the NAT44 instance for server port 5000. The range is from 1 to 65535. The default port is 554. **Caution** The option of specifying a server port) is currently not supported. Even if you configure some port, RTSP works only on the default port (554).
Step 5	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# end` `or` `RP/0/RP0/CPU0:router(config-cgn)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.

	Command or Action	Purpose
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the PPTP ALG for a NAT44 Instance

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-type nat44 nat1**
4. **alg pptpAlg**
5. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGN application and enters CGN configuration mode.
Step 3	**service-type nat44 nat1** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type nat44 nat1`	Configures the service type keyword definition for NAT44 or DS-Lite application.
Step 4	**alg pptpAlg** **Example:** `RP/0/RP0/CPU0:router(config-cgn-nat44)# alg pptpAlg`	Configures the pptp ALG on the CGN instance.
Step 5	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# end` `or` `RP/0/RP0/CPU0:router(config-cgn)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:*

	Command or Action	Purpose
		• Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the TCP Adjustment Value for the Maximum Segment Size

Perform this task to configure the adjustment value for the maximum segment size (MSS) for the VRF. You can configure the TCP MSS adjustment value on each VRF.

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-type nat44 nat1**
4. **inside-vrf** *vrf-name*
5. **protocol tcp**
6. **mss** *size*
7. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGN application and enters CGN configuration mode.
Step 3	**service-type nat44 nat1** **Example:**	Configures the service type keyword definition for CGN NAT44 application.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-cgn)# service-location preferred-active 0/1/CPU0 preferred-standby 0/4/CPU0`	
Step 4	**inside-vrf** *vrf-name* **Example:** `RP/0/RP0/CPU0:router(config-cgn-nat44)# inside-vrf insidevrf1` `RP/0/RP0/CPU0:router(config-cgn-invrf)#`	Configures the inside VRF for the CGN instance named cgn1 and enters CGN inside VRF configuration mode.
Step 5	**protocol tcp** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf)# protocol tcp` `RP/0/RP0/CPU0:router(config-cgn-invrf-proto)#`	Configures the TCP protocol session and enters CGN inside VRF AFI protocol configuration mode.
Step 6	**mss** *size* **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-afi-proto)# mss 1100`	Configures the adjustment MSS value as 1100 for the inside VRF.
Step 7	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-proto)# end` or `RP/0/RP0/CPU0:router(config-cgn-invrf-proto)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the Refresh Direction for the Network Address Translation

Perform this task to configure the NAT mapping refresh direction as outbound for TCP and UDP traffic.

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-type nat44 nat1**
4. **refresh-direction Outbound**
5. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGN application and enters CGN configuration mode.
Step 3	**service-type nat44 nat1** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type nat44 nat1`	Configures the service type keyword definition for NAT44 application.
Step 4	**refresh-direction Outbound** **Example:** `RP/0/RP0/CPU0:router(config-cgn-nat44)# protocol tcp` `RP/0/RP0/CPU0:router(config-cgn-proto)#refresh-direction Outbound`	Configures the NAT mapping refresh direction as outbound for the CGN instance named cgn1.
Step 5	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# end` `or` `RP/0/RP0/CPU0:router(config-cgn)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.

	Command or Action	Purpose
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the Carrier Grade NAT for Static Port Forwarding

Perform this task to configure CGN for static port forwarding for reserved or nonreserved port numbers.

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-type nat44 nat1**
4. **inside-vrf** *vrf-name*
5. **protocol tcp**
6. **static-forward inside**
7. **address** *address* **port** *number*
8. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGN application and enters CGN configuration mode.
Step 3	**service-type nat44 nat1** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type nat44 nat1`	Configures the service type keyword definition for NAT44 application.
Step 4	**inside-vrf** *vrf-name* **Example:** `RP/0/RP0/CPU0:router(config-cgn-nat44)# inside-vrf insidevrf1` `RP/0/RP0/CPU0:router(config-cgn-invrf)#`	Configures the inside VRF for the CGN instance named cgn1 and enters CGN inside VRF configuration mode.
Step 5	**protocol tcp** **Example:**	Configures the TCP protocol session and enters CGN inside VRF AFI protocol configuration mode.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-cgn-invrf)# protocol tcp` `RP/0/RP0/CPU0:router(config-cgn-invrf-proto)#`	
Step 6	**static-forward inside** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-proto)# static-forward inside` `RP/0/RP0/CPU0:router(config-cgn-ivrf-sport-inside)#`	Configures the CGN static port forwarding entries on reserved or nonreserved ports and enters CGN inside static port inside configuration mode.
Step 7	**address** *address* **port** *number* **Example:** `RP/0/RP0/CPU0:router(config-cgn-ivrf-sport-inside)# address 1.2.3.4 port 90`	Configures the CGN static port forwarding entries for the inside VRF.
Step 8	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn-ivrf-sport-inside)# end` `or` `RP/0/RP0/CPU0:router(config-cgn-ivrf-sport-inside)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the Dynamic Port Ranges for NAT44

Perform this task to configure dynamic port ranges for TCP, UDP, and ICMP ports. The default value range of 0 to 1023 is preserved and not used for dynamic translations. Therefore, if the value of **dynamic port range start** is not configured explicitly, the dynamic port range value starts at 1024.

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-type nat44 nat1**

4. **dynamic port range start** *value*
5. **end** or **commit**
6.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGN application and enters CGN configuration mode.
Step 3	**service-type nat44 nat1** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type nat44 nat1`	Configures the service type keyword definition for NAT44 application.
Step 4	**dynamic port range start** *value* **Example:** `RP/0/RP0/CPU0:router(config-cgn-nat44)# dynamic port range start 666`	Configures the value of dynamic port range start for a CGN NAT 44 instance. The value can range from 1 to 65535. The default value is 1024.
Step 5	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn-ivrf-sport-inside)# end` `or` `RP/0/RP0/CPU0:router(config-cgn-ivrf-sport-inside)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

	Command or Action	Purpose
Step 6	**Example:**	

Example

What to do next

-

Configuring 1:1 Redundancy

Perform the following steps to configure 1:1 redundancy on CGSE or CGSE Plus cards.

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-location preferred-active** *node-id* **[preferred-standby** *node-id***]**
4. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGN application and enters CGN configuration mode.
Step 3	**service-location preferred-active** *node-id* **[preferred-standby** *node-id***]** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-location preferred-active 0/1/CPU0 preferred-standby 0/4/CPU0`	Configures the active and standby locations for the CGN application.
Step 4	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# end` `or` `RP/0/RP0/CPU0:router(config-cgn)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:*

	Command or Action	Purpose
		• Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Multiple Public Address Pools

Perform the following steps to configure multiple public address pools for an inside VRF.

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-type nat44 nat1**
4. **inside-vrf** *vrf-name*
5. **map** [**outside-vrf** *outside-vrf-name*] **address-pool** *address/prefix*
6. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGN application and enters CGN configuration mode.
Step 3	**service-type nat44 nat1** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type nat44 nat1`	Configures the service type keyword definition for CGN NAT44 application.

	Command or Action	Purpose
Step 4	**inside-vrf** *vrf-name* **Example:** `RP/0/RP0/CPU0:router(config-cgn-nat44)# inside-vrf insidevrf1` `RP/0/RP0/CPU0:router(config-cgn-invrf)#`	Configures an inside VRF named insidevrf1 and enters CGN inside VRF configuration mode.
Step 5	**map** [**outside-vrf** *outside-vrf-name*] **address-pool** *address/prefix* **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf)# map outside-vrf outside vrf1 address-pool 10.10.0.0/16` or `RP/0/RP0/CPU0:router(config-cgn-invrf)# map address-pool 100.1.0.0/16`	Configures an inside VRF to an outside VRF and address pool mapping.
Step 6	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-afi)# end` or `RP/0/RP0/CPU0:router(config-cgn-invrf-afi)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Port Limit per VRF

Perform the following steps to configure the port limit per VRF:

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-type nat44 nat1**
4. **portlimit** *value*

5. **inside-vrf** *vrf-name*
6. **portlimit** *value*
7. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGN application and enters CGN configuration mode.
Step 3	**service-type nat44 nat1** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type nat44 nat1`	Configures the service type keyword definition for CGN NAT44 application.
Step 4	**portlimit** *value* **Example:** `RP/0/RP0/CPU0:router(config-cgn-nat44)# portlimit 100`	Limits the number of entries per address for each subscriber of the system
Step 5	**inside-vrf** *vrf-name* **Example:** `RP/0/RP0/CPU0:router(config-cgn-nat44)# inside-vrf insidevrf1` `RP/0/RP0/CPU0:router(config-cgn-invrf)#`	Configures an inside VRF named insidevrf1 and enters CGN inside VRF configuration mode.
Step 6	**portlimit** *value* **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf)# portlimit 300`	Sets the port limit per VRF. The port limit value per VRF (300) overrides the port limit value specified globally (100) . The port limit value per VRF (300) is applied to all the subscribers in the insidevrf1.
Step 7	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# end` `or` `RP/0/RP0/CPU0:router(config-cgn)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.

	Command or Action	Purpose
		• Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Same Address Pool for Different NAT Instances

Perform the following steps to configure same address pool for different NAT instances. Note that in the following sample configuration, there are 2 NAT instances, **cgn1** and **cgn2**.

SUMMARY STEPS

1. **configure**
2. **service cgn** *cgn1*
3. **service-type nat44***nat1*
4. **inside-vrf** *insidevrf1*
5. **map address-pool** *address/prefix*
6. **end** or **commit**
7. **configure**
8. **service cgn** *cgn2*
9. **service-type nat44 nat2**
10. **inside-vrf** *insidevrf2*
11. **map address-pool** *address/prefix*
12. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**service cgn** *cgn1* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGN application and enters CGN configuration mode.
Step 3	**service-type nat44***nat1* **Example:**	Configures the service type keyword definition for CGN NAT44 application.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-cgn)# service-type nat44 nat1`	
Step 4	**inside-vrf** *insidevrf1* **Example:** `RP/0/RP0/CPU0:router(config-cgn-nat44)# inside-vrf insidevrf1` `RP/0/RP0/CPU0:router(config-cgn-invrf)#`	Configures an inside VRF named insidevrf1 and enters CGN inside VRF configuration mode.
Step 5	**map address-pool** *address/prefix* **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf)# map address-pool 100.1.0.0/16`	Configures an inside VRF to an outside VRF and address pool mapping.
Step 6	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-afi)# end` `or` `RP/0/RP0/CPU0:router(config-cgn-invrf-afi)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.
Step 7	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 8	**service cgn** *cgn2* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn2` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures another NAT instance **cgn2** for the CGN application and enters CGN configuration mode. The NAT instance, cgn2, can access the same address pool created in the first NAT instance, **cgn1**. Hence you do not have to specify a different address pool here.
Step 9	**service-type nat44 nat2** **Example:**	Configures the service type for CGN NAT44 application.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-cgn)# service-type nat44 nat1`	
Step 10	**inside-vrf** *insidevrf2* **Example:** `RP/0/RP0/CPU0:router(config-cgn-nat44)# inside-vrf insidevrf2` `RP/0/RP0/CPU0:router(config-cgn-invrf)#`	Configures an inside VRF named insidevrf2 and enters CGN inside VRF configuration mode.
Step 11	**map address-pool** *address/prefix* **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf)# map address-pool 100.1.0.0/16`	Configures an inside VRF to an outside VRF and address pool mapping.
Step 12	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# end` `or` `RP/0/RP0/CPU0:router(config-cgn)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring High Availability of Data Path Service Virtual Interface (SVI)

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-type tunnel v6rd** *instance-name*
4. **end** or **commit**
5. **address-family ipv4**
6. **interface ServiceApp41**
7. **address-family ipv6**

8. **interface ServiceApp61**
9. **exit**
10. **datapath-test**
11. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config-if)# service cgn cgn1`	Configures the instance named cgn1 for the CGv6 application, and enters CGv6 configuration mode.
Step 3	**service-type tunnel v6rd** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type tunnel v6rd 6rd1` `RP/0/RP0/CPU0:router(config-cgn-tunnel-6rd)`	Configures the service-type as tunnel v6rd, and the instance name as 6rd1.
Step 4	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn-tunnel-v6rd)# end` or `RP/0/RP0/CPU0:router(config-cgn-tunnel-v6rd)# commit`	Saves configuration changes. • When you issue the end command, the system prompts you to commit changes: `Uncommitted changes found, commit them before exiting (yes/no/cancel)? [cancel]:` • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.
Step 5	**address-family ipv4** **Example:**	Enters the address family IPv4 configuration mode.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-cgn-tunnel-6rd)# address-family ipv4`	
Step 6	**interface ServiceApp41** **Example:** `RP/0/RP0/CPU0:router(config-cgn-tunnel-6rd-afi)# interface ServiceApp41`	Specifies the ServiceApp on which IPv4 traffic enters and leaves.
Step 7	**address-family ipv6** **Example:** `RP/0/RP0/CPU0:router(config-cgn-6rd-afi)# address-family ipv6`	Enters the address family IPv6 configuration mode.
Step 8	**interface ServiceApp61** **Example:** `RP/0/RP0/CPU0:router(config-cgn-6rd-afi)# interface ServiceApp61`	Specifies the ServiceApp on which IPv6 traffic enters and leaves.
Step 9	**exit** **Example:** `RP/0/RP0/CPU0:router(config-cgn-6rd-afi)# exit` `RP/0/RP0/CPU0:router(config-cgn-6rd)#`	Exits the address family configuration mode.
Step 10	**datapath-test** **Example:** `RP/0/RP0/CPU0:router(config-cgn-6rd)# datapath-test`	Specifies the ServiceApp on which IPv6 traffic enters and leaves.
Step 11	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn-tunnel-v6rd)# end` or `RP/0/RP0/CPU0:router(config-cgn-tunnel-v6rd)# commit`	Saves configuration changes. • When you issue the end command, the system prompts you to commit changes: `Uncommitted changes found, commit them before exiting (yes/no/cancel)? [cancel]:` • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the Export and Logging for the Network Address Translation Table Entries

Configuring the Server Address and Port for Netflow Logging

Perform this task to configure the server address and port to log network address translation (NAT) table entries for Netflow logging.

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-type nat44 nat1**
4. **inside-vrf** *vrf-name*
5. **external-logging netflowv9**
6. **server**
7. **address** *address* **port** *number*
8. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGN application and enters CGN configuration mode.
Step 3	**service-type nat44 nat1** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type nat44 nat1`	Configures the service type keyword definition for NAT44 application.
Step 4	**inside-vrf** *vrf-name* **Example:** `RP/0/RP0/CPU0:router(config-cgn)# inside-vrf insidevrf1` `RP/0/RP0/CPU0:router(config-cgn-invrf)#`	Configures the inside VRF for the CGN instance named cgn1 and enters CGN inside VRF configuration mode.
Step 5	**external-logging netflowv9** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf)# external-logging netflowv9` `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog)#`	Configures the external-logging facility for the CGN instance named cgn1 and enters CGN inside VRF address family external logging configuration mode.
Step 6	**server** **Example:**	Configures the logging server information for the IPv4 address and port for the server that is used for the

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog)# server` `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)#`	netflowv9-based external-logging facility and enters CGN inside VRF address family external logging server configuration mode.
Step 7	**address** *address* **port** *number* **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)# address 2.3.4.5 port 45`	Configures the IPv4 address and port number 45 to log Netflow entries for the NAT table.
Step 8	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)# end` `or` `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the Path Maximum Transmission Unit for Netflow Logging

Perform this task to configure the path maximum transmission unit (MTU) for the netflowv9-based external-logging facility for the inside VRF.

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-type nat44 nat1**
4. **inside-vrf** *vrf-name*
5. **external-logging netflowv9**
6. **server**
7. **path-mtu** *value*
8. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGN application and enters CGN configuration mode.
Step 3	**service-type nat44 nat1** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type nat44 nat1`	Configures the service type keyword definition for NAT44 application.
Step 4	**inside-vrf** *vrf-name* **Example:** `RP/0/RP0/CPU0:router(config-cgn)# inside-vrf insidevrf1` `RP/0/RP0/CPU0:router(config-cgn-invrf)#`	Configures the inside VRF for the CGN instance named cgn1 and enters CGN inside VRF configuration mode.
Step 5	**external-logging netflowv9** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf)# external-logging netflowv9` `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog)#`	Configures the external-logging facility for the CGN instance named cgn1 and enters CGN inside VRF address family external logging configuration mode.
Step 6	**server** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog)# server` `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)#`	Configures the logging server information for the IPv4 address and port for the server that is used for the netflowv9-based external-logging facility and enters CGN inside VRF address family external logging server configuration mode.
Step 7	**path-mtu** *value* **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)# path-mtu 2900`	Configures the path MTU with the value of 2900 for the netflowv9-based external-logging facility.
Step 8	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)# end` `or` `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:*

	Command or Action	Purpose
		• Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the Refresh Rate for Netflow Logging

Perform this task to configure the refresh rate at which the Netflow-v9 logging templates are refreshed or resent to the Netflow-v9 logging server.

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-type nat44 nat1**
4. **inside-vrf** *vrf-name*
5. **external-logging netflowv9**
6. **server**
7. **refresh-rate** *value*
8. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGN application and enters CGN configuration mode.
Step 3	**service-type nat44 nat1** **Example:**	Configures the service type keyword definition for NAT44 application.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-cgn)# service-type nat44 nat1`	
Step 4	**inside-vrf** *vrf-name* **Example:** `RP/0/RP0/CPU0:router(config-cgn)# inside-vrf insidevrf1` `RP/0/RP0/CPU0:router(config-cgn-invrf)#`	Configures the inside VRF for the CGN instance named cgn1 and enters CGN inside VRF configuration mode.
Step 5	**external-logging netflowv9** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf)# external-logging netflowv9` `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog)#`	Configures the external-logging facility for the CGN instance named cgn1 and enters CGN inside VRF address family external logging configuration mode.
Step 6	**server** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog)# server` `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)#`	Configures the logging server information for the IPv4 address and port for the server that is used for the netflowv9-based external-logging facility and enters CGN inside VRF address family external logging server configuration mode.
Step 7	**refresh-rate** *value* **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)# refresh-rate 50`	Configures the refresh rate value of 50 to log Netflow-based external logging information for an inside VRF.
Step 8	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)# end` or `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring Session-Logging for a NAT44 or DS-Lite Instance

Perform this task to enable session-logging if destination IP and Port information needs to logged in the Netflow records for each NAT44 or DS-Lite instance.

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-type nat44 nat1** or **service-type ds-lite ds-lite1**
4. **inside-vrf** *vrf-name*
5. **external-logging netflowv9** or **external-loging netflow9**
6. **server**
7. **session-logging**
8. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGN application and enters CGN configuration mode.
Step 3	**service-type nat44 nat1** or **service-type ds-lite ds-lite1** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type nat44 nat1` `or` `RP/0/RP0/CPU0:router(config-cgn)# service-type ds-lite ds-lite1`	Configures the service type keyword definition for NAT44 or DS-Lite application.
Step 4	**inside-vrf** *vrf-name* **Example:** `RP/0/RP0/CPU0:router(config-cgn)# inside-vrf insidevrf1` `RP/0/RP0/CPU0:router(config-cgn-invrf)#`	Configures the inside VRF for the CGN instance named cgn1 and enters CGN inside VRF configuration mode.
Step 5	**external-logging netflowv9** or **external-loging netflow9** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf)# external-logging netflowv9` `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog)#` `or` `RP/0/RP0/CPU0:router(config-cgn)# external-logging netflow9` `RP/0/RP0/CPU0:router(config-cgn-ds-lite-extlog)#`	Configures the external-logging facility for the NAT44 or DS-Lite instance.

	Command or Action	Purpose
Step 6	**server** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog)# server` `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)#` `or` `RP/0/RP0/CPU0:router(config-cgn-ds-lite-extlog)# server` `RP/0/RP0/CPU0:router(config-cgn-ds-lite-extlog-server)#`	Configures the logging server information for the IPv4 address and port for the server that is used for the netflow-v9 based external-logging facility.
Step 7	**session-logging** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)# session-logging` `or` `RP/0/RP0/CPU0:router(config-cgn-ds-lite-extlog-server)# session-logging`	Configures the session logging for a NAT44 or DS-Lite instance.
Step 8	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)# end` `or` `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)# commit` `RP/0/RP0/CPU0:router(config-cgn-ds-lite-extlog-server)# end` `or` `RP/0/RP0/CPU0:router(config-cgn-ds-lite-extlog-server)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the Timeout for Netflow Logging

Perform this task to configure the frequency in minutes at which the Netflow-V9 logging templates are to be sent to the Netflow-v9 logging server.

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-type nat44 nat1**
4. **inside-vrf** *vrf-name*
5. **external-logging netflowv9**
6. **server**
7. **timeout***value*
8. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGN application and enters CGN configuration mode.
Step 3	**service-type nat44 nat1** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type nat44 nat1`	Configures the service type keyword definition for NAT44 application.
Step 4	**inside-vrf** *vrf-name* **Example:** `RP/0/RP0/CPU0:router(config-cgn)# inside-vrf insidevrf1` `RP/0/RP0/CPU0:router(config-cgn-invrf)#`	Configures the inside VRF for the CGN instance named cgn1 and enters CGN inside VRF configuration mode.
Step 5	**external-logging netflowv9** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf)# external-logging netflowv9` `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog)#`	Configures the external-logging facility for the CGN instance named cgn1 and enters CGN inside VRF address family external logging configuration mode.
Step 6	**server** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog)# server` `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)#`	Configures the logging server information for the IPv4 address and port for the server that is used for the netflowv9-based external-logging facility and enters CGN inside VRF address family external logging server configuration mode.
Step 7	**timeout***value* **Example:**	Configures the timeout value of 50 for Netflow logging of NAT table entries for an inside VRF.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)# timeout 50`	
Step 8	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)# end` `or` `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring NAT44 on ISM

Perform these tasks to configure NAT44 on ISM.

Configuring the Application Service Virtual Interface (NAT44)

Perform this task to configure the application service virtual interface (SVI) to forward data traffic.

SUMMARY STEPS

1. **configure**
2. **interface ServiceApp** *value*
3. **service cgn** *instance-name* **service-type** *nat44*
4. **vrf** *vrf-name*
5. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:**	Enters global configuration mode.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router# configure`	
Step 2	**interface ServiceApp** *value* **Example:** `RP/0/RP0/CPU0:router(config)# interface ServiceApp 1` `RP/0/RP0/CPU0:router(config-if)#`	Configures the application SVI as 1 and enters interface configuration mode.
Step 3	**service cgn** *instance-name* **service-type** *nat44* **Example:** `RP/0/RP0/CPU0:router(config-if)# service cgn cgn1`	Configures the instance named cgn1 for the CGN application and enters CGN configuration mode.
Step 4	**vrf** *vrf-name* **Example:** `RP/0/RP0/CPU0:router(config-if)# vrf insidevrf1`	Configures the VPN routing and forwarding (VRF) for the Service Application interface
Step 5	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-if)# end` `or` `RP/0/RP0/CPU0:router(config-if)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring a NAT44 Instance

Perform this task to configure a NAT44 instance.

Note The system does not support deleting VRF on live traffic in the following scenarios:

- If you are in the global configuration mode.
- If you are within the CGN instance.
- If you are in the static route table.

SUMMARY STEPS

1. **configure**
2. **service cgn nat44***instance-name*
3. **service-location preferred-active** *VSM location*
4. **service-type nat44 nat1**
5. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**service cgn nat44***instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGv6 NAT44 application and enters CGv6 configuration mode.
Step 3	**service-location preferred-active** *VSM location* **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-location preferred-active 0/3/CPU0`	Configures the NAT preferred active VSM location.
Step 4	**service-type nat44 nat1** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type nat44 nat1`	Configures the service type keyword definition for CGv6 NAT44 application.
Step 5	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# end` `or` `RP/0/RP0/CPU0:router(config-cgn)# commit`	Saves configuration changes. • When you issue the end command, the system prompts you to commit changes: `Uncommitted changes found, commit them before exiting (yes/no/cancel)?` `[cancel]:`

	Command or Action	Purpose
		• Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring an Inside and Outside Address Pool Map (NAT44)

Perform this task to configure an inside and outside address pool map with the following scenarios:

Note Do not configure multiple outside address-pools to be mapped to a single inside-vrf. If you have multiple outside address-pools to be mapped, then create multiple inside-vrfs and map each outside address-pool to a single inside-vrf inside the NAT44 configuration.

- The designated address pool is used for CNAT.
- One inside VRF is mapped to only one outside VRF.
- Multiple non-overlapping address pools can be used in a specified outside VRF mapped to different inside VRF.
- Max Outside public pool per CGSE/CGN instance is 64 K or 65536 addresses. That is, if a /16 address pool is mapped, then we cannot map any other pool to that particular CGSE.
- Multiple inside vrf cannot be mapped to same outside address pool.
- While Mapping Outside Pool Minimum value for prefix is 16 and maximum value is 26.

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-type nat44 nat1**
4. **inside-vrf** *vrf-name*
5. **map** [**outside-vrf** *outside-vrf-name*] **address-pool** *address/prefix*
6. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGN application and enters CGN configuration mode.
Step 3	**service-type nat44 nat1** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type nat44 nat1`	Configures the service type keyword definition for CGN NAT44 application.
Step 4	**inside-vrf** *vrf-name* **Example:** `RP/0/RP0/CPU0:router(config-cgn-nat44)# inside-vrf insidevrf1` `RP/0/RP0/CPU0:router(config-cgn-invrf)#`	Configures an inside VRF named insidevrf1 and enters CGN inside VRF configuration mode.
Step 5	**map** [**outside-vrf** *outside-vrf-name*] **address-pool** *address/prefix* **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf)# map outside-vrf outside vrf1 address-pool 10.10.0.0/16` `or` `RP/0/RP0/CPU0:router(config-cgn-invrf)# map address-pool 100.1.0.0/16`	Configures an inside VRF to an outside VRF and address pool mapping.
Step 6	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-afi)# end` `or` `RP/0/RP0/CPU0:router(config-cgn-invrf-afi)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.

	Command or Action	Purpose
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Policy Functions

Configuring Port Limit per Subscriber

Perform this task to restrict the number of ports used by an IPv6 address.

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-type nat64 stateful** *instance-name*
4. **portlimit** *value*
5. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGv6 application and enters CGv6 configuration mode.
Step 3	**service-type nat64 stateful** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type nat64 stateful nat64-inst` `RP/0/RP0/CPU0:router(config-cgn-nat64-stateful)#`	Configures the service type keyword definition for CGv6 Stateful NAT64 application.
Step 4	**portlimit** *value* **Example:** `RP/0/RP0/CPU0:router(config-cgn-nat64-stateful)#portlimit 66` `RP/0/RP0/CPU0:router(config-cgn-nat64-stateful)`	Configures a value to restrict the number of ports used by an IPv6 address.
Step 5	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn-nat64-stateful)# end`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes:

	Command or Action	Purpose
	or RP/0/RP0/CPU0:router(config-cgn-nat64-stateful)# commit	*Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the Timeout Value for ICMP, TCP and UDP Sessions

Perform this task to configure the timeout value for ICMP, TCP or UDP sessions for a Dual Stack Lite (DS Lite) instance:

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-type ds-lite** *instance-name*
4. **protocol tcp session {active | initial} timeout** *value* or **protocol {icmp | udp} timeout** ***value***
5. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** RP/0/RP0/CPU0:router# configure	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** RP/0/RP0/CPU0:router(config)# service cgn cgn1 RP/0/RP0/CPU0:router(config-cgn)#	Configures the instance named cgn1 for the CGv6 application and enters CGv6 configuration mode.
Step 3	**service-type ds-lite** *instance-name* **Example:**	Configures the service type keyword definition for CGv6 DS-Lite application.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-cgn)# service-type ds-lite ds-lite-inst` `RP/0/RP0/CPU0:router(config-cgn-ds-lite)#`	
Step 4	**protocol tcp session {active | initial} timeout** *value* or **protocol {icmp | udp} timeout** ***value*** **Example:** `RP/0/RP0/CPU0:router(config-cgn-ds-lite)#protocol tcp session active timeout 90` `or` `protocol icmp timeout 90` `RP/0/RP0/CPU0:router(config-cgn-ds-lite)`	Configures the initial and active session timeout values for TCP. Configures the timeout value in seconds for ICMP and UDP.
Step 5	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn-ds-lite)# end` `or` `RP/0/RP0/CPU0:router(config-cgn-ds-lite)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the FTP ALG for NAT44 Instance

Perform this task to configure the FTP ALG for the specified NAT44 instance.

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-type nat44 nat1**
4. **alg activeFTP**
5. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGN application and enters CGN configuration mode.
Step 3	**service-type nat44 nat1** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type nat44 nat1`	Configures the service type keyword definition for NAT44 application.
Step 4	**alg activeFTP** **Example:** `RP/0/RP0/CPU0:router(config-cgn-nat44)# alg activeFTP`	Configures the FTP ALG on the NAT44 instance.
Step 5	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# end` `or` `RP/0/RP0/CPU0:router(config-cgn)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the RTSP ALG for NAT44 Instance

Perform this task to configure the ALG for the rtsp for the specified NAT44 instance. RTSP packets are usually destined to port 554. But this is not always true because RTSP port value is configurable.

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-type nat44 nat1**
4. **alg rtsp [server-port]** *value*
5. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGN application and enters CGN configuration mode.
Step 3	**service-type nat44 nat1** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type nat44 nat1`	Configures the service type keyword definition for NAT44application.
Step 4	**alg rtsp [server-port]** *value* **Example:** `RP/0/RP0/CPU0:router(config-cgn-nat44)# alg rtsp server-port 5000`	Configures the rtsp ALG on the NAT44 instance for server port 5000. The range is from 1 to 65535. The default port is 554. **Caution** The option of specifying a server port) is currently not supported. Even if you configure some port, RTSP works only on the default port (554).
Step 5	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# end` `or` `RP/0/RP0/CPU0:router(config-cgn)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.

	Command or Action	Purpose
		• Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the PPTP ALG for a NAT44 Instance

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-type nat44 nat1**
4. **alg pptpAlg**
5. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGN application and enters CGN configuration mode.
Step 3	**service-type nat44 nat1** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type nat44 nat1`	Configures the service type keyword definition for NAT44 or DS-Lite application.
Step 4	**alg pptpAlg** **Example:** `RP/0/RP0/CPU0:router(config-cgn-nat44)# alg pptpAlg`	Configures the pptp ALG on the CGN instance.
Step 5	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# end` `or` `RP/0/RP0/CPU0:router(config-cgn)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?*

	Command or Action	Purpose
		[cancel]: • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

TCP Maximum Segment Size Adjustment

When a host initiates a TCP session with a server, the host negotiates the IP segment size by using the maximum segment size (MSS) option. The value of the MSS option is determined by the maximum transmission unit (MTU) that is configured on the host.

Static Port Forwarding

Static port forwarding helps in associating a private IP address and port with a statically allocated public IP and port. After you have configured static port forwarding, this association remains intact and does not get removed due to timeouts until the CGSE is rebooted. In case of redundant CGSE cards, it remains intact until both of the CGSEs are reloaded together or the router is reloaded. There are remote chances that after a reboot, this association might change. This feature helps in cases where server applications running on the private network needs access from public internet.

Configuring Dynamic Port Range

Perform this task to configure a dynamic port range.

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-type nat64 stateful** *instance-name*
4. **dynamic-port-range start** *port-number*
5. **end**or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance for the CGv6 application and enters CGv6 configuration mode.
Step 3	**service-type nat64 stateful** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type nat64 stateful nat64-inst` `RP/0/RP0/CPU0:router(config-cgn-nat64-stateful)`	Configures the service type keyword definition for CGv6 Stateful NAT64 application.
Step 4	**dynamic-port-range start** *port-number* **Example:** `RP/0/RP0/CPU0:router(config-cgn-nat64-stateful)#dynamic-port-range start 66` `RP/0/RP0/CPU0:router(config-cgn-nat64-stateful)`	Configures the port range from 1 to 65535.
Step 5	**end**or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn-nat64-stateful)# end` `or` `RP/0/RP0/CPU0:router(config-cgn-nat64-stateful)# commit`	Saves configuration changes. • When you issue the end command, the system prompts you to commit changes: `Uncommitted changes found, commit them before exiting (yes/no/cancel)?` `[cancel]:` • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring External Logging for the NAT Table Entries

Perform the following to configure external logging for NAT table entries.

Netflow Logging

Perform the following tasks to configure Netflow Logging for NAT table entries.

Configuring the Server Address and Port for Netflow Logging

Perform this task to configure the server address and port to log network address translation (NAT) table entries for Netflow logging.

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-type nat44 nat1**
4. **inside-vrf** *vrf-name*
5. **external-logging netflowv9**
6. **server**
7. **address** *address* **port** *number*
8. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGN application and enters CGN configuration mode.
Step 3	**service-type nat44 nat1** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type nat44 nat1`	Configures the service type keyword definition for NAT44 application.
Step 4	**inside-vrf** *vrf-name* **Example:** `RP/0/RP0/CPU0:router(config-cgn)# inside-vrf insidevrf1` `RP/0/RP0/CPU0:router(config-cgn-invrf)#`	Configures the inside VRF for the CGN instance named cgn1 and enters CGN inside VRF configuration mode.
Step 5	**external-logging netflowv9** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf)# external-logging netflowv9` `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog)#`	Configures the external-logging facility for the CGN instance named cgn1 and enters CGN inside VRF address family external logging configuration mode.

	Command or Action	Purpose
Step 6	**server** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog)# server` `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)#`	Configures the logging server information for the IPv4 address and port for the server that is used for the netflowv9-based external-logging facility and enters CGN inside VRF address family external logging server configuration mode.
Step 7	**address** *address* **port** *number* **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)# address 2.3.4.5 port 45`	Configures the IPv4 address and port number 45 to log Netflow entries for the NAT table.
Step 8	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)# end` `or` `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the Path Maximum Transmission Unit for Netflow Logging

Perform this task to configure the path maximum transmission unit (MTU) for the netflowv9-based external-logging facility for the inside VRF.

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-type nat44 nat1**
4. **inside-vrf** *vrf-name*
5. **external-logging netflowv9**
6. **server**

7. **path-mtu** *value*
8. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGN application and enters CGN configuration mode.
Step 3	**service-type nat44 nat1** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type nat44 nat1`	Configures the service type keyword definition for NAT44 application.
Step 4	**inside-vrf** *vrf-name* **Example:** `RP/0/RP0/CPU0:router(config-cgn)# inside-vrf insidevrf1` `RP/0/RP0/CPU0:router(config-cgn-invrf)#`	Configures the inside VRF for the CGN instance named cgn1 and enters CGN inside VRF configuration mode.
Step 5	**external-logging netflowv9** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf)# external-logging netflowv9` `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog)#`	Configures the external-logging facility for the CGN instance named cgn1 and enters CGN inside VRF address family external logging configuration mode.
Step 6	**server** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog)# server` `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)#`	Configures the logging server information for the IPv4 address and port for the server that is used for the netflowv9-based external-logging facility and enters CGN inside VRF address family external logging server configuration mode.
Step 7	**path-mtu** *value* **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)# path-mtu 2900`	Configures the path MTU with the value of 2900 for the netflowv9-based external-logging facility.
Step 8	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)# end`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes:

	Command or Action	Purpose
	or RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)# commit	*Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the Refresh Rate for Netflow Logging

Perform this task to configure the refresh rate at which the Netflow-v9 logging templates are refreshed or resent to the Netflow-v9 logging server.

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-type nat44 nat1**
4. **inside-vrf** *vrf-name*
5. **external-logging netflowv9**
6. **server**
7. **refresh-rate** *value*
8. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** RP/0/RP0/CPU0:router# configure	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** RP/0/RP0/CPU0:router(config)# service cgn cgn1 RP/0/RP0/CPU0:router(config-cgn)#	Configures the instance named cgn1 for the CGN application and enters CGN configuration mode.

	Command or Action	Purpose
Step 3	**service-type nat44 nat1** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type nat44 nat1`	Configures the service type keyword definition for NAT44 application.
Step 4	**inside-vrf** *vrf-name* **Example:** `RP/0/RP0/CPU0:router(config-cgn)# inside-vrf insidevrf1` `RP/0/RP0/CPU0:router(config-cgn-invrf)#`	Configures the inside VRF for the CGN instance named cgn1 and enters CGN inside VRF configuration mode.
Step 5	**external-logging netflowv9** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf)# external-logging netflowv9` `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog)#`	Configures the external-logging facility for the CGN instance named cgn1 and enters CGN inside VRF address family external logging configuration mode.
Step 6	**server** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog)# server` `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)#`	Configures the logging server information for the IPv4 address and port for the server that is used for the netflowv9-based external-logging facility and enters CGN inside VRF address family external logging server configuration mode.
Step 7	**refresh-rate** *value* **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)# refresh-rate 50`	Configures the refresh rate value of 50 to log Netflow-based external logging information for an inside VRF.
Step 8	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)# end` `or` `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.

	Command or Action	Purpose
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the Timeout for Netflow Logging

Perform this task to configure the frequency in minutes at which the Netflow-V9 logging templates are to be sent to the Netflow-v9 logging server.

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-type nat44 nat1**
4. **inside-vrf** *vrf-name*
5. **external-logging netflowv9**
6. **server**
7. **timeout***value*
8. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGN application and enters CGN configuration mode.
Step 3	**service-type nat44 nat1** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type nat44 nat1`	Configures the service type keyword definition for NAT44 application.
Step 4	**inside-vrf** *vrf-name* **Example:** `RP/0/RP0/CPU0:router(config-cgn)# inside-vrf insidevrf1` `RP/0/RP0/CPU0:router(config-cgn-invrf)#`	Configures the inside VRF for the CGN instance named cgn1 and enters CGN inside VRF configuration mode.

	Command or Action	Purpose
Step 5	**external-logging netflowv9** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf)# external-logging netflowv9` `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog)#`	Configures the external-logging facility for the CGN instance named cgn1 and enters CGN inside VRF address family external logging configuration mode.
Step 6	**server** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog)# server` `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)#`	Configures the logging server information for the IPv4 address and port for the server that is used for the netflowv9-based external-logging facility and enters CGN inside VRF address family external logging server configuration mode.
Step 7	**timeout***value* **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)# timeout 50`	Configures the timeout value of 50 for Netflow logging of NAT table entries for an inside VRF.
Step 8	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)# end` `or` `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Syslog Logging

Perform the following tasks to configure Syslog Logging for NAT table entries.

Configuring the Server Address and Port for Syslog Logging

Perform this task to configure the server address and port to log DS-Lite entries for Syslog logging.

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-type ds-lite** *instance_name*
4. **external-logging syslog**
5. **server**
6. **address***address***port***number*
7. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGv6 application and enters CGv6 configuration mode.
Step 3	**service-type ds-lite** *instance_name* **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type ds-lite ds-lite1`	Configures the service type keyword definition for the DS-Lite application.
Step 4	**external-logging syslog** **Example:** `RP/0/RP0/CPU0:router(config-cgn-ds-lite)#external-logging syslog` `RP/0/RP0/CPU0:router(config-cgn-ds-lite-extlog)#`	Configures the external-logging facility for the CGv6 instance named cgn1 and enters CGv6 external logging configuration mode.
Step 5	**server** **Example:** `RP/0/RP0/CPU0:router(config-cgn-ds-lite-extlog)# server` `RP/0/RP0/CPU0:router(config-cgn-ds-lite-extlogserver)#`	Configures the logging server information for the IPv4 address and port for the server that is used for the syslog-based external-logging facility and enters CGv6 external logging server configuration mode.
Step 6	**address***address***port***number* **Example:** `RP/0/RP0/CPU0:router(config-cgn-ds-lite-extlogserver)#address 2.3.4.5 port 45`	Configures the IPv4 address and port number 45 to log Netflow entries.
Step 7	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn-ds-lite-extlogserver)#end`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes:

	Command or Action	Purpose
	or RP/0/RP0/CPU0:router(config-cgn-ds-lite-extlogserver)#commit	*Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the Host-Name for Syslog Logging

Perform this task to configure the host name to be filled in the Netflow header for the syslog logging.

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-type nat44 nat1**
4. **inside-vrf** *vrf-name*
5. **external-logging syslog**
6. **server**
7. **host-name***name*
8. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** RP/0/RP0/CPU0:router# configure	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** RP/0/RP0/CPU0:router(config)# service cgn cgn1 RP/0/RP0/CPU0:router(config-cgn)#	Configures the instance named cgn1 for the CGv6 application and enters CGv6 configuration mode.

	Command or Action	Purpose
Step 3	**service-type nat44 nat1** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type nat44 nat1`	Configures the service type keyword definition for CGv6 NAT44 application.
Step 4	**inside-vrf** *vrf-name* **Example:** `RP/0/RP0/CPU0:router(config-cgn-nat44)# inside-vrf insidevrf1` `RP/0/RP0/CPU0:router(config-cgn-invrf)#`	Configures the inside VRF for the CGv6 instance named cgn1 and enters CGv6 inside VRF configuration mode.
Step 5	**external-logging syslog** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf)# external-logging syslog` `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog)#`	Configures the external-logging facility for the CGv6 instance named cgn1 and enters CGv6 inside VRF address family external logging configuration mode.
Step 6	**server** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog)# server` `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)#`	Configures the logging server information for the IPv4 address and port for the server that is used for the syslog-based external-logging facility and enters CGv6 inside VRF address family external logging server configuration mode.
Step 7	**host-name***name* **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)# host-name host1`	Configures the host name for the syslog-based external-logging facility.
Step 8	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)# end` `or` `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.

	Command or Action	Purpose
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the Path Maximum Transmission Unit for Syslog Logging

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-type nat44 nat1**
4. **inside-vrf** *vrf-name*
5. **external-logging syslog**
6. **server**
7. **path-mtu***value*
8. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGv6 application and enters CGv6 configuration mode.
Step 3	**service-type nat44 nat1** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type nat44 nat1`	Configures the service type keyword definition for CGv6 NAT44 application.
Step 4	**inside-vrf** *vrf-name* **Example:** `RP/0/RP0/CPU0:router(config-cgn-nat44)# inside-vrf insidevrf1` `RP/0/RP0/CPU0:router(config-cgn-invrf)#`	Configures the inside VRF for the CGv6 instance named cgn1 and enters CGv6 inside VRF configuration mode.
Step 5	**external-logging syslog** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf)# external-logging syslog` `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog)#`	Configures the external-logging facility for the CGv6 instance named cgn1 and enters CGv6 inside VRF address family external logging configuration mode.

	Command or Action	Purpose
Step 6	**server** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog)# server` `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)#`	Configures the logging server information for the IPv4 address and port for the server that is used for the syslog-based external-logging facility and enters CGv6 inside VRF address family external logging server configuration mode.
Step 7	**path-mtu***value* **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)# path-mtu 200`	Configures the path MTU with the value of 200 for the syslog-based external-logging facility.
Step 8	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)# end` `or` `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Bulk Port Allocation

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-type nat44 nat1**
4. **inside-vrf** *vrf-name*
5. **bulk-port-alloc size** *number of ports*
6. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGv6 application and enters CGv6 configuration mode.
Step 3	**service-type nat44 nat1** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type nat44 nat1`	Configures the service type keyword definition for CGv6 NAT44 application.
Step 4	**inside-vrf** *vrf-name* **Example:** `RP/0/RP0/CPU0:router(config-cgn-nat44)# inside-vrf insidevrf1` `RP/0/RP0/CPU0:router(config-cgn-nat44-invrf)#`	Configures the inside VRF for the CGv6 instance named cgn1 and enters CGv6 inside VRF configuration mode.
Step 5	**bulk-port-alloc size** *number of ports* **Example:** `RP/0/RP0/CPU0:router(config-cgn-nat44-invrf-)# bulk-port-alloc size 64` `RP/0/RP0/CPU0:router(config-cgn-nat44-invrf)`	Allocate ports in bulk to reduce Netflow/Syslog data volume.
Step 6	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn-nat44-invrf)# end` `or` `RP/0/RP0/CPU0:router(config-cgn-nat44-invrf)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes.

	Command or Action	Purpose
		• Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Destination-Based Logging for NAT44

Perform these tasks to configure destination-based logging for NAT table entries.

Configuring the Session-Logging for Netflow Logging

Perform this task to configure session-logging if destination IP and Port information needs to logged in the Netflow records.

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-type nat44 nat1**
4. **inside-vrf** *vrf-name*
5. **external-logging netflow**
6. **server**
7. **session-logging**
8. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGv6 application and enters CGv6 configuration mode.
Step 3	**service-type nat44 nat1** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type nat44 nat1`	Configures the service type keyword definition for CGv6 NAT44 application.
Step 4	**inside-vrf** *vrf-name* **Example:** `RP/0/RP0/CPU0:router(config-cgn-nat44)# inside-vrf insidevrf1` `RP/0/RP0/CPU0:router(config-cgn-nat44-invrf)#`	Configures the inside VRF for the CGv6 instance named cgn1 and enters CGv6 inside VRF configuration mode.

	Command or Action	Purpose
Step 5	**external-logging netflow** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf)# external-logging netflow version 9` `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog)#`	Configures the external-logging facility for the NAT44 instance.
Step 6	**server** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog)# server` `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)#`	Configures the logging server information for the IPv4 address and port for the server that is used for the netflow-v9 based external-logging facility.
Step 7	**session-logging** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)# session-logging`	Configures the session logging for a NAT44 instance.
Step 8	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)# end` `or` `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring the Session-Logging for Syslog Logging

Perform this task to configure session-logging if destination IP and Port information needs to logged in the Netflow records.

SUMMARY STEPS

1. **configure**

2. **service cgn** *instance-name*
3. **service-type nat44 nat1**
4. **inside-vrf** *vrf-name*
5. **external-logging syslog**
6. **server**
7. **session-logging**
8. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGv6 application and enters CGv6 configuration mode.
Step 3	**service-type nat44 nat1** **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type nat44 nat1`	Configures the service type keyword definition for CGv6 NAT44 application.
Step 4	**inside-vrf** *vrf-name* **Example:** `RP/0/RP0/CPU0:router(config-cgn-nat44)# inside-vrf insidevrf1` `RP/0/RP0/CPU0:router(config-cgn-nat44-invrf)#`	Configures the inside VRF for the CGv6 instance named cgn1 and enters CGv6 inside VRF configuration mode.
Step 5	**external-logging syslog** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf)# external-logging syslog` `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog)#`	Configures the external-logging facility for the NAT44 instance.
Step 6	**server** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog)# server` `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)#`	Configures the logging server information for the IPv4 address and port for the server that is used for the netflow-v9 based external-logging facility.
Step 7	**session-logging** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)# session-logging`	Configures the session logging for a NAT44 instance.

	Command or Action	Purpose
Step 8	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)# end` `or` `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Example

Configuring the Carrier Grade Service Engine

Hardware:

- CGSE hardware in chassis
- Latest **uboot** and **mans** images in CGSE

Software:

- Load **hfr-mini-px.vm** and activate it
- Load **hfr-services-px.pie** and activate it
- Load **hfr-fpd-px.pie** and activate it

Before you begin

These are the prerequisite components for configuring the carrier grade service engine.

Bringing Up the CGSE Board

- After installing the cgn service pie (the pie installation is similar to any other CRS pie), ensure that the uboot version (fpga2, fpga3, fpga4, fpga5) is 0.559 & MANS FPGA version is 0.41014 as depicted below.

```
RP/0/RP0/CPU0:#admin
RP/0/RP0/CPU0:(admin)#show hw-module fpd location 0/2/cpu0

==================================== ========================================
                                     Existing Field Programmable Devices
                                     ========================================
                                       HW                          Current SW Upg/
Location     Card Type                Version Type Subtype Inst   Version    Dng?
============ ======================== ======= ==== ======= ==== =========== ====
--------------------------------------------------------------------------------
0/1/CPU0     CRS-CGSE-PLIM              0.88  lc   fpga2   0       0.559    No
                                              lc   fpga3   0       0.559    No
                                              lc   fpga4   0       0.559    No
                                              lc   fpga5   0       0.559    No
                                              lc   fpga1   0       0.41014  No
                                              lc   rommonA 0       1.52     No
                                              lc   rommon  0       1.52     Yes
```

Note Latest uboot version is 559 & MANS is 0.41

Note If one or more FPD needs an upgrade, then this can be accomplished using the following steps. Make sure that the fpd pie is loaded and activated. If found different, follow the upgrade procedure in *Line Card Upgrade*.

- After insertion, the card remains in "IOS XR RUN" state until you install the appropriate cgn service pie.
- After installing the cgn service pie, the card goes to "FAILED" state until you complete the configuration mentioned in next step. These log messages appear on the console.

```
LC/0/3/CPU0:Sep 28 23:36:36.815 : plim_services[241]: plim_services_init[2063] Uknown
role Retrying.., Role = -7205769247857836031
LC/0/3/CPU0:Sep 28 23:37:59.341 : plim_services[241]: service_download_thread[3873] App
 img download max-retries exhausted, 'plim_services' detected the 'warning' condition
'Operation not okay'
LC/0/3/CPU0:Sep 28 23:37:59.342 : plim_services[241]: plim_services_tile_failed[752]
TILE0 failed
RP/0/RP1/CPU0:Sep 28 23:38:18.494 : invmgr[240]: %PLATFORM-INV-6-NODE_STATE_CHANGE :
Node: 0/3/0, state: FAILED
```

- After Successful Boot Up:

```
RP/0/RP0/CPU0:router#show platform
Sun Dec 20 07:15:38.893 UTC
Node            Type            PLIM            State           Config State
--------------------------------------------------------------------------------
0/0/CPU0        MSC             Services Plim   IOS XR RUN      PWR,NSHUT,MON
0/0/0           MSC(SPA)        CGSE-TILE       OK              PWR,NSHUT,MON
0/1/CPU0        MSC             Jacket Card     IOS XR RUN      PWR,NSHUT,MON
0/1/0           MSC(SPA)        8X1GE           OK              PWR,NSHUT,MON
```

- Control connection to CGSE, One ServiceInfra Interface per CGSE & IPv4 address of local significance. Minimum of two valid IPv4 unicast addresses are required for each ServiceInfra SVI. The Serviceinfra interface removal/modification needs CGSE LC reload.

```
outer(config)
interface ServiceInfra1
ipv4 address 3.1.1.2 255.255.255.252
service-location 0/0/CPU0
logging events link-status
commit

router(config)
hw-module service cgn location 0/0/CPU0
commit
```

Note This configuration has to be replicated for Standby CGSE Card. The serviceinfra IP has to be different.

- Specify the service role(cgn) for the given CGSE location

  You need to reload the card. It takes about 15minutes.

```
router#
hw-module location 0/0/CPU0 reload
WARNING: This will take the requested node out of service.
Do you wish to continue?[confirm(y/n)] y
```

Configuring the Predefined Mode for NAT44

Perform these tasks to configure the predefined mode for NAT44.

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-type nat44** *nat1*
4. **inside-vrf** *vrf-name*
5. **map address-pool** *address/prefix*
6. **nat-mode**
7. **predefined** *ipaddress/prefix*
8. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:**	Configures the instance named cgn1 for the CGN application and enters CGN configuration mode.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	
Step 3	**service-type nat44** *nat1* **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type nat44 nat1`	Configures the service type keyword definition for CGN NAT44 application.
Step 4	**inside-vrf** *vrf-name* **Example:** `RP/0/RP0/CPU0:router(config-cgn-nat44)# inside-vrf insidevrf1` `RP/0/RP0/CPU0:router(config-cgn-invrf)#`	Configures an inside VRF named *insidevrf1* and enters CGv6 inside VRF configuration mode.
Step 5	**map address-pool** *address/prefix* **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf)# map address-pool 10.10.0.0/16`	Maps an inside VRF to an outside VRF and address pool mapping.
Step 6	**nat-mode** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf)# nat-mode`	Specifies the predefined mode for NAT44.
Step 7	**predefined** *ipaddress/prefix* **Example:** `RP/0/RP1/CPU0:router(config-cgn-invrf-natmode)# predefined private-pool 192.1.106.0/24` `RP/0/RP1/CPU0:router(config-cgn-invrf-natmode)# predefined private-pool 192.1.107.0/26` `RP/0/RP1/CPU0:router(config-cgn-invrf-natmode)# predefined private-pool 192.1.107.128/26`	Specifies the private address range for the predefined mode. You can specify a minimum of one address range to eight addess ranges.
Step 8	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-natmode)# end` `or` `RP/0/RP0/CPU0:router(config-cgn-invrf-natmode)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes.

	Command or Action	Purpose
		• Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring IPv4/IPv6 Stateless Translator (XLAT)

These are the sequence of steps for XLAT configuration:

1. Divert the IPv4 traffic to the IPv4 ServiceApp.
2. Divert the IPv6 traffic to the IPv6 ServiceApp.
3. Configure one CGN instance per CGSE.
4. Configure multiple XLAT instances per CGN instance.
5. Configure IPv4 and IPv6 Service Apps.
6. Configure CGN instance.
7. Configure XLAT instances.
8. Associate IPv4 and IPv6 ServiceApps to XLAT instance.

XLAT ServiceApp Configuration

1. IPv4 ServiceApp
   - Configure Traffic Type – nat64_stless
   - Configure IPv4 address
   - Configure static route to divert specific IPv4 subnets (corresponding to IPv6 hosts) to the IPv4 ServiceApp

```
conf t
int ServiceApp4
    service cgn cgn1 service-type nat64 stateless
    ipv4 add 2.0.0.1/24
    commit
exit

router static
  address-family ipv4 unicast
  136.136.136.0/24 ServiceApp4 2.0.0.2
   commit
   exit
end
```

2. IPv6 ServiceApp

- Configure Type – nat64_stless
- Configure IPv6 address
- Configure static route to divert IPv6 traffic corresponding to XLAT prefix to the IPv6 ServiceApp

```
conf t
int serviceApp6
    service cgn cgn1service-type nat64 stateless
    ipv6 address 2001:db8:fe00::1/40
    commit
exit

router static
  address-family ipv6 unicast
  2001:db8:ff00::/40 ServiceApp6 2001:db8:fe00::2
  commit
  exit
end
```

XLAT Instance Configuration

- IPv4 ServiceApp name
  - Service App on which IPv4 traffic enters/leaves
- IPv6 ServiceApp name
  - Service App on which IPv6 traffic enters/leaves
- XLAT prefix
  - IPv6 prefix corresponding to XLAT translation
- Ubit enabled/disabled
  - whether bits 64..71 are reserved or can be used for xlat purposes
- IPv4 & IPv6 TCP MSS configuration
  - IPv4 TCP traffic's MSS value will be set to the smaller of (incoming MSS value)
  - IPv6 TCP traffic's MSS value will be set to the smaller of (incoming MSS value)
- Traceroute pool
  - Non Translatable IPv6 source addresses are translated to the IPv4 addresses in this range using a hash mechanism
  - Algorithm to chose IPv4 address from traceroute pool

    TTL based – Chose address based on hop count of the pkt

    Hash based – Hash IPv6 Source Address and use it for selection

    Random – Randomly select an IPv4 address
- IPv4 TOS Setting

- By default IPv4 TOS field is copied from IPv6 Traffic Class field
- This value can be overridden based on the configured TOS value

- IPv6 Traffic Class Setting
  - By default IPv6 Traffic Class field is copied from IPv4 TOS field
  - This value can be overridden based on the configured Traffic Class value

- IPv4 DF override
  - When translating a IPv6 packet when the no Fragment Header IPv4 DF bit is set to 1.
  - We can override this and set the DF bit to 0, if incoming IPv6 packets are smaller than 1280 bytes.
  - This is to prevent path-mtu blackholing issues.

```
conf t
service cgn cgn1
service-type nat64 stateless xlat1
              ipv6-prefix 2001:db8:ff00::/40
              ubit-reserved
              address-family ipv4
                   interface ServiceApp4
                   tcp mss 1200
                   tos 64
              address-family ipv6
                    interface ServiceApp6
                    tcp mss 1200
                    traffic-class 32
                    df-override
              traceroute translation
                   address-pool 202.1.1.0/24
                   algorithm Hash
```

Line Card Upgrade

UPGRADE FROM_ UBOOT to 559 & MANS FPGA to 0.41014

SUMMARY STEPS

1. Load the fpd pie.
2. Uboot the line card.
3. Wait for the ready for UBOOT log message on the console.
4. Go to admin mode on the node and upgrade the FPGA MANS.
5. Also upgrade these locations for Uboot:
6. Reload the card after the successful upgrade operation.
7. After the card comes up, check for the uboot version . This can be done using the following command from the admin mode.

DETAILED STEPS

Step 1 Load the fpd pie.

Step 2 Uboot the line card.

Example:

```
hw-module location 0/2/CPU0 uboot-mode
WARNING: This will bring the requested node's PLIM to uboot mode.
Do you wish to continue?[confirm(y/n)]y
```

Step 3 Wait for the ready for UBOOT log message on the console.

Example:

```
RP/0/RP0/CPU0:#LC/0/2/CPU0:Sep 29 02:38:40.418 : plim_services[239]:
 tile_fsm_uboot_doorbell_handler[3222] Plim moved to uboot-mode and ready for UBOOT upgrade
```

Step 4 Go to admin mode on the node and upgrade the FPGA MANS.

Example:

```
upgrade hw-module fpd fpga1_location <>
```

Step 5 Also upgrade these locations for Uboot:

Example:

```
upgrade hw-module fpd fpga2 location <>
upgrade hw-module fpd fpga3 location <>
upgrade hw-module fpd fpga4_location <>
upgrade hw-module fpd fpga5_location <>
```

Step 6 Reload the card after the successful upgrade operation.

Example:

```
hw-module location <> reload
```

Step 7 After the card comes up, check for the uboot version . This can be done using the following command from the admin mode.

Example:

```
show hw-module fpd location <>
```

Configuring IPv6 Rapid Development

These steps describe the configuration of IPv6 Rapid Development application.

Refer topic *6rd CPE/RG configuration parameters* and *6rd BR (CGSE) configuration parameters* to know about 6rd configuration parameters.

1.

- Create a CGN instance per CGSE

```
router(config)#
service cgn demo
 service-location preferred-active 0/0/CPU0
```

- An IPv4 SVI is created to carry IPv4 pkt into the CGSE for Decapsulation and is handed over to native IPv6 via IPv6 SVI. Service-type should be “tunnel v6rd”

```
router(config)#
interface ServiceApp4
 ipv4 address 1.1.1.1 255.255.255.252
```

```
 service cgn demo
service-type tunnel v6rd
 logging events link-status
```

- An IPv6 SVI is created to carry IPv6 pkt into the CGSE for Encapsulation and is handed over to IPv4 N/W via IPv4 SVI. Service-type should be “tunnel v6rd”

```
router(config)#
interface ServiceApp6
 ipv4 address 5000::1/126
 service cgn demo service-type tunnel v6rd
 logging events link-status
```

- Configure 6rd instance (string “6rd1” in this example). There can be 64 6rd instances per CGSE/Chassis.
- Configure 6rd Prefix, BR source IPv4 address & unicast IPv6 address in a single commit.
- **address-family** command binds IPv4 & IPv6 Serviceapp interface to a particular 6rd instance 6rd1, for transmitting and receiving 6rd traffic.

```
router(config)#

service cgn demo
 service-type tunnel v6rd 6rd1
  br
   ipv6-prefix 2001:B000::/28
   source-address 100.1.1.1
   unicast address 2001:B006:4010:1010::1
  !
  address-family ipv4
   interface ServiceApp4
  !
  address-family ipv6
   interface ServiceApp6
```

Note Unicast address specifies a unique IPv6 address for a particular CGSE. This is used as a source IPv6 address while replying to IPv6 ICMP queries destined for BR IPv6 anycast address.

The Unicast address also provides the source IPv6 address during IPv4 ICMP translation to IPv6 ICMP.

2. You can configure routes to the CGSE using these steps.
   - To divert the traffic towards CGSE which is destined for BR

```
router(config)#

router static
 address-family ipv4 unicast
  100.1.1.1/32 1.1.1.2  (Serviceapp4 NextHop)
```

   - Packets destined to 6rd prefix are routed to CGSE

```
Router#show route ipv6
S    2001:b000::/28 is directly connected,00:13:44, ServiceApp6
S    2001:b006:4010:1010::/60 is directly connected,00:19:24, Null0
S    2001:b006:4010:1010::/128 is directly connected,00:13:44, ServiceApp6
S    2001:b006:4010:1010::1/128 is directly connected,00:13:44, ServiceApp6
```

```
C    5000::/64 is directly connected,00:13:44, ServiceApp6
L    5000::1/128 is directly connected,00:13:44, ServiceApp6
C    2001:db8::/64 is directly connected,01:23:55, GigE0/1/1/4
L    2001:db8::2/128 is directly connected,01:23:55, GigE0/1/1/4
```

3. This step shows the output of **show cgn tunnel v6rd 6rd1 statistics** command.

```
RP/0/RP0/CPU0:#show cgn tunnel v6rd 6rd1 statistics


Tunnel 6rd configuration
========================
Tunnel 6rd name: 6rd1
IPv6 Prefix/Length: 2001:db8::/32
Source address: 9.1.1.1
BR Unicast address: 2001:db8:901:101::1
IPv4 Prefix length: 0
IPv4 Suffix length: 0
TOS: 0, TTL: 255, Path MTU: 1280

Tunnel 6rd statistics
=====================

IPv4 to IPv6
============
Incoming packet count  : 0 (Total No. of Protocol pkts 41
 non Protocol 41)
Incoming tunneled packets count  : 0 (Total No. of Protocol pkts 41
 non Protocol 41)
Decapsulated packets  : 0
ICMP translation count  : 0 (ICMPv4 TO ICMPv6 translated count)
Insufficient IPv4 payload drop count  : 0 (Payload should carry IPv6 header)
Security check failure drops  : 0
No DB entry drop count  : 0 (6rd config is incomplete/missing)
Unsupported protocol drop count  : 0 (IPv4 protocol type is not 41 (IPv6))
Invalid IPv6 source prefix drop count  : 0 (IPv6 Source from RG doesn't have 6rd
 prefix)

IPv6 to IPv4
============
Incoming packet count  : 0
Encapsulated packets count  : 0
No DB drop count  : 0 (6rd config is not complete/missing)
Unsupported protocol drop count  : 0 (Non ICMP pkts destined to IPv6 BR
 anycast/unicast address)

IPv4 ICMP
==========
Incoming packets count  : 0
Reply packets count  : 0
Throttled packet count  : 0 (ICMP throttling in CGSE 64  PKTS/sec
Nontranslatable drops  : 0 (ICMPv4 error pkt (ipv4->TL) at least
 72 bytes)
Unsupported icmp type drop count  : 0 (As per
http://tools.ieft.org/html/draft-ieft-behave-v6v4-xlate-22 )

IPv6 ICMP
==========
Incoming packets count                          : 0
Reply packets count                             : 0
Packet Too Big generated packets count          : 0
Packet Too Big not generated packets count      : 0
NA generated packets count                      : 0
TTL expiry generated packets count              : 0
```

```
Unsupported icmp type drop count                : 0 (As per
http://tools.ieft.org/html/draft-ieft-behave-v6v4-xlate-22)
Throttled packet count                          : 0 (ICMP throttling in CSGE 64
 pkts/core)

IPv4 to IPv6 Fragments
======================
Incoming fragments count                        : 0 (No. of IPv4 Fragments Came in)
Reassembled packet count                        : 0 (No. of Pkts Reassembled from
 Fragments )
Reassembled fragments count                      : 0 (No. of Fragments Reassembled)
ICMP incoming fragments count                   : 0 (No. of ICMP Fragments Came in)
Total fragment drop count                       : 0
Fragments dropped due to timeout                : 0 (Fragment dropped due to
 reassembly timeout)
Reassembly throttled drop count                 : 0 (Fragments throttled)
Duplicate fragments drop count                  : 0
Reassembly disabled drop count                  : 0 (Number of fragments dropped
 while re-assembly is disabled.)
No DB entry fragments drop count                : 0 (6rd Config is incomplete
 /missing)
Fragments dropped due to security check failure  : 0
Insufficient IPv4 payload fragment drop count    : 0 (1st Fragment should have IPv6
 header)
Unsupported protocol fragment drops             : 0 (IPv4 protocol type is not 41
 (IPv6) & non ICMP)
Invalid IPv6 prefix fragment drop count         : 0 (IPv6 Source from RG doesn't have
 6rd prefix)
=============================================================================
IPv6 to IPv4 Fragments
======================
Incoming ICMP fragment count                    : 0

=============================================================================================
```

4. Clear all the 6rd counters using the clear cgn tunnel v6rd 6rd1 statistics command.

```
RP/0/RP0/CPU0:BR1#clear cgn tunnel v6rd 6rd1 statistics
```

Ping to BR Anycast Address

- IPv6 Ping from RG to BR Anycast Address

```
etc/init.d/service_wan_ipv6 # ping 2001:B006:4010:1010::
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:B006:4010:1010::, timeout is 2 seconds:
PING 2001:B006:4010:1010::(2001:B006:4010:1010::)56 data bytes
64 bytes from 2001:B006:4010:1010::1 : seq=1 ttl=62 time=1.122 ms
64 bytes from 2001:B006:4010:1010::1 : seq=2 ttl=62 time=0.914 ms

--- 2001:B006:4010:1010:: ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
```

Note Reply will configure IPv6 unicast address as Src address (2001:B006:4010:1010::1).

```
RP/0/RP0/CPU0:BR1#show cgn tunnel v6rd 6rd1 statistics
IPv6 to IPv4
============
Incoming packet count                            : 5
```

```
IPv6 ICMP
==========
Incoming packets count                          : 5
Reply packets count                             : 5
```

Enable Additional 6rd Features

- Common 6rd IPv4 Prefix & Suffix Length
  - **IPv4 Prefix Length** : This common prefix can be provisioned on the router and therefore need not be carried in the IPv6 destination to identify a tunnel endpoint.
  - **IPv4 Suffix Length** : All the 6RD CEs and the BR can agree on a common tail portion of the V4 address to identify a tunnel endpoint.

Note All the BR parameters have to be given in Single Commit.

- p
- 6rd Tunnel TTL and TOS
  - By default the IPv6 Traffic class and Hoplimit field will be copied to the IPv4 TTL and TOS fields respectively. This default behavior MAY be overridden by above configuration.
  - tos value is in decimal

```
service cgn demo
   service-type tunnel v6rd 6rd1
    tos 160
    ttl 100
    commit
```

- Setting 6rd Tunnel Path MTU
  - By default the 6rd Tunnel MTU value is 1280.

```
service cgn demo
   service-type tunnel v6rd 6rd1
    path-mtu 1480
    commit
```

- Enabling reassembly of Fragmented Tunnel Packets.
- Fragmented Tunneled IPv4 packets are reassembled by BR before decapsulation.

```
service cgn demo
   service-type tunnel v6rd 6rd1
    reassembly-enable
    commit

RP/0/RP0/CPU0:BR1#show cgn tunnel v6rd 6rd1 statistics
Incoming fragments count                        : 2
Reassembled packet count                        : 1
Reassembled fragments count                      : 2
ICMP incoming fragments count                   : 0
Total fragment drop count                       : 0
Fragments dropped due to timeout                : 0
Duplicate fragments drop count                  : 0
```

```
No DB entry fragments drop count                 : 0
Fragments dropped due to security check failure  : 0
Insufficient IPv4 payload fragment drop count    : 0
Unsupported protocol fragment drops              : 0
Invalid IPv6 prefix fragment drop count          : 0
Incoming ICMP fragment count                     : 0
```

- ICMP Throttling
  - By default CGSE throttles 1 per core (we have 64 cores in CGSE)

```
RP/0/RP0/CPU0:BR1#config
RP/0/RP0/CPU0:BR1(config)#service cgn cgn1
 RP/0/RP0/CPU0:BR1(config-cgn)#protocol icmp rate-limit ?
  <0-65472>  ICMP rate limit per second, should be multiple of 64
commit
```

- Reset DF bit
  - Tunneled IPv4 packets from BR will have DF bit reset (0) which will allow fragmentation in the path to RG.
  - By default it is set to 1 to support Anycast routing

```
service cgn demo
   service-type tunnel v6rd 6rd1
    reset-df-bit
    commit
```

- Additional Information:
  - IPv6 Rapid Deployment on IPv4 Infrastructures (6rd) – http://tools.ietf.org/html/rfc5969
  - ICMPv4 to ICMPv6 Translation as per http://tools.ietf.org/html/draft-ietf-behave-v6v4-xlate-22
  - Basic Transition Mechanisms for IPv6 Hosts and Routers", RFC 4213, October 2005.
- "An Anycast Prefix for 6to4 Relay Routers", RFC 3068, June 2001.
- “Security Considerations for 6to4", RFC 3964, December 2004.

For line card upgrade procedure, refer Line Card Upgrade, page 64.

Configuring Dual Stack Lite Instance

Perform this task to configure dual stack lite application.

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-location preferred-active** *node-id* [**preferred-standby** *node-id*]
4. **service-type ds-lite** *instance*
5. **portlimit** *value*
6. **bulk-port-alloc size** *value*
7. **map address-pool** address
8. **aftr-tunnel-endpoint-address** <*address*>

9. **address-family ipv4**
10. **interface ServiceApp41**
11. **address-family ipv6**
12. **interface ServiceApp61**
13. **protocol tcp**
14. **session {initial | active} timeout** *seconds*
15. **mss***size*
16. **external-logging netflow9**
17. **server**
18. **address** *A.B.C.D* **port** *port-number*
19. **external-logging syslog**
20. **server**
21. **address** *A.B.C.D* **port** *port-number*
22. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGN application and enters CGN configuration mode.
Step 3	**service-location preferred-active** *node-id* [**preferred-standby** *node-id*] **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-location preferred-active 0/2/CPU0 preferred-standby 0/4/CPU0`	Specifies the global command applied per cgn instance. It initiates the particular instance of the cgn application on the active and standby locations.
Step 4	**service-type ds-lite** *instance* **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type ds-lite dsl1`	Configures the service type keyword definition for the DS LITE application.
Step 5	**portlimit** *value* **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type ds-lite dsl1`	Configures the service type keyword definition for the DS LITE application.
Step 6	**bulk-port-alloc size** *value* **Example:**	Enables bulk port allocation and sets bulk size that is used to reduce logging data volume.

	Command or Action	Purpose
	`RP/0/RP0/CPU0:router(config-cgn-ds-lite)# bulk-port-alloc size 128`	
Step 7	**map address-pool** address **Example:** `RP/0/RP0/CPU0:router(config-cgn-ds-lite)# map address-pool 52.52.52.0/24`	Specifies the address pool for the DS LITE instance. **Note** 52.52.52.0/24 is the IPv4 public address pool assigned to the DS Lite instance.
Step 8	**aftr-tunnel-endpoint-address** <*address*> **Example:** `RP/0/RP0/CPU0:router(config-cgn-ds-lite)# aftr-tunnel-endpoint address 3001:DB8:EOE:E01::`	Specifies the IPv6 address of the tunnel end point. The IPv4 elements must address their IPV6 packets to this address.
Step 9	**address-family ipv4** **Example:** `RP/0/RP0/CPU0:router(config-cgn-ds-lite)# address-family ipv4`	Enters the address family IPv4 configuration mode.
Step 10	**interface ServiceApp41** **Example:** `RP/0/RP0/CPU0:router(config-cgn-ds-lite-afi)# interface ServiceApp41`	Specifies the ServiceApp on which IPv4 traffic enters and leaves.
Step 11	**address-family ipv6** **Example:** `RP/0/RP0/CPU0:router(config-cgn-ds-lite)# address-family ipv6`	Enters the address family IPv6 configuration mode.
Step 12	**interface ServiceApp61** **Example:** `RP/0/RP0/CPU0:router(config-cgn-ds-lite-afi)# interface ServiceApp61`	Specifies the ServiceApp on which IPv6 traffic enters and leaves.
Step 13	**protocol tcp** **Example:** `RP/0/RP0/CPU0:router(config-cgn-ds-lite-afi)# protocol tcp`	Configures the TCP protocol session.
Step 14	**session {initial | active} timeout** *seconds* **Example:** `RP/0/RP0/CPU0:router(config-cgn-proto)# session initial timeout 90`	This command configures the timeout value in seconds for ICMP,TCP or UDP sessions for a service instance. For TCP and UDP, you can configure the initial and active session timeout values. For ICMP, there are no such options. This configuration is applicable to all the IPv4 addresses that belong to a particular service instance. This example configures the initial session timeout value as 90 for the TCP session.

	Command or Action	Purpose
Step 15	**mss***size* **Example:** `RP/0/RP0/CPU0:router(config-cgn-proto)# mss 1100`	Configures the adjustment MSS value as 1100.
Step 16	**external-logging netflow9** **Example:** `RP/0/RP0/CPU0:router(config-cgn-ds-lite)# external-logging netflowv9`	Configures the external-logging facility for the DS LITE instance named dsl1 and enters the external logging configuration mode.
Step 17	**server** **Example:** `RP/0/RP0/CPU0:router(config-cgn-ds-lite-extlog)# server`	Configures the logging server information for the IPv4 address and port for the server that is used for the netflow-v9 based external-logging facility and enters external logging server configuration mode.
Step 18	**address** *A.B.C.D* **port** *port-number* **Example:** `RP/0/RP0/CPU0:router(config-cgn-ds-lite-extlog-server)# address 90.1.1.1 port 99`	Configures the netflow server address and port number to use for netflow version 9 based external logging facility for DS LITE instance.
Step 19	**external-logging syslog** **Example:** `RP/0/RP0/CPU0:router(config-cgn-ds-lite)# external-logging syslog`	Configures the external-logging facility for the DS LITE translation entries that can be logged in syslog servers to analyze and debug the information.
Step 20	**server** **Example:** `RP/0/RP0/CPU0:router(config-cgn-ds-lite-extlog)# server` `RP/0/RP0/CPU0:router(config-cgn-ds-lite-extlog-server)#`	Configures the logging server information for the IPv4 address and port for the server that is used for the syslog based external-logging facility.
Step 21	**address** *A.B.C.D* **port** *port-number* **Example:** `RP/0/RP0/CPU0:router(config-cgn-ds-lite-extlog-server)# address 90.1.1.1 port 514`	Configures the syslog server address and port number to use for syslog based external logging facility for DS LITE instance.
Step 22	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn-ds-lite-extlog-server)# end` `or` `RP/0/RP0/CPU0:router(config-cgn-ds-lite-extlog-server)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.

	Command or Action	Purpose
		• Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring PCP Server for NAT44 Instance

Perform this task to configure PCP server for a NAT44 instance:

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-type nat44** *nat1*
4. **inside-vrf** *vrf-name*
5. **pcp-server**
6. **address** *ipv4-address*
7. **port** *port-number*
8. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGN application and enters CGN configuration mode.
Step 3	**service-type nat44** *nat1* **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type nat44 nat1`	Configures the service type keyword definition for CGN NAT44 application.

	Command or Action	Purpose
Step 4	**inside-vrf** *vrf-name* **Example:** `RP/0/RP0/CPU0:router(config-cgn)# inside-vrf insidevrf1` `RP/0/RP0/CPU0:router(config-cgn-invrf)#`	Configures the inside VRF for the CGN instance named cgn1 and enters CGN inside VRF configuration mode.
Step 5	**pcp-server** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf)# pcp-serevr` `RP/0/RP0/CPU0:router(config-cgn-invrf)#`	Configures the PCP server for a NAT44 instance.
Step 6	**address** *ipv4-address* **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf)# address 10.256.24.192` `RP/0/RP0/CPU0:router(config-cgn-invrf)#`	Configures the address of the PCP server.
Step 7	**port** *port-number* **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf)# port 66`	Configures the port number of the PCP server. The range is from 1 to 65535. The default port is 5351.
Step 8	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)# end` `or` `RP/0/RP0/CPU0:router(config-cgn-invrf-af-extlog-server)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode. • Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuring PCP Server for DS-Lite Instance

Perform this task to configure PCP server for a DS-Lite instance:

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-type nat44** *nat1*
4. **pcp-server**
5. **port** *port-number*
6. **end** or **commit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters global configuration mode.
Step 2	**service cgn** *instance-name* **Example:** `RP/0/RP0/CPU0:router(config)# service cgn cgn1` `RP/0/RP0/CPU0:router(config-cgn)#`	Configures the instance named cgn1 for the CGN application and enters CGN configuration mode.
Step 3	**service-type nat44** *nat1* **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type nat44 nat1`	Configures the service type keyword definition for CGN NAT44 application.
Step 4	**pcp-server** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf)# pcp-serevr` `RP/0/RP0/CPU0:router(config-cgn-invrf)#`	Configures the PCP server for a NAT44 instance.
Step 5	**port** *port-number* **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf)# port 66`	Configures the port number of the PCP server. The range is from 1 to 65535. The default port is 5351.
Step 6	**end** or **commit** **Example:** `RP/0/RP0/CPU0:router(config-cgn-invrf)# end` `or` `RP/0/RP0/CPU0:router(config-cgn-invrf)# commit`	Saves configuration changes. • When you issue the **end** command, the system prompts you to commit changes: *Uncommitted changes found, commit them before exiting (yes/no/cancel)?* *[cancel]:* • Entering **yes** saves configuration changes to the running configuration file, exits the configuration session, and returns the router to EXEC mode.

	Command or Action	Purpose
		• Entering **no** exits the configuration session and returns the router to EXEC mode without committing the configuration changes. • Entering **cancel** leaves the router in the current configuration session without exiting or committing the configuration changes. • Use the **commit** command to save the configuration changes to the running configuration file and remain within the configuration session.

Configuration Examples for Implementing the Carrier Grade NAT

This section provides the following configuration examples for CGN:

Note Do not configure multiple outside address-pools to be mapped to a single inside-vrf. If you have multiple outside address-pools to be mapped, then create multiple inside-vrfs and map each outside address-pool to a single inside-vrf inside the NAT44 configuration.

Configuring a Different Inside VRF Map to a Different Outside VRF: Example

This example shows how to configure a different inside VRF map to a different outside VRF and different outside address pools:

```
service cgn cgn1
inside-vrf insidevrf1
map outside-vrf outsidevrf1 address-pool 100.1.1.0/24
!
!
inside-vrf insidevrf2
map outside-vrf outsidevrf2 address-pool 100.1.2.0/24
!
service-location preferred-active 0/2/cpu0 preferred-standby 0/3/cpu0
!
interface ServiceApp 1
vrf insidevrf1
ipv4 address 210.1.1.1 255.255.255.0
service cgn cgn1
!
router static
vrf insidevrf1
0.0.0.0/0 serviceapp 1
!
!
interface ServiceApp 2
vrf insidevrf2
```

```
ipv4 address 211.1.1.1 255.255.255.0
service cgn cgn1
service-type nat44 nat1
!
router static
vrf insidevrf2
0.0.0.0/0 serviceapp 2
!
!
interface ServiceApp 3
vrf outsidevrf1
ipv4 address 1.1.1.1 255.255.255.0
service cgn cgn1
service-type nat44 nat1
!
router static
vrf outsidevrf1
100.1.1.0/24 serviceapp 3
!
!
interface ServiceApp 4
vrf outsidevrf2
ipv4 address 2.2.2.1 255.255.255.0
service cgn cgn1
service-type nat44 nat1
!
router static
vrf outsidevrf2
100.1.2.0/24 serviceapp 4
!
```

Configuring a Different Inside VRF Map to a Same Outside VRF: Example

This example shows how to configure a different inside VRF map to the same outside VRF but with different outside address pools:

```
service cgn cgn1
inside-vrf insidevrf1
map outside-vrf outsidevrf1 address-pool 100.1.1.0/24
!
inside-vrf insidevrf2
map outside-vrf outsidevrf1 address-pool 200.1.1.0/24
!
!
service-location preferred-active 0/2/cpu0 preferred-standby 0/3/cpu0
!
interface ServiceApp 1
vrf insidevrf1
ipv4 address 1.1.1.1 255.255.255.0
service cgn cgn1
!
router static
vrf insidevrf1
0.0.0.0/0 serviceapp 1
!
!
interface ServiceApp 2
vrf insidevrf2
ipv4 address 2.1.1.1 255.255.255.0
service cgn cgn1
!
router static
vrf insidevrf2
```

```
0.0.0.0/0 serviceapp 2
!
!
interface ServiceApp 3
vrf outsidevrf1
ipv4 address 100.1.1.1 255.255.255.0
service cgn cgn1
!
router static
vrf outsidevrf1
100.1.1.0/24 serviceapp 3
200.1.1.0/24 serviceapp 3
!
```

Configuring ACL for a Infrastructure Service Virtual Interface: Example

In the following example output, the IP address 1.1.1.1 is used by the SVI on the MSC side and IP address 1.1.1.2 is used in the CGSE PLIM.

```
RP/0/RP0/CPU0:router# configure
RP/0/RP0/CPU0:router(config)# ipv4 access-list ServiceInfraFilter
RP/0/RP0/CPU0:router(config)# 100 permit ipv4 host 1.1.1.1 any
RP/0/RP0/CPU0:router(config)# 101 permit ipv4 host 1.1.1.2 any

RP/0/RP0/CPU0:router(config)# interface ServiceInfra1
RP/0/RP0/CPU0:router(config-if)# ipv4 address 1.1.1.1 255.255.255.192 service-location
0/1/CPU0
RP/0/RP0/CPU0:router(config-if)# ipv4 access-group ServiceInfraFilter egress
```

Use the **show controllers services boot-params** command to verify the IP addresses of SVI and the CGSE PLIM.

```
RP/0/RP0/CPU0:router# show controllers services boot-params location 0/1/CPU0

===========================================
Boot Params
===========================================
Phase of implmentation   : 1
Application              : CGN
MSC ipv4 addddress       : 1.1.1.1
Octeon ipv4 addddress    : 1.1.1.2
ipv4netmask              : 255.255.255.252
```

NAT44 Configuration: Example

This example shows a NAT44 sample configuration:

```
IPv4: 40.22.22.22/16
!
interface Loopback40
 description IPv4 Host for NAT44
 ipv4 address 40.22.22.22 255.255.0.0
!
interface Loopback41
 description IPv4 Host for NAT44
 ipv4 address 41.22.22.22 255.255.0.0
!
interface GigabitEthernet0/3/0/0.1
 description Connected to P2_CRS-8 GE 0/6/5/0.1
 ipv4 address 10.222.5.22 255.255.255.0
 encapsulation dot1q 1
!
```

```
router static
 address-family ipv4 unicast
  180.1.0.0/16 10.222.5.2
  181.1.0.0/16 10.222.5.2
!
!
Hardware Configuration for CSGE:
!
vrf InsideCustomer1
 address-family ipv4 unicast
 !
!
vrf OutsideCustomer1
 address-family ipv4 unicast
 !
!
hw-module service cgn location 0/3/CPU0
!
service-plim-ha location 0/3/CPU0 datapath-test
service-plim-ha location 0/3/CPU0 core-to-core-test
service-plim-ha location 0/3/CPU0 pci-test
service-plim-ha location 0/3/CPU0 coredump-extraction
!
!
interface GigabitEthernet0/6/5/0.1
 vrf InsideCustomer1
 ipv4 address 10.222.5.2 255.255.255.0
 encapsulation dot1q 1
!
interface GigabitEthernet0/6/5/1.1
 vrf OutsideCustomer1
 ipv4 address 10.12.13.2 255.255.255.0
 encapsulation dot1q 1
!
interface ServiceApp1
 vrf InsideCustomer1
 ipv4 address 1.1.1.1 255.255.255.252
 service cgn cgn1 service-type nat44
!
interface ServiceApp2
 vrf OutsideCustomer1
 ipv4 address 2.1.1.1 255.255.255.252
 service cgn cgn1 service-type nat44
!
interface ServiceInfra1
 ipv4 address 75.75.75.75 255.255.255.0
 service-location 0/3/CPU0
!
!
router static
 !
vrf InsideCustomer1
  address-family ipv4 unicast
   0.0.0.0/0 ServiceApp1
   40.22.0.0/16 10.222.5.22
   41.22.0.0/16 10.222.5.22
   181.1.0.0/16 vrf OutsideCustomer1 GigabitEthernet0/6/5/1.1 10.12.13.1
  !
 !
 vrf OutsideCustomer1
  address-family ipv4 unicast
   40.22.0.0/16 vrf InsideCustomer1 GigabitEthernet0/6/5/0.1 10.222.5.22
   41.22.0.0/16 vrf InsideCustomer1 GigabitEthernet0/6/5/0.1 10.222.5.22
   100.0.0.0/24 ServiceApp2
```

```
   180.1.0.0/16 10.12.13.1
   181.1.0.0/16 10.12.13.1
  !
 !
!
CGSE Configuration:
service cgn cgn1
 service-location preferred-active 0/3/CPU0
 service-type nat44 nat44
  portlimit 200
  alg ActiveFTP
  inside-vrf InsideCustomer1
   map outside-vrf OutsideCustomer1 address-pool 100.0.0.0/24
   protocol tcp
    static-forward inside
     address 41.22.22.22 port 80
    !
   !
   protocol icmp
    static-forward inside
     address 41.22.22.22 port 80
    !
   !
   external-logging netflow version 9
    server
     address 172.29.52.68 port 2055
     refresh-rate 600
     timeout 100 !
   !
  !
 !
!
IPv4: 180.1.1.1/16
!
interface Loopback180
 description IPv4 Host for NAT44
 ipv4 address 180.1.1.1 255.255.0.0
!
interface Loopback181
 description IPv4 Host for NAT44
 ipv4 address 181.1.1.1 255.255.0.0
!
interface GigabitEthernet0/6/5/1.1
 ipv4 address 10.12.13.1 255.255.255.0
 encapsulation dot1q 1
!
router static
 address-family ipv4 unicast
  40.22.0.0/16 10.12.13.2
  41.22.0.0/16 10.12.13.2
  100.0.0.0/24 10.12.13.2 !
!
```

NAT64 Stateful Configuration on CGSE Plus: Example

This example shows a NAT64 Stateful sample configuration on CGSE Plus.

```
IPv6 Configuration:
interface Loopback210
description IPv6 Host for NAT64 CGSE Plus
ipv6 address 2001:db8:1c0:2:2100::/64
ipv6 enable
!
```

```
interface GigabitEthernet0/3/0/0.20
description Connected to P2_CRS-8 GE 0/6/5/0.20
ipv6 address 2010::22/64
ipv6 enable
dot1q vlan 20
!
router static
!
address-family ipv6 unicast
2001:db8:100::/40 2010::2
!!
CGSE Plus Hardware Configuration:
hw-module service cgn location 0/3/CPU0
!
service-plim-ha location 0/3/CPU0 datapath-test
service-plim-ha location 0/3/CPU0 core-to-core-test
service-plim-ha location 0/3/CPU0 pci-test
service-plim-ha location 0/3/CPU0 coredump-extraction
!
interface GigabitEthernet0/6/5/0.20
description Connected to PE22_C12406 GE 0/3/0/0.20
ipv6 address 2010::2/64
ipv6 enable
dot1q vlan 20
!
interface GigabitEthernet0/6/5/1.20
description Connected to P1_CRS-8 GE 0/6/5/1.20
ipv4 address 10.97.97.2 255.255.255.0
dot1q vlan 20
!
interface ServiceApp4
ipv4 address 7.1.1.1 255.255.255.252
service cgn cgn1 service-type nat64 stateful
!
interface ServiceApp6
ipv6 address 2011::1/64
service cgn cgn1 service-type nat64 stateful

!
interface ServiceInfra1
ipv4 address 75.75.75.75 255.255.255.0
service-location 0/3/CPU0
!
router static
address-family ipv4 unicast
192.0.2.0/24 ServiceApp4
198.51.100.0/24 10.97.97.1
!
address-family ipv6 unicast
2001:db8:100::/40 ServiceApp6
2001:db8:1c0:2::/64 2010::22
!!
CGSE Plus Configuration:
service cgn cgn1
service-location preferred-active 0/3/CPU0
!
service-type nat64 stateful nat64cgseplus
ipv6-prefix 2001:db8:100::/40
address-family ipv4
tos 64
interface ServiceApp4
tcp mss 1200
!
address-family ipv6
```

```
interface ServiceApp6
traffic-class 32
tcp mss 1200
df-override
!

IPv4 Hardware Configuration:
interface Loopback251
description IPv4 Host for NAT64
ipv4 address 198.51.100.2 255.255.255.0
!
interface GigabitEthernet0/6/5/1.20
description Connected to P2_CRS-8 GE 0/6/5/1.20
ipv4 address 10.97.97.1 255.255.255.0
dot1q vlan 20
!
router static
address-family ipv4 unicast
192.0.2.0/24 10.97.97.2 !
!
```

NAT64 Stateless Configuration: Example

This example shows a NAT64 Stateless sample configuration.

```
IPv6 Configuration:
interface Loopback210
 description IPv6 Host for NAT64 XLAT
 ipv6 address 2001:db8:1c0:2:2100::/64
 ipv6 enable
!
interface GigabitEthernet0/3/0/0.20
 description Connected to P2_CRS-8 GE 0/6/5/0.20
 ipv6 address 2010::22/64
 ipv6 enable
 dot1q vlan 20
!
router static
 !
 address-family ipv6 unicast
  2001:db8:100::/40 2010::2
!
!
CGSE Hardware Configuration:
hw-module service cgn location 0/3/CPU0
!
service-plim-ha location 0/3/CPU0 datapath-test
service-plim-ha location 0/3/CPU0 core-to-core-test
service-plim-ha location 0/3/CPU0 pci-test
service-plim-ha location 0/3/CPU0 coredump-extraction
!
interface GigabitEthernet0/6/5/0.20
 description Connected to PE22_C12406 GE 0/3/0/0.20
 ipv6 address 2010::2/64
 ipv6 enable
 dot1q vlan 20
!
interface GigabitEthernet0/6/5/1.20
 description Connected to P1_CRS-8 GE 0/6/5/1.20
 ipv4 address 10.97.97.2 255.255.255.0
 dot1q vlan 20
```

```
!
interface ServiceApp4
 ipv4 address 7.1.1.1 255.255.255.252
 service cgn cgn1 service-type nat64 stateless
!
interface ServiceApp6
 ipv6 address 2011::1/64
 service cgn cgn1 service-type nat64 stateless
!
interface ServiceInfra1
 ipv4 address 75.75.75.75 255.255.255.0
 service-location 0/3/CPU0
!
router static
 address-family ipv4 unicast
  192.0.2.0/24 ServiceApp4
  198.51.100.0/24 10.97.97.1
 !
 address-family ipv6 unicast
  2001:db8:100::/40 ServiceApp6
  2001:db8:1c0:2::/64 2010::22
!
!
CGSE Configuration:
service cgn cgn1
 service-location preferred-active 0/3/CPU0
 !
 service-type nat64 stateless xlat
  ipv6-prefix 2001:db8:100::/40
  address-family ipv4
   tos 64
   interface ServiceApp4
   tcp mss 1200
  !
  address-family ipv6
   interface ServiceApp6
   traffic-class 32
   tcp mss 1200
   df-override
  !
  traceroute translation
   address-pool 202.1.1.0/24
   algorithm Hash
 !
!
IPv4 Hardware Configuration:
interface Loopback251
 description IPv4 Host for NAT64 XLAT
 ipv4 address 198.51.100.2 255.255.255.0
!
interface GigabitEthernet0/6/5/1.20
 description Connected to P2_CRS-8 GE 0/6/5/1.20
 ipv4 address 10.97.97.1 255.255.255.0
 dot1q vlan 20
!
router static
 address-family ipv4 unicast
  192.0.2.0/24 10.97.97.2 !
!
```

Predefined NAT Configuration: Example

This example shows how to configure the predefined NAT for NAT44:

```
service cgn cgn1
service-location preferred-active 0/2/CPU0
service-type nat44 nat1
 inside-vrf red
   map outside-vrf blue address-pool 100.0.0.0/24
   nat-mode
    predefined private-pool 103.1.106.0/24
    predefined private-pool 103.1.107.0/26
    predefined private-pool 103.1.107.128/26
    predefined private-pool 103.1.108.0/23
    predefined private-pool 103.1.112.0/22
    predefined private-pool 103.1.116.0/24
    predefined private-pool 103.1.117.64/26
    predefined private-pool 103.1.117.192/26
```

DS Lite Configuration: Example

IPv6 ServiceApp and Static Route Configuration

```
conf
int serviceApp61
    service  cgn  cgn1 service-type  ds-lite
    ipv6 address 2001:202::/32
    commit
exit

router static
  address-family ipv6 unicast
  3001:db8:e0e:e01::/128 ServiceApp61 2001:202::2
  commit
  exit
end
```

IPv4 ServiceApp and Static Route Configuration

```
conf
int serviceApp41
    service  cgn  cgn1 service-type  ds-lite
    ipv4 add 41.41.41.1/24
    commit
exit

router static
  address-family ipv4 unicast
  52.52.52.0/24 ServiceApp41 41.1.1.2
  commit
  exit
end
```

DS Lite Configuration

```
service cgn cgn1
 service-location preferred-active 0/2/CPU0 preferred-standby 0/4/CPU0
  service-type ds-lite dsl1
    portlimit 200
    bulk-port-alloc size 128
    map address-pool 52.52.52.0/24
    aftr-tunnel-endpoint-address 3001:DB8:E0E:E01::
    address-family ipv4
      interface ServiceApp41
    address-family ipv6
```

```
    interface ServiceApp61
   protocol tcp
     session init timeout 300
     session active timeout 400
     mss 1200
 external-logging netflow9
    server
     address 90.1.1.1 port 99
  external-logging syslog
   server
     address 90.1.1.1 port 514
```

Bulk Port Allocation and Syslog Configuration: Example

```
service cgn cgn2
 service-type nat44 natA
  inside-vrf broadband
   map address-pool 100.1.2.0/24
   external-logging syslog
    server
     address 20.1.1.2 port 514
    !
   !
   bulk-port-alloc size 64
  !
 !
```

Predefined NAT Configuration: Example

This example shows how to configure the predefined NAT for NAT44:

```
service cgn cgn1
service-location preferred-active 0/2/CPU0
service-type nat44 nat1
 inside-vrf red
   map outside-vrf blue address-pool 100.0.0.0/24
   nat-mode
    predefined private-pool 103.1.106.0/24
    predefined private-pool 103.1.107.0/26
    predefined private-pool 103.1.107.128/26
    predefined private-pool 103.1.108.0/23
    predefined private-pool 103.1.112.0/22
    predefined private-pool 103.1.116.0/24
    predefined private-pool 103.1.117.64/26
    predefined private-pool 103.1.117.192/26
```

PPTP ALG Configuration: Example

NAT44 Instance

```
service cgn cgn1
 service-location preferred-active 0/1/CPU0
 service-type nat44 inst1
  alg pptpAlg
```

-

DBL Configuration: Example

NAT44 Instance

```
service cgn cgn1
 service-type nat44 nat1
  inside-vrf ivrf
   external-logging netflow version 9
    server
     address x.x.x.x port x
      session-logging
```

DS-Lite Instance

```
service cgn cgn1
service-type ds-lite ds-lite1
 external-logging netflow9
   server session-logging
```

PCP Server Configuration: Example

NAT44 Instance

```
service cgn cgn1
service-type nat44 nat1
 inside-vrf ivrf
  pcp-server address 10.25.1256.34 port 66
```

DS-Lite Instance

```
service cgn cgn1
service-type ds-lite ds-lite-inst
 pcp-server port 66
```

Services Redundancy Configuration (Active/Standby): Example

Active Configuration

```
conf t
interface ServiceInfra 1
service-location 0/1/CPU0
ipv4 address 50.1.1.1/24
exit

hw-module service cgn location 0/1/CPU0
commit
exit
```

Stand By Configuration

```
conf t
interface ServiceInfra 2
service-location 0/2/CPU0
ipv4 address 100.1.1.1/24
exit

hw-module service cgn location 0/2/CPU0
commit
```

```
exit
conf t
service cgn cgn1
 service-location preferred-active 0/1/CPU0 preferred-standby 0/2/CPU0
commit
exit
```

Configuration of Multiple Address Pools: Example

In the following example, 2 address pools are being configured for the same inside VRF.

```
service cgn cgn1
service-type nat44 nat1
inside-vrf insidevrf1
map address-pool 100.1.1.0/24
map address-pool 100.1.2.0/24
```

Configuration of Port Limit per VRF: Example

In the following example the portlimit value 40 overrides the portlimit value 200.

```
service cgn cgn1
service-location preferred-active 0/3/CPU0
service-type nat44 nat44
portlimit 100
inside-vrf InsideCustomer1
 portlimit 300
```

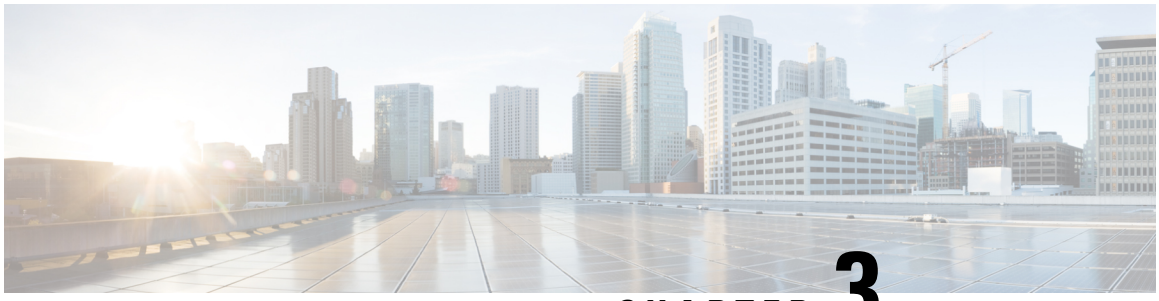
Configuration of Same Public Address Pool across Different NAT Instances: Example

In the following example, both the NAT instances use the same address pool.

```
service cgn cgn1
service-type nat44 nat1
inside-vrf insidevrf1
map address-pool 100.1.1.0/24
end
configure
service cgn cgn2
service-type nat44 nat2
inside-vrf insidevrf2
map address-pool 100.1.1.0/24
end
```

High Availability on data Path SVI: Example

```
service cgn cgn1
   service-type tunnel v6rd 6rd1
     address-family ipv4
       interface ServiceApp 100
        datapath-test shut-down-on-failure
```



CHAPTER 3

External Logging

External logging configures the export and logging of the NAT table entries, private bindings that are associated with a particular global IP port address, and to use Netflow to export the NAT table entries.

Bulk Port Allocation

The creation and deletion of NAT sessions lead to creation of logs. If logs of all such translations are stored, then a huge volume of data is created. This data is stored on a NetFlow or a Syslog collector. To reduce the volume of this data, a block of ports is allocated. If bulk port allocation is enabled, as soon as a subscriber creates the first session, a number of contiguous external ports are allocated. To indicate this allocation, a bulk allocation message is created in the log.

Note The bulk allocation message is created only during the first session. Rest of the sessions use one of the allocated ports. Hence no logs are created for them.

A bulk delete message is created in the log when the subscriber deletes all the sessions that are using the allocated ports.

Another pool of ports is allocated only if the number of simultaneous sessions is more than N where N is the size of the bulkk allocation. The size of the pool can be configured from the CLI.

Restrictions for Bulk Port Allocation

The restrictions for bulk port allocation are as follows:

- The value for the size of bulk allocation can be 16, 32, 64, 128, 256, 512, 1024, 2048 and 4096. For optimum results, it is recommended that you set this size to half of the port limit.
- If the size of bulk allocation is changed, then all the current dynamic transactions will be deleted. Hence it is advisable to change the bulk port allocation size (only if necessary) during a maintenance window.

- The port numbers below the value of dynamic-port-range start value (which is 1024 by default), are not allocated in bulk.
- The algorithm that is used to allocate a public address to a user remains the same.
- When bulk allocation is enabled, session logging is not available.
- When bulk allocation is enabled, the translation record will not contain information about L4 protocol.
- Bulk port allocation features is not supported in NAT64 stateful application.

Session logging

In general, NAT translation entries contain information about private source IP, port and translated public IP and port. However, there could be cases when the destination IP address (public IP address) and port may also be needed. In such cases, session logging has to be enabled so that Netflow or Syslog translation records include these values as well.

Syslog Logging

Perform the following tasks to configure Syslog Logging for NAT table entries.

Restrictions for Syslog

The restrictions for syslog are as follows:

- Syslog is supported over UDP only.
- Syslog is supported in ASCII format only.
- You cannot log onto multiple collectors or relay agents.
- All the messages comply to RFC 3954 except for the timestamp format. Timestamp is represented in a simpler way as explained later in this section.
- Syslog shall be supported for DS-Lite and NAT444 as of now. Support for NAT64 is not yet available.
- The Syslog collector shall be assumed to be in the default VRF. If not, appropriate ACLs have to configured and applied on to Service Infra interface to divert the Syslog packets from default VRF to specific VRF through which the collector can be reached.

Syslog Message Format

In general, the syslog message is made up of header, structured data, and msg fields. However, in the CGv6 applications, the structured data is not used.

Header

The header fields shall be as per the RFC 5424. Fields shall be separated by ' ' (white space) as per the RFC.

The header consists of the following fields:

Field	Description
Priority	• The priority value represents both the facility and severity. • Ensure that the severity code is set to Informational for all the messages at value 6.
Version	• This field denotes the version of the specification of the syslog protocol. • In CGv6 application, the version value is set to 1.
Timestamp	• This field is needed to trace the port usage. • The format is <year> <mon> <day> <hh:mm:ss>. • Ensure that the syslog collector converts the time to local time whenever needed. **Note** The timestamp is always reported in GMT/UTC irrespective of the time zone configured on the device.
Hostname	• This field is used to identify the device that sent the syslog message. • While configuring the syslog server, ensure that the host name does not exceed 31 characters. • The default value for the host name is '-'.
App name and PROC ID	These fields are not included. In ASCII format, '-' is included for these fields.
MSG ID	• This field identifies the type of the syslog message. • In the ASCII format, the values for NAT44 and DS Lite messages are NAT44 and DS LITE respectively.

Structured Data

It is not used.

MSG

This field consists of the information about the NAT44 or DS Lite events. In a single UDP packet, there could be one or more MSG fields each enclosed in [] brackets. The MSG field has many sub fields as it has a common structure across different records (for both NAT44 and DS Lite). Note, that, depending on the event, some of

the fields may not be applicable. For example, fields such as 'Original Source IPv6' address are not applicable for all NAT44 events. In such cases, the inapplicable fields will be replaced by '-'.

The syntax of the MSG part is as follows:

[EventName <L4> <Original Source IP> <Inside VRF Name> <Original Source IPv6> < Translated Source IP> <Original Port> <Translated First Source Port> <Translated Last Source Port> <Destination IP> <Destination Port>]

The descriptions of the fields in this format are as follows:

Field	Description
EventName	Select any one of the values for EventName from the following based on the event: • UserbasedA: User-based port assignment • SessionbasedA: Session-based port assignment • SessionbasedAD: Session-based port assignment with destination information Note: SessionbasedAD is used only if session logging is enabled. Also, session-logging and bulk port allocation are mutually exclusive. • UserbasedW: User-based port withdrawal • SessionbasedW: Session-based port withdrawal • SessionbasedWD: Session-based port withdrawal with destination information • Portblockrunout: Ports exhausted
L4	Specifies the identifier for the transport layer protocol. Select any one of the values for L4 from the following: • 1 for ICMP • 6 for TCP • 17 for UDP • 47 for GRE
Original Source IP	Specify the private IPv4 address.
Inside VRF Name	The Inside vrf is essential to identify the subscriber. Even though multiple subscribers connected to the router might have the same source IP, they might be placed in different VRFs. Hence the VRF name and the original source IP together helps to identify a subscriber. Ensure that the VRF name is not more than 32 characters in length.

Field	Description
Original Source IPv6	Specifies the IPv6 source address of the tunnel in case of DS Lite.
Translated Source IP	Specifies the public IPv4 address post translation.
Original Port	Specifies the source port number before translation. This is not applicable for the UserbasedA and UserbasedW events.
Translated First Source Port	Specifies the first source port after translation.
Translated Last Source Port	Specifies the last source port after translation. This is applicable only for the UserbasedA and UserbasedW events.
Destination IP	Specifies the destination IP recorded in the syslogs for the SessionbasedAD and SessionbasedWD events.
Destination Port	Specifies the destination port recorded in the syslogs for the SessionbasedAD and SessionbasedWD events.

Let us look at an example for NAT444 user-based UDP port translation mapping:

```
[UserbasedA - 10.0.0.1 Broadband - 100.1.1.1 - 2048 3071 - -]
```

The description for this example is as follows:

Value	Description
UserbasedA	Event Name
10.0.0.1	Original Source IP
Broadband	Inside VRF name
100.1.1.1	Translated Source IP
2048	Translated First Source Port
3071	Translated Last Source Port

Note The number of MSG fields in an UDP packet are determined by the following factors:

- The space available in the UDP packet depends on MTU.
- The translation events pertaining to MSG records in a given packet must have happened within a second (starting from the time at which the first event of that packet happened).

Netflow v9 Support

The NAT64 stateful, NAT44, and DS Lite features support Netflow for logging of the translation records.. The Netflow uses binary format and hence requires software to parse and present the translation records. However, for the same reason, Netflow requires lesser space than Syslog to preserve the logs.

Considerations

The considerations for NetFlow are as follows:

- NetFlow V9 is supported over UDP.
- You cannot log onto multiple collectors or relay agents.
- All the messages comply to RFC 3954.

NetFlow Record Format

As NetFlow V9 is based on templates, the record format contains a packet header and templates or data records based on templates.

Header

All the fields of the header follow the format prescribed in RFC 3954. The source ID field is composed of the IPv4 address of ServiceInfra interface (of the card) and specific CPU-core that is generating the record. The collector device can use the combination of the sourceIPv4Address field plus the Source ID field to associate an incoming NetFlow export packet with a unique instance of NetFlow on a particular device.

Templates for NAT44

The templates are defined and used for logging various NAT64 stateful, NAT44 and DS Lite events as follows. The templates may change in future software releases. Hence it is advised that the Netflow collector software is designed to understand the templates as distributed by the router and accordingly parse the records.

Options Templates

The translation entries consist of VRF IDs which might be incomprehensible to a user. To simplify this process, the CGv6 applications send the options templates along with the data templates.

Options template is a special type of data record that indicates the format of option data related to the process of NetFlow. The options data consist of the mapping between VRF Ids and VRF names. By parsing and using this data, the NetFlow collectors can modify the translation entries by adding VRF names instead of VRF IDs.

The value for the Template ID of options template is 1 where as the value of the Template ID for data template is 0. For more information on Options template, see RFC3954.

Events

The events and the corresponding template details are described in the following table:

Event	Template ID	Bulk Port Allocation	Destination/Session Logging	Field Name	IANA IPFIX ID	Size in bytes	Description
Nat444 translation create event	256	Disabled	Disabled	ingressVRFID	234	4	ID of the Ingress VRF
				egressVRFID	235	4	ID of the Egress VRF
				sourceIPv4Address (pre-NAT)	8	4	Original Source IPv4 address
				postNATSourceIPv4 Address	225	4	Post NAT (outside) source IPV4 address
				sourceTransportPort (pre NAT)	7	2	Original source port
				sourceTransportPort	7	2	Post NAT (translated) source port
				protocolIdentifier	4	1	L4 protocol identifier

Event	Template ID	Bulk Port Allocation	Destination/Session Logging	Field Name	IANA IPFIX ID	Size in bytes	Description
Nat444 session create event - session based (with destination)	271	Disabled	Enabled	ingressVRFID	234	4	ID of the Ingress VRF
				egressVRFID	235	4	ID of the Egress VRF
				sourceIPv4Address	8	4	Original source IPV4 address
				postNATSourceIPv4Address	225	4	Post NAT (outside) source IPV4 address
				sourceTransportPort	7	2	Original Source Port
				sourceTransportPort	7	2	Post NAT (translated) source port
				destinationIPv4Address	12	4	Destination IP address
				destinationTransportPort	11	2	Destination port
				protocolIdentifier	4	1	L4 protocol identifier

Event	Template ID	Bulk Port Allocation	Destination/Session Logging	Field Name	IANA IPFIX ID	Size in bytes	Description
Nat444 translation create event - user based	265	Enabled	Disabled	ingressVRFID	234	4	ID of the Ingress VRF
				egressVRFID	235	4	ID of the Egress VRF
				sourceIPv4Address	8	4	Original source IPV4 address
				postNATSourceIPv4Address	225	4	Post NAT (outside) source IPV4 address
				postNATPortBlockStart	361	2	Start of Post NAT (translated) source port block
				postNATPortBlockEnd	362	2	End of Post NAT source port block

Event	Template ID	Bulk Port Allocation	Destination/Session Logging	Field Name	IANA IPFIX ID	Size in bytes	Description
Nat444 translation delete event	257	Disabled		ingressVRFID	234	4	ID of the Ingress VRF
				sourceIPv4Address	8	4	Original source IPV4 address
				sourceTransportPort	7	2	Original source port
				protocolIdentifier	4	1	L4 protocol identifier
Nat444 session delete event - session based (with destination)	272	Disabled	Enabled	ingressVRFID	234	4	ID of the Ingress VRF
				sourceIPv4Address	8	4	Original source IPV4 address
				destinationIPv4Address	12	4	Destination IP address
				sourceTransportPort	7	2	Post NAT (translated) source port
				destinationTransportPort	11	2	Destination port
				protocolIdentifier	4	1	L4 protocol identifier

Event	Template ID	Bulk Port Allocation	Destination/Session Logging	Field Name	IANA IPFIX ID	Size in bytes	Description
Nat444 translation delete event - user based	266	Disabled	Disabled	ingressVRFID	234	4	ID of the Ingress VRF
				sourceIPv4Address	8	4	Original source IPV4 address
				postNATPortBlockStart	361	2	Start of Post NAT (translated) source port block. Note this is not defined by IANA yet.
DS-Lite translation create event	267	Disabled	Disabled	ingressVRFID	234	4	ID of the Ingress VRF
				egressVRFID	235	4	ID of the Egress VRF

Event	Template ID	Bulk Port Allocation	Destination/Session Logging	Field Name	IANA IPFIX ID	Size in bytes	Description
				Pre NAT Source IPv4 Address	8	4	Original source IPV4 address. This field is valid only when session logging is enabled. Else, it will be reported as 0
				Pre NAT Source IPv6 Address	27	16	IPv6 address of the B4 element (Tunnel source)
				postNATSourceIPv4Address	225	4	Post NAT (outside) source IPV4 address
				sourceTransportPort	7	2	Original source port
				sourceTransportPort	227	2	Post NAT (translated) source port

Event	Template ID	Bulk Port Allocation	Destination/Session Logging	Field Name	IANA IPFIX ID	Size in bytes	Description
DS-Lite session create event - session based (with destination)	273	Disabled	Enabled	ingressVRFID	234	4	ID of the Ingress VRF
				egressVRFID	235	4	ID of the Egress VRF
				sourceIPv4Address	8	4	Original source IPV4 address
				sourceIPv6Address	27	16	IPv6 address of the B4 element (Tunnel source)
				postNATSourceIPv4Address	225	4	Post NAT (outside) source IPV4 address
				sourceTransportPort	7	2	Original source port
				sourceTransportPort	227	2	Post NAT (translated) source port
				destinationIPv4Address	12	4	Destination IP address
				destinationTransportPort	11	2	Destination port
				protocolIdentifier	4	1	L4 protocol identifier

Event	Template ID	Bulk Port Allocation	Destination/Session Logging	Field Name	IANA IPFIX ID	Size in bytes	Description
DS-Lite translation create event - user based	269	Enabled	Disabled	ingressVRFID	234	4	ID of the Ingress VRF
				egressVRFID	235	4	ID of the Egress VRF
				sourceIPv4Address	8	4	Original source IPV4 address. This field is valid only when session logging is enabled. Else, it will be reported as 0
				sourceIPv6Address	27	16	IPv6 address of the B4 element (Tunnel source)

Event	Template ID	Bulk Port Allocation	Destination/Session Logging	Field Name	IANA IPFIX ID	Size in bytes	Description
DS-Lite translation create event - user based				postNATSourceIPv4Address	225	4	Post NAT (outside) source IPV4 address
				postNATPortBlockStart	361	2	Start of Post NAT (translated) source port block
				postNATPortBlockEnd	362	2	End of Post NAT source port block
DS-Lite translation delete event	270	Disabled	Disabled	ingressVRFID	234	4	ID of the Ingress VRF
				sourceIPv4Address			Original source IPV4 address
				sourceIPv6Address			IPv6 address of the B4 element (Tunnel source)
				sourceTransportPort			Original source port
				protocolIdentifier			L4 protocol identifier

Event	Template ID	Bulk Port Allocation	Destination/Session Logging	Field Name	IANA IPFIX ID	Size in bytes	Description
DS-Lite session delete event - session based (with destination)				ingressVRFID	234	4	ID of the Ingress VRF
				sourceIPv4Address	8	4	Original source IPV4 address
				sourceIPv6Address	27	16	IPv6 address of the B4 element (Tunnel source)
				sourceTransportPort	7	2	Original source port
				protocolIdentifier	4	1	L4 protocol identifier
DS-Lite translation delete event - user based	270	Disabled	Disabled	ingressVRFID	234	4	ingressVRFID
				sourceIPv4Address	8	4	Original source IPV4 address
				sourceIPv6Address	27	16	IPv6 address of the B4 element (Tunnel source)
				postNATPortBlockStart	361	2	Start of Post NAT (translated) source port block

Event	Template ID	Bulk Port Allocation	Destination/Session Logging	Field Name	IANA IPFIX ID	Size in bytes	Description
NAT64 stateful translation create event	258	Disabled	Disabled	sourceIPv6Address	27	16	Source IPv6 address
				postNATSourceIPv4Address	225	4	Post NAT (outside) source IPV4 address
				sourceTransportPort	7	2	Original source port
				sourceTransportPort	227	2	Post NAT (translated) source port
				protocolIdentifier	4	1	L4 protocol identifier

Event	Template ID	Bulk Port Allocation	Destination/Session Logging	Field Name	IANA IPFIX ID	Size in bytes	Description
NAT64 stateful session create event - session based (with destination)	260	Disabled	Enabled	sourceIPv6Address	27	16	Source IPv6 address (pre translation)
				postNATSourceIPv4Address	225	4	Post NAT (outside) source IPV4 address
				destinationIPv6Address	28	16	Destination IPv6 address (pre translation)
				Post translation Destination IP address	226	4	Destination IPv4 address (post translation)
				sourceTransportPort	7	2	Original source port
				sourceTransportPort	7	2	Post NAT (translated) source port
				destinationTransportPort	11	2	Destination port
				protocolIdentifier	4	1	L4 protocol identifier

Event	Template ID	Bulk Port Allocation	Destination/Session Logging	Field Name	IANA IPFIX ID	Size in bytes	Description
NAT64 translation delete event	259	Disabled	Disabled	sourceIPv6Address	27	16	IPv6 address of the B4 element (Tunnel source)
				sourceTransportPort	7	2	Original source port
				protocolIdentifier	4	1	L4 protocol identifier
NAT64 stateful session delete event - session based (with destination)	261	Disabled	Enabled	sourceIPv6Address	27	16	IPv6 address of the B4 element (Tunnel source)
				destinationIPv6Address	28	16	Destination IPv6 address (pre translation)
				sourceTransportPort	7	2	Original source port
				destinationTransportPort	11	2	Destination port
				protocolIdentifier	4	1	L4 protocol identifier

Frequently Asked Questions (FAQs)

This section provides answers to the following frequently asked questions on external logging.

Q: How to trace a subscriber by using the NAT logs?

A: In order to trace a subscriber, you should know the public source IP address (post NAT source address), post NAT source port, protocol, and the time of usage. With these parameters, the steps to trace a subscriber are as follows:

1. Search for the create event that has the matching public IP address, post NAT Source IP address (postNATSourceIPv4Address) and protocol, egress VRF ID/Name and the time of the usage. Ensure that the time of the create-event is the same or earlier than the time of usage reported. You may not find the protocol entry or the exact post NAT source port in the logs if bulk allocation is enabled. In such cases, find the create-event whose **Post NAT Port Block Start** and **Post NAT Port Block End** values include the post NAT source port. The **Pre NAT source IP address** along with the corresponding **ingress VRF ID/Name** will identify the subscriber.
2. The corresponding delete record may be found optionally to confirm that the subscriber was using the specified public IP and port during the time of the reported usage.

Q: The Netflow records provide VRF IDs for ingress and egress VRFs. How will I know the VRF names?

A: The following are the two ways to find the VRF name from the VRF ID.

1. Use the command show rsi vrf-id <vrf-id> on the Router console to find VRF-ID to VRF-NAME associations.
2. The CGv6 applications periodically send out option templates containing the VRF-ID to VRF-NAME mapping. The Netflow collector software presents the information with VRF-Names rather than VRF IDs.

Q: Does the time format in Syslog or Netflow account for Day light saving?

A: The Syslog and Netflow formats report time corresponding to GMT/UTC. The Netflow header contains the time in seconds that elapsed since EPOCH whereas the Syslog header contains time in human readable formats. In both cases, the day light saving is not accounted. The Netflow/Syslog collectors have to make that adjustments if needed.

Q: Since the Netflow and Syslog use UDP, how can we know if a packet containing translation record was lost?

A: The Netflow header contains a field called Sequence Number. This number is indicates the count of the packet coming from each Source ID. The Netflow collector traces the Seqence Number pertaining to each unique Source ID. The sequence numbers should be increased by one for each packet sent out by the Source. If the collector ever receives two successive packets with the same Source ID, but with a Sequence number difference of more than 1, it indicate a packet loss. However, currently, no such mechanism exists for Syslog.

Q: What is the use of session-logging?

A: Session logging includes destination IP and port number as well. Though this information is not directly useful in tracing the subscriber, in some cases, this information may be useful or may be mandated by the legal authorities. There are cases where, legal authorities may not have the post NAT source 'port', however may know the destination IP address (and optionally destination port, such as IP address and port of an e-mail server). In the absence of post NAT source port information, a list of subscribers who used the specified public IP during that time may have to be pruned further based on the destination IP and port information.

Q: How does the bulk port allocation reduce data volume of translation logs?

A: With bulk port allocation, subscribers are allocated a range of contiguous ports on a public IP. Quite often, a subscriber will need more ports than just one. Especially AJAX based web pages and other web applications simultaneously open several ports. In such cases, pre-allocated ports are used and only one log entry is made

that specifies the range of ports allocated to the user. Hence, bulk port allocation significantly reduces log data volume and hence the demand on storage space needed for the translation logs.

Q: What else can be done to reduce log data volume?

A: Predefined NAT is an option that can be used to eliminate the logging altogether. The Predefined NAT translates private IP address to public IP address and a certain port range by using an algorithm. Hence there is no need to keep track of NAT entries.