Implementing the Carrier Grade NAT on Cisco IOS XR Software

This module describes how to implement the Carrier Grade NAT (CGN) on Cisco IOS XR software.

Contents

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.

Carrier Grade NAT Overview and Benefits

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

- Carrier Grade NAT Overview, page 2
- Benefits of Carrier Grade NAT, page 3
- NAT and NAPT Overview, page 3
- Network Address and Port Mapping, page 4

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.

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.

Information About Implementing Carrier Grade NAT

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

- Implementing NAT with ICMP, page 5
- Implementing NAT with TCP, page 5
- Implementing NAT 44 over ISM, page 6
- Implementing NAT 64 over ISM, page 9
- Double NAT 444, page 12
- Address Family Translation, page 12
- Policy Functions, page 12
- Cisco Carrier-Grade Service Engine (CGSE) Applications, page 13

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, page 5
- Implementing NAT with TCP, page 5

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.

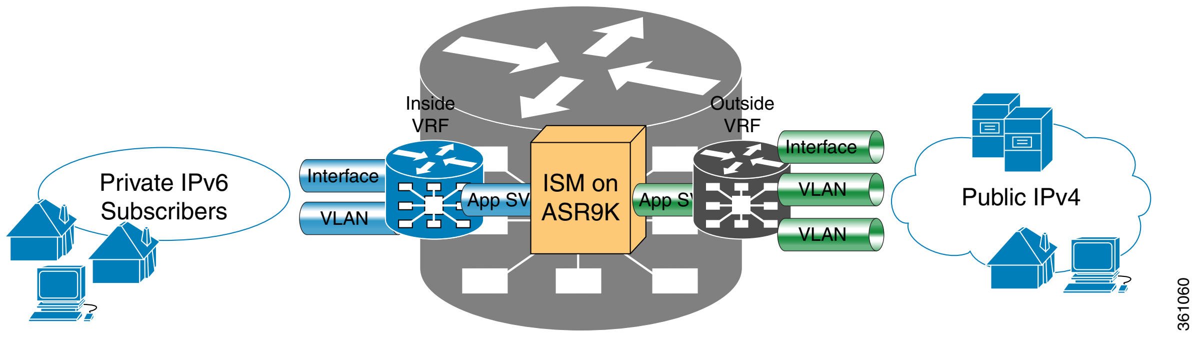
The 3-way handshake corresponds to method of connection initiation.

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

The following figure illustrates the implementation of NAT 44 over ISM.

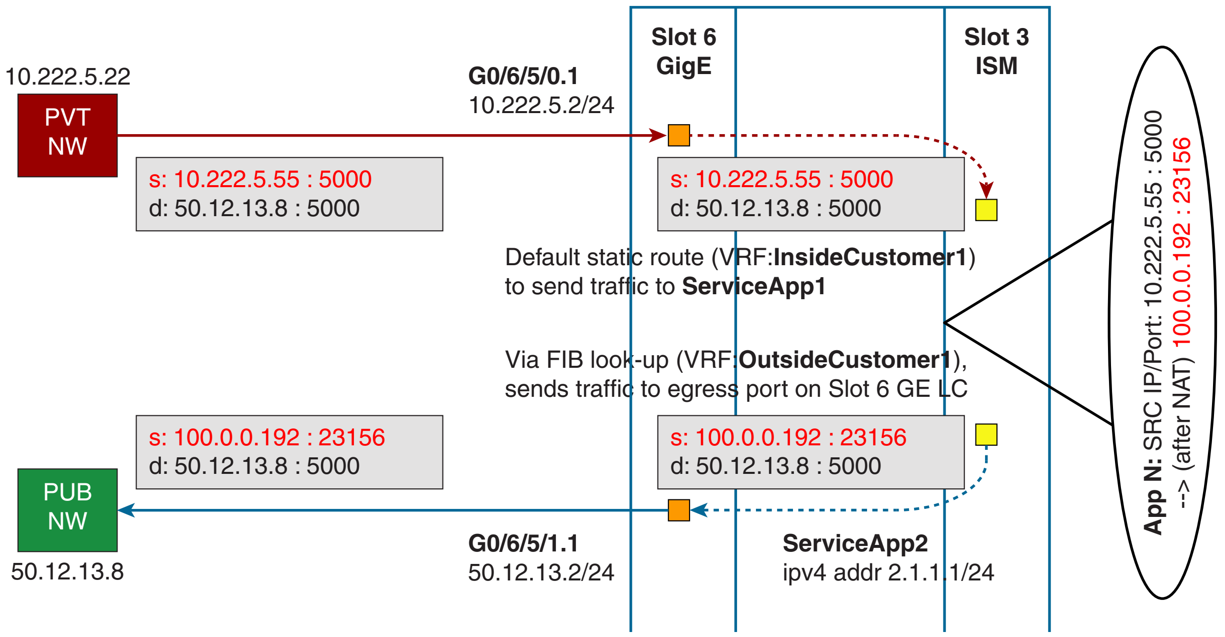


The components of this illustration are as follows:

- 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:

Step 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.

Step 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.

Step 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.

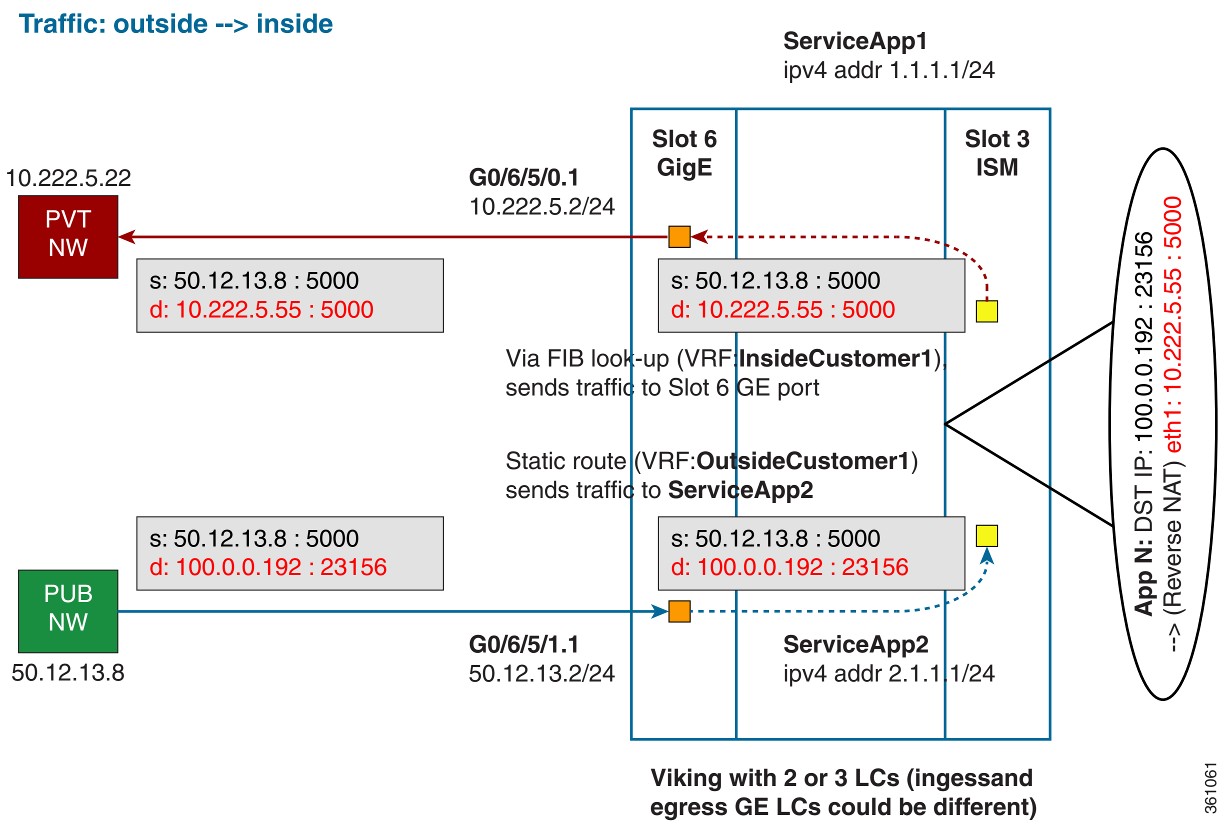
Step 4 The packet is forwarded by AppSVI1 through a default static route (ivrf1). The destination address and the port get translated because of the CGN configuration applied on ISM.

Step 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.

Step 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.

Step 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:

Step 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.

Step 2 The packet enters through the ingress port on the Gigabit Ethernet (GigE) interface at Slot 0.

Step 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).

Step 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.

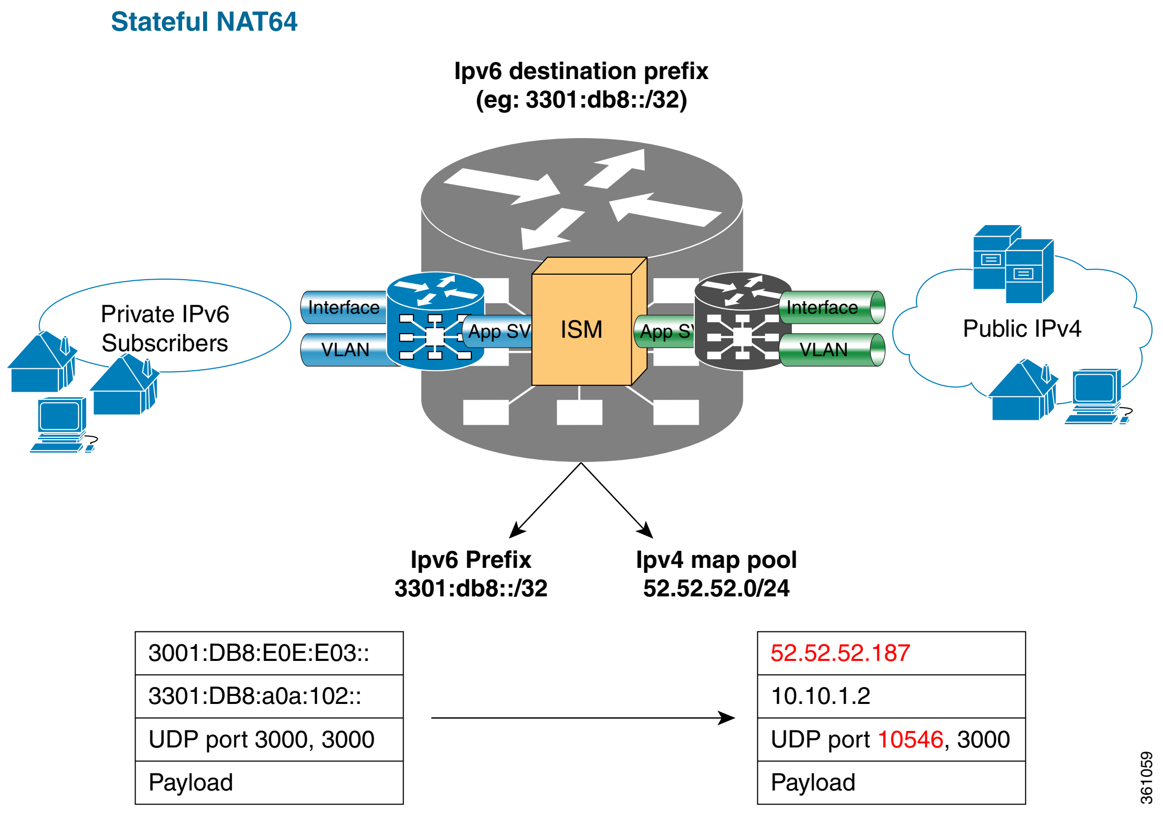
Step 5 The ISM applies NAT44 to the packet. The CGN determines the destination address from the FIB Look Up. It pushes the packet to the egress port.

Step 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.

Step 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.

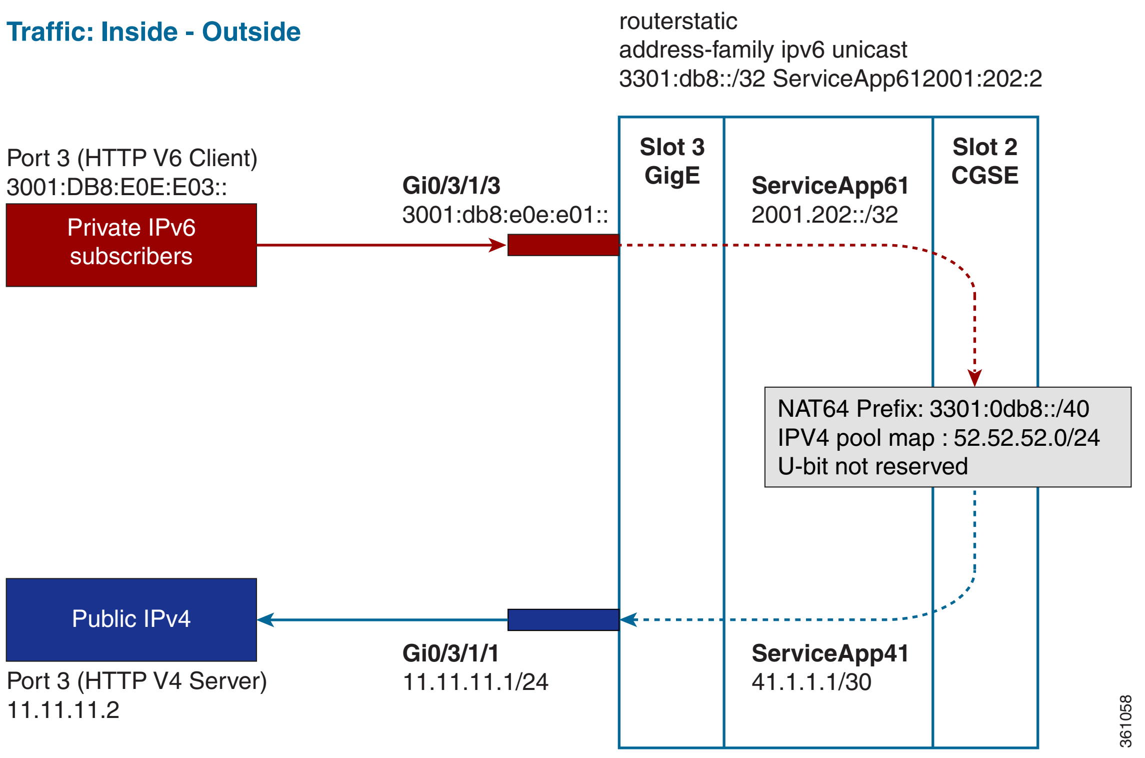


The components of this implementation are as follows:

- 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:

Step 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.

Step 2 The packet enters through the ingress port on the Gigabit Ethernet (GigE) interface at Slot 3.

Step 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.

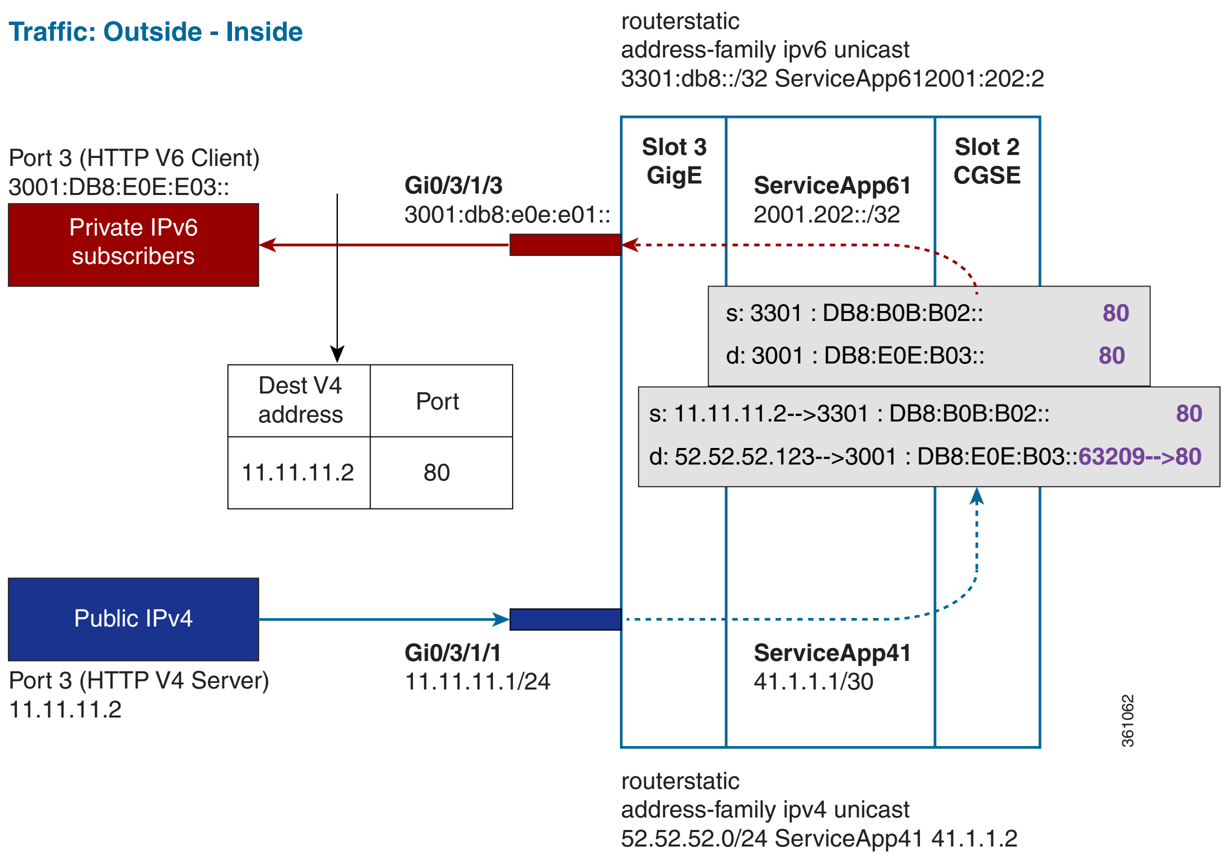
Step 4 The packet is forwarded by AppSVI1 through a default static route. The destination address and the port get translated because of the CGN configuration applied on ISM.

Step 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.

Step 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.

Step 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.


Traffic: Outside - Inside
routerstatic
address-family ipv6 unicast
3301:db8::/32 ServiceApp612001:202:2
Port 3 (HTTP V6 Client)
3001:DB8:E0E:E03::
Private IPv6 subscribers
Gi0/3/1/3
3001:db8:e0e:e01::
Slot 3 GigE
ServiceApp61
2001.202::/32
Slot 2 CGSE
s: 3301 : DB8:B0B:B02:: 80
d: 3001 : DB8:E0E:B03:: 80
Dest V4 address
Port
11.11.11.2
80
s: 11.11.11.2-->3301 : DB8:B0B:B02:: 80
d: 52.52.52.123-->3001 : DB8:E0E:B03::63209-->80
Public IPv4
Port 3 (HTTP V4 Server)
11.11.11.2
Gi0/3/1/1
11.11.11.1/24
ServiceApp41
41.1.1.1/30
361062
routerstatic
address-family ipv4 unicast
52.52.52.0/24 ServiceApp41 41.1.1.2

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

Step 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.

Step 2 The packet enters through the ingress port on the Gigabit Ethernet (GigE) interface at Slot 3.

Step 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.

Step 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.

Step 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.

Step 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.

Step 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.

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.

Policy Functions

Application Level Gateway

The application level gateway (ALG) deals with the applications that are embedded in the IP address payload. Active FTP and RTSP ALG are supported.

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.

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.

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.

Cisco Carrier-Grade Service Engine (CGSE) Applications

A Carrier-Grade Services Engine (CGSE) is a physical line interface module (PLIM). 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.

IPv4/IPv6 Stateless Translator (XLAT)

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.

Advantages of XLAT

These are the advantages of a stateless translator:

- No states maintained in a SL-XLAT. Also, it supports 1:1 IPv6 to IPv4 address mappings. This means that one IPv4 address is consumed for each IPv6 to IPv4 translation.

- It supports asymmetric packet flows. Because it is stateless, it is not necessary to pin individual session flows in both directions to a particular SL-XLAT vehicle.
- It offers basic IP transit between IPv4 and IPv6 networks.

IPv6 Rapid Depolyment (6RD)

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

6RD Definitions

- **6RD CE /RG/CPE:** The 6rd "Customer Edge" router that sits between an IPv6-enabled site and an IPv4-enabled SP network. In the context of residential broadband deployment, this is referred to as the Residential Gateway (RG) or Customer Premises Equipment (CPE) or Internet Gateway Device (IGD). This router has a 6rd tunnel interface acting as an endpoint for the IPv6 in IPv4 encapsulation and forwarding, with at least one 6rd CE LAN side interface and 6rd CE WAN side interface, respectively.
- **6RD Border Relay (BR):** A 6rd-enabled Border Relay router located at the service provider's premises. The 6rd BR router has at least one IPv4 interface, a 6rd tunnel interface for multi-point tunneling, and at least one IPv6 interface that is reachable through the IPv6 Internet or IPv6-enabled portion of the SP network. A router running IOS can also be a 6RD BR.
- **6RD Delegated Prefix:** The IPv6 prefix determined by the 6rd CE device for use by hosts within the customer site.
- **6RD Prefix (SP Prefix) :** An IPv6 prefix selected by the service provider for use by a 6rd domain. There is exactly one 6rd prefix for a given 6rd domain.
- **CE LAN side :** The functionality of a 6rd CE that serves the Local Area Network (LAN) or customer-facing side of the CE. The CE LAN side interface is fully IPv6 enabled.
- **CE WAN side :** The functionality of a 6rd CE that serves the Wide Area Network (WAN) or Service Provider- facing side of the CE. The CE WAN side is IPv4 only.
- **BR IPv4 address :** The IPv4 address of the 6rd Border Relay for a given 6rd domain. This IPv4 address is used by the CE to send packets to a BR in order to reach IPv6 destinations outside of the 6rd domain.

CE IPv4 address : The IPv4 address given to the CE as part of normal IPv4 Internet access (configured through DHCP, PPP, or otherwise). This address may be global or private within the 6rd domain. This address is used by a 6rd CE to create the 6rd delegated prefix, as well as to send and receive IPv4-encapsulated IPv6 packets.

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 platform. CGSE (Carrier Grade Service Engine) has four octeons and supports 20 Gbps full duplex traffic. It works on Linux operating system and traffic into CGSE is forwarded using serviceApp interfaces. SVIs (Service Virtual Interfaces) are configured to enable traffic to flow in and out of CGSE.

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 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.

Functionalities Supported in Stateful NAT64

These functionalities are supported in NAT64 implementation:

- TCP, UDP, and ICMP protocol NAT64
- IPv4 to IPv6 header translation and vice versa
- End point independent mapping
- Address dependant filtering
- Multiple Address Pools
- Well known prefix handling
- Netflowv9 logging
- TCP/UDP/ICMP fragments handling
- IP options and ICMP error handling
- Protocol based session timers
- Destination based session timers
- Hairpinning
- DNS64 being decoupled and NAT64 working with decoupled DNS64
- Multiple NAT64 instances each having configurable options
- XML support for configuration and show commands
- CLI consistent with other CGv6 applications

Note A maximum of 64 NAT64 instances are supported in the NAT64 configuration.

These are the configuration parameters for a NAT64 instance:

- NAT64 Prefix—Indicates IPv6 prefix (for mapping destination IPv4 address – default WKP 64:FF9B::/96)
- NAT64 Prefix Length—Indicates IPv6 NAT64 prefix length (/32, to /96)
- NAT64 IPv4 map address pool—Indicates outside IPv4 address space for this NAT64 instance
- IPv4 serviceApp and IPv6 serviceApp interfaces for the instance
- u-bit-reserved flag—When this configuration is enabled, bits in the range 64-71 in the IPv6 addresses are reserved for several purposes including U-Bit. These bits are not used for translation purposes.

- Static port configuration—This is a protocol based configuration which specifies source IPv6 address that needs static mapping to the outside IPv4
- Protocol based timeouts—Indicates active/init timeouts and per destination IP/port timeouts
- tcp mss—Indicates the tcp mss value to be used while translating packets
- tos, traffic class, df override related flags similar to NAT64 stateless
- Netflow information
- Address dependant filtering enabling
- Port limit
- Destination based active timeouts
- Fragment handling timeouts.

Dual Stack Lite Feature

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.

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.

Attributes of Syslog Collector

1. Syslog is supported in ASCII format only.
2. Logging to multiple syslog collectors (or relay agents) is not supported.
3. Syslog is supported for DS-Lite and NAT444 in the Cisco IOS XR Software Release 4.2.1.

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.

Implementing Carrier Grade NAT on Cisco IOS XR Software

The following configuration tasks are required to implement CGN on Cisco IOS XR software:

- Getting Started with the Carrier Grade NAT, page 18
- Configuring the Service Type Keyword Definition, page 24
- Configuring the Policy Functions for the Carrier Grade NAT, page 27
- Configuring the Carrier Grade Service Engine, page 50
- Configuring IPv4/IPv6 Stateless Translator (XLAT), page 52
- Configuring IPv6 Rapid Development, page 54
- Configuring Dual Stack Lite Instance, page 60

Getting Started with the Carrier Grade NAT

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

- Configuring the Service Role, page 18
- Configuring the Service Instance and Location for the Carrier Grade NAT, page 20
- Configuring the Service Virtual Interfaces, page 21

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.
		– 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.

	Command or Action	Purpose
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]:`
		– 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, page 21
- Configuring the Application Service Virtual Interface, page 23

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, page 66.

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.

	Command or Action	Purpose

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

Configuring the Application Service Virtual Interface

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.
		– 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]:`
		– 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

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

- 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.

	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 the Policy Functions for the Carrier Grade NAT

- Configuring the Port Limit Per Subscriber, page 28
- Configuring the Timeout Value for the Protocol, page 29
- Configuring the Application Level Gateway, page 34
- Configuring the TCP Adjustment Value for the Maximum Segment Size, page 35
- Configuring the Refresh Direction for the Network Address Translation, page 37
- Configuring the Carrier Grade NAT for Static Port Forwarding, page 39
- Configuring the Dynamic Port Ranges for NAT44, page 40

Configuring the Port Limit Per Subscriber

Perform this task to configure the port limit per subscriber for the system that includes TCP, UDP, and ICMP.

SUMMARY STEPS

1. **configure**
2. **service cgn** *instance-name*
3. **service-type nat44 nat1**
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 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	**portlimit** *value* **Example:** `RP/0/RP0/CPU0:router(config-cgn-nat44)# portlimit 10`	Limits the number of entries per address for each subscriber of the system

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 Timeout Value for the Protocol

- Configuring the Timeout Value for the ICMP Protocol, page 29
- Configuring the Timeout Value for the TCP Session, page 31
- Configuring the Timeout Value for the UDP Session, page 32

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.

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.

	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 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.
		– 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 sessions. 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 active timeout 90`	Configures the timeout value as 90 for the UDP session. The example shows how to configure the active 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 Application Level Gateway

Perform this task to configure the application level gateway (ALG) for the rtsp for the specified CGN 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 CGN NAT44 application.

	Command or Action	Purpose
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 CGN instance named cgn1 for server port 5000. The default 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.
		• 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:
	`RP/0/RP0/CPU0:router(config-cgn)#`
	`service-location preferred-active 0/1/CPU0`
	`preferred-standby 0/4/CPU0`
		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 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.

	Command or Action	Purpose
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 CGN 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.
		• 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 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 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.

	Command or Action	Purpose

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

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	**`dynamic port range start`** *`value`*
	
	Example:
	`RP/0/RP0/CPU0:router(config-cgn-nat44)# dynamic port range start 1024`
		Configures the value of **dynamic port range start** for a CGN NAT 44 instance. The value can range from 1 to 65535.

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.

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 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	**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	**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 CGN 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	**`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]:`
		– 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:
	`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)# 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 netflow-v9 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 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 CGN 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.

Configuring the Carrier Grade Service Engine

Prerequisites:

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

Hardware:

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

Software:

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

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.

```
router(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 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

Step 1 Load the fpd pie.

Step 2 Uboot the line card.

```
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.

```
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.

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

Step 5 Also upgrade these locations for Uboot:

```
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.

```
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.

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

Configuring IPv6 Rapid Development

These steps describe the configuration of IPv6 Rapid Development application.

Step 1 These are the 6rd CPE/RG configuration parameters.

SP Prefix	2001:B000::/28
V4 Common Prefix length	0
V4 Common Suffix length	0
RG/CPE Delegated 6RD prefix	2001:B000:a010:1010::/60

CE1 (V4) tunnel transport source	10.1.1.1
BR (V4) tunnel transport address	100:1:1:1
*Static Routes	::/0 -> 6rd-virtual-int0 via 2001:B006:4010:1010::/ (default route) 2001:B000::/28 -> 6rd-virtual-int0 (direct connect to 6rd) 2001:B000:a010:1010::/60-> Null0 (delegated prefix null route) 2001:B000:a010:1010::/64 -> Ethernet0 (LAN interface)

Step 2 These are the 6rd BR (CGSE) configuration parameters.

SP Prefix	2001:B000::/28
V4 Common Prefix length	0
V4 Common Suffix length	0
BR Delegated 6RD prefix	2001:B006:4010:1010::/60
BR (V4) source address	100:1:1:1
*Static Routes	100:1:1:1/32-> Serviceapp4 2001:B000::/28 -> Serviceapp6 2001:B006:4010:1010::/60 -> Null0 (BR delegated prefix null route) 2001:B006:4010:1010::/128 -> Serviceapp6 (BR anycast reachability route) 2001:B006:4010:1010::1/128 -> Serviceapp6 (BR unicast reachability route)

- 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.

Step 3 You can configure routes to the CGSE using these steps.

- To divert the traffic towards CGSE which is destined for BR

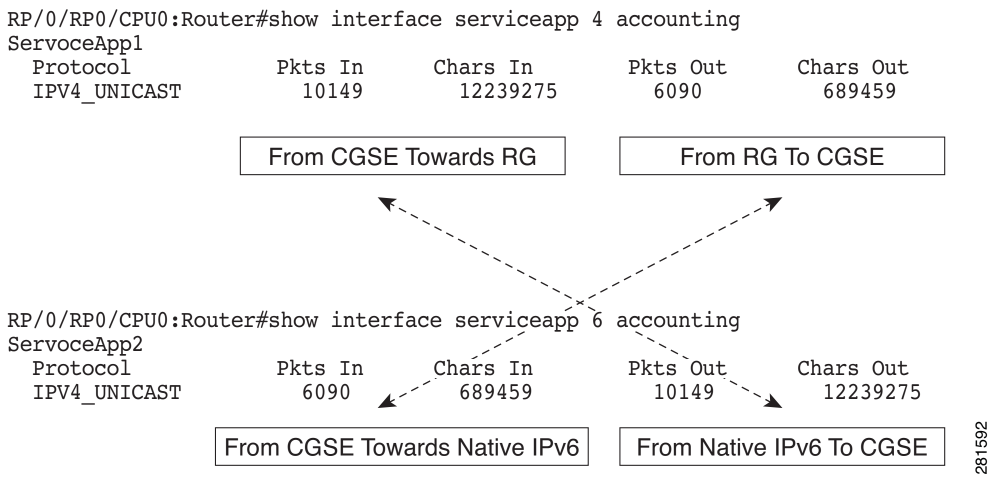
```
router(config)#

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
```

Step 4 This step illustrates the **show interface serviceapp 4 accounting** command.



```
RP/0/RP0/CPU0:Router#show interface serviceapp 4 accounting
ServoceApp1
  Protocol           Pkts In      Chars In        Pkts Out      Chars Out
  IPV4_UNICAST         10149       12239275         6090          689459

RP/0/RP0/CPU0:Router#show interface serviceapp 6 accounting
ServoceApp2
  Protocol           Pkts In      Chars In        Pkts Out      Chars Out
  IPV4_UNICAST         6090         689459         10149        12239275
```

a. 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

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

Step 5 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 Note : All the BR parameters have to be given in Single Commit.

- 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 54.

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-name*

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 90.1.1.1 port 99**
19. **external-logging syslog**
20. **server**
21. **address 90.1.1.1 port 514**
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.

	Command or Action	Purpose
Step 4	**service-type ds-lite** *instance* **Example:** `RP/0/RP0/CPU0:router(config-cgn)# service-type ds-lite` *`ds11`*	Configures the service type keyword definition for the DS LITE application.
Step 5	**portlimit** *value* **Example:** `RP/0/RP0/CPU0:router(config-cgn-ds-lite)# portlimit 200`	Specifies the maximum ports for a given IPV4 private address. It provides the limits for the number of entries per address for each subscriber of the system.
Step 6	**bulk-port-alloc size** *value* **Example:** `RP/0/RP0/CPU0:router(config-cgn-ds-lite)# bulk-port-alloc size 128`	Enables bulk port allocation and sets bulk size that is used to reduce logging data volume.
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.

	Command or Action	Purpose
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.
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.

	Command or Action	Purpose

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.
		– 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:

- Configuring a Different Inside VRF Map to a Different Outside VRF: Example, page 65
- Configuring a Different Inside VRF Map to a Same Outside VRF: Example, page 66
- Configuring ACL for a Infrastructure Service Virtual Interface: Example, page 66
- NAT44 Configuration: Example, page 67
- NAT64 Stateless Configuration: Example, page 70
- DS Lite Configuration: Example, page 72
- Bulk port allocation and Syslog Configuration: Example, page 73

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
```

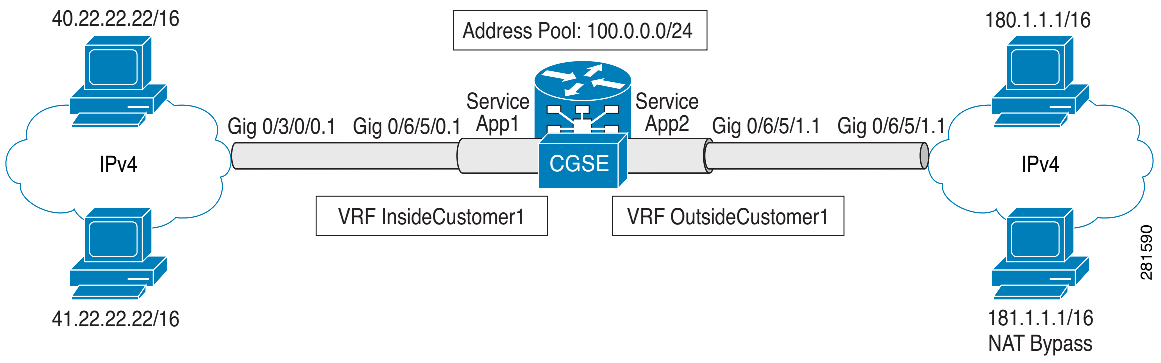
Use the **show controllers services boot-params** command to verify the IP addresses of SVI and the CGSE PLIM.

```
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

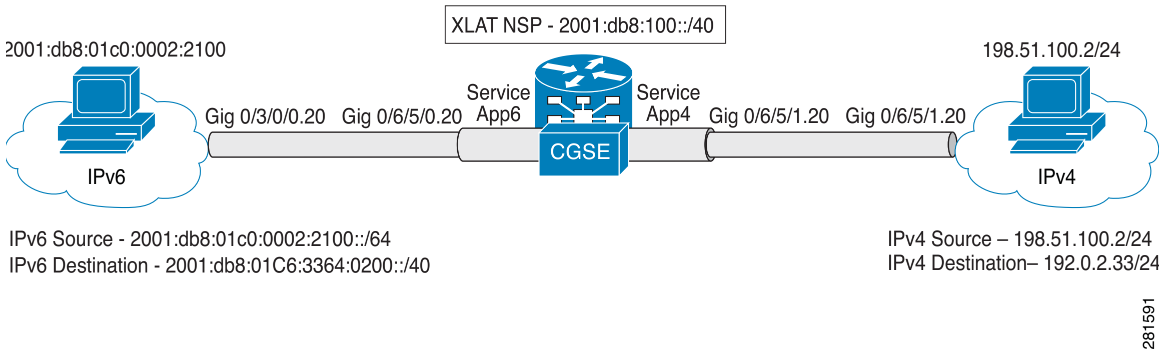
This example shows a NAT44 sample configuration:



```
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
 dot1q vlan 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
 dot1q vlan 1
!
interface GigabitEthernet0/6/5/1.1
 vrf OutsideCustomer1
 ipv4 address 10.12.13.2 255.255.255.0
 dot1q vlan 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
 dot1q vlan 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 Stateless Configuration: Example

This example shows a NAT64 Stateless sample configuration.


XLAT NSP - 2001:db8:100::/40
2001:db8:01c0:0002:2100
198.51.100.2/24
Service App6
Service App4
Gig 0/3/0/0.20
Gig 0/6/5/0.20
Gig 0/6/5/1.20
Gig 0/6/5/1.20
CGSE
IPv6
IPv4
IPv6 Source - 2001:db8:01c0:0002:2100::/64
IPv6 Destination - 2001:db8:01C6:3364:0200::/40
IPv4 Source – 198.51.100.2/24
IPv4 Destination– 192.0.2.33/24
281591

```
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 !
!
```

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
  !
 !
```

Additional References

For additional information related to Implementing the Carrier Grade NAT, see the following references:

Related Documents

Related Topic	Document Title
Cisco IOS XR Carrier Grade NAT commands	*Cisco IOS XR Carrier Grade NAT Command Reference for the Cisco CRS Router*
Cisco CRS Router getting started material	*Cisco IOS XR Getting Started Guide for the Cisco CRS Router*
Information about user groups and task IDs	*Configuring AAA Services on Cisco IOS XR Software* module of the *Cisco IOS XR System Security Configuration Guide*

Standards

Standards[1]	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	—

1. Not all supported standards are listed.

MIBs

MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs[1]	Title
RFC 4787	*Network Address Translation (NAT) Behavioral Requirements for Unicast UDP*
RFC 5382	*NAT Behavioral Requirements for TCP*
RFC 5508	*NAT Behavioral Requirements for ICMP*

1. Not all supported RFCs are listed.

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport