Configuring Stateful Network Address Translation 64

The Stateful NAT64 feature provides a translation mechanism that translates IPv6 packets into IPv4 packets and vice versa. This mechanism focuses on enabling connectivity across IPv6 and IPv4 domains. There are other methods like dual stacking and tunnelling but stateful NAT64 has certain advantages over them as mentioned in RFC 6144.

Restrictions for Configuring Stateful Network Address Translation 64

- Applications without a corresponding application-level gateway (ALG) may not work properly with the Stateful NAT64 translator.
- IP Multicast is not supported.
- The translation of IPv4 options, IPv6 routing headers, hop-by-hop extension headers, destination option headers, and source routing headers is not supported.
- When traffic flows from IPv6 to IPv4, the destination IP address that you have configured must match a stateful prefix to prevent hairpinning loops. However, the source IP address (source address of the IPv6 host) must not match the stateful prefix. If the source IP address matches the stateful prefix, packets are dropped.

  Hairpinning allows two endpoints inside Network Address Translation (NAT) to communicate with each other, even when the endpoints use only each other's external IP addresses and ports for communication.
- Only TCP and UDP Layer 4 protocols are supported for header translation.
- Routemaps are not supported.

- If a static mapping host-binding entry exists for an IPv6 host, the IPv4 nodes can initiate communication. In dynamic mapping, IPv4 nodes can initiate communication only if a host-binding entry is created for the IPv6 host through a previously established connection to the same or a different IPv4 host.

  Dynamic mapping rules that use Port-Address Translation (PAT), host-binding entries cannot be created because IPv4-initiated communication not possible through PAT.
- Configuring NAT44 and NAT64 on the same interface is not recommended. Applying such a configuration could potentially impact the functionality of both NAT44 and NAT64. If such a configuration is applied, then you must remove both the configurations and re-apply the desired configuration.
- Address Only Translation is not supported.
- Post NAT fragmentation is not supported. If a packet exceeds the maximum transmission unit (MTU) after the translation, the packet will be dropped.
- NAT64 is not supported with object group-based ACLs.
- Overlapping address translation within the same VRF is not supported. You can configure only a single NAT64 rule for an IPv6 host within a given VRF. This implies that a given IPv6 host can only access IPv4 services hosted on either of the default or nondefault VRFs, but not both. However, overlapping address translation across different VRFs is supported, and NAT64 supports the translation of IPv6 hosts with overlapping addresses which are associated with different VRFs.

Note For Domain Name System (DNS) traffic to work, you must have a separate working installation of DNS64.

Information About Stateful Network Address Translation 64

Stateful Network Address Translation 64

The Stateful NAT64 feature provides a translation mechanism that translates IPv6 packets into IPv4 packets and vice versa.

Stateful NAT64 supports Internet Control Message Protocol (ICMP), TCP, and UDP traffic. Packets that are generated in an IPv6 network and are destined for an IPv4 network are routed within the IPv6 network towards the Stateful NAT64 translator. Stateful NAT64 translates the packets and forwards them as IPv4 packets through the IPv4 network. The process is reversed for traffic that is generated by hosts connected to the IPv4 network and destined for an IPv6 receiver.

The Stateful NAT64 translation is not symmetric, because the IPv6 address space is larger than the IPv4 address space and a one-to-one address mapping is not possible. Before it can perform an IPv6 to an IPv4 translation, Stateful NAT64 requires a state that binds the IPv6 address and the TCP/UDP port to the IPv4 address. The binding state is either statically configured or dynamically created when the first packet that flows from the IPv6 network to the IPv4 network is translated. After the binding state is created, packets flowing in both directions are translated. In dynamic binding, Stateful NAT64 supports communication initiated by the IPv6-only node toward an IPv4-only node. Static binding supports communication initiated by an IPv4-only node to an IPv6-only node and vice versa. Stateful NAT64 with NAT overload or Port Address Translation (PAT) provides a 1:*n* mapping between IPv4 and IPv6 addresses.

When an IPv6 node initiates traffic through Stateful NAT64, and the incoming packet does not have an existing state and the following events happen:

- The source IPv6 address (and the source port) is associated with an IPv4 configured pool address (and port, based on the configuration).
- The destination IPv6 address is translated mechanically based on RFC 7915 using either the configured NAT64 stateful prefix or the Well Known Prefix (WKP).
- The packet is translated from IPv6 to IPv4 and forwarded to the IPv4 network.

When an incoming packet is stateful (if a state exists for an incoming packet), NAT64 identifies the state and uses the state to translate the packet.

Prefix Format for Stateful Network Address Translation 64

A set of bits at the start of an IPv6 address is called the format prefix. Prefix length is a decimal value that specifies how many of the leftmost contiguous bits of an address comprise the prefix.

When packets flow from the IPv6 to the IPv4 direction, the IPv4 host address is derived from the destination IP address of the IPv6 packet that uses the prefix length. When packets flow from the IPv4 to the IPv6 direction, the IPv4 host address is constructed using the stateful prefix.

According to the IETF address format RFC 7915 a u-bit (bit 70) defined in the IPv6 architecture should be set to zero. For more information on the u-bit usage, see RFC 2464. The reserved octet, also called u-octet, is reserved for compatibility with the host identifier format defined in the IPv6 addressing architecture. RFC 6052 section 2.2 describes the address encoding/u-bit semantics. When constructing an IPv6 packet, the translator has to make sure that the u-bits are not tampered with and are set to the value suggested by RFC 2373. The suffix will be set to all zeros by the translator. IETF recommends that the 8 bits of the u-octet (bit range 64–71) be set to zero.

Well-Known Prefix

Well-Known Prefix 64:FF9B::/96 is supported for Stateful NAT64. During a stateful translation, if no stateful prefix is configured (either on the interface or globally), the WKP prefix is used to translate the IPv4 host addresses.

Performance and Scale Numbers for Stateful Nat64

Stateful NAT64 module is capable of performing translation and forwarding in the hardware at half line-rate, by programming the relevant hardware tables with the forwarding and rewrite information.

The maximum number of bidirectional translations supported on the hardware level is 3000. Packet flows that cannot be translated in the hardware will be translated-forwarded at the software level at a reduced throughput

Stateful IPv4-to-IPv6 Packet Flow

The packet flow of IPv4-initiated packets for stateful NAT64 is as follows:

- The destination address is routed to a NAT Virtual Interface (NVI).

A virtual interface is created when stateful NAT64 is configured. For stateful NAT64 translation to work, all packets must get routed to the NVI. When you configure an address pool, a route is automatically added to all IPv4 addresses in the pool. This route automatically points to the NVI.

- The IPv4-initiated packet hits static or dynamic binding.

  Static address bindings are created by the stateful NAT64 translator when you configure a static rule. Dynamic address bindings are created by stateful NAT64 translator when the IPv6-to-IPv4 traffic matches a configured dynamic non-overload rule. This creates a binding between the given IPv6 address and the corresponding IPv4 address from the associated IPv4 address pool.

- A session is created based on the translation information.

All subsequent IPv4-initiated packets are translated based on the previously created session.

Stateful NAT64 supports endpoint-dependent filtering for the IPv4-to-IPv6 packet flow with PAT configuration. In a stateful NAT64 PAT configuration, the packet flow must have originated from the IPv6 realm and created the state information in NAT64 state tables. Packets from the IPv4 side that do not have a previously created state are dropped. Endpoint-independent filtering is supported with static Network Address Translation (NAT) and non-PAT configurations.

Stateful IPv6-to-IPv4 Packet Flow

The stateful IPv6-initiated packet flow is as follows:

- The destination address is routed to a NVI.

  A virtual interface is created when stateful NAT64 is configured. A route is added to destinations that match well-known prefix (WKP) or network specific prefix (NSP) pointing to the NVI.

- A packet is considered eligible for translation when it matches a static or dynamic rule. Dynamic rule match is successful when a packet is permitted by the associated access control list. An IPv4 address and port is associated to the IPv6 destination address based on the matched rule. The IPv6 packet is translated and the IPv4 packet is formed by using the following methods:
  - Extracting the destination IPv4 address by stripping the prefix from the IPv6 address. The source address is replaced by the allocated IPv4 address (and port).
  - The rest of the fields are translated from IPv6-to-IPv4 to form a valid IPv4 packet.

- A new NAT64 translation is created in the session database and in the bind database. The pool and port databases are updated depending on the configuration. The return traffic and the subsequent traffic of the IPv6 packet flow will use this session database entry for translation.

VRF-Aware Network Address Translation 64

NAT64 is typically configured to operate across the default or the global routing domain. Translation between the IPv6 and IPv4 packets occur in the default Virtual routing and forwarding (VRF) domain. However, certain scenarios such as the BGP EVPN VXLAN fabric with Dynamic NAT64 require NAT64 to operate within a specific nondefault VRF. A common scenario would be enabling shared service access for private networks that have overlapping address space. In such a scenario, the private networks can be placed in different VRFs and access to the global service can be achieved by configuring VRF-aware NAT64 rules. These rules aid in mapping the overlapping private address to a unique global address. VRF-awareness enables NAT64 to carry out address and port translation, and forwarding by considering the VRF in which these IPv4 and IPv6 networks exist.

Supported Deployments for VRF-Aware NAT64

In this release, VRF-aware NAT64 supports the following NAT64 translations:

- **VRF to Global NAT64 translation**:

  VRF to Global translation is between an IPv6 network that is associated with a specific user-defined VRF and an IPv4 network that is associated with the default VRF.

- **Intra-VRF NAT64 translation**:

  Intra-VRF NAT64 translation is between an IPv6 network and an IPv4 network, both within the same nondefault VRF instance. In the intra-VRF NAT64 translation, local and global addresses for the hosts are isolated to their respective VRFs, and the translated addresses may overlap each other.

  To separate the address space for the translated addresses among the VRFs, use the **match-in-vrf** keyword in the NAT64 mapping command

NAT64 Commands to Configure VRF-Awareness

The following NAT64 commands make the translations VRF-aware.

Command	Usage
nat64 prefix stateful *ipv6_prefix length* [**vrf** *vrf_name*]	Use this command in the global configuration mode to provision a VRF-aware stateful prefix configuration.
nat64 v6v4 static *ipv6_addr ipv4_addr*[**vrf** *vrf_name* [**match-in-vrf**]]	Use this command to provision a VRF-aware static rule. To enable intra-VRF translation, use the **match-in-vrf** keyword.
nat64 v6v4 list *acl_name* **pool** *pool_name* [**vrf** *vrf_name* [**match-in-vrf]**] [**overload**]	Use this command to provision a VRF-aware dynamic rule. To enable intra-VRF translation, use the **match-in-vrf** keyword. **Note** Same ACL or pool cannot be shared across different dynamic rules. However, overlapping IPv6 address translation can be achieved through different ACLs.

Related Show Commands

Use the following **show** commands to display the VRF association details:

- **show nat64 statistics**
- **show nat64 prefix stateful global**
- **show nat64 prefix stateful static-routes**

Verify the VRF-aware NAT64 translations using the following **show** commands:

- **show nat64 translations** [**vrf** *vrf_name*]
- **show nat64 statistics prefix stateful** *ipv6_prefix* [**vrf** *vrf_name*]
- **show nat64 prefix stateful static-routes prefix** *ipv6_prefix* [**vrf** *vrf_name*]

How to Configure Stateful Network Address Translation 64

Based on your network configuration, you can configure static, dynamic, or dynamic Port Address Translation (PAT) Stateful NAT64.

Note You need to configure at least one of the configurations described in the following tasks for Stateful NAT64 to work.

Configuring Static Stateful Network Address Translation 64

You can configure a static IPv6 address to an IPv4 address and vice versa. Optionally, you can configure static Stateful NAT64 with or without ports. Perform this task to configure static Stateful NAT64.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ipv6 unicast-routing** **Example:** `Device(config)# ipv6 unicast-routing`	Enables the forwarding of IPv6 unicast datagrams.
Step 4	**interface** *type number* **Example:** `Device(config)# interface gigabitethernet 0/0/0`	Configures an interface type and enters interface configuration mode.
Step 5	**description** *string* **Example:** `Device(config-if)# description interface facing ipv6`	Adds a description to an interface configuration.

	Command or Action	Purpose
Step 6	**vrf forwarding** *vrf-name* **Example:** `Device(config-if)# vrf forwarding blue`	Associates a VRF with the IPv6 interface.
Step 7	**ipv6 enable** **Example:** `Device(config-if)# ipv6 enable`	Enables IPv6 processing on an interface.

Step 8	**ipv6 address** {*ipv6-address/prefix-length* | *prefix-name sub-bits/prefix-length*}
	Example:
	`Device(config-if)# ipv6 address 2001:DB8:1::1/96`
		Configures an IPv6 address based on an IPv6 general prefix and enables IPv6 processing on an interface.

Step 9	**nat64 enable** **Example:** `Device(config-if)# nat64 enable`	Enables NAT64 translation on an IPv6 interface.
Step 10	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and enters global configuration mode.
Step 11	**interface** *type number* **Example:** `Device(config)# interface gigabitethernet 1/2/0`	Configures an interface and enters interface configuration mode.
Step 12	**description** *string* **Example:** `Device(config-if)# description interface facing ipv4`	Adds a description to an interface configuration.
Step 13	**ip address** *ip-address mask* **Example:** `Device(config-if)# ip address 209.165.201.1 255.255.255.0`	Configures an IPv4 address for an interface.
Step 14	**nat64 enable** **Example:** `Device(config-if)# nat64 enable`	Enables NAT64 translation on an IPv4 interface.
Step 15	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and enters global configuration mode.

	Command or Action	Purpose
Step 16	**nat64 prefix stateful** *ipv6-prefix/length* [**vrf** *vrf-name*] **Example:** `Device(config)# nat64 prefix stateful 2001:DB8:1::1/96 vrf blue`	Defines the Stateful NAT64 prefix to be added to IPv4 hosts to translate the IPv4 address into an IPv6 address. The Stateful NAT64 prefix can be configured at the global configuration mode or at the interface mode. Use the **vrf** keyword in the global configuration mode to provision a VRF-aware stateful prefix configuration.
Step 17	**nat64 v6v4 static** *ipv6-address ipv4-address* [**vrf** *vrf-name*] **Example:** `Device(config)# nat64 v6v4 static 2001:DB8:1::FFFE 209.165.201.1 vrf blue`	Enables NAT64 IPv6-to-IPv4 static address mapping.

Step 18	**end**
	Example:
	`Device(config)# end`
		Exits global configuration mode and enters privileged EXEC mode.

Configuring Dynamic Stateful Network Address Translation 64

A dynamic Stateful NAT64 configuration provides a one-to-one mapping of IPv6 addresses to IPv4 addresses in the address pool. You can use the dynamic Stateful NAT64 configuration when the number of active IPv6 hosts is less than the number of IPv4 addresses in the pool. Perform this task to configure dynamic Stateful NAT64.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ipv6 unicast-routing** **Example:** `Device(config)# ipv6 unicast-routing`	Enables the forwarding of IPv6 unicast datagrams.
Step 4	**interface** *type number* **Example:**	Configures an interface type and enters interface configuration mode.

	Command or Action	Purpose
	`Device(config)# interface gigabitethernet 0/0/0`	
Step 5	**description** *string* **Example:** `Device(config-if)# description interface facing ipv6`	Adds a description to an interface configuration.
Step 6	**vrf forwarding** *vrf-name* **Example:** `Device(config-if)# vrf forwarding blue`	Associates a VRF with the IPv6 interface.
Step 7	**ipv6 enable** **Example:** `Device(config-if)# ipv6 enable`	Enables IPv6 processing on an interface.

Step 8	**ipv6** {*ipv6-address/prefix-length* | *prefix-name sub-bits/prefix-length*}
	Example:
	`Device(config-if)# ipv6 2001:DB8:1::1/96`
		Configures an IPv6 address based on an IPv6 general prefix and enables IPv6 processing on an interface.

Step 9	**nat64 enable** **Example:** `Device(config-if)# nat64 enable`	Enables Stateful NAT64 translation on an IPv6 interface.
Step 10	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and enters global configuration mode.
Step 11	**interface** *type number* **Example:** `Device(config)# interface gigabitethernet 1/2/0`	Configures an interface type and enters interface configuration mode
Step 12	**description** *string* **Example:** `Device(config-if)# description interface facing ipv4`	Adds a description to an interface configuration.
Step 13	**ip address** *ip-address mask* **Example:** `Device(config-if)# ip address 209.165.201.24 255.255.255.0`	Configures an IPv4 address for an interface.
Step 14	**nat64 enable** **Example:**	Enables Stateful NAT64 translation on an IPv4 interface.

	Command or Action	Purpose
	`Device(config-if)# nat64 enable`	
Step 15	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and enters global configuration mode.
Step 16	**ipv6 access-list** *access-list-name* **Example:** `Device(config)# ipv6 access-list nat64-acl`	Defines an IPv6 access list and enters IPv6 access list configuration mode.
Step 17	**permit ipv6** *ipv6-address* **any** **Example:** `Device(config-ipv6-acl)# permit ipv6 2001:DB8:2::/96 any`	Sets permit conditions for an IPv6 access list.
Step 18	**exit** **Example:** `Device(config-ipv6-acl# exit`	Exits IPv6 access list configuration mode and enters global configuration mode.
Step 19	**nat64 prefix stateful** *ipv6-prefix/length* [**vrf** *vrf-name*] **Example:** `Device(config)# nat64 prefix stateful 2001:DB8:1::1/96 vrf blue`	Enables NAT64 IPv6-to-IPv4 address mapping. Use the **vrf** keyword to provision a VRF-aware stateful prefix configuration.
Step 20	**nat64 v4 pool** *pool-name start-ip-address end-ip-address* **Example:** `Device(config)# nat64 v4 pool pool1 209.165.201.1 209.165.201.254`	Defines the Stateful NAT64 IPv4 address pool.
Step 21	**nat64 v6v4 list** *access-list-name* **pool** *pool-name* [**vrf** *vrf-name* [**match-in-vrf**]] **Example:** `Device(config)# nat64 v6v4 list nat64-acl pool pool1 vrf blue`	Dynamically translates an IPv6 source address to an IPv4 source address and an IPv6 destination address to an IPv4 destination address for NAT64. Use the **vrf** keyword to provision a VRF-aware translation. Use the **match-in-vrf** keyword to enable intra-VRF translation.

Step 22	**end**
	Example:
	`Device(config)# end`
		Exits global configuration mode and enters privileged EXEC mode.

Configuring Dynamic Port Address Translation Stateful NAT64

A Port Address Translation (PAT) or overload configuration is used to multiplex (mapping IPv6 addresses to a single IPv4 pool address) multiple IPv6 hosts to a pool of available IPv4 addresses on a first-come first-served basis. The dynamic PAT configuration conserves the IPv4 address space while providing connectivity to the IPv4 Internet. Configure the **nat64 v6v4 list** command with the **overload** keyword to configure PAT address translation. Perform this task to configure dynamic PAT Stateful NAT64.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ipv6 unicast-routing** **Example:** `Device(config)# ipv6 unicast-routing`	Enables the forwarding of IPv6 unicast datagrams.
Step 4	**interface** *type number* **Example:** `Device(config)# interface gigabitethernet 0/0/0`	Configures an interface type and enters interface configuration mode.
Step 5	**description** *string* **Example:** `Device(config-if)# description interface facing ipv6`	Adds a description to an interface configuration.
Step 6	**vrf forwarding** *vrf-name* **Example:** `Device(config-if)# vrf forwarding blue`	Associates a VRF with the IPv6 interface.
Step 7	**ipv6 enable** **Example:** `Device(config-if)# ipv6 enable`	Enables IPv6 processing on an interface.

Step 8	**ipv6** {*ipv6-address/prefix-length* | *prefix-name sub-bits/prefix-length*}
	Example:
	`Device(config-if)# ipv6 2001:DB8:1::1/96`
		Configures an IPv6 address based on an IPv6 general prefix and enables IPv6 processing on an interface.

	Command or Action	Purpose
Step 9	**nat64 enable** **Example:** `Device(config-if)# nat64 enable`	Enables Stateful NAT64 translation on an IPv6 interface.
Step 10	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and enters global configuration mode.
Step 11	**interface** *type number* **Example:** `Device(config)# interface gigabitethernet 1/2/0`	Configures an interface type and enters interface configuration mode
Step 12	**description** *string* **Example:** `Device(config-if)# description interface facing ipv4`	Adds a description to an interface configuration.
Step 13	**ip address** *ip-address mask* **Example:** `Device(config-if)# ip address 209.165.201.24 255.255.255.0`	Configures an IPv4 address for an interface.
Step 14	**nat64 enable** **Example:** `Device(config-if)# nat64 enable`	Enables Stateful NAT64 translation on an IPv6 interface.
Step 15	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and enters global configuration mode.
Step 16	**ipv6 access-list** *access-list-name* **Example:** `Device(config)# ipv6 access-list nat64-acl`	Defines an IPv6 access list and places the device in IPv6 access list configuration mode.
Step 17	**permit ipv6** *ipv6-address* **any** **Example:** `Device(config-ipv6-acl)# permit ipv6 2001:db8:2::/96 any`	Sets permit conditions for an IPv6 access list.
Step 18	**exit** **Example:** `Device(config-ipv6-acl)# exit`	Exits IPv6 access list configuration mode and enters global configuration mode.

	Command or Action	Purpose
Step 19	**nat64 prefix stateful** *ipv6-prefix/length* [**vrf** *vrf-name*] **Example:** `Device(config)# nat64 prefix stateful 2001:db8:1::1/96 vrf blue`	Enables NAT64 IPv6-to-IPv4 address mapping.
Step 20	**nat64 v4 pool** *pool-name start-ip-address end-ip-address* **Example:** `Device(config)# nat64 v4 pool pool1 209.165.201.1 209.165.201.254`	Defines the Stateful NAT64 IPv4 address pool.
Step 21	**nat64 v6v4 list** *access-list-name* **pool** *pool-name* [**vrf** *vrf-name* [**match-in-vrf**]] **overload** **Example:** `Device(config)# nat64 v6v4 list nat64-acl pool pool1 vrf blue overload`	Enables NAT64 PAT or overload address translation. Use the **vrf** keyword to provision a VRF-aware configuration. Use the **match-in-vrf** option to configure intra-VRF NAT64 PAT.

Step 22	**end**
	Example:
	`Device(config)# end`
		Exits global configuration mode and enters privileged EXEC mode.

Configuring Timeout Functionality

Before you begin

Perform this task to configure idle timeout for dynamically created NAT64 session and bind entries.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

Step 2	**configure terminal**
	Example:
	`Device# configure terminal`
	`Device(config)#`
		Enters global configuration mode.

Step 3	**nat64 translation timeout** { **bind** *time* | **icmp** *time*|**tcp** *time*|**tcp-transient** *time*|**udp** *time* } **Example:**	Configures timeout functionality for binds and session entries.

	Command or Action	Purpose
	`Device(config)#nat64 translation timeout bind 2`	• **bind** *time*: Specifies the timeout for NAT64 binds. The default timeout value is 1 hour. You can change the timeout value using the *time* option. • **icmp** *time*: Specifies the timeout for NAT64 ICMP traffic flow. The default timout value is 60 seconds. You can change the timeout value using the *time* option. • **tcp** *time*: Specifies the timeout for NAT64 TCP traffic flow. The default timeout value is 2 hours. You can change the timeout value using the *time* option. • **tcp-transient** *time*: Specifies the timeout for NAT64 transient TCP traffic flow. The default timeout value is 4 minutes. You can change the timeout value using the *time* option. • **udp** *time*: Specifies the timeout for NAT64 UDP traffic flow. The default timeout value is 5 minutes. You can change the timeout value using the *time* option.

Step 4	**end**
	Example:
	`Device(config)# end`
		Returns to privileged EXEC mode.

Configuring Switch Database Management Template

Use (SDM) templates to configure system resources to optimize support for stateful NAT64.

After you set the template and the system reboots, you can use the **show sdm prefer** privileged EXEC command to verify the new template configuration. If you enter the **show sdm prefer** command before you enter the **reload** privileged EXEC command, the **show sdm prefer** command output displays the template currently in use and the template that will become active after a reload.

Follow these steps to set the SDM template to maximize stateful NAT64 usage:

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:**	Enables privileged EXEC mode. Enter your password, if prompted.

	Command or Action	Purpose
	`Device> enable`	
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**sdm prefer custom acl** **Example:** `Device(config)#sdm prefer custom acl`	Creates a customizable SDM template for ACL features. Enters a submode for customizing features.
Step 4	**pbr** *number-of-entries* **priority** *priority-value* **Example:** `Device(config-sdm-acl)#pbr 27 priority 1`	Specifies the number of entries allotted for PBR or NAT. The value ranges from 2 K to 27 K. The value is rounded up to the next 2 K unit. The priority values range from 1 to 8.
Step 5	**sdm prefer custom commit** **Example:** `Device(config)# sdm prefer custom commit`	Changes the running SDM preferences to the values in the customized template. The new template takes effect in the next reload.

Step 6	**end**
	Example:
	`Device(config)# end`
		Returns to privileged EXEC mode.

Step 7	**reload** **Example:** `Device# reload`	Reloads the device and applies the customized SDM template.

Configuration Examples for Stateful Network Address Translation 64

Example: Configuring Static Stateful Network Address Translation 64

```
Device# configure terminal
Device(config)# ipv6 unicast-routing
Device(config)# interface gigabitethernet 0/0/0
Device(config-if)# description interface facing ipv6
Device(config-if)# ipv6 enable
Device(config-if)# ipv6 address 2001:DB8:1::1/96
Device(confif-if)# nat64 enable
Device(config-fi)# exit
Device(config)# interface gigabitethernet 1/2/0
Device(config-if)# description interface facing ipv4
Device(config-if)# ip address 209.165.201.1 255.255.255.0
Device(config-if)# nat64 enable
Device(config-if)# exit
Device(config)# nat64 prefix stateful 2001:DB8:1::1/96
Device(config)# nat64 v6v4 static 2001:DB8:1::FFFE 209.165.201.1
Device(config)# end
```

Example: Configuring Dynamic Stateful Network Address Translation 64

The following example shows how to configure dynamic Stateful NAT64 with intra-VRF translation enabled for the VRF blue.

```
Device# configure terminal
Device(config)# ipv6 unicast-routing
Device(config)# interface gigabitethernet 0/0/0
Device(config-if)# description interface facing ipv6
Device(config-if)# vrf forwarding blue
Device(config-if)# ipv6 enable
Device(config-if)# ipv6 2001:DB8:1::1/96
Device(config-if)# nat64 enable
Device(config-if)# exit
Device(config)# interface gigabitethernet 1/2/0
Device(config-if)# description interface facing ipv4
Device(config-if)# vrf forwarding blue
Device(config-if)# ip address 209.165.201.24 255.255.255.0
Device(config-if)# nat64 enable
Device(config-if)# exit
Device(config)# ipv6 access-list nat64-acl
Device(config-ipv6-acl)# permit ipv6 2001:db8:2::/96 any
Device(config-ipv6-acl)# exit
Device(config)# nat64 prefix stateful 2001:db8:1::1/96 vrf blue
Device(config)# nat64 v4 pool pool1 209.165.201.1 209.165.201.254
Device(config)# nat64 v6v4 list nat64-acl pool pool1 vrf blue match-in-vrf
Device(config)# end
```

Example: Configuring Dynamic Port Address Translation Stateful NAT64

```
enable
 configure terminal
  ipv6 unicast-routing
  interface gigabitethernet 0/0/0
   description interface facing ipv6
   ipv6 enable
   ipv6 2001:DB8:1::1/96
   nat64 enable
   exit
  interface gigabitethernet 1/2/0
   description interface facing ipv4
   ip address 209.165.201.24 255.255.255.0
   nat64 enable
   exit
```

```
ipv6 access-list nat64-acl
 permit ipv6 2001:db8:2::/96 any
 exit
nat64 prefix stateful 2001:db8:1::1/96
nat64 v4 pool pool1 209.165.201.1 209.165.201.254
nat64 v6v4 list nat64-acl pool pool1 overload
end
```

Feature History for Configuring Stateful Network Address Translation 64

This table provides release and related information for the features explained in this module.

These features are available in all the releases subsequent to the one they were introduced in, unless noted otherwise.

Release	**Feature**	**Feature Information**
Cisco IOS XE Dublin 17.10.1	Configuring Stateful Network Address Translation 64	The Stateful NAT64 feature provides a translation mechanism that translates IPv6 packets into IPv4 packets and vice versa. Packets that are generated in an IPv6 network and are destined for an IPv4 network are routed within the IPv6 network towards the Stateful NAT64 translator. Stateful NAT64 translates the packets and forwards them as IPv4 packets through the IPv4 network. The process is reversed for traffic that is generated by hosts connected to the IPv4 network and destined for an IPv6 receiver.
Cisco IOS XE Dublin 17.12.1	VRF-aware Stateful Network Address Translation 64	NAT64 is made VRF-aware, and it supports the translation in the user-defined VRFs. The following translation scenarios are supported: • VRF to Global: IPv6 device is in a user-defined VRF and it communicates with an IPv4 device that is default VRF. • Intra-VRF: An IPv6 device communicates with an IPv4 device that is in the same user-defined VRF.

Use the Cisco Feature Navigator to find information about platform and software image support.