

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: May 16, 2018

M. Boucadair  
Orange  
S. Sivakumar  
Cisco Systems  
C. Jacquenet  
Orange  
S. Vinapamula  
Juniper Networks  
Q. Wu  
Huawei  
November 12, 2017

**A YANG Data Model for Network Address Translation (NAT) and Network  
Prefix Translation (NPT)  
draft-ietf-opsawg-nat-yang-08**

**Abstract**

For the sake of network automation and the need for programming Network Address Translation (NAT) function in particular, a data model for configuring and managing the NAT is essential. This document defines a YANG module for the NAT function.

NAT44, Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers (NAT64), Customer-side transLAtor (CLAT), Stateless IP/ICMP Translation (SIIT), Explicit Address Mappings for Stateless IP/ICMP Translation (SIIT EAM), and IPv6 Network Prefix Translation (NPTv6) are covered in this document.

**Editorial Note (To be removed by RFC Editor)**

Please update these statements with the RFC number to be assigned to this document:

"This version of this YANG module is part of RFC XXXX;"

"RFC XXXX: A YANG Data Model for Network Address Translation (NAT) and Network Prefix Translation (NPT)";

"reference: RFC XXXX"

**Status of This Memo**

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 16, 2018.

## Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

<a href="#">1.</a>	Introduction . . . . .	<a href="#">3</a>
<a href="#">1.1.</a>	Terminology . . . . .	<a href="#">3</a>
<a href="#">1.2.</a>	Tree Diagrams . . . . .	<a href="#">5</a>
<a href="#">2.</a>	Overview of the NAT YANG Data Model . . . . .	<a href="#">5</a>
<a href="#">2.1.</a>	Overview . . . . .	<a href="#">5</a>
<a href="#">2.2.</a>	Various Translation Flavors . . . . .	<a href="#">6</a>
<a href="#">2.3.</a>	TCP/UDP/ICMP NAT Behavioral Requirements . . . . .	<a href="#">8</a>
<a href="#">2.4.</a>	Other Transport Protocols . . . . .	<a href="#">8</a>
<a href="#">2.5.</a>	IP Addresses Used for Translation . . . . .	<a href="#">8</a>
<a href="#">2.6.</a>	Port Set Assignment . . . . .	<a href="#">8</a>
<a href="#">2.7.</a>	Port-Restricted IP Addresses . . . . .	<a href="#">9</a>
<a href="#">2.8.</a>	NAT Mapping Entries . . . . .	<a href="#">9</a>
<a href="#">2.9.</a>	Resource Limits . . . . .	<a href="#">12</a>
<a href="#">2.10.</a>	Binding the NAT Function to an External Interface . . . . .	<a href="#">15</a>
<a href="#">2.11.</a>	Relationship to NATV2-MIB . . . . .	<a href="#">15</a>
<a href="#">2.12.</a>	Tree Structure . . . . .	<a href="#">16</a>
<a href="#">3.</a>	NAT YANG Module . . . . .	<a href="#">20</a>
<a href="#">4.</a>	Security Considerations . . . . .	<a href="#">68</a>
<a href="#">5.</a>	IANA Considerations . . . . .	<a href="#">69</a>



<a href="#">6.</a>	Acknowledgements	<a href="#">70</a>
<a href="#">7.</a>	References	<a href="#">70</a>
<a href="#">7.1.</a>	Normative References	<a href="#">70</a>
<a href="#">7.2.</a>	Informative References	<a href="#">72</a>
<a href="#">Appendix A.</a>	Sample Examples	<a href="#">74</a>
<a href="#">A.1.</a>	Traditional NAT44	<a href="#">74</a>
<a href="#">A.2.</a>	Carrier Grade NAT (CGN)	<a href="#">75</a>
<a href="#">A.3.</a>	CGN Pass-Through	<a href="#">78</a>
<a href="#">A.4.</a>	NAT64	<a href="#">79</a>
<a href="#">A.5.</a>	Stateless IP/ICMP Translation (SIIT)	<a href="#">79</a>
<a href="#">A.6.</a>	Explicit Address Mappings for Stateless IP/ICMP Translation (EAM SIIT)	<a href="#">80</a>
<a href="#">A.7.</a>	Static Mappings with Port Ranges	<a href="#">84</a>
<a href="#">A.8.</a>	Static Mappings with IP Prefixes	<a href="#">84</a>
<a href="#">A.9.</a>	Destination NAT	<a href="#">85</a>
<a href="#">A.10.</a>	Customer-side Translator (CLAT)	<a href="#">88</a>
<a href="#">A.11.</a>	IPv6 Network Prefix Translation (NPTv6)	<a href="#">88</a>
	Authors' Addresses	<a href="#">91</a>

## [1.](#) Introduction

This document defines a data model for Network Address Translation (NAT) and Network Prefix Translation (NPT) capabilities using the YANG data modeling language [[RFC7950](#)].

Traditional NAT is defined in [[RFC2663](#)], while Carrier Grade NAT (CGN) is defined in [[RFC6888](#)]. Unlike traditional NAT, the CGN is used to optimize the usage of global IP address space at the scale of a domain: a CGN is not managed by end users, but by service providers instead. This document covers both traditional NATs and CGNs.

This document also covers NAT64 [[RFC6146](#)], customer-side translator (CLAT) [[RFC6877](#)], Stateless IP/ICMP Translation (SIIT) [[RFC7915](#)], Explicit Address Mappings for Stateless IP/ICMP Translation (EAM) [[RFC7757](#)], and IPv6 Network Prefix Translation (NPTv6) [[RFC6296](#)]. The full set of translation schemes that are in scope is included in [Section 2.2](#).

Sample examples are provided in [Appendix A](#). These examples are not intended to be exhaustive.

### [1.1.](#) Terminology

This document makes use of the following terms:

- o Basic NAT44: translation is limited to IP addresses alone ([Section 2.1 of \[RFC3022\]](#)).



- o Network Address/Port Translator (NAPT): translation in NAPT is extended to include IP addresses and transport identifiers (such as a TCP/UDP port or ICMP query ID); refer to [Section 2.2 of \[RFC3022\]](#).
- o Destination NAT: is a translation that acts on the destination IP address and/or destination port number. This flavor is usually deployed in load balancers or at devices in front of public servers.
- o Port-restricted IPv4 address: An IPv4 address with a restricted port set. Multiple hosts may share the same IPv4 address; however, their port sets must not overlap [\[RFC7596\]](#).
- o Restricted port set: A non-overlapping range of allowed external ports to use for NAT operation. Source ports of IPv4 packets translated by a NAT must belong to the assigned port set. The port set is used for all port-aware IP protocols [\[RFC7596\]](#).
- o Internal Host: A host that may solicit a NAT or an NPTv6 (or both) capability to send to and receive traffic from the Internet.
- o Internal Address/prefix: The IP address/prefix of an internal host.
- o External Address: The IP address/prefix assigned by a NAT/NPTv6 to an internal host; this is the address that will be seen by a remote host on the Internet.
- o Mapping: denotes a state at the NAT that is necessary for network address and/or port translation.
- o Dynamic implicit mapping: is created implicitly as a side effect of processing a packet (e.g., an initial TCP SYN packet) that requires a new mapping. A validity lifetime is associated with this mapping.
- o Dynamic explicit mapping: is created as a result of an explicit request, e.g., PCP message [\[RFC6887\]](#). A validity lifetime is associated with this mapping.
- o Static explicit mapping: is created using, e.g., a CLI interface. This mapping is likely to be maintained by the NAT function till an explicit action is executed to remove it.

The usage of the term NAT in this document refers to any translation flavor (NAT44, NAT64, etc.) indifferently.



This document uses the term "session" as defined in [[RFC2663](#)] and [[RFC6146](#)] for NAT64.

## 1.2. Tree Diagrams

The meaning of the symbols in these diagrams is as follows:

- o Brackets "[" and "]" enclose list keys.
- o Curly braces "{" and "}" contain names of optional features that make the corresponding node conditional.
- o Abbreviations before data node names: "rw" means configuration (read-write), "ro" state data (read-only).
- o Symbols after data node names: "?" means an optional node, "!" a container with presence, and "\*" denotes a "list" or "leaf-list".
- o Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- o Ellipsis ("...") stands for contents of subtrees that are not shown.

## 2. Overview of the NAT YANG Data Model

### 2.1. Overview

The NAT YANG module is designed to cover dynamic implicit mappings and static explicit mappings. The required functionality to instruct dynamic explicit mappings is defined in separate documents such as [[I-D.boucadair-pcp-yang](#)]. Considerations about instructing explicit dynamic means (e.g., [[RFC6887](#)], [[RFC6736](#)], or [[RFC8045](#)]) are out of scope. As a reminder, REQ-9 of [[RFC6888](#)] requires that a CGN must implement a protocol giving subscribers explicit control over NAT mappings; that protocol should be the Port Control Protocol [[RFC6887](#)].

A single NAT device can have multiple NAT instances; each of these instances can be provided with its own policies (e.g., be responsible for serving a group of hosts). This document does not make any assumption about how internal hosts or flows are associated with a given NAT instance.

The NAT YANG module assumes that each NAT instance can be enabled/disabled, be provisioned with a specific set of configuration data, and maintains its own mapping tables.





The NAT YANG module allows for a NAT instance to be provided with multiple NAT policies (/nat/instances/instance/policy). The document does not make any assumption about how flows are associated with a given NAT policy of a given NAT instance. Classification filters are out of scope.

Defining multiple NAT instances or configuring multiple NAT policies within one single NAT instance is implementation- and deployment-specific.

This YANG module allows to instruct a NAT function to enable the logging feature. Nevertheless, configuration parameters specific to logging protocols are out of the scope of this document.

## **2.2. Various Translation Flavors**

The following translation modes are supported:

- o Basic NAT44
- o NAPT
- o Destination NAT
- o Port-restricted NAT
- o Stateful NAT64
- o SIIT
- o CLAT
- o EAM
- o NPTv6
- o Combination of Basic NAT/NAPT and Destination NAT
- o Combination of port-restricted and Destination NAT
- o Combination of NAT64 and EAM
- o Stateful and Stateless NAT64

[I-D.ietf-softwire-dslite-yang] specifies an extension to the NAT YANG module to support DS-Lite.

The YANG "feature" statement is used to indicate which of the different translation modes is relevant for a specific data node. Table 1 lists defined features:



Translation Mode		YANG Feature
Basic NAT44		basic-nat44
NAPT		napt44
Destination NAT		dst-nat
Stateful NAT64		nat64
Stateless IPv4/IPv6 translation		siit
CLAT		clat
EAM		eam
NPTv6		nptv6

Table 1: YANG NAT Features

The following translation modes do not require defining dedicated features:

- o Port-restricted NAT: This mode corresponds to supplying port restriction policies to a NAPT or NAT64 (port-set-restrict).
- o Combination of Basic NAT/NAPT and Destination NAT: This mode corresponds to setting 'dst-nat-enable' for Basic NAT44 or NAPT.
- o Combination of port-restricted and Destination NAT: This mode can be achieved by configuring a NAPT with port restriction policies (port-set-restrict) together with a destination IP address pool (dst-ip-address-pool).
- o Combination of NAT64 and EAM: This mode corresponds to configuring static mappings for NAT64.
- o Stateful and stateless NAT64: A NAT64 implementation can be instructed to behave in the stateless mode for a given prefix by setting the parameter (nat64-prefixes/stateless-enable). A NAT64 implementation may behave in both stateful and stateless modes if, in addition to appropriately setting the parameter (nat64-prefixes/stateless-enable), an external IPv4 address pool is configured.

The NAT YANG module allows to retrieve the capabilities of a NAT instance (including, list of supported translation modes, list of supported protocols, port restriction support status, supported NAT mapping types, supported NAT filtering types, port range allocation support status, port parity preservation support status, port preservation support status, the behavior for handling fragments (all, out-of-order, in-order)).



### **2.3. TCP/UDP/ICMP NAT Behavioral Requirements**

This document assumes NAT behavioral recommendations for UDP [[RFC4787](#)], TCP [[RFC5382](#)], and ICMP [[RFC5508](#)] are enabled by default.

Furthermore, the NAT YANG module relies upon the recommendations detailed in [[RFC6888](#)] and [[RFC7857](#)].

### **2.4. Other Transport Protocols**

The module is structured to support other protocols than UDP, TCP, and ICMP. The mapping table is designed so that it can indicate any transport protocol. For example, this module may be used to manage a DCCP-capable NAT that adheres to [[RFC5597](#)].

Future extensions can be defined to cover NAT-related considerations that are specific to other transport protocols such as SCTP [[I-D.ietf-tsvwg-natsupp](#)]. Typically, the mapping entry can be extended to record two optional SCTP-specific parameters: Internal Verification Tag (Int-VTag) and External Verification Tag (Ext-VTag).

Also, the module allows to enable translation for these protocols when required (/nat/instances/instance/policy/transport-protocols).

### **2.5. IP Addresses Used for Translation**

The NAT YANG module assumes that blocks of IP external addresses (external-ip-address-pool) can be provisioned to the NAT function. These blocks may be contiguous or not.

This behavior is aligned with [[RFC6888](#)] which specifies that a NAT function should not have any limitations on the size or the contiguity of the external address pool. In particular, the NAT function must be configurable with contiguous or non-contiguous external IPv4 address ranges. To accommodate traditional NAT, the module allows for a single IP address to be configured for external-ip-address-pool.

Likewise, one or multiple IP address pools may be configured for Destination NAT (dst-ip-address-pool).

### **2.6. Port Set Assignment**

Port numbers can be assigned by a NAT individually (that is, a single port is assigned on a per session basis). Nevertheless, this port allocation scheme may not be optimal for logging purposes ([Section 12 of \[RFC6269\]](#)). Therefore, a NAT function should be able to assign port sets (e.g., [[RFC7753](#)]) to optimize the volume of the logging



data (REQ-14 of [\[RFC6888\]](#)). Both allocation schemes are supported in the NAT YANG module.

When port set assignment is activated (i.e., `port-allocation-type==port-range-allocation`), the NAT can be provided with the size of the port set to be assigned (`port-set-size`).

## **2.7. Port-Restricted IP Addresses**

Some NATs require to restrict the source port numbers (e.g., Lightweight 4over6 [\[RFC7596\]](#), MAP-E [\[RFC7597\]](#)). Two schemes of port set assignments (`port-set-restrict`) are supported in this document:

- o Simple port range: is defined by two port values, the start and the end of the port range [\[RFC8045\]](#).
- o Algorithmic: an algorithm is defined in [\[RFC7597\]](#) to characterize the set of ports that can be used.

## **2.8. NAT Mapping Entries**

A TCP/UDP mapping entry maintains an association between the following information:

```
(internal-src-address, internal-src-port) (internal-dst-address,  
internal-dst-port) <=> (external-src-address, external-src-port)  
(external-dst-address, external-dst-port)
```

An ICMP mapping entry maintains an association between the following information:

```
(internal-src-address, internal-dst-address, internal ICMP/ICMPv6  
identifier) <=> (external-src-address, external-dst-address,  
external ICMP/ICMPv6 identifier)
```

As a reminder, all the ICMP Query messages contain an 'Identifier' field, which is referred to in this document as the 'ICMP Identifier'.

To cover TCP, UDP, and ICMP, the NAT YANG module assumes the following structure of a mapping entry:

`type`: Indicates how the mapping was instantiated. For example, it may indicate whether a mapping is dynamically instantiated by a packet or statically configured.

`transport-protocol`: Indicates the transport protocol (e.g., UDP, TCP, ICMP) of a given mapping.





**internal-src-address:** Indicates the source IP address/prefix as used by an internal host.

**internal-src-port:** Indicates the source port number (or ICMP identifier) as used by an internal host.

**external-src-address:** Indicates the source IP address/prefix as assigned by the NAT.

**external-src-port:** Indicates the source port number (or ICMP identifier) as assigned by the NAT.

**internal-dst-address:** Indicates the destination IP address/prefix as used by an internal host when sending a packet to a remote host.

**internal-dst-port:** Indicates the destination port number as used by an internal host when sending a packet to a remote host.

**external-dst-address:** Indicates the destination IP address/prefix used by a NAT when processing a packet issued by an internal host towards a remote host.

**external-dst-port:** Indicates the destination port number used by a NAT when processing a packet issued by an internal host towards a remote host.

In order to cover both NAT64 and NAT44 flavors in particular, the NAT mapping structure allows to include an IPv4 or an IPv6 address as an internal IP address. Remaining fields are common to both NAT schemes.

For example, the mapping that will be created by a NAT64 upon receipt of a TCP SYN from source address 2001:db8:aaaa::1 and source port number 25636 to destination IP address 2001:db8:1234::198.51.100.1 and destination port number 8080 is shown in Table 2. This example assumes EDM (Endpoint-Dependent Mapping).



Mapping Entry Attribute	Value
type	dynamic implicit mapping
transport-protocol	6 (TCP)
internal-src-address	2001:db8:aaaa::1
internal-src-port	25636
external-src-address	T (an IPv4 address configured on the NAT64)
external-src-port	t (a port number that is chosen by the NAT64)
internal-dst-address	2001:db8:1234::198.51.100.1
internal-dst-port	8080
external-dst-address	198.51.100.1
external-dst-port	8080

Table 2: Example of an EDM NAT64 Mapping

The mappings that will be created by a NAT44 upon receipt of an ICMP request from source address 198.51.100.1 and ICMP identifier (ID1) to destination IP address 198.51.100.11 is depicted in Table 3. This example assumes EIM (Endpoint-Independent Mapping).

Mapping Entry Attribute	Value
type	dynamic implicit mapping
transport-protocol	1 (ICMP)
internal-src-address	198.51.100.1
internal-src-port	ID1
external-src-address	T (an IPv4 address configured on the NAT44)
external-src-port	ID2 (an ICMP identifier that is chosen by the NAT44)

Table 3: Example of an EIM NAT44 Mapping Entry

The mapping that will be created by a NAT64 (EIM mode) upon receipt of an ICMP request from source address 2001:db8:aaaa::1 and ICMP identifier (ID1) to destination IP address 2001:db8:1234::198.51.100.1 is shown in Table 4.



Mapping Entry	Value
Attribute	
type	dynamic implicit mapping
transport-protocol	58 (ICMPv6)
internal-src-address	2001:db8:aaaa::1
internal-src-port	ID1
external-src-address	T (an IPv4 address configured on the NAT64)
external-src-port	ID2 (an ICMP identifier that is chosen by the NAT64)

Table 4: Example of an EIM NAT64 Mapping Entry

Note that a mapping table is maintained only for stateful NAT functions. Particularly:

- o No mapping table is maintained for NPTv6 given that it is stateless and transport-agnostic.
- o The double translations are stateless in CLAT if a dedicated IPv6 prefix is provided for CLAT. If not, a stateful NAT44 will be required.
- o No per-flow mapping is maintained for EAM [[RFC7757](#)].
- o No mapping table is maintained for Stateless IPv4/IPv6 translation. As a reminder, in such deployments internal IPv6 nodes are addressed using IPv4-translatable IPv6 addresses, which enable them to be accessed by IPv4 nodes [[RFC6052](#)].

## 2.9. Resource Limits

In order to comply with CGN deployments in particular, the NAT YANG module allows limiting the number of external ports per subscriber (port-quota) and the amount of state memory allocated per mapping and per subscriber (mapping-limits and connection-limits). According to [[RFC6888](#)], the model allows for the following:

- o Per-subscriber limits are configurable by the NAT administrator.
- o Per-subscriber limits are configurable independently per transport protocol.
- o Administrator-adjustable thresholds to prevent a single subscriber from consuming excessive CPU resources from the NAT (e.g., rate-



limit the subscriber's creation of new mappings) can be configured.

Table 5 lists the various limits that can be set using the NAT YANG module. Once a limit is reached, packets that would normally trigger new port mappings or be translated because they match existing mappings, are dropped by the translator.

Limit	Description
port-quota	Specifies a port quota to be assigned per subscriber. It corresponds to the maximum number of ports to be used by a subscriber. The port quota can be configured to apply to all protocols or to a specific protocol. Distinct port quota may be configured per protocol.
fragments-limit	In order to prevent denial of service attacks that can be caused by fragments, this parameter is used to limit the number of out-of-order fragments that can be handled by a translator.
mapping-limits	This parameter can be used to control the maximum number of subscribers that can be serviced by a NAT instance (limit-subscriber) and the maximum number of address and/or port mappings that can be maintained by a NAT instance (limit-address-mappings and limit-port-mappings). Also, limits specific to protocols (e.g., TCP, UDP, ICMP) can also be specified (limit-per-protocol).
connection-limits	In order to prevent exhausting the resources of a NAT implementation and to ensure fairness usage among subscribers, various rate-limits can be specified. Rate-limiting can be enforced per subscriber (limit-subscriber), per NAT instance (limit-per-instance), and/or be specified for each supported protocol (limit-per-protocol).

Table 5: NAT Limits





Table 6 describes limits, that once exceeded, will trigger notifications to be generated:

Notification Threshold	Description
high-threshold	Used to notify high address utilization of a given pool. When exceeded, a nat-pool-event notification will be generated.
low-threshold	Used to notify low address utilization of a given pool. An administrator is supposed to configure low-threshold so that it can reflect an abnormal usage of NAT resources. When exceeded, a nat-pool-event notification will be generated.
notify-addresses-usage	Used to notify high address utilization of all pools configured to a NAT instance. When exceeded, a nat-instance-event will be generated.
notify-ports-usage	Used to notify high port allocation taking into account all pools configured to a NAT instance. When exceeded, a nat-instance-event notification will be generated.

Table 6: Notification Thresholds

In order to prevent from generating frequent notifications, the NAT YANG module supports the following limits (Table 7) used to control how frequent notifications can be generated. That is, notifications are subject to rate-limiting imposed by these intervals.



Interval	Description
notify-pool-usage/notify-interval	Indicates the minimum number of seconds between successive notifications for a given address pool.
notification-limits/notify-interval	Indicates the minimum number of seconds between successive notifications for a NAT instance.

Table 7: Notification Intervals

### 2.10. Binding the NAT Function to an External Interface

The model is designed to specify an external realm on which the NAT function must be applied (external-realm). The module supports indicating an interface as an external realm, but the module is extensible so that other choices can be indicated in the future (e.g., Virtual Routing and Forwarding (VRF) instance).

Distinct external realms can be provided as a function of the NAT policy (see for example, [Section 4 of \[RFC7289\]](#)).

If no external realm is provided, this assumes that the system is able to determine the external interface (VRF instance, etc.) on which the NAT will be applied. Typically, the WAN and LAN interfaces of a CPE are determined by the CPE.

### 2.11. Relationship to NATV2-MIB

Section of 5.1 of [\[RFC7659\]](#) indicates that the NATV2-MIB assumes that the following information is configured on the NAT by some means, not specified in [\[RFC7659\]](#):

- o The set of address realms to which the device connect.
- o For the CGN case, per-subscriber information including subscriber index, address realm, assigned prefix or address, and (possibly) policies regarding address pool selection in the various possible address realms to which the subscriber may connect.
- o The set of NAT instances running on the device, identified by NAT instance index and name.



- o The port mapping, filtering, pooling, and fragment behavior for each NAT instance.
- o The set of protocols supported by each NAT instance.
- o Address pools for each NAT instance, including for each pool the pool index, address realm, and minimum and maximum port number.
- o Static address and port mapping entries.

All the above parameters can be configured by means of the NAT YANG module.

Unlike the NATV2-MIB, the NAT YANG module allows to configure multiple policies per NAT instance.

## 2.12. Tree Structure

The tree structure of the NAT YANG module is provided below:

```

module: ietf-nat
  +--rw nat
    +--rw instances
      +--rw instance* [id]
        +--rw id                               uint32
        +--rw name?                           string
        +--rw enable?                         boolean
        +--ro capabilities
          | +--ro nat-flavor*                   identityref
          | +--ro transport-protocols* [protocol-id]
          | | +--ro protocol-id                uint8
          | | +--ro protocol-name?             string
          | +--ro restricted-port-support?      boolean
          | +--ro static-mapping-support?       boolean
          | +--ro port-randomization-support?   boolean
          | +--ro port-range-allocation-support? boolean
          | +--ro port-preservation-suport?     boolean
          | +--ro port-parity-preservation-support? boolean
          | +--ro address-roundrobin-support?   boolean
          | +--ro paired-address-pooling-support? boolean
          | +--ro endpoint-independent-mapping-support? boolean
          | +--ro address-dependent-mapping-support? boolean
          | +--ro address-and-port-dependent-mapping-support? boolean
          | +--ro endpoint-independent-filtering-support? boolean
          | +--ro address-dependent-filtering?   boolean
          | +--ro address-and-port-dependent-filtering? boolean
          | +--ro fragment-behavior?            enumeration
        +--rw nat-pass-through* [id] {basic-nat44 or napt44 or dst-nat}?

```



```

| +--rw id          uint32
| +--rw prefix      inet:ip-prefix
| +--rw port?       inet:port-number
+--rw policy* [id]
| +--rw id          uint32
| +--rw clat-parameters {clat}?
| | +--rw clat-ipv6-prefixes* [ipv6-prefix]
| | | +--rw ipv6-prefix      inet:ipv6-prefix
| | +--rw ipv4-prefixes* [ipv4-prefix]
| |   +--rw ipv4-prefix      inet:ipv4-prefix
| +--rw nptv6-prefixes* [internal-ipv6-prefix] {nptv6}?
| | +--rw internal-ipv6-prefix      inet:ipv6-prefix
| | +--rw external-ipv6-prefix      inet:ipv6-prefix
| +--rw eam* [ipv4-prefix] {eam}?
| | +--rw ipv4-prefix      inet:ipv4-prefix
| | +--rw ipv6-prefix      inet:ipv6-prefix
| +--rw nat64-prefixes* [nat64-prefix] {siit or nat64 or clat}?
| | +--rw nat64-prefix      inet:ipv6-prefix
| | +--rw destination-ipv4-prefix* [ipv4-prefix]
| | | +--rw ipv4-prefix      inet:ipv4-prefix
| | +--rw stateless-enable?      boolean
| +--rw external-ip-address-pool* [pool-id] {basic-nat44 or
napt44 or nat64}?
| | +--rw pool-id          uint32
| | +--rw external-ip-pool      inet:ipv4-prefix
| +--rw port-set-restrict {napt44 or nat64}?
| | +--rw (port-type)?
| |   +--:(port-range)
| |     | +--rw start-port-number?      inet:port-number
| |     | +--rw end-port-number?        inet:port-number
| |     +--:(port-set-algo)
| |       +--rw psid-offset?            uint8
| |       +--rw psid-len                uint8
| |       +--rw psid                    uint16
| +--rw dst-nat-enable?            boolean {basic-nat44 or
napt44}?
| +--rw dst-ip-address-pool* [pool-id] {dst-nat}?
| | +--rw pool-id          uint32
| | +--rw dst-in-ip-pool?    inet:ip-prefix
| | +--rw dst-out-ip-pool    inet:ip-prefix
| +--rw transport-protocols* [protocol-id] {napt44 or nat64 or
dst-nat}?
| | +--rw protocol-id      uint8
| | +--rw protocol-name?    string
| +--rw subscriber-mask-v6?      uint8
| +--rw subscriber-match* [match-id] {basic-nat44 or napt44 or
dst-nat}?
| | +--rw match-id          uint32

```



			+-rw subnet	inet:ip-prefix
			+-rw address-allocation-type?	enumeration
nat64}?			+-rw port-allocation-type?	enumeration {napt44 or
			+-rw mapping-type?	enumeration {napt44 or
nat64}?				

```

nat64}?
| +--rw filtering-type?          enumeration {napt44 or
nat64}?
| +--rw fragment-behavior?      enumeration {napt44 or
nat64}?
| +--rw port-quota* [quota-type] {napt44 or nat64}?
| | +--rw port-limit?          uint16
| | +--rw quota-type           uint8
| +--rw port-set {napt44 or nat64}?
| | +--rw port-set-size        uint16
| | +--rw port-set-timeout?    uint32
| +--rw timers {napt44 or nat64}?
| | +--rw udp-timeout?          uint32
| | +--rw tcp-idle-timeout?     uint32
| | +--rw tcp-trans-open-timeout? uint32
| | +--rw tcp-trans-close-timeout? uint32
| | +--rw tcp-in-syn-timeout?   uint32
| | +--rw fragment-min-timeout? uint32
| | +--rw icmp-timeout?         uint32
| | +--rw per-port-timeout* [port-number]
| | | +--rw port-number        inet:port-number
| | | +--rw timeout            uint32
| | +--rw hold-down-timeout?    uint32
| | +--rw hold-down-max?        uint32
| +--rw fragments-limit?        uint32
| +--rw algs* [name]
| | +--rw name                  string
| | +--rw transport-protocol?   uint32
| | +--rw dst-transport-port
| | | +--rw start-port-number?   inet:port-number
| | | +--rw end-port-number?     inet:port-number
| | +--rw src-transport-port
| | | +--rw start-port-number?   inet:port-number
| | | +--rw end-port-number?     inet:port-number
| | +--rw status?               boolean
| +--rw all-algs-enable?         boolean
| +--rw notify-pool-usage {basic-nat44 or napt44 or nat64}?
| | +--rw pool-id?              uint32
| | +--rw high-threshold?        percent
| | +--rw low-threshold?         percent
| | +--rw notify-interval?       uint32
| +--rw external-realm
| | +--rw (realm-type)?
| | | +--:(interface)
| | | +--rw external-interface?  if:interface-ref
+--rw mapping-limits {napt44 or nat64}?
| +--rw limit-subscribers?       uint32
| +--rw limit-address-mapings?   uint32
| +--rw limit-port-mappings?     uint32

```

```
dst-nat}? | ---rw limit-per-protocol* [protocol-id] {napt44 or nat64 or
|          ---rw protocol-id      uint8
```

```

|      +--rw limit?          uint32
+--rw connection-limits {basic-nat44 or napt44 or nat64}?
|  +--rw limit-per-subscriber?  uint32
|  +--rw limit-per-instance     uint32
|  +--rw limit-per-protocol* [protocol-id] {napt44 or nat64}?
|      +--rw protocol-id      uint8
|      +--rw limit?           uint32
+--rw notification-limits
|  +--rw notify-interval?      uint32 {basic-nat44 or napt44
or nat64}?
|  +--rw notify-addresses-usage? percent {basic-nat44 or napt44
or nat64}?
|  +--rw notify-ports-usage?   percent {napt44 or nat64}?
+--rw logging-enable?         boolean {basic-nat44 or napt44 or
nat64}?
+--rw mapping-table {basic-nat44 or napt44 or nat64 or clat or
dst-nat}?
|  +--rw mapping-entry* [index]
|      +--rw index              uint32
|      +--rw type?              enumeration
|      +--rw transport-protocol? uint8
|      +--rw internal-src-address? inet:ip-prefix
|      +--rw internal-src-port
|          | +--rw start-port-number? inet:port-number
|          | +--rw end-port-number?   inet:port-number
|      +--rw external-src-address? inet:ip-prefix
|      +--rw external-src-port
|          | +--rw start-port-number? inet:port-number
|          | +--rw end-port-number?   inet:port-number
|      +--rw internal-dst-address? inet:ip-prefix
|      +--rw internal-dst-port
|          | +--rw start-port-number? inet:port-number
|          | +--rw end-port-number?   inet:port-number
|      +--rw external-dst-address? inet:ip-prefix
|      +--rw external-dst-port
|          | +--rw start-port-number? inet:port-number
|          | +--rw end-port-number?   inet:port-number
|      +--rw lifetime?          uint32
+--ro statistics
    +--ro discontinuity-time      yang:date-and-time
    +--ro traffic-statistics
        | +--ro sent-packets?      yang:zero-based-
counter64
        | +--ro sent-bytes?        yang:zero-based-
counter64
        | +--ro rcvd-packets?      yang:zero-based-
counter64
        | +--ro rcvd-bytes?        yang:zero-based-

```

counter64	+--ro dropped-packets?	yang:zero-based-
counter64	+--ro dropped-bytes?	yang:zero-based-
counter64	+--ro dropped-fragments?	yang:zero-based-
counter64 {napt44 or nat64}?	+--ro dropped-address-limit-packets?	yang:zero-based-
counter64 {basic-nat44 or napt44 or nat64}?	+--ro dropped-address-limit-bytes?	yang:zero-based-
counter64 {basic-nat44 or napt44 or nat64}?	+--ro dropped-address-packets?	yang:zero-based-
counter64 {basic-nat44 or napt44 or nat64}?	+--ro dropped-address-bytes?	yang:zero-based-
counter64 {basic-nat44 or napt44 or nat64}?		

```

        | +--ro dropped-port-limit-packets?      yang:zero-based-
counter64 {napt44 or nat64}?
        | +--ro dropped-port-limit-bytes?        yang:zero-based-
counter64 {napt44 or nat64}?
        | +--ro dropped-port-packets?            yang:zero-based-
counter64 {napt44 or nat64}?
        | +--ro dropped-port-bytes?              yang:zero-based-
counter64 {napt44 or nat64}?
        | +--ro dropped-subscriber-packets?      yang:zero-based-
counter64 {basic-nat44 or napt44 or nat64}?
        | +--ro dropped-subscriber-bytes?        yang:zero-based-
counter64 {basic-nat44 or napt44 or nat64}?
        +--ro mappings-statistics
        | +--ro total-address-mappings?          yang:gauge32 {basic-nat44 or
napt44 or nat64 or clat or dst-nat}?
        | +--ro total-port-mappings?            yang:gauge32 {napt44 or
nat64}?
        | +--ro total-per-protocol* [protocol-id] {napt44 or nat64}?
        |   +--ro protocol-id      uint8
        |   +--ro total?            yang:gauge32
        +--ro pools-stats {basic-nat44 or napt44 or nat64}?
        +--ro addresses-allocated?  yang:gauge32
        +--ro addresses-free?       yang:gauge32
        +--ro ports-stats {napt44 or nat64}?
        | +--ro ports-allocated?    yang:gauge32
        | +--ro ports-free?         yang:gauge32
        +--ro per-pool-stats* [pool-id] {basic-nat44 or napt44 or
nat64}?
        +--ro pool-id                uint32
        +--ro discontinuity-time      yang:date-and-time
        +--ro pool-stats
        | +--ro addresses-allocated?  yang:gauge32
        | +--ro addresses-free?       yang:gauge32
        +--ro port-stats {napt44 or nat64}?
        +--ro ports-allocated?        yang:gauge32
        +--ro ports-free?             yang:gauge32

notifications:
  +---n nat-pool-event {basic-nat44 or napt44 or nat64}?
  | +--ro id                -> /nat/instances/instance/id
  | +--ro policy-id?        -> /nat/instances/instance/policy/id
  | +--ro pool-id           -> /nat/instances/instance/policy/
external-ip-address-pool/pool-id
  | +--ro notify-pool-threshold  percent
  +---n nat-instance-event {basic-nat44 or napt44 or nat64}?
  +--ro id                    -> /nat/instances/instance/id
  +--ro notify-addresses-threshold? percent
  +--ro notify-ports-threshold?  percent

```

### **3. NAT YANG Module**

```
<CODE BEGINS> file "ietf-nat@2017-11-13.yang"

module ietf-nat {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-nat";

  //namespace to be assigned by IANA
```

```
prefix "nat";
```

```
import ietf-inet-types { prefix inet; }  
import ietf-yang-types { prefix yang; }  
import ietf-interfaces { prefix if; }
```

```
organization
```

```
  "IETF OPSAWG (Operations and Management Area Working Group)";
```

```
contact
```

```
  "WG Web:  <https://datatracker.ietf.org/wg/opsawg/>
```

```
  WG List:  <mailto:opsawg@ietf.org>
```

```
  WG Chair: Ignas Bagdonas  
            <mailto:ibagdona@gmail.com>
```

```
  WG Chair: Joe Clarke  
            <mailto:jclarke@cisco.com>
```

```
  WG Chair: Tianran Zhou  
            <mailto:zhoutianran@huawei.com>
```

```
  Editor:  Mohamed Boucadair  
           <mailto:mohamed.boucadair@orange.com>
```

```
  Editor:  Senthil Sivakumar  
           <mailto:ssenthil@cisco.com>
```

```
  Editor:  Christian Jacquenet  
           <mailto:christian.jacquenet@orange.com>
```

```
  Editor:  Suresh Vinapamula  
           <mailto:sureshk@juniper.net>
```

```
  Editor:  Qin Wu  
           <mailto:bill.wu@huawei.com>";
```

```
description
```

```
  "This module is a YANG module for NAT implementations  
  (including NAT44 and NAT64 flavors)."
```

```
  Copyright (c) 2017 IETF Trust and the persons identified as  
  authors of the code.  All rights reserved.
```

```
  Redistribution and use in source and binary forms, with or  
  without modification, is permitted pursuant to, and subject  
  to the license terms contained in, the Simplified BSD License
```





set forth in [Section 4.c](#) of the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>).

This version of this YANG module is part of RFC XXXX; see the RFC itself for full legal notices.";

```
revision 2017-11-13 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Data Model for Network Address Translation
      (NAT) and Network Prefix Translation (NPT)";
}

/*
 * Definitions
 */

typedef percent {
  type uint8 {
    range "0 .. 100";
  }
  description
    "Percentage";
}

/*
 * Features
 */

feature basic-nat44{
  description
    "Basic NAT44 translation is limited to IP addresses alone.";
  reference
    "RFC 3022: Traditional IP Network Address Translator
      (Traditional NAT)";
}

feature napt44 {
  description
    "Network Address/Port Translator (NAPT): translation is
      extended to include IP addresses and transport identifiers
      (such as a TCP/UDP port or ICMP query ID).";
  reference
    "RFC 3022: Traditional IP Network Address Translator
      (Traditional NAT)";
}
```



```
feature dst-nat {
  description
    "Destination NAT is a translation that acts on the destination
    IP address and/or destination port number. This flavor is
    usually deployed in load balancers or at devices
    in front of public servers.";
}

feature nat64 {
  description
    "NAT64 translation allows IPv6-only clients to contact IPv4
    servers using unicast UDP, TCP, or ICMP. One or more
    public IPv4 addresses assigned to a NAT64 translator are
    shared among several IPv6-only clients.";
  reference
    "RFC 6146: Stateful NAT64: Network Address and Protocol
    Translation from IPv6 Clients to IPv4 Servers";
}

feature siit {
  description
    "The Stateless IP/ICMP Translation Algorithm (SIIT), which
    translates between IPv4 and IPv6 packet headers (including
    ICMP headers).

    In the stateless mode, an IP/ICMP translator converts IPv4
    addresses to IPv6 and vice versa solely based on the
    configuration of the stateless IP/ICMP translator and
    information contained within the packet being translated.

    The translator must support the stateless address mapping
    algorithm defined in RFC6052, which is the default behavior.";
  reference
    "RFC 7915: IP/ICMP Translation Algorithm";
}

feature clat {
  description
    "CLAT is customer-side translator that algorithmically
    translates 1:1 private IPv4 addresses to global IPv6 addresses,
    and vice versa.

    When a dedicated /64 prefix is not available for translation
    from DHCPv6-PD, the CLAT may perform NAT44 for all IPv4 LAN
    packets so that all the LAN-originated IPv4 packets appear
    from a single IPv4 address and are then statelessly translated
    to one interface IPv6 address that is claimed by the CLAT via
    the Neighbor Discovery Protocol (NDP) and defended with
```



```
        Duplicate Address Detection.";
    reference
        "RFC 6877: 464XLAT: Combination of Stateful and Stateless
        Translation";
}

feature eam {
    description
        "Explicit Address Mapping (EAM) is a bidirectional coupling
        between an IPv4 Prefix and an IPv6 Prefix.";
    reference
        "RFC 7757: Explicit Address Mappings for Stateless IP/ICMP
        Translation";
}

feature nptv6 {
    description
        "NPTv6 is a stateless transport-agnostic IPv6-to-IPv6
        prefix translation.";
    reference
        "RFC 6296: IPv6-to-IPv6 Network Prefix Translation";
}

/*
 * Identities
 */

identity nat-type {
    description
        "Base identity for nat type.";
}

identity basic-nat44 {
    base nat:nat-type;
    description
        "Identity for Basic NAT support.";
    reference
        "RFC 3022: Traditional IP Network Address Translator
        (Traditional NAT)";
}

identity napt44 {
    base nat:nat-type;
    description
        "Identity for NAPT support.";
    reference
        "RFC 3022: Traditional IP Network Address Translator
        (Traditional NAT)";
}
```



```
}
```

```
identity dst-nat {  
  base nat:nat-type;  
  description  
    "Identity for Destination NAT support."  
}
```

```
identity nat64 {  
  base nat:nat-type;  
  description  
    "Identity for NAT64 support."  
  reference  
    "RFC 6146: Stateful NAT64: Network Address and Protocol  
      Translation from IPv6 Clients to IPv4 Servers";  
}
```

```
identity siit {  
  base nat:nat-type;  
  description  
    "Identity for SIIT support."  
  reference  
    "RFC 7915: IP/ICMP Translation Algorithm";  
}
```

```
identity clat {  
  base nat:nat-type;  
  description  
    "Identity for CLAT support."  
  reference  
    "RFC 6877: 464XLAT: Combination of Stateful and Stateless  
      Translation";  
}
```

```
identity eam {  
  base nat:nat-type;  
  description  
    "Identity for EAM support."  
  reference  
    "RFC 7757: Explicit Address Mappings for Stateless IP/ICMP  
      Translation";  
}
```

```
identity nptv6 {  
  base nat:nat-type;  
  description  
    "Identity for NPTv6 support."  
  reference
```





```
    "RFC 6296: IPv6-to-IPv6 Network Prefix Translation";
}

/*
 * Grouping
 */

grouping port-number {
  description
    "Individual port or a range of ports.
    When only start-port-number is present,
    it represents a single port.";

  leaf start-port-number {
    type inet:port-number;
    description
      "Begining of the port range.";
    reference
      "Section 3.2.9 of RFC 8045";
  }

  leaf end-port-number {
    type inet:port-number;

    must ". >= ../start-port-number"
    {
      error-message
        "The end-port-number must be greater than or
        equal to start-port-number.";
    }
    description
      "End of the port range.";
    reference
      "Section 3.2.10 of RFC 8045";
  }
}

grouping port-set {
  description
    "Indicates a set of ports.

    It may be a simple port range, or use the Port Set ID (PSID)
    algorithm to represent a range of transport layer
    ports which will be used by a NAPT.";

  choice port-type {
    default port-range;
    description
```



```
"Port type: port-range or port-set-algo.";
case port-range {
  uses port-number;
}

case port-set-algo {
  leaf psid-offset {
    type uint8 {
      range 0..15;
    }

    description
      "The number of offset bits (a.k.a., 'a' bits).

      Specifies the numeric value for the excluded port
      range/offset bits.

      Allowed values are between 0 and 15 ";

    reference
      "Section 5.1 of RFC 7597";
  }

  leaf psid-len {
    type uint8 {
      range 0..15;
    }
    mandatory true;

    description
      "The length of PSID, representing the sharing
      ratio for an IPv4 address.

      (also known as 'k').

      The address-sharing ratio would be 2^k.";
    reference
      "Section 5.1 of RFC 7597";
  }

  leaf psid {
    type uint16;
    mandatory true;
    description
      "Port Set Identifier (PSID) value, which
      identifies a set of ports algorithmically.";
    reference
      "Section 5.1 of RFC 7597";
```



```
    }
  }
  reference
    "Section 7597: Mapping of Address and Port with
      Encapsulation (MAP-E)";
}
}

grouping mapping-entry {
  description
    "NAT mapping entry.

    If an attribute is not stored in the mapping/session table,
    this means the corresponding fields of a packet that
    matches this entry is not rewritten by the NAT or this
    information is not required for NAT filtering purposes.";

  leaf index {
    type uint32;
    description
      "A unique identifier of a mapping entry. This identifier can be
      automatically assigned by the NAT instance or be explicitly
      configured.";
  }

  leaf type {
    type enumeration {
      enum "static" {
        description
          "The mapping entry is explicitly configured
          (e.g., via command-line interface).";
      }

      enum "dynamic-implicit" {
        description
          "This mapping is created implicitly as a side effect
          of processing a packet that requires a new mapping.";
      }

      enum "dynamic-explicit" {
        description
          "This mapping is created as a result of an explicit
          request, e.g., a PCP message.";
      }
    }
  }
  description
```



```
        "Indicates the type of a mapping entry. E.g.,
        a mapping can be: static, implicit dynamic
        or explicit dynamic.";
    }

    leaf transport-protocol {
        type uint8;
        description
            "Upper-layer protocol associated with this mapping.
            Values are taken from the IANA protocol registry.
            For example, this field contains 6 (TCP) for a TCP
            mapping or 17 (UDP) for a UDP mapping.

            If this leaf is not instantiated, then the mapping
            applies to any protocol.";
    }

    leaf internal-src-address {
        type inet:ip-prefix;
        description
            "Corresponds to the source IPv4/IPv6 address/prefix
            of the packet received on an internal
            interface.";
    }

    container internal-src-port {
        description
            "Corresponds to the source port of the packet received
            on an internal interface.

            It is used also to indicate the internal source ICMP
            identifier.

            As a reminder, all the ICMP Query messages contain
            an 'Identifier' field, which is referred to in this
            document as the 'ICMP Identifier'.";

        uses port-number;
    }

    leaf external-src-address {
        type inet:ip-prefix;
        description
            "Source IP address/prefix of the packet sent on an
            external interface of the NAT.";
    }

    container external-src-port {
```





```
description
  "Source port of the packet sent on an external
  interface of the NAT.

  It is used also to indicate the external source ICMP
  identifier.";

  uses port-number;
}

leaf internal-dst-address {
  type inet:ip-prefix;
  description
    "Corresponds to the destination IP address/prefix
    of the packet received on an internal interface
    of the NAT.

    For example, some NAT implementations support
    the translation of both source and destination
    addresses and ports, sometimes referred to
    as 'Twice NAT'.";
}

container internal-dst-port {
  description
    "Corresponds to the destination port of the
    IP packet received on the internal interface.

    It is used also to include the internal
    destination ICMP identifier.";

  uses port-number;
}

leaf external-dst-address {
  type inet:ip-prefix;
  description
    "Corresponds to the destination IP address/prefix
    of the packet sent on an external interface
    of the NAT.";
}

container external-dst-port {
  description
    "Corresponds to the destination port number of
    the packet sent on the external interface
    of the NAT."
```



It is used also to include the external destination ICMP identifier.";

uses port-number;  
}

leaf lifetime {  
 type uint32;  
 units "seconds";  
 description  
 "When specified, it is used to track the connection that is fully-formed (e.g., once the three-way handshake TCP is completed) or the duration for maintaining an explicit mapping alive. The mapping entry will be removed by the NAT instance once this lifetime is expired.

When reported in a get operation, the lifetime indicates the remaining validity lifetime.

Static mappings may not be associated with a lifetime. If no lifetime is associated with a static mapping, an explicit action is required to remove that mapping.";

}  
}

/\*  
 \* NAT Module  
 \*/

container nat {  
 description  
 "NAT module";

container instances {  
 description  
 "NAT instances";

list instance {  
 key "id";  
  
 description  
 "A NAT instance. This identifier can be automatically assigned or explicitly configured.";

leaf id {  
 type uint32;  
 must ". >= 1";



```
    description
      "NAT instance identifier.

      The identifier must be greater than zero as per RFC 7659.";
    reference
      "RFC 7659: Definitions of Managed Objects for Network
        Address Translators (NATs)";
  }

  leaf name {
    type string;
    description
      "A name associated with the NAT instance.";
    reference
      "RFC 7659: Definitions of Managed Objects for Network
        Address Translators (NATs)";
  }

  leaf enable {
    type boolean;
    description
      "Status of the the NAT instance.";
  }

  container capabilities {
    config false;

    description
      "NAT capabilities";

    leaf-list nat-flavor {
      type identityref {
        base nat-type;
      }
      description
        "Type of NAT.";
    }
  }

  list transport-protocols {
    key protocol-id;

    description
      "List of supported protocols.";

    leaf protocol-id {
      type uint8;
      mandatory true;
      description
```



```
    "Upper-layer protocol associated with this mapping.

    Values are taken from the IANA protocol registry:
    https://www.iana.org/assignments/protocol-numbers/
    protocol-numbers.xhtml

    For example, this field contains 6 (TCP) for a TCP
    mapping or 17 (UDP) for a UDP mapping.";
}

leaf protocol-name {
    type string;
    description
        "The name of the Upper-layer protocol associated
        with this mapping.

        Values are taken from the IANA protocol registry:
        https://www.iana.org/assignments/protocol-numbers/
        protocol-numbers.xhtml

        For example, TCP, UDP, DCCP, and SCTP.";
}

leaf restricted-port-support {
    type boolean;
    description
        "Indicates source port NAT restriction support.";
    reference
        "RFC 7596: Lightweight 4over6: An Extension to
        the Dual-Stack Lite Architecture.";
}

leaf static-mapping-support {
    type boolean;
    description
        "Indicates whether static mappings are supported.";
}

leaf port-randomization-support {
    type boolean;
    description
        "Indicates whether port randomization is supported.";
    reference
        "Section 4.2.1. of RFC 4787.";
}

leaf port-range-allocation-support {
```





```
    type boolean;
    description
      "Indicates whether port range allocation is supported.";
    reference
      "Section 1.1 of RFC 7753.";
  }

  leaf port-preservation-suport {
    type boolean;
    description
      "Indicates whether port preservation is supported.";
    reference
      "Section 4.2.1. of RFC 4787.";
  }

  leaf port-parity-preservation-support {
    type boolean;
    description
      "Indicates whether port parity preservation is
        supported.";
    reference
      "Section 8 of RFC 7857.";
  }

  leaf address-roundrobin-support {
    type boolean;
    description
      "Indicates whether address allocation round robin is
        supported.";
  }

  leaf paired-address-pooling-support {
    type boolean;
    description
      "Indicates whether paired-address-pooling is
        supported";
    reference
      "REQ-2 of RFC 4787.";
  }

  leaf endpoint-independent-mapping-support {
    type boolean;
    description
      "Indicates whether endpoint-independent-
        mapping in Section 4 of RFC 4787 is
        supported.";
    reference
      "Section 4 of RFC 4787.";
  }
```



```
}

leaf address-dependent-mapping-support {
  type boolean;
  description
    "Indicates whether address-dependent-mapping is
    supported.";
  reference
    "Section 4 of RFC 4787.";
}

leaf address-and-port-dependent-mapping-support {
  type boolean;
  description
    "Indicates whether address-and-port-dependent-mapping is
    supported.";
  reference
    "Section 4 of RFC 4787.";
}

leaf endpoint-independent-filtering-support {
  type boolean;
  description
    "Indicates whether endpoint-independent-filtering is
    supported.";
  reference
    "Section 5 of RFC 4787.";
}

leaf address-dependent-filtering {
  type boolean;
  description
    "Indicates whether address-dependent-filtering is
    supported.";
  reference
    "Section 5 of RFC 4787.";
}

leaf address-and-port-dependent-filtering {
  type boolean;
  description
    "Indicates whether address-and-port-dependent is
    supported.";
  reference
    "Section 5 of RFC 4787.";
}

leaf fragment-behavior {
```



```
type enumeration {
  enum "unsupported" {
    description
      "No capability to translate incoming fragments.
       All received fragments are dropped.";
  }

  enum "in-order" {
    description
      "The NAT instance is able to translate fragments only if
       they are received in order. That is, in particular the
       header is in the first packet. Fragments received
       out of order are dropped. ";
  }

  enum "out-of-order" {
    description
      "The NAT instance is able to translate a fragment even
       if it is received out of order.

       This behavior is the one recommended in RFC4787";
    reference
      "REQ-14 of RFC 4787";
  }
}
description
  "The fragment behavior is the NAT instance's capability to
   translate fragments received on the external interface of
   the NAT.";
}

list nat-pass-through {
  if-feature "basic-nat44 or napt44 or dst-nat";
  key id;

  description
    "IP prefix NAT pass through.";

  leaf id {
    type uint32;
    description
      "An identifier of the IP prefix pass through.";
  }

  leaf prefix {
    type inet:ip-prefix;
```



```
    mandatory true;
    description
      "The IP addresses that match should not be translated.

      According to REQ#6 of RFC6888, it must be possible to
      administratively turn off translation for specific
      destination addresses and/or ports.";
    reference
      "REQ#6 of RFC6888.";
  }

  leaf port {
    type inet:port-number;
    description
      "According to REQ#6 of RFC6888, it must be possible to
      administratively turn off translation for specific
      destination addresses and/or ports.

      If no prefix is defined, the NAT pass through bound
      to a given port applies for any destination address.";

    reference
      "REQ#6 of RFC6888.";
  }
}

list policy {
  key id;
  description
    "NAT parameters for a given instance";

  leaf id {
    type uint32;
    description
      "An identifier of the NAT policy.
      it must be unique within the NAT instance.";
  }
}

container clat-parameters {
  if-feature clat;
  description
    "CLAT parameters.";

  list clat-ipv6-prefixes {
    key ipv6-prefix;
    description
      "464XLAT double translation treatment is stateless when a
      dedicated /64 is available for translation on the CLAT.
```





```
    Otherwise, the CLAT will have both stateful and stateless
    since it requires NAT44 from the LAN to a single IPv4
    address and then stateless translation to a single
    IPv6 address.";
  reference
    "RFC 6877: 464XLAT: Combination of Stateful and Stateless
    Translation";

  leaf ipv6-prefix {
    type inet:ipv6-prefix;
    description
      "An IPv6 prefix used for CLAT.";
  }
}

list ipv4-prefixes {
  key ipv4-prefix;
  description
    "Pool of IPv4 addresses used for CLAT.
    192.0.0.0/29 is the IPv4 service continuity prefix.";
  reference
    "RFC 7335: IPv4 Service Continuity Prefix";

  leaf ipv4-prefix {
    type inet:ipv4-prefix;
    description
      "464XLAT double translation treatment is
      stateless when a dedicated /64 is available
      for translation on the CLAT. Otherwise, the
      CLAT will have both stateful and stateless
      since it requires NAT44 from the LAN to
      a single IPv4 address and then stateless
      translation to a single IPv6 address.
      The CLAT performs NAT44 for all IPv4 LAN
      packets so that all the LAN-originated IPv4
      packets appear from a single IPv4 address
      and are then statelessly translated to one
      interface IPv6 address that is claimed by
      the CLAT.

      An IPv4 address from this pool is also
      provided to an application that makes
      use of literals.";

    reference
      "RFC 6877: 464XLAT: Combination of Stateful and Stateless
      Translation";
  }
}
```



```
    }
  }

  list nptv6-prefixes {
    if-feature nptv6;
    key internal-ipv6-prefix ;
    description
      "Provides one or a list of (internal IPv6 prefix,
       external IPv6 prefix) required for NPTv6.

       In its simplest form, NPTv6 interconnects two network
       links, one of which is an 'internal' network link
       attached to a leaf network within a single
       administrative domain and the other of which is an
       'external' network with connectivity to the global
       Internet.";
    reference
      "RFC 6296: IPv6-to-IPv6 Network Prefix Translation";

    leaf internal-ipv6-prefix {
      type inet:ipv6-prefix;
      mandatory true;
      description
        "An IPv6 prefix used by an internal interface of NPTv6.";
      reference
        "RFC 6296: IPv6-to-IPv6 Network Prefix Translation";
    }

    leaf external-ipv6-prefix {
      type inet:ipv6-prefix;
      mandatory true;
      description
        "An IPv6 prefix used by the external interface of NPTv6.";
      reference
        "RFC 6296: IPv6-to-IPv6 Network Prefix Translation";
    }
  }

  list eam {
    if-feature eam;
    key ipv4-prefix;
    description
      "The Explicit Address Mapping Table, a conceptual
       table in which each row represents an EAM.

       Each EAM describes a mapping between IPv4 and IPv6
       prefixes/addresses.";
    reference
```



```
    "Section 3.1 of RFC 7757.";

leaf ipv4-prefix {
    type inet:ipv4-prefix;
    mandatory true;
    description
        "The IPv4 prefix of an EAM.";
    reference
        "Section 3.2 of RFC 7757.";
}

leaf ipv6-prefix {
    type inet:ipv6-prefix;
    mandatory true;
    description
        "The IPv6 prefix of an EAM.";
    reference
        "Section 3.2 of RFC 7757.";
}
}

list nat64-prefixes {
    if-feature "siit or nat64 or clat";
    key nat64-prefix;
    description
        "Provides one or a list of NAT64 prefixes
         with or without a list of destination IPv4 prefixes.

         Destination-based Pref64::/n is discussed in
         Section 5.1 of \[RFC7050\]). For example:
         192.0.2.0/24 is mapped to 2001:db8:122:300::/56.
         198.51.100.0/24 is mapped to 2001:db8:122::/48.";
    reference
        "Section 5.1 of RFC7050.";

leaf nat64-prefix {
    type inet:ipv6-prefix;
    mandatory true;
    description
        "A NAT64 prefix. Can be Network-Specific Prefix (NSP) or
         Well-Known Prefix (WKP).

         Organizations deploying stateless IPv4/IPv6 translation
         should assign a Network-Specific Prefix to their
         IPv4/IPv6 translation service.

         For stateless NAT64, IPv4-translatable IPv6 addresses
         must use the selected Network-Specific Prefix.
```



```
        Both IPv4-translatable IPv6 addresses and IPv4-converted
        IPv6 addresses should use the same prefix.";
    reference
        "Sections 3.3 and 3.4 of RFC 6052.";
}

list destination-ipv4-prefix {
    key ipv4-prefix;
    description
        "An IPv4 prefix/address.";

    leaf ipv4-prefix {
        type inet:ipv4-prefix;
        description
            "An IPv4 address/prefix.";
    }
}

leaf stateless-enable {
    type boolean;
    default false;
    description
        "Enable explicitly stateless NAT64.";
}

list external-ip-address-pool {
    if-feature "basic-nat44 or napt44 or nat64";
    key pool-id;

    description
        "Pool of external IP addresses used to service internal
        hosts.

        A pool is a set of IP prefixes.";

    leaf pool-id {
        type uint32;
        must ". >= 1";
        description
            "An identifier that uniquely identifies the address pool
            within a NAT instance.

            The identifier must be greater than zero as per
            RFC 7659.";
    }
    reference
        "RFC 7659: Definitions of Managed Objects for
        Network Address Translators (NATs)";
}
```





```
    }

    leaf external-ip-pool {
      type inet:ipv4-prefix;
      mandatory true;
      description
        "An IPv4 prefix used for NAT purposes.";
    }
  }

  container port-set-restrict {
    if-feature "napt44 or nat64";
    description
      "Configures contiguous and non-contiguous port ranges.

      The port set is used to restrict the external source
      port numbers used by the translator.";

    uses port-set;
  }

  leaf dst-nat-enable {
    if-feature "basic-nat44 or napt44";
    type boolean;
    default false;
    description
      "Enable/Disable destination NAT.

      A NAT44 may be configured to enable Destination
      NAT, too.";
  }

  list dst-ip-address-pool {
    if-feature dst-nat;
    key pool-id;
    description
      "Pool of IP addresses used for destination NAT.";

    leaf pool-id {
      type uint32;
      description
        "An identifier of the address pool.";
    }

    leaf dst-in-ip-pool {
      type inet:ip-prefix;
      description
        "Is used to identify an internal IP prefix/address
```



```
        to be translated.";
    }

    leaf dst-out-ip-pool {
        type inet:ip-prefix;
        mandatory true;
        description
            "IP address/prefix used for destination NAT.";
    }
}

list transport-protocols {
    if-feature "napt44 or nat64 or dst-nat";
    key protocol-id;

    description
        "Configure the transport protocols to be handled by
        the translator.

        TCP and UDP are supported by default.";

    leaf protocol-id {
        type uint8;
        mandatory true;
        description
            "Upper-layer protocol associated with this mapping.

            Values are taken from the IANA protocol registry:
            https://www.iana.org/assignments/protocol-numbers/
            protocol-numbers.xhtml

            For example, this field contains 6 (TCP) for a TCP
            mapping or 17 (UDP) for a UDP mapping.";
    }

    leaf protocol-name {
        type string;
        description
            "The name of the Upper-layer protocol associated
            with this mapping.

            Values are taken from the IANA protocol registry:
            https://www.iana.org/assignments/protocol-numbers/
            protocol-numbers.xhtml

            For example, TCP, UDP, DCCP, and SCTP.";
    }
}
```



```
leaf subscriber-mask-v6 {
  type uint8 {
    range "0 .. 128";
  }

  description
    "The subscriber-mask is an integer that indicates
    the length of significant bits to be applied on
    the source IPv6 address (internal side) to
    unambiguously identify a CPE.

    Subscriber-mask is a system-wide configuration
    parameter that is used to enforce generic
    per-subscriber policies (e.g., port-quota).

    The enforcement of these generic policies does not
    require the configuration of every subscriber's
    prefix.

    Example: suppose the 2001:db8:100:100::/56 prefix
    is assigned to a NAT64 serviced CPE. Suppose also
    that 2001:db8:100:100::1 is the IPv6 address used
    by the client that resides in that CPE. When the
    NAT64 receives a packet from this client,
    it applies the subscriber-mask (e.g., 56) on
    the source IPv6 address to compute the associated
    prefix for this client (2001:db8:100:100::/56).
    Then, the NAT64 enforces policies based on that
    prefix (2001:db8:100:100::/56), not on the exact
    source IPv6 address."
}

list subscriber-match {
  if-feature "basic-nat44 or napt44 or dst-nat";
  key match-id;

  description
    "IP prefix match.
    A subscriber is identified by a subnet."

  leaf match-id {
    type uint32;
    description
      "An identifier of the subscriber match."
  }

  leaf subnet {
    type inet:ip-prefix;
```



```
    mandatory true;
    description
      "The IP address subnets that match
      should be translated. E.g., all addresses
      that belong to the 192.0.2.0/24 prefix must
      be processed by the NAT.";
  }
}

leaf address-allocation-type {
  type enumeration {
    enum "arbitrary" {
      if-feature "basic-nat44 or napt44 or nat64";
      description
        "Arbitrary pooling behavior means that the NAT
        instance may create the new port mapping using any
        address in the pool that has a free port for the
        protocol concerned.";
    }

    enum "roundrobin" {
      if-feature "basic-nat44 or napt44 or nat64";
      description
        "Round robin allocation.";
    }

    enum "paired" {
      if-feature "napt44 or nat64";
      description
        "Paired address pooling informs the NAT
        that all the flows from an internal IP
        address must be assigned the same external
        address. This is the recommended behavior for
        NAPT/NAT64.";
      reference
        "RFC 4787: Network Address Translation (NAT)
        Behavioral Requirements for Unicast UDP";
    }
  }
  description
    "Specifies how external IP addresses are allocated.";
}

leaf port-allocation-type {
  if-feature "napt44 or nat64";
  type enumeration {
    enum "random" {
      description
```





```
        "Port randomization is enabled. A NAT port allocation
        scheme should make it hard for attackers to guess
        port numbers";
    reference
        "REQ-15 of RFC 6888";
}

enum "port-preservation" {
    description
        "Indicates whether the NAT should preserve the internal
        port number.";
}

enum "port-parity-preservation" {
    description
        "Indicates whether the NAT should preserve the port
        parity of the internal port number.";
}

enum "port-range-allocation" {
    description
        "Indicates whether the NAT assigns a range of ports
        for an internal host. This scheme allows to minimize
        log volume.";
    reference
        "REQ-14 of RFC 6888";
}
}
description
    "Indicates the type of port allocation.";
}

leaf mapping-type {
    if-feature "napt44 or nat64";
    type enumeration {
        enum "eim" {
            description
                "endpoint-independent-mapping.";
            reference
                "Section 4 of RFC 4787.";
        }

        enum "adm" {
            description
                "address-dependent-mapping.";
            reference
                "Section 4 of RFC 4787.";
        }
    }
}
```



```
    enum "edm" {
      description
        "address-and-port-dependent-mapping.";
      reference
        "Section 4 of RFC 4787.";
    }
  }
  description
    "Indicates the type of a NAT mapping.";
}

leaf filtering-type {
  if-feature "napt44 or nat64";
  type enumeration {
    enum "eif" {
      description
        "endpoint-independent-filtering.";
      reference
        "Section 5 of RFC 4787.";
    }

    enum "adf" {
      description
        "address-dependent-filtering.";
      reference
        "Section 5 of RFC 4787.";
    }

    enum "edf" {
      description
        "address-and-port-dependent-filtering";
      reference
        "Section 5 of RFC 4787.";
    }
  }
  description
    "Indicates the type of a NAT filtering.";
}

leaf fragment-behavior {
  if-feature "napt44 or nat64";
  type enumeration {
    enum "drop-all" {
      description
        "All received fragments are dropped.";
    }

    enum "in-order" {
```



```
    description
      "Translate fragments only if they are received
       in order.";
  }

  enum "out-of-order" {
    description
      "Translate a fragment even if it is received out
       of order.

       This behavior is the recommended behavior.";
    reference
      "REQ-14 of RFC 4787";
  }
}
description
  "The fragment behavior instructs the NAT about the
   behavior to follow to translate fragments received
   on the external interface of the NAT.";
}

list port-quota {
  if-feature "napt44 or nat64";
  key quota-type;
  description
    "Configures a port quota to be assigned per subscriber.
     It corresponds to the maximum number of ports to be
     used by a subscriber.";

  leaf port-limit {
    type uint16;
    description
      "Configures a port quota to be assigned per subscriber.
       It corresponds to the maximum number of ports to be
       used by a subscriber.";
    reference
      "REQ-4 of RFC 6888.";
  }

  leaf quota-type {
    type uint8;
    description
      "Indicates whether the port quota applies to
       all protocols (0) or to a specific protocol.";
  }
}

container port-set {
```



```
when "../port-allocation-type = 'port-range-allocation'";

if-feature "napt44 or nat64";
description
  "Manages port-set assignments.";

leaf port-set-size {
  type uint16;
  mandatory true;
  description
    "Indicates the size of assigned port sets.";
}

leaf port-set-timeout {
  type uint32;
  units "seconds";
  description
    "inactivity timeout for port sets.";
}
}

container timers {
  if-feature "napt44 or nat64";
  description
    "Configure values of various timeouts.";

  leaf udp-timeout {
    type uint32;
    units "seconds";
    default 300;
    description
      "UDP inactivity timeout. That is the time a mapping
       will stay active without packets traversing the NAT.";
    reference
      "RFC 4787: Network Address Translation (NAT)
       Behavioral Requirements for Unicast UDP";
  }

  leaf tcp-idle-timeout {
    type uint32;
    units "seconds";
    default 7440;
    description
      "TCP Idle timeout should be 2 hours and 4 minutes.";
    reference
      "RFC 5382: NAT Behavioral Requirements for TCP";
  }
}
```





```
leaf tcp-trans-open-timeout {
  type uint32;
  units "seconds";
  default 240;
  description
    "The value of the transitory open connection
    idle-timeout.

    Section 2.1 of \[RFC7857\] clarifies that a NAT
    should provide different configurable
    parameters for configuring the open and
    closing idle timeouts.

    To accommodate deployments that consider
    a partially open timeout of 4 minutes as being
    excessive from a security standpoint, a NAT may
    allow the configured timeout to be less than
    4 minutes.

    However, a minimum default transitory connection
    idle-timeout of 4 minutes is recommended.";
  reference
    "Section 2.1 of RFC 7857.";
}

leaf tcp-trans-close-timeout {
  type uint32;
  units "seconds";
  default 240;
  description
    "The value of the transitory close connection
    idle-timeout.

    Section 2.1 of \[RFC7857\] clarifies that a NAT
    should provide different configurable
    parameters for configuring the open and
    closing idle timeouts.";
  reference
    "Section 2.1 of RFC 7857.";
}

leaf tcp-in-syn-timeout {
  type uint32;
  units "seconds";
  default 6;
  description
    "A NAT must not respond to an unsolicited
    inbound SYN packet for at least 6 seconds
```



```
        after the packet is received.  If during
        this interval the NAT receives and translates
        an outbound SYN for the connection the NAT
        must silently drop the original unsolicited
        inbound SYN packet.";
    reference
        "RFC 5382 NAT Behavioral Requirements for TCP";
}

leaf fragment-min-timeout {
    when "../..//fragment-behavior='out-of-order'";
    type uint32;
    units "seconds";
    default 2;
    description
        "As long as the NAT has available resources,
        the NAT allows the fragments to arrive
        over fragment-min-timeout interval.
        The default value is inspired from RFC6146.";
}

leaf icmp-timeout {
    type uint32;
    units "seconds";
    default 60;
    description
        "An ICMP Query session timer must not expire
        in less than 60 seconds. It is recommended
        that the ICMP Query session timer be made
        configurable";
    reference
        "RFC 5508: NAT Behavioral Requirements for ICMP";
}

list per-port-timeout {
    key port-number;
    description
        "Some NATs are configurable with short timeouts
        for some ports, e.g., as 10 seconds on
        port 53 (DNS) and NTP (123) and longer timeouts
        on other ports.";

    leaf port-number {
        type inet:port-number;
        description
            "A port number.";
    }
}
```



```
leaf timeout {
  type uint32;
  units "seconds";
  mandatory true;
  description
    "Timeout for this port number";
}
}
```

```
leaf hold-down-timeout {
  type uint32;
  units "seconds";
  default 120;
  description
    "Hold down timer.

    Ports in the hold down pool are not reassigned until
    hold-down-timeout expires.

    The length of time and the maximum number of ports in
    this state must be configurable by the administrator.

    This is necessary in order to prevent collisions
    between old and new mappings and sessions. It ensures
    that all established sessions are broken instead of
    redirected to a different peer.";
  reference
    "REQ#8 of RFC 6888.";
}
```

```
leaf hold-down-max {
  type uint32;
  description
    "Maximum ports in the Hold down timer pool.

    Ports in the hold down pool are not reassigned
    until hold-down-timeout expires.

    The length of time and the maximum
    number of ports in this state must be
    configurable by the administrator.
    This is necessary in order
    to prevent collisions between old
    and new mappings and sessions. It ensures
    that all established sessions are broken
    instead of redirected to a different peer.";
  reference
    "REQ#8 of RFC 6888.";
}
```



```
    }  
  }  
  
  leaf fragments-limit{  
    when "../fragment-behavior='out-of-order'";  
    type uint32;  
    description  
      "Limits the number of out of order fragments that can  
       be handled.";  
    reference  
      "Section 11 of RFC 4787.";  
  }  
  
  list algs {  
    key name;  
    description  
      "ALG-related features.";  
  
    leaf name {  
      type string;  
      description  
        "The name of the ALG.";  
    }  
  
    leaf transport-protocol {  
      type uint32;  
      description  
        "The transport protocol used by the ALG  
         (e.g., TCP, UDP).";  
    }  
  
    container dst-transport-port {  
      uses port-number;  
      description  
        "The destination port number(s) used by the ALG.  
         For example,  
         - 21 for the FTP ALG  
         - 53 for the DNS ALG.";  
    }  
  
    container src-transport-port {  
      uses port-number;  
      description  
        "The source port number(s) used by the ALG.";  
    }  
  
    leaf status {  
      type boolean;
```





```
        description
            "Enable/disable the ALG.";
    }
}

leaf all-algs-enable {
    type boolean;
    description
        "Enable/disable all ALGs.

        When specified, this parameter overrides the one
        that may be indicated, eventually, by the 'status'
        of an individual ALG.";
}

container notify-pool-usage {
    if-feature "basic-nat44 or napt44 or nat64";
    description
        "Notification of pool usage when certain criteria
        are met.";

    leaf pool-id {
        type uint32;
        description
            "Pool-ID for which the notification criteria
            is defined";
    }

    leaf high-threshold {
        type percent;
        description
            "Notification must be generated when the defined high
            threshold is reached.

            For example, if a notification is required when the
            pool utilization reaches 90%, this configuration
            parameter must be set to 90.

            0% indicates that no high threshold is enabled.";
    }

    leaf low-threshold {
        type percent;
        must ". >= ../high-threshold" {
            error-message
                "The upper port number must be greater than or
                equal to lower port number.";
        }
    }
}
```



```
    description
      "Notification must be generated when the defined low
       threshold is reached.

       For example, if a notification is required when the
       pool utilization reaches below 10%, this
       configuration parameter must be set to 10";
  }

  leaf notify-interval {
    type uint32 {
      range "1 .. 3600";
    }
    units "seconds";
    default '20';
    description
      "Minimum number of seconds between successive
       notifications for this pool.";

    reference
      "RFC 7659: Definitions of Managed Objects for
       Network Address Translators (NATs)";
  }
}

container external-realm {
  description
    "Identifies the external realm of the NAT instance.";

  choice realm-type {
    description
      "Can be an interface, VRF instance, etc.";

    case interface {
      description
        "External interface.";

      leaf external-interface {
        type if:interface-ref;
        description
          "Name of the external interface.";
      }
    }
  }
}

container mapping-limits {
```



```
if-feature "napt44 or nat64";
description
  "Information about the configuration parameters that
   limits the mappings based upon various criteria.";

leaf limit-subscribers {
  type uint32;
  description
    "Maximum number of subscribers that can be serviced
     by a NAT instance.

     A subscriber is identified by a given prefix.";
  reference
    "RFC 7659: Definitions of Managed Objects for
     Network Address Translators (NATs)";
}

leaf limit-address-mappings {
  type uint32;
  description
    "Maximum number of address mappings that can be
     handled by a NAT instance.

     When this limit is reached, packets that would
     normally trigger translation, will be dropped.";
  reference
    "RFC 7659: Definitions of Managed Objects
     for Network Address Translators
     (NATs)";
}

leaf limit-port-mappings {
  type uint32;
  description
    "Maximum number of port mappings that can be handled
     by a NAT instance.

     When this limit is reached, packets that would
     normally trigger translation, will be dropped.";
  reference
    "RFC 7659: Definitions of Managed Objects for
     Network Address Translators (NATs)";
}

list limit-per-protocol {
  if-feature "napt44 or nat64 or dst-nat";
  key protocol-id;
```



```
    description
      "Configure limits per transport protocol";

    leaf protocol-id {
      type uint8;
      mandatory true;
      description
        "Upper-layer protocol associated with this mapping.

        Values are taken from the IANA protocol registry:
        https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml

        For example, this field contains 6 (TCP) for a TCP
        mapping or 17 (UDP) for a UDP mapping.";
    }

    leaf limit {
      type uint32;
      description
        "Maximum number of protocol-specific NAT mappings
        per instance.";
    }
  }
}

container connection-limits {
  if-feature "basic-nat44 or napt44 or nat64";
  description
    "Information about the configuration parameters that
    rate limit the translation based upon various criteria.";

  leaf limit-per-subscriber {
    type uint32;
    units "bits/second";
    description
      "Rate-limit the number of new mappings and sessions
      per subscriber.";
  }

  leaf limit-per-instance {
    type uint32;
    units "bits/second";
    mandatory true;
    description
      "Rate-limit the number of new mappings and sessions
      per instance.";
  }
}
```





```
list limit-per-protocol {
  if-feature "napt44 or nat64";
  key protocol-id;
  description
    "Configure limits per transport protocol";

  leaf protocol-id {
    type uint8;
    mandatory true;
    description
      "Upper-layer protocol associated with this mapping.

      Values are taken from the IANA protocol registry:
      https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml

      For example, this field contains 6 (TCP) for a TCP
      mapping or 17 (UDP) for a UDP mapping.";
  }

  leaf limit {
    type uint32;
    description
      "Rate-limit the number of protocol-specific mappings
      and sessions per instance.";
  }
}

container notification-limits {
  description "Sets notification limits.";

  leaf notify-interval {
    if-feature "basic-nat44 or napt44 or nat64";
    type uint32 {
      range "1 .. 3600";
    }
    units "seconds";
    default '10';
    description
      "Minimum number of seconds between successive
      notifications for this NAT instance.";
    reference
      "RFC 7659: Definitions of Managed Objects
      for Network Address Translators (NATs)";
  }

  leaf notify-addresses-usage {
```



```
    if-feature "basic-nat44 or napt44 or nat64";
    type percent;
    description
      "Notification of address mappings usage over
       the whole NAT instance.

       Notification must be generated when the defined
       threshold is reached.

       For example, if a notification is required when
       the address mappings utilization reaches 90%,
       this configuration parameter must be set
       to 90.";
  }

  leaf notify-ports-usage {
    if-feature "napt44 or nat64";
    type percent;
    description
      "Notification of port mappings usage over the
       whole NAT instance.

       Notification must be generated when the defined
       threshold is reached.

       For example, if a notification is required when
       the port mappings utilization reaches 90%, this
       configuration parameter must be set to 90.";
  }
}

leaf logging-enable {
  if-feature "basic-nat44 or napt44 or nat64";
  type boolean;
  description
    "Enable logging features.";
  reference
    "Section 2.3 of RFC 6908 and REQ-12 of RFC6888.";
}

container mapping-table {
  if-feature "basic-nat44 or napt44 " +
    "or nat64 or clat or dst-nat";
  description
    "NAT mapping table. Applicable for functions which maintain
     static and/or dynamic mappings, such as NAT44, Destination
     NAT, NAT64, or CLAT.";
```



```
list mapping-entry {
  key "index";
  description "NAT mapping entry.";
  uses mapping-entry;
}

container statistics {
  config false;

  description
    "Statistics related to the NAT instance.";

  leaf discontinuity-time {
    type yang:date-and-time;
    mandatory true;
    description
      "The time on the most recent occasion at which the NAT
      instance suffered a discontinuity. This must be
      initialized when the NAT instance is configured
      or rebooted.";
  }

  container traffic-statistics {
    description
      "Generic traffic statistics.";

    leaf sent-packets {
      type yang:zero-based-counter64;
      description
        "Number of packets sent.";
    }

    leaf sent-bytes {
      type yang:zero-based-counter64;
      units 'bytes';
      description
        "Counter for sent traffic in bytes.";
    }

    leaf rcvd-packets {
      type yang:zero-based-counter64;
      description
        "Number of received packets.";
    }

    leaf rcvd-bytes {
      type yang:zero-based-counter64;
```



```
    units 'bytes';
    description
      "Counter for received traffic in bytes.";
  }

  leaf dropped-packets {
    type yang:zero-based-counter64;
    description
      "Number of dropped packets.";
  }

  leaf dropped-bytes {
    type yang:zero-based-counter64;
    units 'bytes';
    description
      "Counter for dropped traffic in bytes.";
  }

  leaf dropped-fragments {
    if-feature "napt44 or nat64";
    type yang:zero-based-counter64;
    description
      "Number of dropped fragments on the external realm.";
  }

  leaf dropped-address-limit-packets {
    if-feature "basic-nat44 or napt44 or nat64";
    type yang:zero-based-counter64;
    description
      "Number of dropped packets because an address limit
        is reached.";
  }

  leaf dropped-address-limit-bytes {
    if-feature "basic-nat44 or napt44 or nat64";
    type yang:zero-based-counter64;
    units 'bytes';
    description
      "Counter of dropped packets because an address limit
        is reached, in bytes.";
  }

  leaf dropped-address-packets {
    if-feature "basic-nat44 or napt44 or nat64";
    type yang:zero-based-counter64;
    description
      "Number of dropped packets because no address is
        available for allocation.";
```





```
}

leaf dropped-address-bytes {
  if-feature "basic-nat44 or napt44 or nat64";
  type yang:zero-based-counter64;
  units 'bytes';
  description
    "Counter of dropped packets because no address is
    available for allocation, in bytes.";
}

leaf dropped-port-limit-packets {
  if-feature "napt44 or nat64";
  type yang:zero-based-counter64;
  description
    "Number of dropped packets because a port limit
    is reached.";
}

leaf dropped-port-limit-bytes {
  if-feature "napt44 or nat64";
  type yang:zero-based-counter64;
  units 'bytes';
  description
    "Counter of dropped packets because a port limit
    is reached, in bytes.";
}

leaf dropped-port-packets {
  if-feature "napt44 or nat64";
  type yang:zero-based-counter64;
  description
    "Number of dropped packets because no port is
    available for allocation.";
}

leaf dropped-port-bytes {
  if-feature "napt44 or nat64";
  type yang:zero-based-counter64;
  units 'bytes';
  description
    "Counter of dropped packets because no port is
    available for allocation, in bytes.";
}

leaf dropped-subscriber-packets {
  if-feature "basic-nat44 or napt44 or nat64";
  type yang:zero-based-counter64;
```



```
    description
      "Number of dropped packets because the subscriber
        limit per instance is reached.";
  }

  leaf dropped-subscriber-bytes {
    if-feature "basic-nat44 or napt44 or nat64";
    type yang:zero-based-counter64;
    units 'bytes';
    description
      "Counter of dropped packets because the subscriber
        limit per instance is reached, in bytes.";
  }
}

container mappings-statistics {
  description
    "Mappings statistics.";

  leaf total-address-mappings {
    if-feature "basic-nat44 or napt44 " +
      "or nat64 or clat or dst-nat";
    type yang:gauge32;
    description
      "Total number of address mappings present at a given
        time. It includes both static and dynamic mappings.";
    reference
      "Section 3.3.8 of RFC 7659";
  }

  leaf total-port-mappings {
    if-feature "napt44 or nat64";
    type yang:gauge32;
    description
      "Total number of NAT port mappings present at
        a given time. It includes both static and dynamic
        mappings.";
    reference
      "Section 3.3.9 of RFC 7659";
  }
}

list total-per-protocol {
  if-feature "napt44 or nat64";
  key protocol-id;
  description
    "Total mappings for each enabled/supported protocol.";

  leaf protocol-id {
```



```
    type uint8;
    mandatory true;
    description
      "Upper-layer protocol associated with this mapping.
       For example, this field contains 6 (TCP) for a TCP
       mapping or 17 (UDP) for a UDP mapping.";
  }

  leaf total {
    type yang:gauge32;
    description
      "Total number of a protocol-specific mappings present
       at a given time. The protocol is identified by
       protocol-id.";
  }
}

container pools-stats {
  if-feature "basic-nat44 or napt44 or nat64";
  description
    "Statistics related to address/prefix pools
     usage";

  leaf addresses-allocated {
    type yang:gauge32;
    description
      "Number of all allocated addresses.";
  }

  leaf addresses-free {
    type yang:gauge32;
    description
      "Number of unallocated addresses of all pools at
       a given time. The sum of unallocated and allocated
       addresses is the total number of addresses of
       the pools.";
  }
}

container ports-stats {
  if-feature "napt44 or nat64";

  description
    "Statistics related to port numbers usage.";

  leaf ports-allocated {
    type yang:gauge32;
    description
```



```
        "Number of allocated ports from all pools.";
    }

    leaf ports-free {
        type yang:gauge32;
        description
            "Number of unallocated addresses from all pools.";
    }
}

list per-pool-stats {
    if-feature "basic-nat44 or napt44 or nat64";
    key "pool-id";
    description
        "Statistics related to address/prefix pool usage";

    leaf pool-id {
        type uint32;
        description
            "Unique Identifier that represents a pool of
            addresses/prefixes.";
    }

    leaf discontinuity-time {
        type yang:date-and-time;
        mandatory true;
        description
            "The time on the most recent occasion at which this
            pool counters suffered a discontinuity. This must
            be initialized when the address pool is
            configured.";
    }
}

container pool-stats {
    description
        "Statistics related to address/prefix pool usage";

    leaf addresses-allocated {
        type yang:gauge32;
        description
            "Number of allocated addresses from this pool.";
    }

    leaf addresses-free {
        type yang:gauge32;
        description
            "Number of unallocated addresses in this pool.";
    }
}
```





```
    }

    container port-stats {
      if-feature "napt44 or nat64";
      description
        "Statistics related to port numbers usage.";

      leaf ports-allocated {
        type yang:gauge32;
        description
          "Number of allocated ports from this pool.";
      }

      leaf ports-free {
        type yang:gauge32;
        description
          "Number of unallocated addresses from this pool.";
      }
    }
  }
}

/*
 * Notifications
 */

notification nat-pool-event {
  if-feature "basic-nat44 or napt44 or nat64";
  description
    "Notifications must be generated when the defined high/low
    threshold is reached. Related configuration parameters
    must be provided to trigger the notifications.";

  leaf id {
    type leafref {
      path "/nat/instances/instance/id";
    }
    mandatory true;
    description
      "NAT instance Identifier.";
  }

  leaf policy-id {
    type leafref {
```



```
    path "/nat/instances/instance/policy/id";
  }

  description
    "Policy Identifier.";
}

leaf pool-id {
  type leafref {
    path
      "/nat/instances/instance/policy/"
      + "external-ip-address-pool/pool-id";
  }
  mandatory true;
  description
    "Pool Identifier.";
}

leaf notify-pool-threshold {
  type percent;
  mandatory true;
  description
    "A threshold (high-threshold or low-threshold) has
    been fired.";
}
}

notification nat-instance-event {
  if-feature "basic-nat44 or napt44 or nat64";
  description
    "Notifications must be generated when notify-addresses-usage
    and/or notify-ports-usagethreshold are reached.";

  leaf id {
    type leafref {
      path "/nat/instances/instance/id";
    }
    mandatory true;
    description
      "NAT instance Identifier.";
  }

  leaf notify-addresses-threshold {
    type percent;
    description
      "The notify-addresses-usage threshold has been fired.";
  }
}
```



```
    leaf notify-ports-threshold {  
      type percent;  
      description  
        "The notify-ports-usage threshold has been fired.";  
    }  
  }  
}  
<CODE ENDS>
```

#### 4. Security Considerations

Security considerations related to address and prefix translation are discussed in [\[RFC6888\]](#), [\[RFC6146\]](#), [\[RFC6877\]](#), [\[RFC7757\]](#), and [\[RFC6296\]](#).

The YANG module defined in this document is designed to be accessed via network management protocols such as NETCONF [\[RFC6241\]](#) or RESTCONF [\[RFC8040\]](#). The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [\[RFC6242\]](#). The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [\[RFC5246\]](#).

The NETCONF access control model [\[RFC6536\]](#) provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

All data nodes defined in the YANG module which can be created, modified and deleted (i.e., config true, which is the default) are considered sensitive. Write operations (e.g., edit-config) applied to these data nodes without proper protection can negatively affect network operations. The NAT YANG module allows to set parameters to prevent a user from aggressively using NAT resources (port-quota), rate-limit connections as a guard against Denial-of-Service, or to enable notifications so that appropriate measures are enforced to anticipate traffic drops. Nevertheless, an attacker who is able to access to the NAT can undertake various attacks, such as:

- o Set a high or low resource limit to cause a DoS attack:

- \* /nat/instances/instance/policy/port-quota
- \* /nat/instances/instance/policy/fragments-limit
- \* /nat/instances/instance/mapping-limits
- \* /nat/instances/instance/connection-limits



- o Set a low notification threshold to cause useless notifications to be generated:
  - \* /nat/instances/instance/policy/notify-pool-usage/high-threshold
  - \* /nat/instances/instance/notification-limits/notify-addresses-usage
  - \* /nat/instances/instance/notification-limits/notify-ports-usage
- o Set an arbitrarily high threshold, which may lead to the deactivation of notifications:
  - \* /nat/instances/instance/policy/notify-pool-usage/high-threshold
  - \* /nat/instances/instance/notification-limits/notify-addresses-usage
  - \* /nat/instances/instance/notification-limits/notify-ports-usage
- o Set a low notification interval and a low notification threshold to induce useless notifications to be generated:
  - \* /nat/instances/instance/policy/notify-pool-usage/notify-interval
  - \* /nat/instances/instance/notification-limits/notify-interval
- o Access to privacy data maintained in the mapping table. Such data can be misused to track the activity of a host:
  - \* /nat/instances/instance/mapping-table

## 5. IANA Considerations

This document requests IANA to register the following URI in the "IETF XML Registry" [[RFC3688](#)]:

URI: urn:ietf:params:xml:ns:yang:ietf-nat  
Registrant Contact: The IESG.  
XML: N/A; the requested URI is an XML namespace.

This document requests IANA to register the following YANG module in the "YANG Module Names" registry [[RFC7950](#)].





```
name: ietf-nat
namespace: urn:ietf:params:xml:ns:yang:ietf-nat
prefix: nat
reference: RFC XXXX
```

## **6. Acknowledgements**

Many thanks to Dan Wing and Tianran Zhou for the review.

Thanks to Juergen Schoenwaelder for the comments on the YANG structure and the suggestion to use NMDA.

Thanks to Lee Howard and Jordi Palet for the CLAT comments, Fred Baker for the NPTv6 comments, Tore Anderson for EAM SIIT review, and Kristian Poscic for the CGN review.

Special thanks to Maros Marsalek and Marek Gradzki for sharing their comments based on the FD.io implementation of an earlier version of this module.

Rajiv Asati suggested to clarify how the module applies for both stateless and stateful NAT64.

Juergen Schoenwaelder provided an early yangdoctors review. Many thanks to him.

## **7. References**

### **7.1. Normative References**

- [RFC3688] Mealling, M., "The IETF XML Registry", [BCP 81](#), [RFC 3688](#), DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC4787] Audet, F., Ed. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", [BCP 127](#), [RFC 4787](#), DOI 10.17487/RFC4787, January 2007, <<https://www.rfc-editor.org/info/rfc4787>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", [RFC 5246](#), DOI 10.17487/RFC5246, August 2008, <<https://www.rfc-editor.org/info/rfc5246>>.
- [RFC5382] Guha, S., Ed., Biswas, K., Ford, B., Sivakumar, S., and P. Srisuresh, "NAT Behavioral Requirements for TCP", [BCP 142](#), [RFC 5382](#), DOI 10.17487/RFC5382, October 2008, <<https://www.rfc-editor.org/info/rfc5382>>.



- [RFC5508] Srisuresh, P., Ford, B., Sivakumar, S., and S. Guha, "NAT Behavioral Requirements for ICMP", [BCP 148](#), [RFC 5508](#), DOI 10.17487/RFC5508, April 2009, <<https://www.rfc-editor.org/info/rfc5508>>.
- [RFC6052] Bao, C., Huitema, C., Bagnulo, M., Boucadair, M., and X. Li, "IPv6 Addressing of IPv4/IPv6 Translators", [RFC 6052](#), DOI 10.17487/RFC6052, October 2010, <<https://www.rfc-editor.org/info/rfc6052>>.
- [RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", [RFC 6146](#), DOI 10.17487/RFC6146, April 2011, <<https://www.rfc-editor.org/info/rfc6146>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", [RFC 6241](#), DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", [RFC 6242](#), DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC6296] Wasserman, M. and F. Baker, "IPv6-to-IPv6 Network Prefix Translation", [RFC 6296](#), DOI 10.17487/RFC6296, June 2011, <<https://www.rfc-editor.org/info/rfc6296>>.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", [RFC 6536](#), DOI 10.17487/RFC6536, March 2012, <<https://www.rfc-editor.org/info/rfc6536>>.
- [RFC6877] Mawatari, M., Kawashima, M., and C. Byrne, "464XLAT: Combination of Stateful and Stateless Translation", [RFC 6877](#), DOI 10.17487/RFC6877, April 2013, <<https://www.rfc-editor.org/info/rfc6877>>.
- [RFC6888] Perreault, S., Ed., Yamagata, I., Miyakawa, S., Nakagawa, A., and H. Ashida, "Common Requirements for Carrier-Grade NATs (CGNs)", [BCP 127](#), [RFC 6888](#), DOI 10.17487/RFC6888, April 2013, <<https://www.rfc-editor.org/info/rfc6888>>.
- [RFC7596] Cui, Y., Sun, Q., Boucadair, M., Tsou, T., Lee, Y., and I. Farrer, "Lightweight 4over6: An Extension to the Dual-Stack Lite Architecture", [RFC 7596](#), DOI 10.17487/RFC7596, July 2015, <<https://www.rfc-editor.org/info/rfc7596>>.



- [RFC7597] Troan, O., Ed., Dec, W., Li, X., Bao, C., Matsushima, S., Murakami, T., and T. Taylor, Ed., "Mapping of Address and Port with Encapsulation (MAP-E)", [RFC 7597](#), DOI 10.17487/RFC7597, July 2015, <<https://www.rfc-editor.org/info/rfc7597>>.
- [RFC7757] Anderson, T. and A. Leiva Popper, "Explicit Address Mappings for Stateless IP/ICMP Translation", [RFC 7757](#), DOI 10.17487/RFC7757, February 2016, <<https://www.rfc-editor.org/info/rfc7757>>.
- [RFC7857] Penno, R., Perreault, S., Boucadair, M., Ed., Sivakumar, S., and K. Naito, "Updates to Network Address Translation (NAT) Behavioral Requirements", [BCP 127](#), [RFC 7857](#), DOI 10.17487/RFC7857, April 2016, <<https://www.rfc-editor.org/info/rfc7857>>.
- [RFC7915] Bao, C., Li, X., Baker, F., Anderson, T., and F. Gont, "IP/ICMP Translation Algorithm", [RFC 7915](#), DOI 10.17487/RFC7915, June 2016, <<https://www.rfc-editor.org/info/rfc7915>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", [RFC 7950](#), DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", [RFC 8040](#), DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.

## **7.2. Informative References**

- [I-D.boucadair-pcp-yang]  
Boucadair, M., Jacquenet, C., Sivakumar, S., and S. Vinapamula, "YANG Modules for the Port Control Protocol (PCP)", [draft-boucadair-pcp-yang-05](#) (work in progress), October 2017.
- [I-D.ietf-softwire-dslite-yang]  
Boucadair, M., Jacquenet, C., and S. Sivakumar, "YANG Data Modules for the DS-Lite", [draft-ietf-softwire-dslite-yang-07](#) (work in progress), October 2017.
- [I-D.ietf-tsvwg-natsupp]  
Stewart, R., Tuexen, M., and I. Ruengeler, "Stream Control Transmission Protocol (SCTP) Network Address Translation Support", [draft-ietf-tsvwg-natsupp-11](#) (work in progress), July 2017.



- [RFC2663] Srisuresh, P. and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", [RFC 2663](#), DOI 10.17487/RFC2663, August 1999, <<https://www.rfc-editor.org/info/rfc2663>>.
- [RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", [RFC 3022](#), DOI 10.17487/RFC3022, January 2001, <<https://www.rfc-editor.org/info/rfc3022>>.
- [RFC5597] Denis-Courmont, R., "Network Address Translation (NAT) Behavioral Requirements for the Datagram Congestion Control Protocol", [BCP 150](#), [RFC 5597](#), DOI 10.17487/RFC5597, September 2009, <<https://www.rfc-editor.org/info/rfc5597>>.
- [RFC6269] Ford, M., Ed., Boucadair, M., Durand, A., Levis, P., and P. Roberts, "Issues with IP Address Sharing", [RFC 6269](#), DOI 10.17487/RFC6269, June 2011, <<https://www.rfc-editor.org/info/rfc6269>>.
- [RFC6736] Brockners, F., Bhandari, S., Singh, V., and V. Fajardo, "Diameter Network Address and Port Translation Control Application", [RFC 6736](#), DOI 10.17487/RFC6736, October 2012, <<https://www.rfc-editor.org/info/rfc6736>>.
- [RFC6887] Wing, D., Ed., Cheshire, S., Boucadair, M., Penno, R., and P. Selkirk, "Port Control Protocol (PCP)", [RFC 6887](#), DOI 10.17487/RFC6887, April 2013, <<https://www.rfc-editor.org/info/rfc6887>>.
- [RFC7289] Kuarsingh, V., Ed. and J. Cianfarani, "Carrier-Grade NAT (CGN) Deployment with BGP/MPLS IP VPNs", [RFC 7289](#), DOI 10.17487/RFC7289, June 2014, <<https://www.rfc-editor.org/info/rfc7289>>.
- [RFC7335] Byrne, C., "IPv4 Service Continuity Prefix", [RFC 7335](#), DOI 10.17487/RFC7335, August 2014, <<https://www.rfc-editor.org/info/rfc7335>>.
- [RFC7659] Perreault, S., Tsou, T., Sivakumar, S., and T. Taylor, "Definitions of Managed Objects for Network Address Translators (NATs)", [RFC 7659](#), DOI 10.17487/RFC7659, October 2015, <<https://www.rfc-editor.org/info/rfc7659>>.





- [RFC7753] Sun, Q., Boucadair, M., Sivakumar, S., Zhou, C., Tsou, T., and S. Perreault, "Port Control Protocol (PCP) Extension for Port-Set Allocation", [RFC 7753](#), DOI 10.17487/RFC7753, February 2016, <<https://www.rfc-editor.org/info/rfc7753>>.
- [RFC8045] Cheng, D., Korhonen, J., Boucadair, M., and S. Sivakumar, "RADIUS Extensions for IP Port Configuration and Reporting", [RFC 8045](#), DOI 10.17487/RFC8045, January 2017, <<https://www.rfc-editor.org/info/rfc8045>>.

## **Appendix A. Sample Examples**

This section provides a non-exhaustive set of examples to illustrate the use of the NAT YANG module.

### **A.1. Traditional NAT44**

Traditional NAT44 is a Basic NAT44 or NAPT that is used to share the same IPv4 address among hosts that are owned by the same subscriber. This is typically the NAT that is embedded in CPE devices.

This NAT is usually provided with one single external IPv4 address; disambiguating connections is achieved by rewriting the source port number. The XML snippet to configure the external IPv4 address in such case together with a mapping entry is depicted below:

```
<instances>
  <instance>
    <id>1</id>
    <name>NAT_Subscriber_A</name>
    ....
    <external-ip-address-pool>
      <pool-id>1</pool-id>
      <external-ip-pool>
        192.0.2.1
      </external-ip-pool>
    </external-ip-address-pool>
    ....
    <mapping-table>
      ....
      <external-src-address>
        192.0.2.1
      </external-src-address>
      ....
    </mapping-table>
  </instance>
</instances>
```



The following shows the XML excerpt depicting a dynamic UDP mapping entry maintained by a traditional NAPT44. In reference to this example, the UDP packet received with a source IPv4 address (192.0.2.1) and source port number (1568) is translated into a UDP packet having a source IPv4 address (198.51.100.1) and source port (15000). The remaining lifetime of this mapping is 300 seconds.

```
<mapping-entry>
  <index>15</index>
  <type>
    dynamic-explicit
  </type>
  <transport-protocol>
    17
  </transport-protocol>
  <internal-src-address>
    192.0.2.1
  </internal-src-address>
  <internal-src-port>
    <start-port-number>
      1568
    </start-port-number>
  </internal-src-port>
  <external-src-address>
    198.51.100.1
  </external-src-address>
  <external-src-port>
    <start-port-number>
      15000
    </start-port-number>
  </external-src-port>
  <lifetime>
    300
  </lifetime>
</mapping-entry>
```

## **[A.2.](#) Carrier Grade NAT (CGN)**

The following XML snippet shows the example of the capabilities supported by a CGN as retrieved using NETCONF.



```
<capabilities
  <nat-flavor>
    napt44
  </nat-flavor>
  <restricted-port-support>
    false
  </restricted-port-support>
  <static-mapping-support>
    true
  </static-mapping-support>
  <port-randomization-support>
    true
  </port-randomization-support>
  <port-range-allocation-support>
    true
  </port-range-allocation-support>
  <port-preservation-suport>
    true
  </port-preservation-suport>
  <port-parity-preservation-support>
    false
  </port-parity-preservation-support>
  <address-roundrobin-support>
    true
  </address-roundrobin-support>
  <paired-address-pooling-support>
    true
  </paired-address-pooling-support>
  <endpoint-independent-mapping-support>
    true
  </endpoint-independent-mapping-support>
  <address-dependent-mapping-support>
    false
  </address-dependent-mapping-support>
  <address-and-port-dependent-mapping-support>
    false
  </address-and-port-dependent-mapping-support>
  <endpoint-independent-filtering-support>
    true
  </endpoint-independent-filtering-support>
  <address-dependent-filtering>
    false
  </address-dependent-filtering>
  <address-and-port-dependent-filtering>
    false
  </address-and-port-dependent-filtering>
</capabilities>
```



The following XML snippet shows the example of a CGN that is provisioned with one contiguous pool of external IPv4 addresses (192.0.2.0/24). Further, the CGN is instructed to limit the number of allocated ports per subscriber to 1024. Ports can be allocated by the CGN by assigning ranges of 256 ports (that is, a subscriber can be allocated up to four port ranges of 256 ports each).

```
<instances>
  <instance>
    <id>1</id>
    <name>myCGN</name>
    ....
    <external-ip-address-pool>
      <pool-id>1</pool-id>
      <external-ip-pool>
        192.0.2.0/24
      </external-ip-pool>
    </external-ip-address-pool>
    <port-quota>
      <port-limit>
        1024
      </port-limit>
      <quota-type >
        all
      </quota-type >
    </port-quota>
    <port-allocation-type>
      port-range-allocation
    </port-allocation-type>
    <port-set>
      <port-set-size>
        256
      </port-set-size>
    </port-set>
    ....
  </instance>
</instances>
```

An administrator may decide to allocate one single port range per subscriber (port range of 1024 ports) as shown below:





```
<instances>
  <instance>
    <id>1</id>
    <name>myotherCGN</name>
    ....
    <external-ip-address-pool>
      <pool-id>1</pool-id>
      <external-ip-pool>
        192.0.2.0/24
      </external-ip-pool>
    </external-ip-address-pool>
    <port-quota>
      <port-limit>
        1024
      </port-limit>
      <quota-type >
        all
      </quota-type >
    </port-quota>
    <port-allocation-type>
      port-range-allocation
    </port-allocation-type>
    <port-set>
      <port-set-size>
        1024
      </port-set-size>
      ....
    </port-set>
    ....
  </instance>
</instances>
```

### A.3. CGN Pass-Through

Figure 1 illustrates an example of the CGN pass-through feature.

X1:x1		X1':x1'		X2:x2
+---+from X1:x1		+---+from X1:x1		+---+
C     to X2:x2		C     to X2:x2		S
l   >>>>>>>>>>		C   >>>>>>>>>>		e
i		G		r
e   <<<<<<<<<<		N   <<<<<<<<<<		v
n   from X2:x2		from X2:x2		e
t     to X1:x1		to X1:x1		r
+---+		+---+		+---+

Figure 1: CGN Pass-Through



For example, in order to disable NAT for communications issued by the client (192.0.2.25), the following configuration parameter must be set:

```
<nat-pass-through>
  ...
  <prefix>192.0.2.25</prefix>
  ...
</nat-pass-through>
```

#### [A.4.](#) NAT64

Let's consider the example of a NAT64 that should use 2001:db8:122:300::/56 to perform IPv6 address synthesis [[RFC6052](#)]. The XML snippet to configure the NAT64 prefix in such case is depicted below:

```
<nat64-prefixes>
  <nat64-prefix>
    2001:db8:122:300::/56
  </nat64-prefix>
</nat64-prefixes>
```

Let's now consider the example of a NAT64 that should use 2001:db8:122::/48 to perform IPv6 address synthesis [[RFC6052](#)] only if the destination address matches 198.51.100.0/24. The XML snippet to configure the NAT64 prefix in such case is shown below:

```
<nat64-prefixes>
  <nat64-prefix>
    2001:db8:122::/48
  </nat64-prefix>
  <destination-ipv4-prefix>
    <ipv4-prefix>
      198.51.100.0/24
    </ipv4-prefix>
  </destination-ipv4-prefix>
</nat64-prefixes>
```

#### [A.5.](#) Stateless IP/ICMP Translation (SIIT)

Let's consider the example of a stateless translator that is configured with 2001:db8:100::/40 to perform IPv6 address synthesis [[RFC6052](#)]. Similar to the NAT64 case, the XML snippet to configure the NAT64 prefix in such case is depicted below:



```
<nat64-prefixes>
  <nat64-prefix>
    2001:db8:100::/40
  </nat64-prefix>
</nat64-prefixes>
```

When the translator receives an IPv6 packet, for example, with a source address (2001:db8:1c0:2:21::) and destination address (2001:db8:1c6:3364:2::), it extracts embedded IPv4 addresses following [RFC6052](#) rules with 2001:db8:100::/40 as the NSP:

- o 192.0.2.33 is extracted from 2001:db8:1c0:2:21::
- o 198.51.100.2 is extracted from 2001:db8:1c6:3364:2::

The translator transforms the IPv6 header into an IPv4 header using the IP/ICMP Translation Algorithm [[RFC7915](#)]. The IPv4 packets will include 192.0.2.33 as the source address and 198.51.100.2 as the destination address.

Alos, a NAT64 can be instructed to behave in the stateless mode by providing the following configuration. The same NAT64 prefix is used for constructing both IPv4-translatable IPv6 addresses and IPv4-converted IPv6 addresses ([Section 3.3 of \[RFC6052\]](#)).

```
<nat64-prefixes>
  <nat64-prefix>
    2001:db8:122:300::/56
  </nat64-prefix>
  <stateless-enable>
    true
  </stateless-enable>
</nat64-prefixes>
```

#### **[A.6.](#) Explicit Address Mappings for Stateless IP/ICMP Translation (EAM SIIT)**

As specified in [[RFC7757](#)], an EAM consists of an IPv4 prefix and an IPv6 prefix. Let's consider the set of EAM examples in Figure 2.



+-----+-----+	
IPv4 Prefix	IPv6 Prefix
+-----+-----+	
192.0.2.1	2001:db8:aaaa::
192.0.2.2/32	2001:db8:bbbb::b/128
192.0.2.16/28	2001:db8:cccc::/124
192.0.2.128/26	2001:db8:dddd::/64
192.0.2.192/29	2001:db8:eeee:8::/62
192.0.2.224/31	64:ff9b::/127
+-----+-----+	

Figure 2: EAM Examples ([RFC7757](#))

The following XML excerpt illustrates how these EAMs can be configured using the YANG NAT module:





```
<eam>
  <ipv4-prefix>
    192.0.2.1
  </ipv4-prefix>
  <ipv6-prefix>
    2001:db8:aaaa::
  </ipv6-prefix>
</eam>
<eam>
  <ipv4-prefix>
    192.0.2.2/32
  </ipv4-prefix>
  <ipv6-prefix>
    2001:db8:bbbb::b/128
  </ipv6-prefix>
</eam>
<eam>
  <ipv4-prefix>
    192.0.2.16/28
  </ipv4-prefix>
  <ipv6-prefix>
    2001:db8:cccc::/124
  </ipv6-prefix>
</eam>
<eam>
  <ipv4-prefix>
    192.0.2.128/26
  </ipv4-prefix>
  <ipv6-prefix>
    2001:db8:dddd::/64
  </ipv6-prefix>
</eam>
<eam>
  <ipv4-prefix>
    192.0.2.192/29
  </ipv4-prefix>
  <ipv6-prefix>
    2001:db8:eeee:8::/62
  </ipv6-prefix>
</eam>
<eam>
  <ipv4-prefix>
    192.0.2.224/31
  </ipv4-prefix>
  <ipv6-prefix>
    64:ff9b::/127
  </ipv6-prefix>
</eam>
```



EAMs may be enabled jointly with statefull NAT64. This example shows a NAT64 function that supports static mappings:

```
<capabilities
  <nat-flavor>
    nat64
  </nat44-flavor>
  <static-mapping-support>
    true
  </static-mapping-support>
  <port-randomization-support>
    true
  </port-randomization-support>
  <port-range-allocation-support>
    true
  </port-range-allocation-support>
  <port-preservation-suport>
    true
  </port-preservation-suport>
  <port-parity-preservation-support>
    false
  </port-parity-preservation-support>
  <address-roundrobin-support>
    true
  </address-roundrobin-support>
  <paired-address-pooling-support>
    true
  </paired-address-pooling-support>
  <endpoint-independent-mapping-support>
    true
  </endpoint-independent-mapping-support>
  <address-dependent-mapping-support>
    false
  </address-dependent-mapping-support>
  <address-and-port-dependent-mapping-support>
    false
  </address-and-port-dependent-mapping-support>
  <endpoint-independent-filtering-support>
    true
  </endpoint-independent-filtering-support>
  <address-dependent-filtering>
    false
  </address-dependent-filtering>
  <address-and-port-dependent-filtering>
    false
  </address-and-port-dependent-filtering>
</capabilities>
```



### [A.7.](#) Static Mappings with Port Ranges

The following example shows a static mapping that instructs a NAT to translate packets issued from 192.0.2.1 and with source ports in the 100-500 range to 198.51.100.1:1100-1500.

```
<mapping-entry>
  <index>1</index>
  <type>static</type>
  <transport-protocol>6</transport-protocol>
  <internal-src-address>
    192.0.2.1
  </internal-src-address>
  <internal-src-port>
    <start-port-number>
      100
    </start-port-number>
    <end-port-number>
      500
    </end-port-number>
  </internal-dst-port>
  <external-src-address>
    198.51.100.1
  </external-src-address>
  <external-src-port>
    <start-port-number>
      1100
    </start-port-number>
    <end-port-number>
      1500
    </end-port-number>
  </external-src-port>
  ...
</mapping-entry>
```

### [A.8.](#) Static Mappings with IP Prefixes

The following example shows a static mapping that instructs a NAT to translate TCP packets issued from 192.0.2.1/24 to 198.51.100.1/24.



```
<mapping-entry>
  <index>1</index>
  <type>static</type>
  <transport-protocol>6</transport-protocol>
  <internal-src-address>
    192.0.2.1/24
  </internal-src-address>
  <external-src-address>
    198.51.100.1/24
  </external-src-address>
  ...
</mapping-entry>
```

#### [A.9.](#) Destination NAT

The following XML snippet shows an example of a destination NAT that is instructed to translate all packets having 192.0.2.1 as a destination IP address to 198.51.100.1.

```
<dst-ip-address-pool>
  <pool-id>1</pool-id>
  <dst-in-ip-pool>
    192.0.2.1
  </dst-in-ip-pool>
  <dst-out-ip-pool>
    198.51.100.1
  </dst-out-ip-pool>
</dst-ip-address-pool>
```

In order to instruct a NAT to translate TCP packets destined to '192.0.2.1:80' to '198.51.100.1:8080', the following XML snippet shows the static mapping to be configured on the NAT:





```
<mapping-entry>
  <index>1</index>
  <type>static</type>
  <transport-protocol>6</transport-protocol>
  <internal-dst-address>
    192.0.2.1
  </internal-dst-address>
  <internal-dst-port>
    <start-port-number>80</start-port-number>
  </internal-dst-port>
  <external-dst-address>
    198.51.100.1
  </external-dst-address>
  <external-dst-port>
    <start-port-number>8080</start-port-number>
  </external-dst-port>
</mapping-entry>
```

In order to instruct a NAT to translate TCP packets destined to '192.0.2.1:80' (http traffic) to 198.51.100.1 and '192.0.2.1:22' (ssh traffic) to 198.51.100.2, the following XML snippet shows the static mappings to be configured on the NAT:



```
<mapping-entry>
  <index>1</index>
  <type>static</type>
  <transport-protocol>6</transport-protocol>
  <internal-dst-address>
    192.0.2.1
  </internal-dst-address>
  <internal-dst-port>
    <start-port-number>
      80
    </start-port-number>
  </internal-dst-port>
  <external-dst-address>
    198.51.100.1
  </external-dst-address>
  ...
</mapping-entry>
<mapping-entry>
  <index>2</index>
  <type>static</type>
  <transport-protocol>
    6
  </transport-protocol>
  <internal-dst-address>
    192.0.2.1
  </internal-dst-address>
  <internal-dst-port>
    <start-port-number>
      22
    </start-port-number>
  </internal-dst-port>
  <external-dst-address>
    198.51.100.2
  </external-dst-address>
  ...
</mapping-entry>
```

The NAT may also be instructed to proceed with both source and destination NAT. To do so, in addition to the above sample to configure destination NAT, the NAT may be provided, for example with a pool of external IP addresses (198.51.100.0/24) to use for source address translation. An example of the corresponding XML snippet is provided hereafter:



```
<external-ip-address-pool>
  <pool-id>1</pool-id>
  <external-ip-pool>
    198.51.100.0/24
  </external-ip-pool>
</external-ip-address-pool>
```

Instead of providing an external IP address to share, the NAT may be configured with static mapping entries that modifies the internal IP address and/or port number.

#### **A.10. Customer-side Translator (CLAT)**

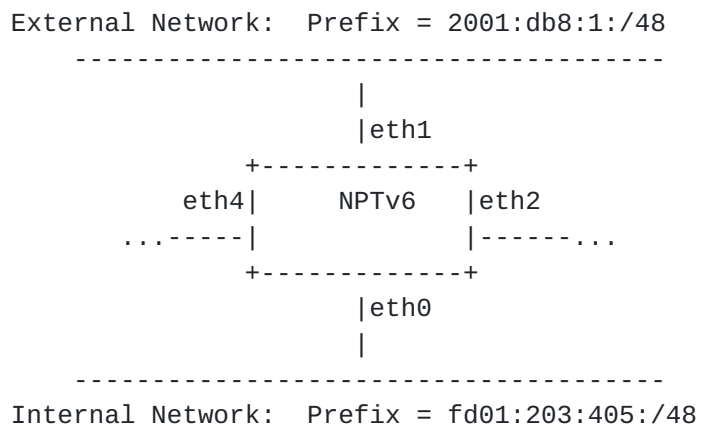
The following XML snippet shows the example of a CLAT that is configured with 2001:db8:1234::/96 as PLAT-side IPv6 prefix and 2001:db8:aaaa::/96 as CLAT-side IPv6 prefix. The CLAT is also provided with 192.0.0.1/32 (which is selected from the IPv4 service continuity prefix defined in [[RFC7335](#)]).

```
<clat-ipv6-prefixes>
  <ipv6-prefix>
    2001:db8:aaaa::/96
  </ipv6-prefix>
</clat-ipv6-prefixes>
<clat-ipv4-prefixes>
  <ipv4-prefix>
    192.0.0.1/32
  </ipv4-prefix>
</clat-ipv4-prefixes>
<nat64-prefixes>
  <nat64-prefix>
    2001:db8:1234::/96
  </nat64-prefix>
</nat64-prefixes>
```

#### **A.11. IPv6 Network Prefix Translation (NPTv6)**

Let's consider the example of a NPTv6 translator that should rewrite packets with the source prefix (fd01:203:405:/48) with the external prefix (2001:db8:1:/48). The internal interface is "eth0" while the external interface is "eth1".





#### Example of NPTv6 ([RFC6296](#))

The XML snippet to configure NPTv6 prefixes in such case is depicted below:

```

<nptv6-prefixes>
  <internal-ipv6-prefix>
    fd01:203:405:/48
  </internal-ipv6-prefix>
  <external-ipv6-prefix>
    2001:db8:1:/48
  </external-ipv6-prefix>
</nptv6-prefixes>
...
<external-realm>
  <external-interface>
    eth1
  </external-interface>
</external-realm>
  
```

Figure 3 shows an example of an NPTv6 that interconnects two internal networks (fd01:203:405:/48 and fd01:4444:5555:/48); each is translated using a dedicated prefix (2001:db8:1:/48 and 2001:db8:6666:/48, respectively).





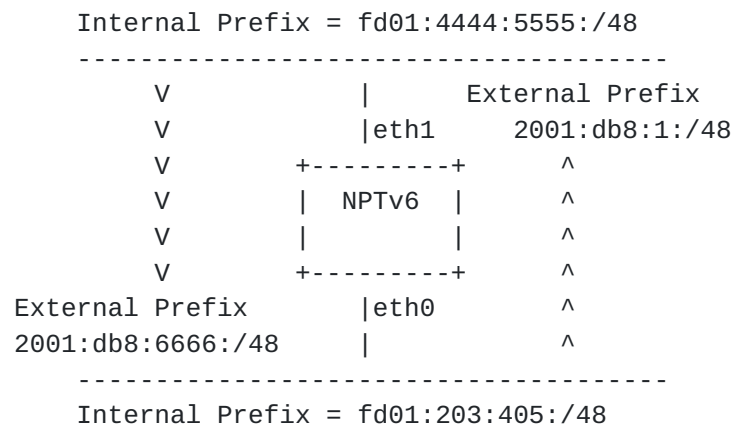


Figure 3: Connecting two Peer Networks ([RFC6296](#))

To that aim, the following configuration is provided to the NPTv6:



```
<policy>
  <id>1</id>
  <nptv6-prefixes>
    <internal-ipv6-prefix>
      fd01:203:405:/48
    </internal-ipv6-prefix>
    <external-ipv6-prefix>
      2001:db8:1:/48
    </external-ipv6-prefix>
  </nptv6-prefixes>
  <external-realm>
    <external-interface>
      eth1
    </external-interface>
  </external-realm>
</policy>
<policy>
  <id>2</id>
  <nptv6-prefixes>
    <internal-ipv6-prefix>
      fd01:4444:5555:/48
    </internal-ipv6-prefix>
    <external-ipv6-prefix>
      2001:db8:6666:/48
    </external-ipv6-prefix>
  </nptv6-prefixes>
  <external-realm>
    <external-interface>
      eth0
    </external-interface>
  </external-realm>
</policy>
```

#### Authors' Addresses

Mohamed Boucadair  
Orange  
Rennes 35000  
France

Email: mohamed.boucadair@orange.com



Senthil Sivakumar  
Cisco Systems  
7100-8 Kit Creek Road  
Research Triangle Park, North Carolina 27709  
USA

Phone: +1 919 392 5158  
Email: ssenthil@cisco.com

Christian Jacquenet  
Orange  
Rennes 35000  
France

Email: christian.jacquenet@orange.com

Suresh Vinapamula  
Juniper Networks  
1133 Innovation Way  
Sunnyvale 94089  
USA

Email: sureshk@juniper.net

Qin Wu  
Huawei  
101 Software Avenue, Yuhua District  
Nanjing, Jiangsu 210012  
China

Email: bill.wu@huawei.com

