Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 17, 2013

E. Abdo
M. Boucadair
J. Queiroz
France Telecom
July 16, 2012

HOST_ID TCP Options: Implementation & Preliminary Test Results
draft-abdo-hostid-tcpopt-implementation-03

Abstract

This memo documents the implementation of the HOST_ID TCP Options. It also discusses the preliminary results of the tests that have been conducted to assess the technical feasibility of the approach as well as its scalability. Several HOST_ID TCP options have been implemented and tested.

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 http://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 January 17, 2013.

Table of Contents

1. Introduction

To ensure IPv4 service continuity, service providers will need to deploy IPv4 address sharing techniques. Several issues are likely to be encountered (refer to [RFC6269] for a detailed survey of the issues) and they may affect the delivery of services that depends on the enforcement of policies based upon the source IPv4 address.

Some of these issues may be mitigated owing to the activation of advanced features. Among the solutions analyzed in [I-D.boucadair-intarea-nat-reveal-analysis], the use of a new TCP option to convey a HOST_ID seems to be a promising solution.

This memo documents some implementation and experimentation efforts that have been conducted to assess the viability of using HOST_ID TCP options at large scale. In particular, this document provides experimentation results related to the support of the HOST_ID TCP Options, the behavior of legacy TCP servers when receiving the HOST_ID TCP options. This draft also discusses the impact of using a HOST_ID TCP options on the time it takes to establish a connection; it also tries to evaluate the impact of the new TCP options on the performance of the CGN. Finally it presents the enforcement policies that could be applied by remote servers based upon the HOST_ID options contents.

2. Objectives

The implementation of several HOST_ID TCP options is primarily meant to:

- Assess the validity of the HOST_ID TCP option approach
- Evaluate the impact on the TCP stack to support the HOST_ID TCP options
- Improve filtering and logging capabilities based upon the contents of the HOST_ID TCP option. This means the enforcement of various policies based upon the content of the HOST_ID TCP option at the server side: Log, Deny, Accept, etc.
- Assess the behavior of legacy TCP servers when receiving a HOST_ID TCP option
- Assess the success ratio of TCP communications when a HOST_ID TCP option is received
- Assess the impact of injecting a HOST_ID TCP option on the time it takes to establish a connection
- Assess the performance impact on the CGN device that has been configured to inject the HOST_ID option

3. NAT Reveal TCP Options: Overview

The original idea of defining a TCP option is documented in [I-D.wing-nat-reveal-option] and denoted as HOST_ID_WING.

An additional TCP option is also considered and denoted as HOST_ID_BOUCADAIR. The main motivation is to cover also the load-balancer use case and provide richer functionality as Forwarded-For HTTP header than HOST_ID_WING can provide.

The following sub-sections provide an overview of these HOST_ID TCP options.

3.1. HOST_ID_WING TCP Option

HOST_ID_WING is defined in [I-D.wing-nat-reveal-option]. Figure 1 shows the format of this option.

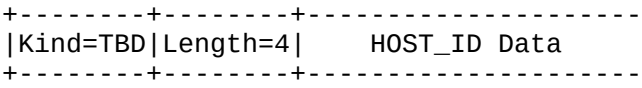


Figure 1: Format of HOST_ID_WING TCP Option

This option must be sent only upon the initial connection request, i.e., in SYN packets as shown in Figure 2

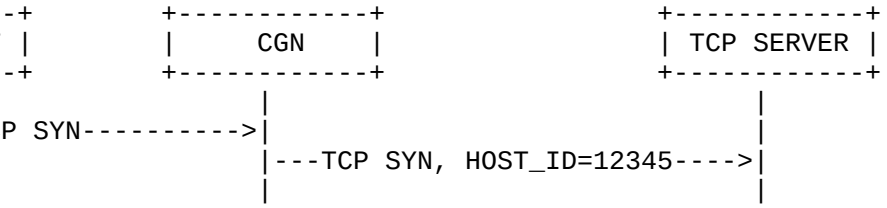


Figure 2: HOST_ID_WING TCP Option: Flow example

3.2. HOST_ID_BOUCADAIR TCP Option

As mentioned above, the HOST_ID_BOUCADAIR TCP Option is inspired from HOST_ID_WING and XFF.

The HOST_ID_BOUCADAIR option is a 10-byte long TCP option, where KIND, Length and lifetime-Origin fields fill one byte each, and HOST_ID data is 7-byte long as shown in Figure 3

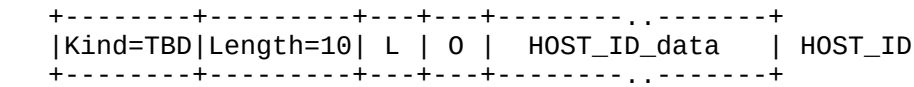


Figure 3: Format of HOST_ID_BOUCADAIR TCP option

- o L: Indicates the validity lifetime of the enclosed data (in the spirit of [RFC6250]). The following values are supported:
  0: Permanent;
  >0:Dynamic; this value indicates the validity time.
- o Origin: Indicates the origin of the data conveyed in the data field. The following values are supported:
  0: Internal Port
  1: Internal IPv4 address
  2: Internal Port: Internal IPv4 address
  3: IPv6 Prefix
  >3: No particular semantic
- o HOST_ID_data depends on the content of the Origin field; padding is required.

Two modes are described below: the SYN mode (Section 3.2.1) and the ACK mode. (Section 3.2.2).

If the ACK mode is used (Section 3.2.2), Figure 4 shows the HOST_ID_ENABLED option (2-bytes long) to be included in the SYN.

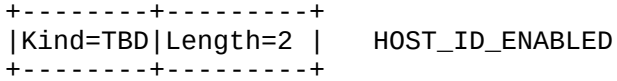


Figure 4: Format of HOST_ID_ENABLED

3.2.1. SYN Mode

This mode is similar to the mode described in Section 3.1. In this mode, HOST_ID_BOUCADAIR is sent in SYN packets.

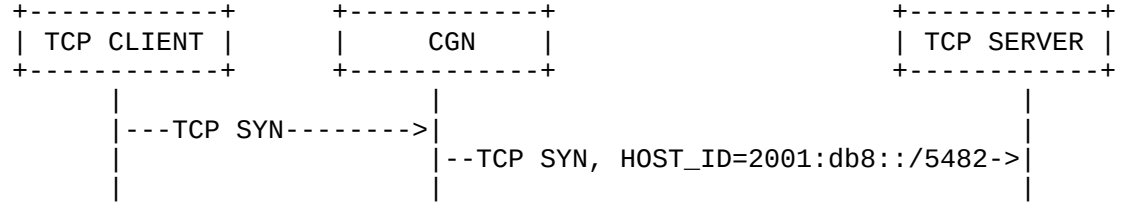


Figure 5: HOST_ID_BOUCADAIR: SYN Mode

3.2.2. ACK Mode

The ACK Mode is as follows (see Figure 6):

- Send HOST_ID_ENABLED (Figure 4) in SYN
- If the remote TCP server supports that option, it must return it in SYNACK
- Then the TCP Client sends an ACK in which the CGN injects HOST_ID_BOUCADAIR (Figure 3)

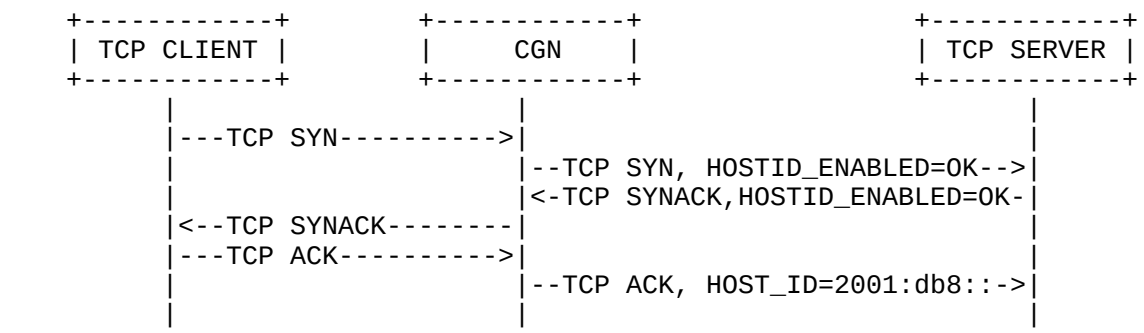


Figure 6: HOST_ID_BOUCADAIR: ACK Mode

4. Overview of the Linux Kernel Modifications

The objective of this phase is to support HOST_ID_WING, HOST_ID_BOUCADAIR and HOST_ID_ENABLED in the SYN mode.

In order to support the injection of the HOST_ID TCP options presented in Section 3, some modifications were applied to the Linux Kernel (more precisely to the TCP stack part of the Kernel). The header file tcp.h, file where are defined the TCP variables and functions, is updated to define the new HOST_ID options' KINDs (option numbers) and Lengths.

Major modifications have been made in the "tcp_output.c" file. This file is responsible for building and transmitting all TCP packets. For each HOST_ID TCP option, the required modifications to increase the header size and to inject KIND, Length and the corresponding HOST_ID data are implemented for the TCP SYN packets.

As we have three different HOST_ID options and as HOST_ID_BOUCADAIR can convey different information the configuration of the HOST_ID options have to be simple with minimal complexity. Since the manipulation of HOST_ID options impacts the Kernel TCP drivers, a suitable solution is to define new sysctl variables (system control variables) that allow the modification of Kernel parameters at

runtime, without having to reboot the machine so that it takes into account a new configuration.

Once modifications have taken place, the Kernel must be recompiled so that the new TCP options are taken into account.

Kernel modifications and recompilation have been done and tested successfully on Fedora and Debian Linux distributions, on different kernel versions.

The following configuration options are supported:

- Enable/Disable injecting the TCP Option
- Support HOST_ID WING, HOST_ID BOUCADAIR and HOST_ID_ENABLED
- When the HOST_ID TCP option is supported, the information to be injected is configurable:
  - Source IPv6 address or the first 56 bits of the address
  - Source IPv4 address
  - Source port number
  - Source IPv4 address and Source port
  - IPv6 address or the first 56 bits of the B4 when DS-Lite is activated

5. Testbed Setup & Configuration

The setup of three testbed configurations have been considered:

1. HOST_ID TCP option is injected by the host itself. No CGN is present in the forwarding path (Figure 7)
2. HOST_ID TCP option is injected by hosts deployed behind a HTTP proxy. No CGN is present in the forwarding path (Figure 8)
3. HOST_ID TCP option is injected by the DS-Lite AFTR element (Figure 9).

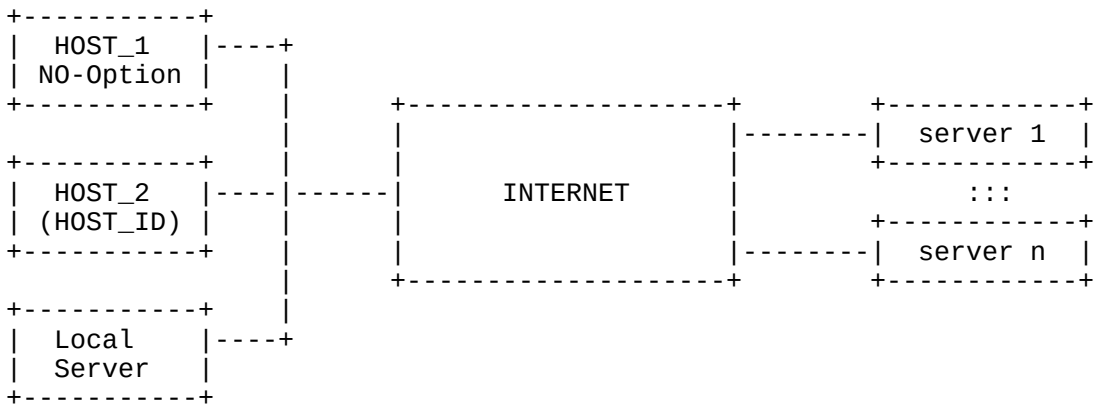


Figure 7: Testbed setup: No Proxy and no CGN

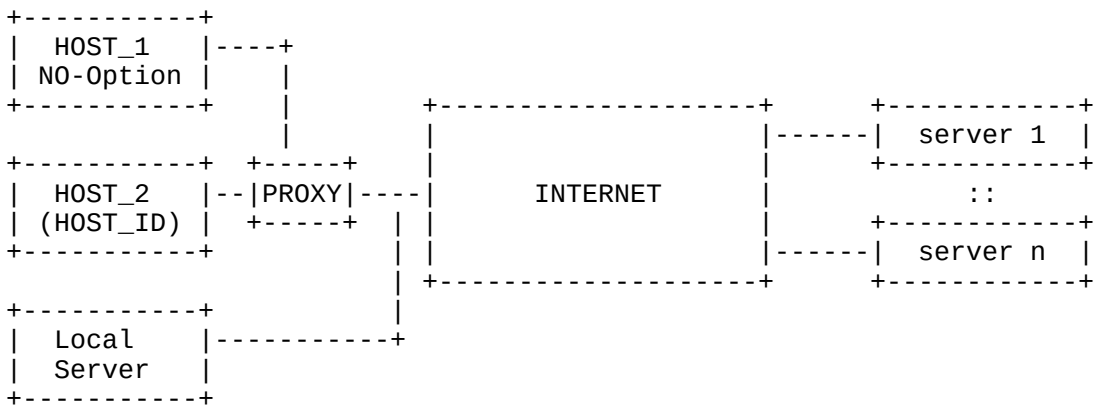


Figure 8: Testbed setup: HTTP Proxy

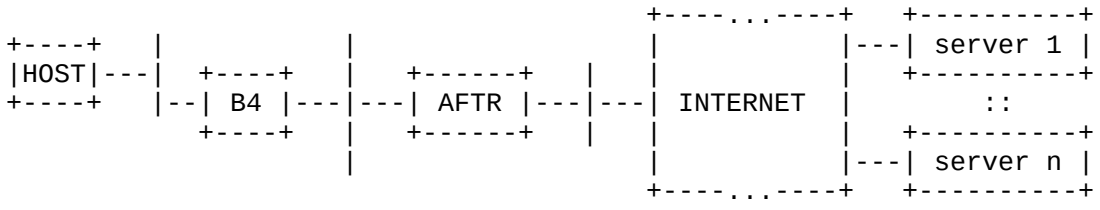


Figure 9: DS-Lite CGN Environment

Figure 7 and Figure 8 are used to assess the behavior of the top 100,000 sites when a HOST_ID option is enabled and to evaluate the impact of the option on both the session establishment delay and the success ratio.

On the other hand, the configuration shown in Figure 9 will be used to evaluate the impact on the CGN performances when HOST_ID TCP option is injected by the CGN.

5.1. Automated TCP Traffic Generator

A Python-encoded robot has been used as the traffic generator. The robot automates the retrieval of HTTP pages identified by URLs, and returns different connection information. The retrieval of pages is based upon Pycurl, a Python interface of libcurl. Libcurl is an URL transfer library that supports different protocols (e.g., HTTP, FTP).

The robot consists of two programs:

1. The first one takes an URL as a input parameter, performs the DNS lookup and then tries to connect to the corresponding machine. It returns either different time values and connection status or an error message with the source of the error in case of connection failure (e.g., DNS error). The TCP connection establishment time is calculated as the difference between the CONNECT_TIME and NAMELOOKUP_TIME where:
   * NAMELOOKUP_TIME is the time it took from the start until the name resolution is completed.
   * CONNECT_TIME is the time it took from the start until the connection to the remote host (or proxy) is completed.
2. The second program aims to increase efficiency and speed of the testing by using a multi-thread technique. It takes the number of threads and an input file listing URLs as parameters. This program prints URLs to an output file with the corresponding connection time. If something wrong happened so that the connection failed, the program returns an error message with the corresponding error type.

5.2. Testing Methodology and Procedure

The testing is done using two machines, one that supports the HOST_ID TCP options and the other that does not. The second machine is used as a reference for the measurements. Testing is performed in parallel on the two machines that are directly connected to the Internet. For each HOST_ID TCP option, the test is repeated many times. The cycle is repeated in different days. Then results are grouped into tables where averages are calculated. The comparison between the different HOST_ID options results is made by using the no-option testing results as a reference.

Testing was also performed behind a proxy (Figure 8) to evaluate the impact of embedding the HOST_ID TCP options on the connection establishment time when a proxy is in the path. When a proxy is

present, the connection delay is impacted (the delay is calculated for the connection between the host and the proxy).

Tests have been conducted from hosts:
1. Connected to an enterprise network
2. In a lab behind a firewall
3. Connected to two (2) commercial ISP networks

5.3. Check HOST_ID TCP Options are Correctly Injected

To check whether the HOST_ID TCP options are correctly injected, the local server in Figure 7 is configured to be reachable from Internet. Packets conveying the HOST_ID TCP options are sent from a host supporting the options. These packets are used without alteration by the local server.

This configuration confirms the packets sent to remote servers conveys HOST_ID TCP options.

5.4. Top Site List

The Alexa top sites list has been used to conduct the HTTP tests.

Anonymous FTP sites list from ftp-sites.org has been used to conduct the FTP tests.

6. Experimentation Results

Various combinations of the HOST_ID TCP options have been tested:
1. HOST_ID_WING
2. HOST_ID_WING has also been adapted to include 32 bits and 64 bits values. No particular impact on session establishment has been observed.
3. HOST_ID_BOUCADAIR (source port)
4. HOST_ID_BOUCADAIR (IPv4 address)
5. HOST_ID_BOUCADAIR (source port:IPv4 address)
6. HOST_ID_BOUCADAIR (IPv6 Prefix)
7. HOST_ID_ENABLED

Both the success ratio and the average time to establish the TCP session are reported below.

6.1. HTTP Experimentation Results

Tests have been conducted from hosts:

1. Connected to an enterprise network
2. Connected to two commercial ISP networks
3. In a lab behind a firewall

6.1.1. Configuration 1: Connected to an enterprise network

The results show that the success ratio for establishing TCP connection with legacy servers is almost the same for all the HOST_ID options as shown in Figure 10, Figure 11 and Figure 12.

6.1.1.1. Results

	NO-OPTION	O-WING	Failure Ratio
Top10	100,00000%	100,00000%	0,00000%
Top100	100,00000%	100,00000%	0,00000%
Top200	100,00000%	100,00000%	0,00000%
Top300	99,66667%	99,66667%	0,00000%
Top400	99,50000%	99,50000%	0,00000%
Top500	99,40000%	99,40000%	0,00000%
Top600	99,50000%	99,50000%	0,00000%
Top700	99,57143%	99,57143%	0,00000%
Top800	99,50000%	99,50000%	0,00000%
Top900	99,44444%	99,44444%	0,00000%
Top1000	99,50000%	99,50000%	0,00000%
Top2000	99,35000%	99,30000%	0,05000%
Top3000	99,10000%	99,06667%	0,03333%
Top4000	99,10000%	99,05000%	0,05000%
Top5000	99,14000%	99,10000%	0,04000%
Top6000	99,21667%	99,18333%	0,03333%
Top7000	99,25714%	99,21429%	0,04286%
Top8000	99,15000%	99,10000%	0,05000%
Top9000	99,16667%	99,12222%	0,04444%
Top10000	99,16000%	99,12000%	0,04000%
Top20000	98,50500%	98,44000%	0,06500%
Top30000	98,21667%	98,11667%	0,10000%
Top40000	98,10750%	98,00750%	0,10000%
Top50000	98,00000%	97,89800%	0,10200%
Top60000	97,95167%	97,85000%	0,10167%
Top70000	97,88857%	97,78857%	0,10000%
Top80000	97,84500%	97,74875%	0,09625%
Top90000	97,79444%	97,69889%	0,09556%
Top100000	97,75100%	97,64800%	0,10300%

Figure 10: Cumulated Success ratio (HOST_ID_WING)

	NO-OPTION	O-WING	Failure Ratio
1-100	100,00%	100,00%	0,00%
101-200	100,00%	100,00%	0,00%
201-300	99,00%	99,00%	0,00%
301-400	99,00%	99,00%	0,00%
401-500	99,00%	99,00%	0,00%
501-600	100,00%	100,00%	0,00%
601-700	100,00%	100,00%	0,00%
701-800	99,00%	99,00%	0,00%
801-900	99,00%	99,00%	0,00%
901-1000	100,00%	100,00%	0,00%
1-1000	99,50%	99,50%	0,00%
1001-2000	99,20%	99,10%	0,10%
2001-3000	98,60%	98,60%	0,00%
3001-4000	99,10%	99,00%	0,10%
4001-5000	99,30%	99,30%	0,00%
5001-6000	99,60%	99,60%	0,00%
6001-7000	99,50%	99,40%	0,10%
7001-8000	98,40%	98,30%	0,10%
8001-9000	99,30%	99,30%	0,00%
9001-10000	99,10%	99,10%	0,00%
10001-20000	97,85%	97,76%	0,90%
20001-30000	97,64%	97,47%	1,70%
30001-40000	97,78%	97,68%	1,00%
40001-50000	97,57%	97,46%	1,10%
50001-60000	97,71%	97,61%	1,00%
60001-70000	97,61%	97,52%	0,90%
70001-80000	97,44%	97,37%	0,70%
80001-90000	97,39%	97,30%	0,90%
90001-100000	97,36%	97,19%	1,70%

Figure 11: TopX000 Success Ratio (HOST_ID_WING)

	NO-OPTION	O-BOUCADAIR	Failure Ratio
1-100	100,00%	100,00%	0,00%
101-200	100,00%	100,00%	0,00%
201-300	99,00%	99,00%	0,00%
301-400	99,00%	99,00%	0,00%
401-500	99,00%	99,00%	0,00%
501-600	100,00%	100,00%	0,00%
601-700	100,00%	100,00%	0,00%
701-800	99,00%	99,00%	0,00%
801-900	99,00%	99,00%	0,00%
901-1000	100,00%	100,00%	0,00%
0-1000	99,50%	99,50%	0,00%
1001-2000	99,20%	99,10%	0,10%
2001-3000	98,60%	98,60%	0,00%
3001-4000	99,30%	99,30%	0,00%
5001-6000	99,60%	99,60%	0,00%
6001-7000	99,50%	99,40%	0,10%
7001-8000	98,40%	98,30%	0,10%
8001-9000	99,30%	99,20%	0,10%
9001-10000	99,10%	99,10%	0,00%
10001-20000	97,85%	97,76%	0,90%
20001-30000	97,64%	97,46%	1,80%
30001-40000	97,78%	97,66%	1,20%
40001-50000	97,57%	97,46%	1,10%
50001-60000	97,71%	97,61%	1,00%
60001-70000	97,61%	97,51%	1,00%
70001-80000	97,44%	97,36%	0,80%
80001-90000	97,39%	97,30%	0,90%
90001-100000	97,36%	97,19%	1,70%

Figure 12: TopX000 Success Ratio (HOST_ID_BOUCADAIR)

6.1.1.2. Analysis

- For the top 100,000 sites, connection failures occur for 2249 HTTP sites. These failures were reported as being caused by DNS issues (servers not mounted), connection timeouts (servers down...), connection resets by peers, connection problems and empty replies from servers. The 2249 failures occur, whether HOST_ID options are injected or not.
- When any HOST_ID TCP option is conveyed, 103 servers did not respond; however when no option is injected, all these servers responded normally.

- Same results were obtained for HOST_ID_WING and HOST_ID_ENABLED.
- Same results were obtained for all the HOST_ID_BOUCADAIR options (source port, IPv6 prefix, etc.).

When HOST_ID_BOUCADAIR is enabled, six (6) additional servers did not respond:

- Three (3) servers (www.teufel.de - www.1001fonts.com - www.sigur-ros.co.uk) did not respond to the SYN packets sent by the host.
- Three (3) servers (www.lawyers.com, www.lexis.com, www.nexis.com) responded with "strange" SYN/ACK packets with same TCP options length including a part of the HOST_ID options that was sent. This part of HOST_ID option caused an erroneous SYN/ACK packet received by the host: in fact the second byte of the HOST_ID part is considered as its length and this length does not really fit with the real length of the part. So the machine does not respond back to the server with an ACK packet. This is why we have no response for these servers.

When HOST_ID_WING or HOST_ID_ENABLED is enabled, also strange SYN/ACKs were received by the host but no errors in these packets (a long series of NOP options). This justifies the connection success for these 2 options.

The results show that including a HOST_ID TCP option does not systematically imply an extra delay for the establishment of the TCP session. Based on the average of session establishment with the top 100 000 sites, the following results have been obtained:

- delay(HOST_ID_WING) < delay(NO_OPTION): 42,55 %
- delay(HOST_ID_BOUCADAIR) < delay(NO_OPTION): 48,16 %
- delay(HOST_ID_ENABLED) < delay(NO_OPTION): 51,28 %

6.1.2. Configuration 2: In a lab behind a firewall

When a HTTP proxy is in the path, the injection of HOST_ID TCP option does not impact the success ratio. This is due to that the HTTP proxy strips the HOST_ID TCP options; these options are not leaked to remote Internet servers. The testing has been done by observing packets received to a server installed with a public IP address (no HOST_ID options were seen in the received SYN packets).

6.1.3. Configuration 3: Connected to two commercial ISP networks

The results obtained when testing was performed by connecting to two ISP networks confirmed the results obtained in the testing described in Section 6.1.1

6.1.4. Additional Results

In one of our testing for top 1000 sites, when padding was badly implemented for HOST_ID_BOUCADAIR (padding was implemented as a prefix so option's Length does not correspond to the real length because the padding was not counted), we got for configuration(1) in the lab and for one of the ISP the following results:

	No-Option	O-BOUCADAIR	Failure Ratio
Top10	100,00000%	100,00000%	0,00000%
Top100	100,00000%	100,00000%	0,00000%
Top200	100,00000%	100,00000%	0,00000%
Top300	100,00000%	99,66667%	0,33333%
Top400	99,75000%	99,00000%	0,75000%
Top500	99,80000%	99,00000%	0,80000%
Top600	99,83333%	98,66667%	1,16667%
Top700	99,85714%	98,14286%	1,71429%
Top800	99,75000%	98,00000%	1,75000%
Top900	99.66667%	97,33333%	2,33333%
Top1000	99,70000%	97,10000%	2,60000%

Cumulated Success ratio (HOST_ID_Boucadair with wrong padding)

The results for HOST_ID_WING for all three configurations are the same as Section 6 (this option was correctly coded). Results obtained for HOST_ID_BOUCADAIR are not the same.

For the configuration (2) behind a firewall, we did not face any rejection because of parsing the TCP options (the HOST_ID options were retrieved from the packet).

6.1.5. Analysis

Configuration (1) in Lab and for one of the two CPEs lead to the results because 2.6% of these 1000 servers perform parsing validation for the received options so when the bad HOST_ID_BOUCADAIR option is sent, 2.6% of the servers treat the received SYN packets as erroneous packets and discard them.

For the connection behind the second ISP, we didn't get a response for any of the servers. After investigation, the reason was that the Box validates the received packets before sending them to the Internet. The erroneous SYN packets holding badly encoded options (HOST_ID_BOUCADAIR in this case) were dropped and no connection was

established. On the other hand, the other box did not validate options length for received packets before sending them to the Internet.

6.2. FTP

Various combinations of the HOST_ID TCP options have been tested:

1. HOST_ID_WING
2. HOST_ID_BOUCADAIR (source port)
3. HOST_ID_BOUCADAIR (source port:IPv4 address)

A list of 5591 FTP servers has been used to conduct these testings. Among this list, only 2045 were reachable:

- Failure to reach 942 FTP servers due to connection timeout
- Failure to reach 1286 FTP servers due to DNS errors
- Failure to reach 717 FTP servers because access was denied
- Could not connect to 500 FTP servers
- Response reading failed for 81 servers
- Bad response from server for 20 servers

When HOST_ID TCP options are injected, 9 errors are observed (connection timeout).

Figure 13 and Figure 14 provide more data about the error distribution.

	NOB	HOST_ID	Failure Ratio
1-100	100%	100%	0,000%
101-200	100%	99%	1,000%
201-300	100%	99%	1,000%
301-400	100%	100%	0,000%
401-500	100%	100%	0,000%
501-600	100%	100%	0,000%
601-700	100%	100%	0,000%
701-800	100%	100%	0,000%
801-900	100%	99%	1,000%
901-1000	100%	99%	1,000%
1001-2000	100%	99,5%	0,500%
2000-2045	100%	100%	0,000%

Figure 13: Cumulated Success Ratio (FTP)

	NOB	HOST_ID	Failure Ratio
first 10	100,000%	100,000%	0,000%
first 100	100,000%	100,000%	0,000%
first 200	100,000%	99,500%	0,500%
first 300	100,000%	99,333%	0,667%
first 400	100,000%	99,500%	0,500%
first 500	100,000%	99,600%	0,400%
first 600	100,000%	99,667%	0,333%
first 700	100,000%	99,714%	0,286%
first 800	100,000%	99,750%	0,250%
first 900	100,000%	99,667%	0,333%
first 1000	100,000%	99,600%	0,400%
first 2000	100,000%	99,550%	0,450%
first 2045	100,000%	99,560%	0,440%

Figure 14: FirstXXX FTP Servers

The results show that including a HOST_ID TCP option does not systematically imply an extra delay for the establishment of the TCP session with remote FTP servers. Based upon the average of the session establishment with the 2045 FTP sites, the following results have been obtained:

- delay(HOST_ID_WING) < delay(NO_OPTION): 49,36585 %
- delay(HOST_ID_BOUCADAIR (source port:IPv4 address)) < delay(NO_OPTION): 48,41076%
- delay(HOST_ID_BOUCADAIR (source port)) < delay(NO_OPTION): 48,43902 %

6.3. SSH

The secure shell service has been tested between a host and a SSH server connected to the same network.

SSH connections have been successfully established with the server for all the HOST_ID TCP options. Same results were obtained using configuration (1) and configuration (2).

6.4. Telnet

Telnet sessions have been successfully initiated for all HOST_ID TCP options with a server (the CGN used in Figure 9).

7. AFTR Module Modifications

This section highlights the support the HOST_ID functionalities in the AFTR element of the DS-Lite model (Figure 9) and presents the testing results in order to conclude about the HOST_ID TCP options impacts on the performance of the CGN.

We used ISC AFTR implementation.

7.1. Specification

All privately-addressed IPv4 packets sent from DS-Lite serviced hosts go through an AFTR device where an isc_aftr daemon program is responsible for establishing the tunnel, configuring network interfaces and processing received packets.

The aftr.c source code controls all functionalities to be included or modified on packets received by the CGN, e.g., patching TCP MSS values, fix MTU, etc.

In order to activate/deactivate such functionalities, the corresponding parameters can be configured in a specific configuration file called "aftr.conf". In this file, other parameters are configured, e.g., the IPv6 addresses assigned to the tunnel endpoint and the global IPv4 address pool maintained by the CGN.

To support the injection of HOST_ID TCP options, "aftr.c" must be updated to inject, retrieve or verify the HOST_ID options depending on the HOST_ID parameters defined in "aftr.conf" file. Four HOST_ID parameters are defined in the configuration file:

1. hostid: to enable the injection, retrieval, matching... of HOST_ID options
2. hostid_wing: to enable injection/verification of HOST_ID_WING - to disable injection or to remove HOST_ID_WING
3. hostid_boucadair: to enable injection/verification of HOST_ID_BOUCADAIR - to disable injection or to remove HOST_ID_BOUCADAIR
4. hostid_enabled: to enable or disable HOST_ID_ENABLED injection

hostid, hostid_wing and hostid_enabled can be simply enabled or disabled. hostid_boucadair can be disabled or enabled with the corresponding Origin as HOST_ID data can be:

- Source Port Number
- Source IPv4 Address
- Source IPv4 Address + Source Port Number

- o 56 bits of Tunnel Softwire IPv6 Source Address.

Based on different HOST_ID parameters, the "aftr.c" code has been modified to control HOST_ID options; the AFTR is able to:

- o Inject the enabled HOST_ID TCP option if it is not already present in the packet
- o Retrieve an existing HOST_ID TCP option if this option is not enabled
- o Check an existing HOST_ID option's content if it is enabled; if the content's verification failed, the AFTR replaces the HOST_ID contents with the suitable information

The implementation takes into consideration the SYN mode for all the HOST_ID options (even for HOST_ID_enabled). The Support of HOST_ID_BOUCADAIR in the ACK mode needs implementation on the server's side and since both Enabled and Boucadair's options have been tested and no impact observed; the ACK mode should not imply any complication in implementation or impact on the performance.

7.2. Verification

The verification of HOST_ID implementation in the CGN has taken place using the testbed setup shown in Figure 9. The host used in this testing is a modified Linux machine that can inject HOST_ID options. The objective of the testing is to verify the different functionalities implemented in the AFTR. Verification has occurred using a local server where all the received packets were observed to make sure that the content of the HOST_ID fields is consistent with the enabled option.

The testing consists in observing the SYN packets (as SYN mode is supported) sent by the host and in comparing these packets to those received by the server. Different combinations of HOST_ID options sent by the host and HOST_ID configured options at the CGN level have been used.

The results show that once the host sends packets without any HOST_ID option injected, the SYN packets received by the server contain the correct option that has been enabled by the CGN (if any). Once HOST_ID_WING or HOST_ID_BOUCADAIR are injected by the host, if the hostid parameter in aftr.conf is enabled, the enabled (in "aftr.conf") HOST_ID option will be injected if not already present, or else its content will be verified and corrected (if wrong); the other disabled option will be discarded if it has already been sent by the host.

One additional case has been tested when both Wing's and Boucadair's HOST_ID options are sent by the host, the contents of the enabled

option are checked and corrected (if wrong), the other option is retrieved from the packet. The two options are dropped from the packet if they are both disabled.

The testing has been repeated for all the HOST_ID options sent by the host and enabled by the CGN. Verification also occurred for HOST_ID_ENABLED option.

7.3. CGN Performance Testing

To conclude about the impact of using HOST_ID, a commercial testing product has been used. This tool supports multiple application protocols such as HTTP and FTP for both IPv4 and IPv6 (including encapsulation). The DS-Lite model can be built directly from a port of this product: IPv4 packets are directly encapsulated in an IPv6 tunnel; the client's port emulates hosts and B4 elements at the same time. This port is directly connected to the AFTR tunnel endpoint. The AFTR's IPv4 interface is connected to the testing product server side where servers are assigned IPv4 addresses.

The testbed setup of this testing is shown in Figure 15:

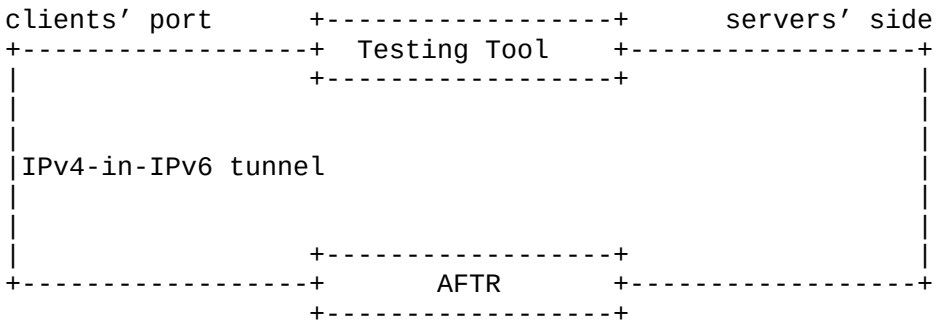


Figure 15: Platform Testbed

7.3.1. Configuration

At the IP level, the testing client port was configured with IPv6 addresses representing the B4. The testing tool also supports the DS-Lite "level" where the number of clients connected to each B4 and their addresses are configured. The AFTR address is defined at this level.

In the current testing, the total number of B4 elements is 5000 behind; One client is connected to each B4 (in total, 5000 clients are configured). However, the number of active users varies from 10 to 100, 500, 1000 and 10,000 during each testing simulation.

From the server standpoint, five servers have been assigned IPv4 addresses. These servers support HTTP and FTP traffic. For each HOST_ID TCP option, the testing was repeated for a different number of active users (N=10, 100, 500, 1000 and 10,000) and for HTTP and FTP traffic.

The HOST_ID options are injected by the CGN.

7.3.2. HTTP Testing

The testing duration was about 50 seconds during which the number of active users varies as a function of time: during the first 10s, the number of active users reaches the maximum and remains the same for the next 20 s. Then it decreases to zero during the next 20s.

Hereafter are provided some testing statistics providing some details about connections' success ratio, latency and other information that can be useful to evaluate the impact of HOST_ID on the CGN.

	No-Opt	O-WING	O-BOUCADAIR3	O-ENABLED
TCP connection established	1378	1267	1363	1369
TCP SYN SENT	1378	1267	1363	1369
Success Ratio	100	100	100	100
TCP Retries	193	193	197	177
TCP timeouts	140	136	152	111
HTTP connect' latencies t=20s	0,11	0,21	0,20	0,1
t=40s	0,40	0,50	0,50	0,45
t=60s	0,60	0,60	0,50	0,6
HTTP throughput received	46,47	45,31	45,88	46,12
TCP Connections Established/s	20,29	19,88	20,06	20,18

Figure 16: Results HTTP (N=10)

	No-Opt	O-WING	O-BOUCADAIR3	O-ENABLED
TCP connection established	1662	1739	1813	1679
TCP SYN SENT	1718	1770	1819	1729
Success Ratio	96	98	99	97
TCP Retries	1577	1569	1783	1576
TCP timeouts	798	806	934	808
HTTP connect' latencies t=20s	1,70	2,00	1,90	1,80
t=30s	3,30	2,40	2,25	3,30
t=40s	4,20	3,70	3,75	4,00
t=50s	5,00	4,80	4,50	5,00
HTTP throughput received	47,56	46,65	48,59	48,06
TCP Connections Established/s	20,94	20,53	21,35	21,19

Figure 17: Results HTTP (N=100)

	No-Opt	O-WING	O-BOUCADAIR3	O-ENABLED
TCP connection established	1956	1923	1944	1873
TCP SYN SENT	2088	2095	2137	1986
Success Ratio	93	91	90	94
TCP Retries	2734	2576	2453	2773
TCP timeouts	1261	1110	995	1213
HTTP connect' latencies t=20s	2,00	1,80	1,50	2,30
t=40s	4,00	3,30	2,80	4,30
t=50s	6,50	6,90	6,00	8,00
HTTP throughput received	70,19	65,00	69,81	67,13
TCP Connections Established/s	30,69	28,41	30,50	29,38

Figure 18: Results HTTP (N=1000)

	No-Opt	O-WING	O-BOUCADAIR4	O-ENABLED
TCP connection established	1576	2000	1796	1998
TCP SYN SENT	2088	2304	2009	2262
Success Ratio	87	86	89	88
TCP Retries	3018	3101	3013	3148
TCP timeouts	1167	1298	1213	1417
HTTP connect' latencies t=20s	2,20	3,00	2,20	2,50
t=40s	3,70	3,00	3,30	3,00
t=60s	7,80	5,00	7,00	5,60
t=70s	9,60	6,00	8,70	7,00
HTTP throughput received	45,00	54,52	51,45	57,20
TCP Connections Established/s	19,98	24,05	22,45	25,04

Figure 19: Results HTTP (N=10000)

7.3.2.1. Analysis of results

The results clearly show that there is no impact of any HOST_ID option on session establishment success ratio, which is quite similar to the success ratio when packets do not hold options or when HOST_ID options are not used. Also, the number of established connections does not decrease when any HOST_ID option is injected, so the CGN performance is not impacted by the fact of adding the HOST_ID options.

Another important factor to study is the latency that can be caused by HOST_ID injection. As the results show, the HTTP connection latency does not increase when HOST_ID is present if we compare the latency measured at different times for the different options.

As a result, we clearly see that the average throughput measured at servers is identical, whether HOST_ID options are used or not (given that the number of session established is quite the same).

Another consequence is that the TCP connection establishment rate at servers is not decreasing when a HOST_ID option is taken into account.

7.3.2.2. Conclusion

The results that have been obtained show that the performance of the CGN is not impacted by HOST_ID option injection even when the number of active users is high (10,000 is not negligible for a CGN run on an ordinary Linux machine): neither the session success ratio, nor the connection latency are impacted by the presence of the HOST_ID in SYN

packets.

7.3.3. FTP

The same testing was also run for FTP traffic. No particular impact on the performance of the CGN has been observed.

8. IPTABLES: Modifications to Enforce Policies at the Server Side

8.1. Overview

iptables module has been updated to:
- Log the content of TCP header with HOST_ID
- Drop packets holding a HOST_ID option
- Match any HOST_ID value
- Drop packets holding a specific HOST_ID value
- Strip any existing HOST_ID option

To support the above functionalities, modification should take into consideration stripping and matching options as described below:

1. To strip the content of any existing HOST_ID option, the shared library "libxt_TCPOPTSTRIP.so" is modified: the HOST_ID_WING and HOST_ID_BOUCADAIR Kinds' numbers were defined in the corresponding source file (libxt_TCPOPTSTRIP.c) with the corresponding names to enforce the iptables stripping rule. After enforcing these changes, the shared library must be created to replace the existing one and to allow applying the rule of stripping of the HOST_ID options. Once modifications have taken place, the following command should be used to strip the HOST_ID options:

```
iptables -t mangle -A INPUT -j TCPOPTSTRIP -p tcp --strip-options
 hostid_wing, hostid_boucadair
```

2. In order to allow blocking, logging or applying any rule based upon the HOST_ID_WING or HOST_ID_BOUCADAIR values or range of values, a HOST_ID shared library must be created to:
   * Match HOST_ID options values entered in corresponding iptables rules,
   * Print the HOST_ID rules on screen,
   * Save values,
   * Check the values (or range values) entered by user if they respect the limit values of these options.

In addition to the shared library: a specific Kernel module must be built to apply HOST_ID matching rules on the packets passing through the network interfaces. This module compares the HOST_ID options' values held by packets with the HOST_ID values specified in the iptables rule table: when a packet matches the HOST_ID's range, the corresponding rule will be applied for this packet. The HOST_ID_WING matching value is 2 bytes long corresponding to HOST_ID_WING data.
The HOST_ID_BOUCADAIR matching value is 8 bytes long corresponding to Lifetime + Origin field (1 byte) and HOST_ID_WING data (7 bytes).

8.2. Validation

After having updated the iptables package with the suitable HOST_ID libraries and module, different HOST_ID policies should be applied and tested on the server side. The testing has been done using a simple configuration as shown below (Figure 20).

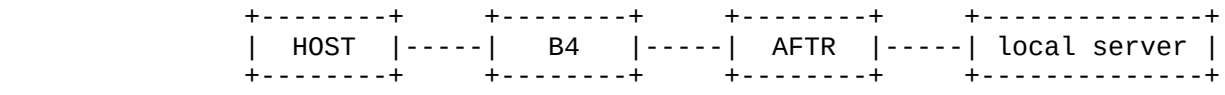


Figure 20: Platform configuration: HOST_ID enforcing policies

In the current testing, the AFTR supports HOST_ID options injection and iptables is modified at the local server. Logging recommendations consists of logging the IPv4 address and the HOST_ID option for each connection. Because HOST_ID is sent only in SYN packets (in the current implementation), only SYN packets will be logged to a specific file called iptables.log: the rsyslog.d must be updated with the corresponding command to log iptables messages into the specific file. Then rsyslog must be reloaded to apply changes.

8.3. Stripping HOST_ID Options

To strip a certain HOST_ID option, TCPOPTSTRIP rule must be called. Verification consists in logging and then checking the SYN packets and more precisely the corresponding TCP options, e.g., the following rules must be applied to strip HOST_ID_WING:

```
iptables -t mangle -A INPUT -j TCPOPTSTRIP -p tcp --strip-options
 hostid_wing
iptables -A INPUT -j LOG --log-tcp-options -p tcp --syn
```

The first rule applies for the mangle table. This table allows

stripping HOST_ID_WING whose role is to remove option Wing's fields and replaces them by NOP options (NOP=No Operation=0x01). The second rule enables the logging of SYN packets with the corresponding TCP options.

After applying these rules (to strip and log HOST_ID_WING) on the local server, we tried to access the server's HTTP pages from the host. The test is repeated several times and a different HOST_ID option is enabled by the AFTR each time.

Then the "iptables.log" file is checked: only one SYN packet is logged with 4 bytes stripped out in the TCP option part. All IPv4 packets going through the AFTR are also logged to be compared with the server's logged stripped packets.

The comparison of the SYN packets logged by the server with the SYN packets sent by the AFTR clearly shows that the stripped option is HOST_ID_WING (all the header fields have been verified to ensure packet matching): the 4 bytes corresponding to the HOST_ID_WING option are replaced with NOP options (each one of the 4 bytes is equal to '1' = NOP).

The same testing was repeated with HOST_ID_BOUCADAIR. The testing shows that the 10 bytes corresponding to this option were successfully stripped.

8.4. Logging a Specific HOST_ID Option Value

The remote server should be able to track connections coming from different clients; it should log packets headers including the HOST_ID TCP option information. This can be enforced using the following command:

```
iptables -t mangle -A INPUT -j TCPOPTSTRIP -p tcp --strip-options
 hostid_wing
```

Now, to log packets matching a certain HOST_ID value or range of values, the following rule must be applied:

```
iptables -A INPUT -p tcp --syn -m hostid --hostid_wing value[:value]
 -j LOG -log-tcp-options
```

This command matches the HOST_ID_WING values held by SYN packets with the specific value [or the specific range of values] determined by

the rule.

The testing configuration in Figure 20 was used. The HOST_ID_WING data are implemented as being the last 16 bits of the IPv4 private source address. When the HOST_ID_WING option is injected by the CGN, if the data field value corresponds to the iptables value (or range of values), the packet header is logged. Otherwise, if the HOST_ID_WING data is said out of range or the packet does not hold the HOST_ID_WING option, the packet is not logged.

The same testing was repeated to match HOST_ID_BOUCADAIR data information:

```
iptables -A INPUT -p tcp --syn -m hostid --hostid_boucadair value
 [:value] -j LOG -log-tcp-options
```

To verify the logging of a specific Boucadair's value, the Boucadair's options holding source IP address (Origin=2) or IPv6 prefix (Origin=4) were tested successfully; these data values are fixed since they depend on the host's address. The two other options that include source port numbers (variable) cannot be tested by value because the port number varies for each connection.

The iptables rules to log HOST_ID_BOUCADAIR range values have been verified successfully for all four HOST_ID_BOUCADAIR options.

8.5. Dropping a specific HOST_ID Option Value

The same testing methodology described in the previous section was repeated to drop packets matching HOST_ID value (or a range of values); e.g. to drop SYN packets matching a particular HOST_ID_WING value:

```
iptables -A INPUT -p tcp --syn -m hostid --hostid_wing value[:value]
 -j DROP
```

In this testing, the HOST_ID_WING option is enabled at the CGN level. After applying the previous rule where Wing's specified value corresponds to the HOST_ID_WING data value (last 16 bits of the host's IPv4 source address), the hosts tries to access HTTP pages of the local server. It sends SYN packets but the server does not respond. Because this packet matches the iptables matching value, the corresponding rule is applied to the SYN packets: a SYN packet is dropped so the host does not receive any packet in return.

When the host is still trying to retrieve pages by sending SYN packets, the command 'iptables -F' will flush all iptables rules. Once applied, this command will let the host retrieve the required pages and the connection is therefore established successfully.

The same testing was repeated for HOST_ID_BOUCADAIR options. SYN packets matching the corresponding rule value or range of values were dropped. Once iptables rules are flushed, connection is established normally.

9. IANA Considerations

This document makes no request of IANA.

10. Security Considerations

Security considerations discussed in [I-D.wing-nat-reveal-option] should be taken into account.

11. Acknowledgments

Many thanks to M. Meulle, P. Ng Tung and L. Valeyre for their help and review. Special thanks to C. Jacquenet for his careful review.

12. References

12.1. Normative References

[I-D.wing-nat-reveal-option]
Yourtchenko, A. and D. Wing, "Revealing hosts sharing an IP address using TCP option", draft-wing-nat-reveal-option-03 (work in progress), December 2011.

[RFC6250] Thaler, D., "Evolution of the IP Model", RFC 6250, May 2011.

12.2. Informative References

[I-D.boucadair-intarea-nat-reveal-analysis]
Boucadair, M., Touch, J., Levis, P., and R. Penno, "Analysis of Solution Candidates to Reveal a Host Identifier in Shared Address Deployments", draft-boucadair-intarea-nat-reveal-analysis-04 (work in

progress), September 2011.

[RFC6269] Ford, M., Boucadair, M., Durand, A., Levis, P., and P. Roberts, "Issues with IP Address Sharing", RFC 6269, June 2011.

Authors' Addresses

Elie Abdo
France Telecom
Issy-les-Moulineaux

Email: elie.abdo@orange.com

Mohamed Boucadair
France Telecom

Email: mohamed.boucadair@orange.com

Jaqueline Queiroz
France Telecom
Issy-les-Moulineaux

Email: jaqueline.queiroz@orange.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: March 5, 2012

M. Boucadair
France Telecom
J. Touch
USC/ISI
P. Levis
France Telecom
R. Penno
Juniper Networks

September 2, 2011

Analysis of Solution Candidates to Reveal a Host Identifier in Shared Address Deployments
draft-boucadair-intarea-nat-reveal-analysis-04

Abstract

This document analyzes a set of solution candidates which have been proposed to mitigate some of the issues encountered when address sharing is used. In particular, this document focuses on means to reveal a host identifier when a Carrier Grade NAT (CGN) or application proxies are involved in the path. This host identifier must be unique to each host under the same shared IP address.

The ultimate goal is to assess the viability of proposed solutions and hopefully to make a recommendation on the more suitable solution(s).

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

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 http://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 March 5, 2012.

Table of Contents

1. Introduction

As reported in [RFC6269], several issues are encountered when an IP address is shared among several subscribers. Examples of such issues are listed below:

- Implicit identification (Section 13.2 of [RFC6269])
- SPAM (Section 13.3 of [RFC6269])
- Blacklisting a mis-behaving user (Section 13.1 of [RFC6269])
- Redirect users with infected machines to a dedicated portal (Section 5.1 of [RFC6269])

The sole use of the IPv4 address is not sufficient to uniquely distinguish a host. As a mitigation, it is tempting to investigate means which would help in disclosing an information to be used by the remote server as a means to uniquely disambiguate packets of hosts using the same IPv4 address.

The risk of not mitigating these issues are: OPEX increase for IP connectivity service providers (costs induced by calls to a hotline), revenue loss for content providers (loss of users audience), customers unsatisfaction (low quality of experience, service segregation, etc.).

1.1. Problem to Be Solved

Observation: Today, servers use the source IPv4 address as an identifier to treat some incoming connections differently. Tomorrow, due to the introduction of CGNs (e.g., NAT44 [I-D.ietf-behave-lsn-requirements], NAT64 [RFC6146]), that address will be shared. In particular, when a server receives packets from the same source address. Because this address is shared, the server does not know which host is the sending host.

Objective: The server should be able to sort out the packets by sending host.

Requirement: The server must have extra information than the source IP address to differentiate the sending host. We call HOST_ID this information.

For all solutions analyzed, we provide answers to the following questions:

What is the HOST_ID? It must be unique to each host under the same IP address. It does not need to be globally unique. Of course, the combination of the (public) IPv4 source address and the identifier (i.e., HOST_ID) ends up being relatively unique. As unique as today’s 32-bit IPv4 addresses which, today, can change

when a host re-connects.

Where is the HOST_ID? (which protocol, which field): If the HOST_ID is put at the IP level, all packets will have to bear the identifier. If it is put at a higher connection-oriented level, the identifier is only needed once in the session establishment phase (for instance TCP three-way-handshake), then, all packets received in this session will be attributed to the HOST_ID designated during the session opening.

Who puts the HOST_ID? For almost all the analyzed solutions, the address sharing function injects the HOST_ID. When there are several address sharing functions in the data path, we describe to what extent the proposed solution is efficient. Another option to avoid potential performance degradation is to let the host inject its HOST_ID but the address sharing function will check its content (just like an IP anti-spoofing function).

What are the security considerations? Security considerations are common to all analyzed solutions (see Section 5). Privacy-related aspect are discussed in Section 1.2.

1.2. HOST_ID and Privacy

HOST_ID provides an additional information to uniquely disambiguate a host among those sharing the same IP address. Unlike URIs, HOST_ID does not leak user’s identity information.

The HOST_ID does not reveal more privacy information than what the source IP address does in a non-shared address environment (see [I-D.morris-privacy-considerations]).

The volatility of the HOST_ID information is similar to the source IP address: a distinct HOST_ID may be used by the address sharing function when the host reboots or gets a new internal IP address. If the HOST_ID is persistent it may be used to track a host (similar to persistent IP addresses).

The trust on the information conveyed in the HOST_ID is likely to be the same as for current practices with the source IP address. In that sense, a HOST_ID can be spoofed as this is also the case for spoofing an IP address.

It is the responsibility of the remote server to rely or not on the content of the HOST_ID to enforce its policies and to log or not the content conveyed in the HOST_ID.

Enabling explicit identification means and adequate security suite is

more robust than relying on source IP address or HOST_ID. But tension may appear between strong privacy and usability (see Section 4.2 of [I-D.iab-privacy-workshop]).

1.3. IPv6 May Also Be Concerned

Issues similar to the ones described in Section 1.1 may be encountered also in an IPv6 environment (e.g., when the same /64 is used among several hosts).

1.4. Purpose and Scope

The purpose of this document is to analyze the solutions that have been proposed so far and to assess to what extent they solve the problem (see Section 1.1).

The purpose of this document is not to argue in favor of mandating the use of a HOST_ID but to document encountered issues, proposed solutions and their limitations.

Only IPv4-based solutions are analyzed in the following sections:

- define a new IP option (Section 3.1)
- define a new TCP option (Section 3.2)
- use the Identification field of IP header (denoted as IP-ID, Section 3.3)
- inject application headers (Section 3.4)
- enable Proxy Protocol ((Section 3.5))
- use of port set (Section 3.6)
- activate HIP (Section 3.7).

2. Recommendations

The following Table 1 summarizes the approaches analyzed in this document.

- "Success ratio" indicates the ratio of successful communications when the option is used. Provided figures are inspired from the results documented in [Options].
- "Deployable today" indicates if the solution can be generalized without any constraint on current architectures and practices.
- "Possible Perf Impact" indicates the level of expected performance degradation. The rationale behind the indicated potential performance degradation is whether the injection requires some treatment at the IP level or not.

o "OS TCP/IP Modif" indicates whether a modification of the OS TCP/IP stack is required at the server side.

	IP Option	TCP Option	IP-ID	HTTP Header (XFF)	Proxy Protocol	Port Set	HIP
UDP	Yes	No	Yes	No	No	Yes	
TCP	Yes	Yes	Yes	No	Yes	Yes	
HTTP	Yes	Yes	Yes	Yes	Yes	Yes	
Encrypted Traffic	Yes	Yes	Yes	No	Yes	Yes	
Success Ratio	30%	99%	100%	100%	Low	100%	Low
Possible Perf Impact	High	Med to High	Low to Med	Med to High	High	No	N/A
OS TCP/IP Modif	Yes	Yes	Yes	No	No	No	
Deployable Today	Yes	Yes	Yes	Yes	No	Yes	No
Notes			(1)	(2)		(1) (3)	(4) (5)

Table 1: Summary of analyzed solutions.

Notes for the above table:
(1) Requires mechanism to advertise NAT is participating in this scheme (e.g., DNS PTR record)
(2) This solution is widely deployed
(3) When the port set is not advertised, the solution is less efficient for third-party services.
(4) Requires the client and the server to be HIP-compliant and HIP infrastructure to be deployed.

(5) If the client and the server are HIP-enabled, the address sharing function does not need to insert a host-hint. If the client is not HIP-enabled, designing the device that performs address sharing to act as a UDP/TCP-HIP relay is not viable.

According to the above table and the analysis elaborated in Section 3:

- IP Option, IP-ID and Proxy Protocol proposals are broken;
- HIP is not largely deployed;
- The use of Port Set may contradict the port randomization [RFC6056] requirement identified in [RFC6269]. This solution can be used by a service provider for the delivery of its own service offerings relying on implicit identification.
- XFF is de facto standard deployed and supported in operational networks (e.g., HTTP Severs, Load-Balancers, etc.).
- From an application standpoint, the TCP Option is superior to XFF since it is not restricted to HTTP. Nevertheless XFF is compatible with the presence of address sharing and load-balancers in the communication path. To provide a similar functionality, the TCP Option may be extended to allow conveying a list of IP addresses to not loose the source IP address in the presence of load-balancers. Note that TCP Option requires the modification of the OS TCP/IP stack of remote servers; which can be seen as a blocking point.

As a conclusion of this analysis, the following recommendation is made:

[Hopefully to be completed]

3. Solutions Analysis

3.1. Define an IP Option

3.1.1. Description

This proposal aims to define an IP option [RFC0791] to convey a "host identifier". This identifier can be inserted by the address sharing function to uniquely distinguish a host among those sharing the same IP address. The option can convey an IPv4 address, the prefix part of an IPv6 address, etc.

Another way for using IP option has been described in Section 4.6 of [RFC3022].

3.1.2. Analysis

Unlike the solution presented in Section 3.2, this proposal can apply for any transport protocol. Nevertheless, it is widely known that routers (and other middle boxes) filter IP options. IP packets with IP options can be dropped by some IP nodes. Previous studies demonstrated that "IP Options are not an option" (Refer to [Not_An_Option], [Options]).

As a conclusion, using an IP option to convey a host-hint is not viable.

3.2. Define a TCP Option

3.2.1. Description

This proposal [I-D.wing-nat-reveal-option] defines a new TCP option called USER_HINT. This option encloses the TCP client's identifier (e.g., the lower 16 bits of their IPv4 address, their VLAN ID, VRF ID, subscriber ID). The address sharing device inserts this TCP option to the TCP SYN packet.

3.2.2. Analysis

The risk related to handling a new TCP option is low as measured in [Options].

[I-D.wing-nat-reveal-option] discusses the interference with other TCP options.

Using a new TCP option to convey the host-hint does not require any modification to the applications but it is applicable only for TCP-based applications. Applications relying on other transport protocols are therefore left unsolved.

Some downsides have been raised against defining a TCP option to reveal a host identity:

- Conveying an IP address in a TCP option may be seen as a violation of OSI layers but since IP addresses are already used for the checksum computation, this is not seen as a blocking point. Moreover, Updated version of [I-D.wing-nat-reveal-option] does not allow anymore to convey an IP address (the HOST_ID is encoded in 16bits).

- TCP option space is limited, and might be consumed by the TCP client. Earlier versions of [I-D.wing-nat-reveal-option] discuss two approaches to sending the HOST_ID: sending the HOST_ID in the TCP SYN (which consumes more bytes in the TCP header of the TCP SYN) and sending the HOST_ID in a TCP ACK (which consumes only two bytes in the TCP SYN). Content providers may find it more desirable to receive the HOST_ID in the TCP SYN, as that more closely preserves the host hint received in the source IP address as per current practices. It is more complicated to implement sending the HOST_ID in a TCP ACK, as it can introduce MTU issues if the ACK packet also contains TCP data, or a TCP segment is lost. The latest specification of the HOST_ID TCP Option, documented at [I-D.wing-nat-reveal-option], allows only to enclose the HOST_ID in the TCP SYN packet.

- When there are several NATs in the path, the original HOST_ID may be lost. In such case, the procedure may not be efficient.

- Interference with current usages such as X-Forwarded-For (see Section 3.4) should be elaborated to specify the behavior of servers when both options are used; in particular specify which information to use: the content of the TCP option or what is conveyed in the application headers.

- When load-balancers or proxies are in the path, this option does not allow to preserve the original source IP address and source. Preserving such information is required for logging purposes for instance.

3.3. Use the Identification Field of IP Header (IP-ID)

3.3.1. Description

IP-ID (Identification field of IP header) can be used to insert an information which uniquely distinguishes a host among those sharing the same IPv4 address. An address sharing function can re-write the IP-ID field to insert a value unique to the host (16 bits are sufficient to uniquely disambiguate hosts sharing the same IP address). Note that this field is not altered by some NATs; hence some side effects such as counting hosts behind a NAT as reported in [Count].

A variant of this approach relies upon the format of certain packets, such as TCP SYN, where the IP-ID can be modified to contain a 16 bit host-hint. Address sharing devices performing this function would require to indicate they are performing this function out of band, possibly using a special DNS record.

3.3.2. Analysis

This usage is not compliant with what is recommended in [I-D.ietf-intarea-ipv4-id-update].

3.4. Inject Application Headers

3.4.1. Description

Another option is to not require any change at the transport nor the IP levels but to convey at the application payload the required information which will be used to disambiguate hosts. This format and the related semantics depend on its application (e.g., HTTP, SIP, SMTP, etc.).

For HTTP, the X-Forwarded-For (XFF) header can be used to display the original IP address when an address sharing device is involved. Service Providers operating address sharing devices can enable the feature of injecting the XFF header which will enclose the original IPv4 address or the IPv6 prefix part. The address sharing device has to strip all included XFF headers before injecting their own. Servers may rely on the contents of this field to enforce some policies such as blacklisting misbehaving users. Note that XFF can also be logged by some servers (this is for instance supported by Apache).

3.4.2. Analysis

Not all applications impacted by the address sharing can support the ability to disclose the original IP address. Only a subset of protocols (e.g., HTTP) can rely on this solution.

For the HTTP case, to prevent users injecting invalid host-hints, an initiative has been launched to maintain a list of trusted ISPs using XFF: See for example the list available at: [Trusted_ISPs] of trusted ISPs as maintained by Wikipedia. If an address sharing device is on the trusted XFF ISPs list, users editing Wikipedia located behind the address sharing device will appear to be editing from their "original" IP address and not from the NATed IP address. If an offending activity is detected, individual hosts can be blacklisted instead of all hosts sharing the same IP address.

XFF header injection is a common practice of load balancers. When a load balancer is in the path, the original content of any included XFF header should not be stripped. Otherwise the information about the "origin" IP address will be lost.

When several address sharing devices are crossed, XFF header can

convey the list of IP addresses. The origin HOST_ID can be exposed to the target server.

XFF also introduces some implementation complexity if the HTTP packet is at or close to the MTU size.

It has been reported that some "poor" implementation may encounter some parsing issues when injecting XFF header.

For encrypted HTTP traffic, injecting XFF header may be broken.

3.5. PROXY Protocol

3.5.1. Description

The solution, referred to as Proxy Protocol [Proxy], does not require any application-specific knowledge. The rationale behind this solution is to prepend each connection with a line reporting the characteristics of the other side's connection as shown in the example below (excerpt from [Proxy]):

```
PROXY TCP4 198.51.100.1 198.51.100.11 56324 443\r\n
```

Upon receipt of a message conveying this line, the server removes the line. The line is parsed to retrieve the transported protocol. The content of this line is recorded in logs and used to enforce policies.

3.5.2. Analysis

This solution can be deployed in a controlled environment but it can not be deployed to all access services available in the Internet. If the remote server does not support the Proxy Protocol, the session will fail. Other complications will raise due to the presence of firewalls for instance.

As a consequence, this solution is broken and can not be recommended.

3.6. Enforce a Source-based Selection Algorithm at the Server Side (Port Set)

3.6.1. Description

This solution proposal does not require any action from the address sharing function to disclose a host identifier. Instead of assuming all the ports are associated with the same host, a random-based algorithm (or any port selection method) is run to generate the set of ports (including the source port of the received packet). The

length of the ports set to be generated by the server may be configurable (e.g., 8, 32, 64, 512, 1024, etc.). Instead of a random-based scheme, the server can use contiguous port ranges to form the port sets.

The server may reduce (or enlarge) the width of the ports set of the misbehaving action is (not) mitigated.

A variant of this proposal is to announce by off-line means the port set assignment policy of an operator. This announcement is not required for the delivery of internal services (i.e., offered by the service provider deploying the address sharing function) relying on implicit identification.

3.6.2. Analysis

In nominal mode, no coordination is required between the address sharing function and the server side but the efficiency of the method depends on the port set selection algorithm.

The method is more efficient if the provider that operates the address sharing device advertises its port assignment policy but this may contradicts the port randomization as identified in [RFC6269].

The method is deterministic for the delivery of services offered by the service provider offering also the IP connectivity service.

3.7. Host Identity Protocol (HIP)

3.7.1. Description

[RFC5201] specifies an architecture which introduces a new namespace to convey an identity information.

3.7.2. Analysis

This solution requires both the client and the server to support HIP [RFC5201]. Additional architectural considerations are to be taken into account such as the key exchanges, etc.

If the address sharing function is required to act as a UDP/TCP-HIP relay, this is not a viable option.

4. IANA Considerations

This document does not require any action from IANA.

5. Security Considerations

The same security concerns apply for the injection of an IP option, TCP option and application-related content (e.g., XFF) by the address sharing device. If the server trusts the content of the HOST_ID field, a third party user can be impacted by a misbehaving user to reveal a "faked" original IP address.

6. Acknowledgments

Many thanks to D. Wing and C. Jacquenet for their review, comments and inputs.

Thanks also to P. McCann, T. Tsou, Z. Dong, B. Briscoe, T. Taylor, M. Blanchet, D. Wing and A. Yourtchenko for the discussions in Prague.

Some of the issues related to defining a new TCP option have been raised by L. Eggert.

7. References

7.1. Normative References

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January 2001.

[RFC6056] Larsen, M. and F. Gont, "Recommendations for Transport-Protocol Port Randomization", BCP 156, RFC 6056, January 2011.

7.2. Informative References

[Count] "A technique for counting NATted hosts", <http://www.cs.columbia.edu/~smb/papers/fnat.pdf>.

[I-D.iab-privacy-workshop] Cooper, A., "Report from the Internet Privacy Workshop", draft-iab-privacy-workshop-00 (work in progress), June 2011.

[I-D.ietf-behave-lsn-requirements]
Perreault, S., Yamagata, I., Miyakawa, S., Nakagawa, A., and H. Ashida, "Common requirements for Carrier Grade NAT (CGN)", draft-ietf-behave-lsn-requirements-03 (work in progress), August 2011.

[I-D.ietf-intarea-ipv4-id-update]
Touch, J., "Updated Specification of the IPv4 ID Field", draft-ietf-intarea-ipv4-id-update-02 (work in progress), March 2011.

[I-D.morris-privacy-considerations]
Aboba, B., Morris, J., Peterson, J., and H. Tschofenig, "Privacy Considerations for Internet Protocols", draft-morris-privacy-considerations-03 (work in progress), March 2011.

[I-D.wing-nat-reveal-option]
Yourtchenko, A. and D. Wing, "Revealing hosts sharing an IP address using TCP option", draft-wing-nat-reveal-option-02 (work in progress), June 2011.

[Not_An_Option]
R. Fonseca, G. Porter, R. Katz, S. Shenker, and I. Stoica,, "IP options are not an option", 2005, <http://www.eecs.berkeley.edu/Pubs/TechRpts/2005/EECS-2005-24.html>.

[Options] Alberto Medina, Mark Allman, Sally Floyd, "Measuring Interactions Between Transport Protocols and Middleboxes", 2005, <http://conferences.sigcomm.org/imc/2004/papers/p336-medina.pdf>.

[Proxy] Tarreau, W., "The PROXY protocol", November 2010, <http://haproxy.1wt.eu/download/1.5/doc/proxy-protocol.txt>.

[RFC5201] Moskowitz, R., Nikander, P., Jokela, P., and T. Henderson, "Host Identity Protocol", RFC 5201, April 2008.

[RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, April 2011.

[RFC6269] Ford, M., Boucadair, M., Durand, A., Levis, P., and P. Roberts, "Issues with IP Address Sharing", RFC 6269, June 2011.

[Trusted_ISPs]
"Trusted XFF list", <http://meta.wikimedia.org/wiki/XFF_project#Trusted_XFF_list>.

Authors' Addresses

Mohamed Boucadair
France Telecom
Rennes, 35000
France

Email: mohamed.boucadair@orange-ftgroup.com

Joe Touch
USC/ISI

Email: touch@isi.edu

Pierre Levis
France Telecom
Caen, 14000
France

Email: pierre.levis@orange-ftgroup.com

Reinaldo Penno
Juniper Networks
1194 N Mathilda Avenue
Sunnyvale, California 94089
USA

Email: rpenno@juniper.net

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 20, 2014

D. Cheng
Huawei Technologies
J. Korhonen
Broadcom
M. Boucadair
France Telecom
S. Sivakumar
Cisco Systems

December 17, 2013

RADIUS Extensions for Port Set Configuration and Reporting
draft-cheng-behave-cgn-cfg-radius-ext-07

Abstract

This document defines new RADIUS attributes that can be used by a device implementing port ranges to communicate with a RADIUS server to configure and/or report TCP/UDP port sets and ICMP identifiers mapping behavior for specific hosts. This mechanism can be used in various deployment scenarios such as CGN, NAT64, Provider WiFi Gateway, etc.

This document does not make any assumption about the deployment context.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

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 http://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 June 20, 2014.

Table of Contents

1. Introduction

In a broadband network, customer information is usually stored on a RADIUS server [RFC2865] and at the time when a user initiates an IP connection request, the RADIUS server will populate the user's configuration information to the Network Access Server (NAS), which is usually co-located with the Border Network Gateway (BNG), after

the connection request is granted. The Carrier Grade NAT (CGN) function may also implemented on the BNG, and therefore CGN TCP/UDP port (or ICMP identifier) mapping behavior can be configured on the RADIUS server as part of the user profile, and populated to the NAS in the same manner. In addition, during the operation, the CGN can also convey port/identifier mapping behavior specific to a user to the RADIUS server, as part of the normal RADIUS accounting process.

The CGN device that communicates with a RADIUS server using RADIUS extensions defined in this document may perform NAT44 [RFC3022], NAT64 [RFC6146], or Dual-Stack Lite AFTR [RFC6333] function.

For the CGN example, when IP packets traverse a CGN, it would perform TCP/UDP source port mapping or ICMP identifier mapping as required. A TCP/ UDP source port or ICMP identifier, along with source IP address, destination IP address, destination port and protocol identifier if applicable, uniquely identify a session. Since the number space of TCP/UDP ports and ICMP identifiers in CGN's external realm is shared among multiple users assigned with the same IPv4 address, the total number of a user's simultaneous IP sessions is likely to subject to port quota.

The attributes defined in this document may also be used to report the assigned port set in some deployment such as Provider Wi-Fi [I-D.gundavelli-v6ops-community-wifi-svcs]. For example, a visiting host can be managed by a CPE which will need to report the assigned port set to the service platform. This is required for identification purposes (see WT-146 for example).

This document proposes three new RADIUS attributes as RADIUS protocol's extensions, and they are used for separate purposes as follows:

- A session limit is configured on a RADIUS server based on service agreement with a subscriber, and this parameter imposes the limit of total number of TCP/UDP ports and/or ICMP identifiers that the subscriber can use. Alternately, a separate session limit may be configured to limit the number of TCP ports, UDP ports, or the sum of the two, and ICMP identifiers, respectively, that the user can use. The session limit is carried by a new RADIUS attribute Port-Session-Limit, which is included in a RADIUS Access-Accept message sent by the RADIUS server to port-based device. This new RADIUS attribute can also be included in a RADIUS CoA message sent by the RADIUS server to the port-based device in order to change the session limit previously configured.

- A port-based device may allocate or de-allocate a set of TCP/UDP ports or ICMP identifiers for a specific subscriber. When it does

so, the associated session range along with the shared IPv4 address can be conveyed to the RADIUS server as part of the accounting process. These parameters are carried by a new RADIUS attribute Port-Session-Range, which is included in a RADIUS Accounting- Request message sent by the port-based device to the RADIUS server.

- A user may require the port-based device to perform port forwarding function, i.e., a port mapping is pre-configured on the port-based so that inbound IP packets sent by some applications from the port-based external realm can pass through that device and reach the user. The port mapping information includes the port-based device internal port, external port, and may also include the associated internal IPv4 or IPv6 address, and is carried by a new RADIUS attribute Port- Forwarding-Map, which is included in a RADIUS Access-Accept message sent by the RADIUS server to the port-based device. This new RADIUS attribute can also be included in a RADIUS CoA message sent by the RADIUS server to the port-based device in order to change the forwarding port mapping previously configured.

2. Terminology

Some terms that are used in this document are listed as follows:

- Session Limit - This is the maximum number of TCP ports, or UDP ports, or the total of the two, or ICMP identifiers, or the total of the three, that a device supporting port ranges can use when performing mapping on TCP/ UDP ports or ICMP identifiers for a specific user.

- Session Range - This specifies a set of TCP/UDP port numbers or ICMP identifiers, indicated by the port/identifier with the smallest numerical number and the port/identifier with the largest numerical number, inclusively.

- Internal IP Address - The IP address that is used as a source IP address in an outbound IP packet sent toward a device supporting port ranges in the internal realm. In IPv4 case, it is typically a private address [RFC1918].

- External IP Address - The IP address that is used as a source IP address in an outbound IP packet after traversing a device supporting port ranges in the external realm. In IPv4 case, it is typically a global and routable IP address.

- Internal Port - The internal port is a UDP or TCP port, or an ICMP identifier, which is allocated by a host or application behind a

device supporting port ranges for an outbound IP packet in the internal realm.

- o External Port - The external port is a UDP or TCP port, or an ICMP identifier, which is allocated by a device supporting port ranges upon receiving an outbound IP packet in the internal realm, and is used to replace the internal port that is allocated by a user or application.

- o External realm - The networking segment where IPv4 public addresses are used in respective of the device supporting port ranges.

- o Internal realm - The networking segment that is behind a device supporting port ranges and where IPv4 private addresses are used.

- o Mapping - This term in this document associates with a device supporting port ranges for a relationship between an internal IP address, internal port and the protocol, and an external IP address, external port, and the protocol.

- o Port-based device - A device that is capable of providing IP address and TCP/UDP port mapping services and in particular, with the granularity of one or more subsets within the 16-bit TCP/UDP port number range. A typical example of this device can be a CGN, CPE, Provider Wi-Fi Gateway, etc.

Note the terms "internal IP address", "internal port", "internal realm", "external IP address", "external port", "external realm", and "mapping" and their semantics are the same as in [RFC6887], and [RFC6888].

3. RADIUS Attributes

[Discussion: should these attributes be allocated from the extended RADIUS attribute code space?]

[Discussion: Should we define a dedicated attribute (port_set_policies) to configure the following policies: (1) enforce port randomization, (2) include/exclude the WKP in the port assignment, (3) preserve parity, (4) quota for explicit port mapping, (5) DSCP marking policy, (6) Port hold down timer, (7) port hold down pool, etc. Perhaps we don’t need to cover all these parameters.]

3.1. Port-Session-Limit Attribute

This attribute is of type complex [RFC6158] and specifies the limit of TCP ports, or UDP ports, or the sum of the two, or ICMP identifiers, or the sum of the three, which is configured on a device supporting port ranges corresponding to a specific subscriber.

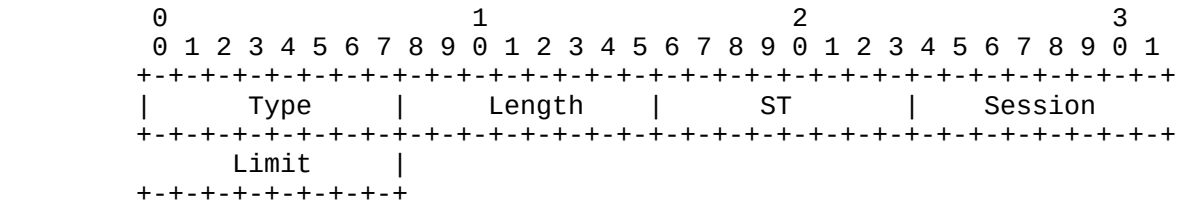
The Port-Session-Limit MAY appear in an Access-Accept packet, it MAY also appear in an Access-Request packet as a hint by the device supporting port ranges, which is co-allocated with the NAS, to the RADIUS server as a preference, although the server is not required to honor such a hint.

The Port-Session-Limit MAY appear in an CoA-Request packet.

The Port-Session-Limit MAY appear in an Accounting-Request packet.

The Port-Session-Limit MUST NOT appear in any other RADIUS packets.

The format of the Port-Session-Limit RADIUS attribute format is shown below. The fields are transmitted from left to right.



Type:

TBA1 for Port-Session-Limit.

Length:

5 octets. This field indicates the total length in octets of this attribute including the Type and the Length field.

ST (Session Type):

This one octet field contains an enumerated value that indicates the applicability of the Session Limit as follows:

0:

The limit as specified is applied to each transport protocol (TCP/UDP) and ICMP Identifiers as a whole.

1:

The limit as specified is applied to TCP and UDP ports.

2:

The limit as specified is applied to TCP ports.

3:

The limit as specified is applied to UDP ports.

4:

The limit as specified is applied to ICMP Identifiers.

5-255:

These values are undefined.

Session Limit:

This field contains the maximum number that is assigned to the transport sessions depending on the value in the Session Type (ST) field, that the specific user can use.

3.2. Port-Session-Range Attribute

This attribute is of type complex [RFC6158] and contains a range of numbers for TCP ports or UDP ports, or both, or for ICMP Identifiers, which has been allocated or de-allocated by a device supporting port ranges for a given subscriber, along with an external IPv4 address that is associated with any TCP/UDP port or ICMP identifier in the range.

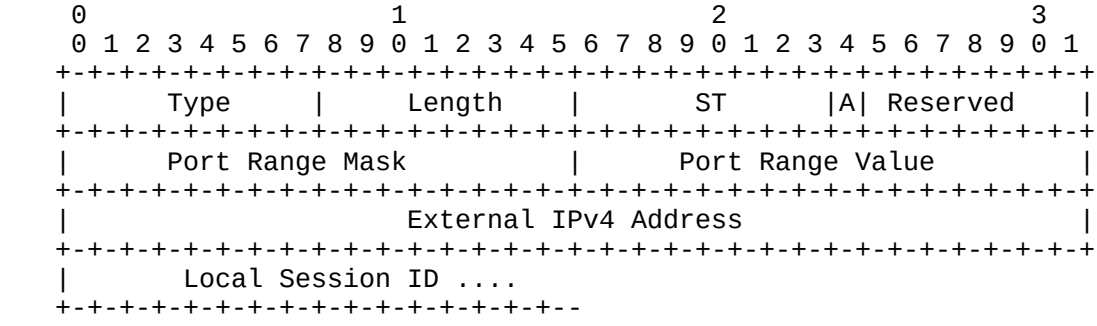
In some CGN deployment scenarios as described such as L2NAT [I-D.miles-behave-l2nat], DS-Extra-Lite [RFC6619] and Lightweight 4over6 [I-D.ietf-softwire-lw4over6], parameters at a customer premise such as MAC address, interface ID, VLAN ID, PPP session ID, IPv6 prefix, VRF ID, etc., may also be required to pass to the RADIUS server as part of the accounting record.

The Port-Session-Range MAY appear in an Accounting-Request packet.

The Port-Session-Range MUST NOT appear in any other RADIUS packets.

The port range follows the encoding specified in [RFC6431]; as such both contiguous and non-contiguous port sets are supported.

The format of the Port-Session-Range RADIUS attribute format is shown below. The fields are transmitted from left to right.



Type:

TBA2 for Port-Session-Range.

Length:

12 octets plus the length of optional field Local Session ID. This field indicates the total length in octets of this attribute including the Type and the Length field.

ST (Session Type):

This one octet field contains an enumerated value that indicates the semantics of the session range. The values follow the Session Type encoding defined in Section 3.1 except that the following values are not valid in scope of this attribute:

0:

The limit as specified is applied to the sum of TCP ports, UDP ports, and ICMP Identifiers as a whole.

A-bit Flag:

This field is set to 0 or 1, indicates that the session range has been allocated or de-allocated, respectively, by the device supporting port ranges.

Reserved:

This field MUST be set to zero by the sender and ignored by the receiver.

Port Range Mask:

The Port Range Mask indicates the position of the bits that are used to build the Port Range Value. By default, no PRM value is assigned. The 1 values in the Port Range Mask indicate by their position the significant bits of the Port Range Value. Refer to [RFC6431] for more details.

Port Range Value:

The PRV indicates the value of the significant bits of the Port Mask. By default, no PRV is assigned. Refer to [RFC6431] for more details.

External IPv4 Address:

This is an optional field. If present, this field contains the IPv4 address assigned to the associated subscriber to be used in the external realm. If set to 0/0, the allocation address policy is local to the device supporting port ranges.

Local Session ID:

This is an optional field and if presents, it contains a local session identifier at the customer premise, such as MAC address, interface ID, VLAN ID, PPP sessions ID, VRF ID, IPv6 address/ prefix, etc. The length of this field equals to the total attribute length minus 12 octets. If this field is not present, the port range policies must be enforced to all subscribers using a local subscriber identifier.

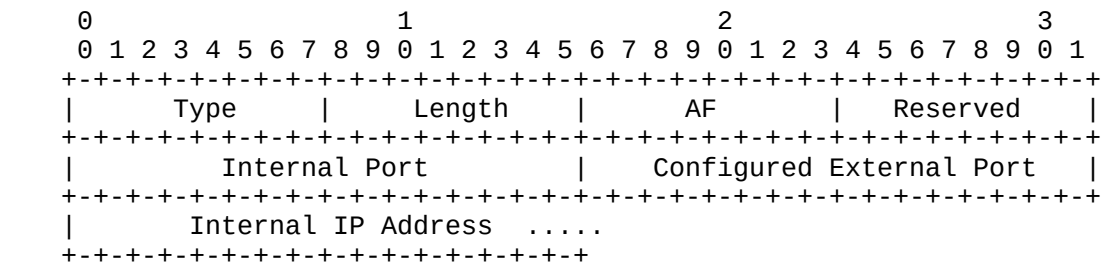
3.3. Port-Forwarding-Map Attribute

This attribute is type of complex [RFC6158] and contains a 16-bit Internal Port that identifies the source TCP/UDP port number of an IP packet sent by the user, or the destination port number of an IP packet destined to the user, and in both cases, the IP packet travels behind the NAT device. Also they contain a 16-bit Configured External Port that identifies the source TCP/UDP port number of an IP packet sent by the user, or the destination port number of an IP packet destined to the user, and in both cases, the IP packet travels outside of the NAT device. In addition, the attribute may contain a 32-bit IPv4 address or a 128-bit IPv6 address, respectively, as their respective NAT mappings internal IP address. Together, the port pair and IP address determine the port mapping rule for a specific IP flow that traverses a NAT device.

The attribute MAY appear in an Access-Accept packet, and may also appear in an Accounting-Request packet. In either case, the attribute MUST NOT appear more than once in a single packet.

The attribute MUST NOT appear in any other RADIUS packets.

The format of the Port-Forwarding-Map RADIUS attribute format is shown below. The fields are transmitted from left to right.



Type:

TBA3 for Port-Forwarding-Map.

Length:

This field indicates the total length in octets of this attribute including the Type and the Length field. Depending on the value of the AF field, the length could be 8, 12 or 24 octets.

AF (Address Family):

This one octet field contains a value that indicates address family of the internal IP address at the mapping as follows:

0:

There is no internal address attached.

1:

The internal address is an IPv4 address.

2:

The internal address is an IPv6 address.

3-255:

Unused.

[Discussion: should we use IANA assigned protocol numbers here?]

Reserved:

This field is set to zero by the sender and ignored by the receiver.

Internal Port:

This field contains the internal port for the CGN mapping.

Configured External Port:

This field contains the external port for the CGN mapping.

Internal IP Address:

This field may or may not present, and when it does, contains the internal IPv4 or IPv6 address for the CGN mapping.

4. Applications, Use Cases and Examples

This section describes some applications and use cases to illustrate the use of the RADIUS port set attributes.

4.1. Managing CGN Port Behavior using RADIUS

In a broadband network, customer information is usually stored on a RADIUS server, and the BNG hosts the NAS. The communication between the NAS and the RADIUS server is triggered by a subscriber when the user signs in to the Internet service, where either PPP or DHCP/DHCPv6 is used. When a user signs in, the NAS sends a RADIUS Access-Request message to the RADIUS server. The RADIUS server validates the request, and if the validation succeeds, it in turn sends back a RADIUS Access-Accept message. The Access-Accept message carries configuration information specific to that user, back to the NAS, where some of the information would pass on to the requesting user via PPP or DHCP/DHCPv6.

A CGN function in a broadband network would most likely reside on a BNG. In that case, parameters for CGN port/identifier mapping behavior for users can be configured on the RADIUS server. When a user signs in to the Internet service, the associated parameters can be conveyed to the NAS, and proper configuration is accomplished on the CGN device for that user.

Also, CGN operation status such as CGN port/identifier allocation and de-allocation for a specific user on the BNG can also be transmitted back to the RADIUS server for accounting purpose using the RADIUS protocol.

RADIUS protocol has already been widely deployed in broadband networks to manage BNG, thus the functionality described in this specification introduces little overhead to the existing network operation.

In the following sub-sections, we describe how to manage CGN behavior using RADIUS protocol, with required RADIUS extensions proposed in Section 3.

4.1.1. Configure CGN Session Limit

In the face of IPv4 address shortage, there are currently proposals to multiplex multiple subscribers' connections over a smaller number of shared IPv4 addresses, such as Carrier Grade NAT [RFC6888], Dual-Stack Lite [RFC6333], NAT64 [RFC6146], etc. As a result, a single IPv4 public address may be shared by hundreds or even thousands of subscribers. As indicated in [RFC6269], it is therefore necessary to impose limits on the total number of ports available to an individual subscriber to ensure that the shared resource, i.e., the IPv4 address remains available in some capacity to all the subscribers using it, and port limiting is also documented in [RFC6888] as a requirement.

There are two practical granularities to impose such a limit. One is to define a session limit that is imposed to the total number of TCP and UDP ports, plus the number of ICMP identifiers, for a specific subscriber. Alternatively, a session limit can be specified for the sum of TCP ports and UDP ports, or a separate session limit for TCP ports and UDP ports, respectively, and another session limit for ICMP identifiers.

The per-subscriber based session limit(s) is configured on a RADIUS server, along with other user information such as credentials. The value of these session limit(s) is based on service agreement and its specification is out of the scope of this document.

When a subscriber signs in to the Internet service successfully, the session limit(s) for the subscriber is passed to the BNG based NAS, where CGN also locates, using a new RADIUS attribute called Port-Session-Limit (defined in Section 3.1), along with other configuration parameters. While some parameters are passed to the subscriber, the session limit(s) is recorded on the CGN device for imposing the usage of TCP/UDP ports and ICMP identifiers for that subscriber.

Figure 1 illustrates how RADIUS protocol is used to configure the maximum number of TCP/UDP ports for a given subscriber on a NAT44 device.

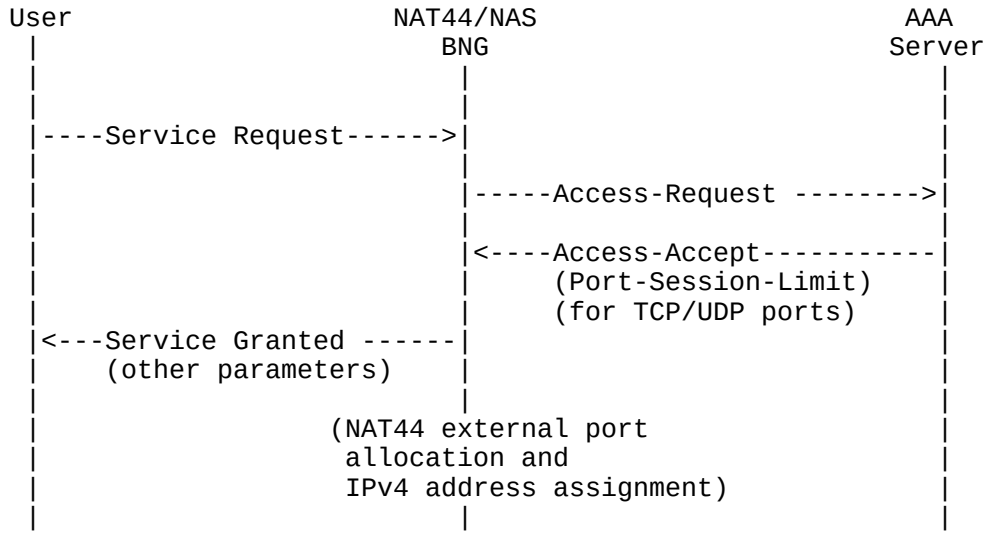


Figure 1: RADIUS Message Flow for Configuring NAT44 Port Limit

The session limit(s) created on a CGN device for a specific user using RADIUS extension may be changed using RADIUS CoA message [RFC5176] that carries the same RADIUS attribute. The CoA message may be sent from the RADIUS server directly to the NAS, which once accepts and sends back a RADIUS CoA ACK message, the new session limit replaces the previous one.

Figure 2 illustrates how RADIUS protocol is used to increase the TCP/UDP port limit from 1024 to 2048 on a NAT44 device for a specific user.

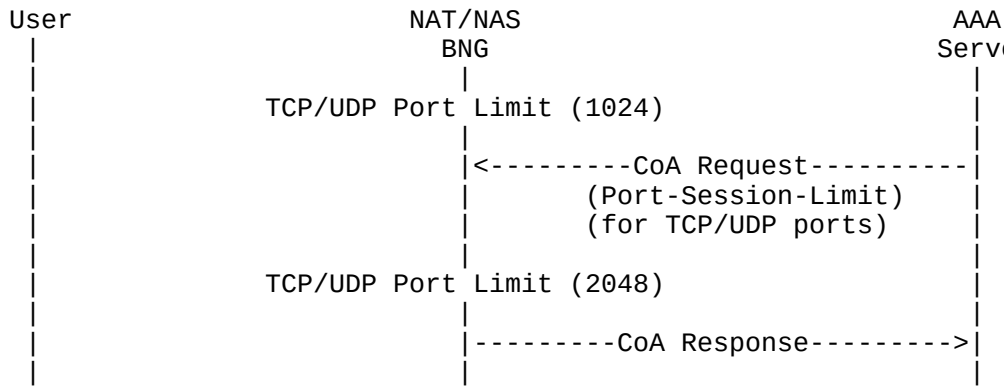


Figure 2: RADIUS Message Flow for changing a user's NAT44 port limit

4.1.2. Report CGN Session Allocation or De-allocation

Upon obtaining the session limit(s) for a subscriber, the CGN device needs to allocate a TCP/UDP port or an ICMP identifiers for the subscriber when receiving a new IP flow sent from that subscriber.

As one practice, a CGN may allocate a bulk of TCP/UDP ports or ICMP identifiers once at a time for a specific user, instead of one port/identifier at a time, and within each session bulk, the ports/identifiers may be randomly distributed or in consecutive fashion. When a CGN device allocates bulk of TCP/UDP ports and ICMP identifiers, the information can be easily conveyed to the RADIUS server by a new RADIUS attribute called the CGN-Session-Range (defined in Section 3.2). The CGN device may allocate one or more TCP/UDP port ranges or ICMP identifier ranges, or generally called session ranges, where each range contains a set of numbers representing TCP/UDP ports or ICMP identifiers, and the total number of sessions must be less or equal to the associated session limit defined for that subscriber. A CGN device may choose to allocate a small session range, and allocate more at a later time as needed; such practice is good because its randomization in nature.

At the same time, the CGN device also needs to decide the shared IPv4 address for that subscriber. The shared IPv4 address and the pre-allocated session range are both passed to the RADIUS server.

When a subscriber initiates an IP flow, the CGN device randomly selects a TCP/UDP port or ICMP identifier from the associated and pre-allocated session range for that subscriber to replace the original source TCP/UDP port or ICMP identifier, along with the replacement of the source IP address by the shared IPv4 address.

A CGN device may decide to "free" a previously assigned set of TCP/ UDP ports or ICMP identifiers that have been allocated for a specific subscriber but not currently in use, and with that, the CGN device must send the information of the de-allocated session range along with the shared IPv4 address to the RADIUS server.

Figure 3 illustrates how RADIUS protocol is used to report a set of ports allocated and de-allocated, respectively, by a NAT44 device for a specific user to the RADIUS server.

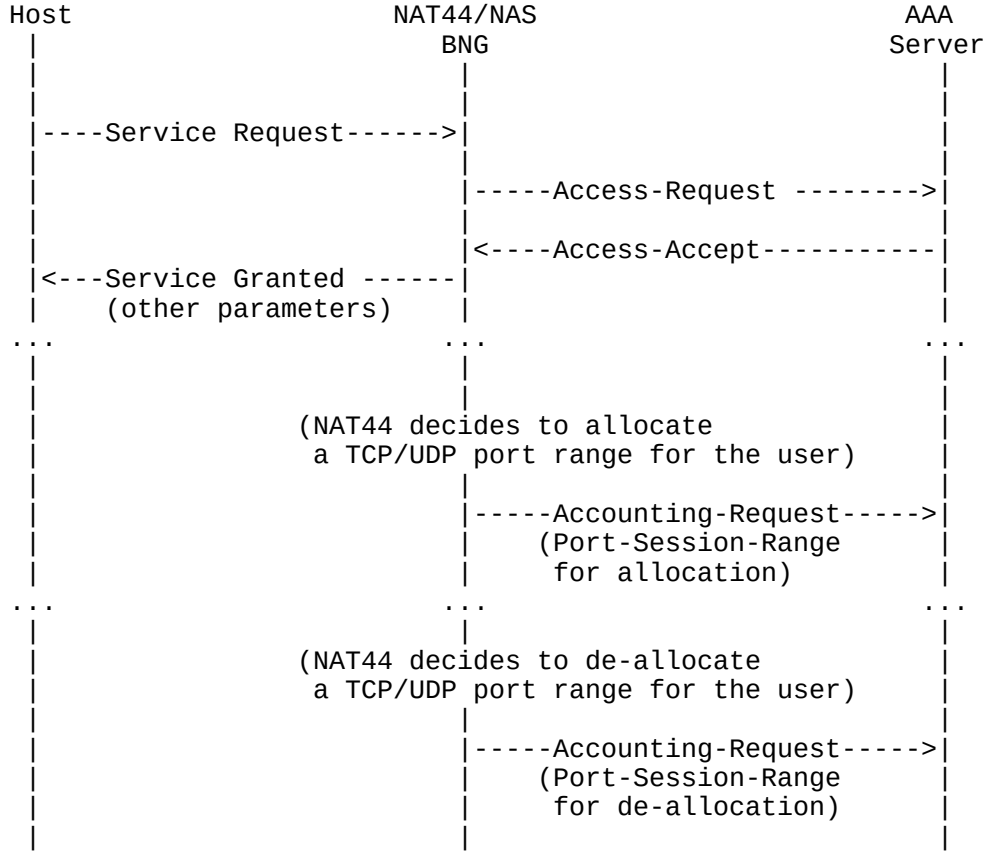


Figure 3: RADIUS Message Flow for reporting NAT44 allocation/de-allocation of a port set

4.1.3. Configure CGN Forwarding Port Mapping

In most scenarios, the port mapping on a NAT device is dynamically created when the IP packets of an IP connection initiated by a user arrives. For some applications, the port mapping needs to be pre-defined allowing IP packets of applications from outside a CGN device to pass through and "port forwarded" to the correct user located behind the CGN device.

Port Control Protocol [RFC6887], provides a mechanism to create a mapping from an external IP address and port to an internal IP address and port on a CGN device just to achieve the "port forwarding" purpose. PCP is a server-client protocol capable of creating or deleting a mapping along with a rich set of features on a CGN device in dynamic fashion. In some deployment, all users need is a few, typically just one pre-configured port mapping for applications such as web cam at home, and the lifetime of such a port mapping remains valid throughout the duration of the customer’s Internet service connection time. In such an environment, it is possible to statically configure a port mapping on the RADIUS server for a user and let the RADIUS protocol to propagate the information to the associated CGN device.

Figure 4 illustrates how RADIUS protocol is used to configure a forwarding port mapping on a NAT44 device by using RADIUS protocol.

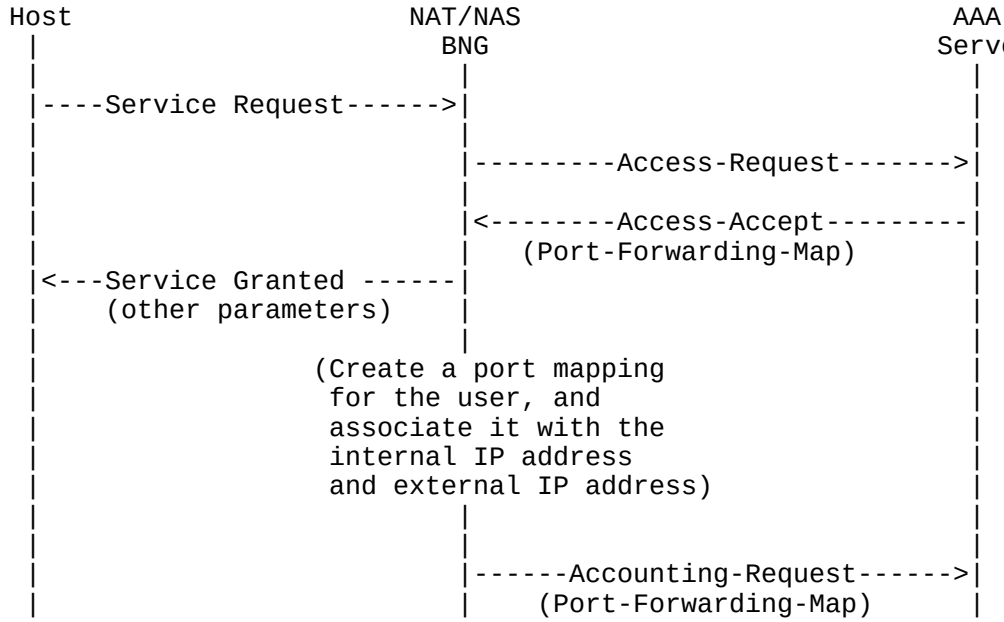


Figure 4: RADIUS Message Flow for configuring a forwarding port mapping

A port forwarding mapping that is created on a CGN device using RADIUS extension as described above may also be changed using RADIUS CoA message [RFC5176] that carries the same RADIUS associate. The CoA message may be sent from the RADIUS server directly to the NAS, which once accepts and sends back a RADIUS CoA ACK message, the new port forwarding mapping then replaces the previous one.

Figure 5 illustrates how RADIUS protocol is used to change an existing port mapping from (a:X) to (a:Y), where "a" is an internal port, and "X" and "Y" are external ports, respectively, for a specific user with a specific IP address

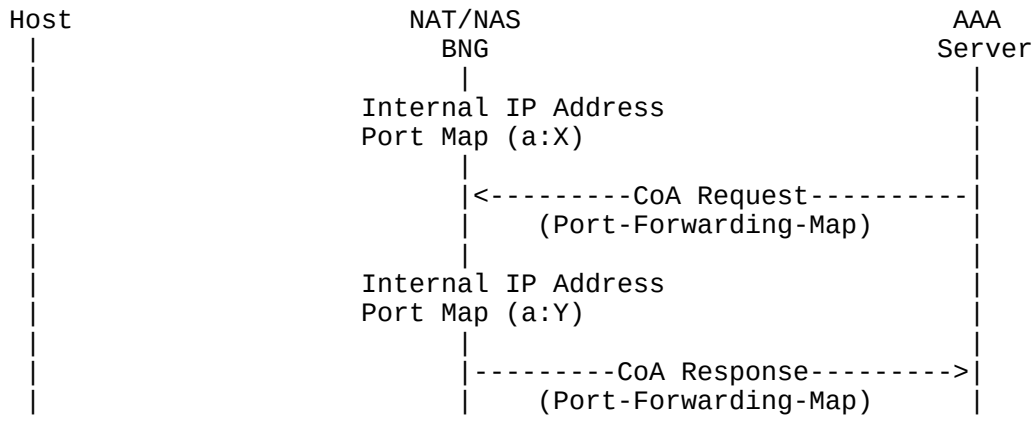


Figure 5: RADIUS Message Flow for changing a user's forwarding port mapping

4.1.4. An Example

An Internet Service Provider (ISP) assigns TCP/UDP 500 ports for the subscriber Joe. This number is the limit that can be used for TCP/UDP ports on a NAT44 device for Joe, and is configured on a RADIUS server. Also, Joe asks for a pre-defined port forwarding mapping on the NAT44 device for his web cam applications (external port 5000 maps to internal port 80).

When Joe successfully connects to the Internet service, the RADIUS server conveys the TCP/UDP port limit (1000) and the forwarding port mapping (external port 5000 to internal port 80) to the NAT44 device, using Port-Session-Limit attribute and Port-Forwarding-Map attribute, respectively, carried by an Access-Accept message to the BNG where NAS and CGN co-located.

Upon receiving the first outbound IP packet sent from Joe's laptop, the NAT44 device decides to allocate a small port pool that contains 40 consecutive ports, from 3500 to 3540, inclusively, and also assign a shared IPv4 address 192.0.2.15, for Joe. The NAT44 device also randomly selects one port from the allocated range (say 3519) and use that port to replace the original source port in outbound IP packets.

For accounting purpose, the NAT44 device passes this port range (3500-3540) and the shared IPv4 address 192.0.2.15 together to the RADIUS server using Port-Session-Range attribute carried by an Accounting-Request message.

When Joe works on more applications with more outbound IP sessions and the port pool (3500-3540) is close to exhaust, the NAT44 device

allocates a second port pool (8500-8800) in a similar fashion, and also passes the new port range (8500-8800) and IPv4 address 192.0.2.15 together to the RADIUS server using Port-Session-Range attribute carried by an Accounting-Request message. Note when the CGN allocates more ports, it needs to assure that the total number of ports allocated for Joe is within the limit.

Joe decides to upgrade his service agreement with more TCP/UDP ports allowed (up to 1000 ports). The ISP updates the information in Joe's profile on the RADIUS server, which then sends a CoA-Request message that carries the Port-Session-Limit attribute with 1000 ports to the NAT44 device; the NAT44 device in turn sends back a CoA-ACK message. With that, Joe enjoys more available TCP/UDP ports for his applications.

When Joe travels, most of the IP sessions are closed with their associated TCP/UDP ports released on the NAT44 device, which then sends the relevant information back to the RADIUS server using Port-Session-Range attribute carried by Accounting-Request message.

Throughout Joe's connection with his ISP Internet service, applications can communicate with his web cam at home from external realm directly traversing the pre-configured mapping on the CGN device.

When Joe disconnects from his Internet service, the CGN device will de-allocate all TCP/UDP ports as well as the port-forwarding mapping, and send the relevant information to the RADIUS server.

4.2. Report Assigned Port Set for a Visiting UE

Figure 6 illustrates an example of the flow exchange which occurs when a visiting UE connects to a CPE offering Wi-Fi service.

For identification purposes (see [RFC6967]), once the CPE assigns a port set, it issues a RADIUS message to report the assigned port set.

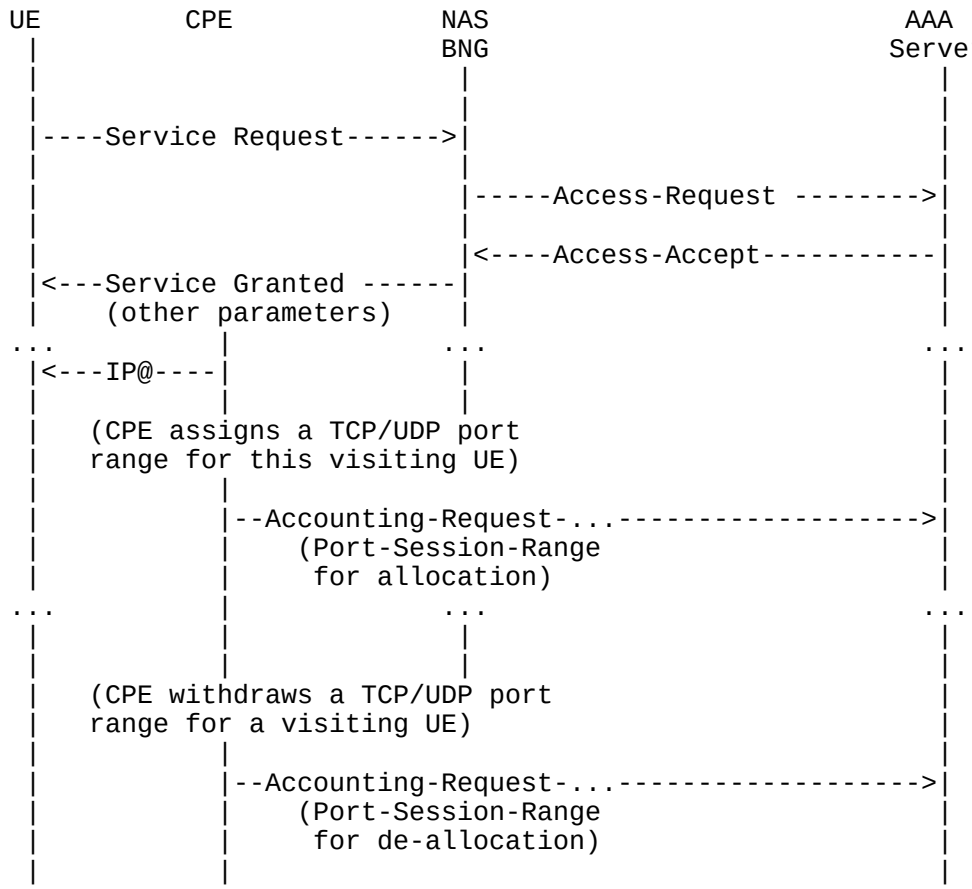


Figure 6: RADIUS Message Flow for reporting CPE allocation/de-allocation of a port set to a visiting UE

5. Table of Attributes

The following table provides a guide as the attributes may be found in which kinds of RADIUS packets, and in what quantity.

Request	Accept	Reject	Challenge	Acct. Request	#	Attribute
0-1	0-1	0	0	0-1	TBA1	Port-Session-Limit
0	0	0	0	0-1	TBA2	Port-Session-Range
0-1	0-1	0	0	0-1	TBA3	Port-Forwarding-Map

The following table defines the meaning of the above table entries.

0 This attribute MUST NOT be present in packet.
0+ Zero or more instances of this attribute MAY be present in packet.
0-1 Zero or one instance of this attribute MAY be present in packet.

6. Security Considerations

This document does not introduce any security issue than what has been identified in [RFC2865].

7. IANA Considerations

7.1. RADIUS Attributes

This document requires new code point assignment for the three new RADIUS attributes as follows:

- Port-Session-Limit
- Port-Session-Range
- Port-Forwarding-Map

7.2. Name Spaces

This document establishes a new name space for Session Type (see Section 3.1 for the initial reservation of values. The allocation of future values is according to RFC Required policy [RFC5226].

8. Acknowledgements

Many thanks to Dan Wing, Roberta Maglione, Daniel Derksen, and David Thaler for their useful comments and suggestions.

9. References

9.1. Normative References

[RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

[RFC5176] Chiba, M., Dommety, G., Eklund, M., Mitton, D., and B. Aboba, "Dynamic Authorization Extensions to Remote Authentication Dial In User Service (RADIUS)", RFC 5176, January 2008.

9.2. Informative References

[I-D.gundavelli-v6ops-community-wifi-svcs]
Gundavelli, S., Grayson, M., Seite, P., and Y. Lee, "Service Provider Wi-Fi Services Over Residential Architectures", draft-gundavelli-v6ops-community-wifi-svcs-06 (work in progress), April 2013.

[I-D.ietf-softwire-lw4over6]
Cui, Y., Qiong, Q., Boucadair, M., Tsou, T., Lee, Y., and I. Farrer, "Lightweight 4over6: An Extension to the DS-Lite Architecture", draft-ietf-softwire-lw4over6-03 (work in progress), November 2013.

[I-D.miles-behave-l2nat]
Miles, D. and M. Townsley, "Layer2-Aware NAT", draft-miles-behave-l2nat-00 (work in progress), March 2009.

[RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January 2001.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, April 2011.

[RFC6158] DeKok, A. and G. Weber, "RADIUS Design Guidelines", BCP 158, RFC 6158, March 2011.

[RFC6269] Ford, M., Boucadair, M., Durand, A., Levis, P., and P. Roberts, "Issues with IP Address Sharing", RFC 6269, June 2011.

[RFC6333] Durand, A., Droms, R., Woodyatt, J., and Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion", RFC 6333, August 2011.

[RFC6431] Boucadair, M., Levis, P., Bajko, G., Savolainen, T., and T. Tsou, "Huawei Port Range Configuration Options for PPP IP Control Protocol (IPCP)", RFC 6431, November 2011.

[RFC6619] Arkko, J., Eggert, L., and M. Townsley, "Scalable Operation of Address Translators with Per-Interface Bindings", RFC 6619, June 2012.

[RFC6887] Wing, D., Cheshire, S., Boucadair, M., Penno, R., and P. Selkirk, "Port Control Protocol (PCP)", RFC 6887, April 2013.

[RFC6888] Perreault, S., Yamagata, I., Miyakawa, S., Nakagawa, A., and H. Ashida, "Common Requirements for Carrier-Grade NATs (CGNs)", BCP 127, RFC 6888, April 2013.

[RFC6967] Boucadair, M., Touch, J., Levis, P., and R. Penno, "Analysis of Potential Solutions for Revealing a Host Identifier (HOST_ID) in Shared Address Deployments", RFC 6967, June 2013.

Authors' Addresses

Dean Cheng
Huawei Technologies
2330 Central Expressway
Santa Clara, California 95050
USA

Email: dean.cheng@huawei.com

Jouni Korhonen
Broadcom
Porkkalankatu 24
FIN-00180 Helsinki
Finland

Email: jouni.nospam@gmail.com

Mohamed Boucadair
France Telecom
Rennes
France

Email: mohamed.boucadair@orange.com

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

Email: ssenthil@cisco.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: June 14, 2015

C. Donley
CableLabs
C. Grundemann
Internet Society
V. Sarawat
K. Sundaresan
CableLabs
O. Vautrin
Juniper Networks

December 11, 2014

Deterministic Address Mapping to Reduce Logging in Carrier Grade NAT Deployments
draft-donley-behave-deterministic-cgn-09

Abstract

In some instances, Service Providers have a legal logging requirement to be able to map a subscriber's inside address with the address used on the public Internet (e.g. for abuse response). Unfortunately, many Carrier Grade NAT logging solutions require active logging of dynamic translations. Carrier Grade NAT port assignments are often per-connection, but could optionally use port ranges. Research indicates that per-connection logging is not scalable in many residential broadband services. This document suggests a way to manage Carrier Grade NAT translations in such a way as to significantly reduce the amount of logging required while providing traceability for abuse response. IPv6 is, of course, the preferred solution. While deployment is in progress, service providers are forced by business imperatives to maintain support for IPv4. This note addresses the IPv4 part of the network when a Carrier Grade NAT solution is in use.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

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 http://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 June 14, 2015.

Table of Contents

1. Introduction

It is becoming increasingly difficult to obtain new IPv4 address assignments from Regional/Local Internet Registries due to depleting supplies of unallocated IPv4 address space. To meet the growing demand for Internet connectivity from new subscribers, devices, and service types, some operators will be forced to share a single public IPv4 address among multiple subscribers using techniques such as Carrier Grade Network Address Translation (CGN) [RFC6264] (e.g., NAT444 [I-D.shirasaki-nat444], DS-Lite [RFC6333], NAT64 [RFC6146] etc.). However, address sharing poses additional challenges to operators when considering how they manage service entitlement, public safety requests, or attack/abuse/fraud reports [RFC6269]. In order to identify a specific user associated with an IP address in response to such a request or for service entitlement, an operator will need to map a subscriber's internal source IP address and source port with the global public IP address and source port provided by the CGN for every connection initiated by the user.

CGN connection logging satisfies the need to identify attackers and respond to abuse/public safety requests, but it imposes significant operational challenges to operators. In lab testing, we have observed CGN log messages to be approximately 150 bytes long for NAT444 [I-D.shirasaki-nat444], and 175 bytes for DS-Lite [RFC6333] (individual log messages vary somewhat in size). Although we are not aware of definitive studies of connection rates per subscriber, reports from several operators in the US sets the average number of connections per household at approximately 33,000 connections per day. If each connection is individually logged, this translates to a data volume of approximately 5 MB per subscriber per day, or about 150 MB per subscriber per month; however, specific data volumes may vary across different operators based on myriad factors. Based on available data, a 1-million subscriber service provider will generate approximately 150 terabytes of log data per month, or 1.8 petabytes per year. Note that many Service Providers compress log data after collection; compression factors of 2:1 or 3:1 are common.

The volume of log data poses a problem for both operators and the public safety community. On the operator side, it requires a significant infrastructure investment by operators implementing CGN. It also requires updated operational practices to maintain the logging infrastructure, and requires approximately 23 Mbps of bandwidth between the CGN devices and the logging infrastructure per 50,000 users. On the public safety side, it increases the time required for an operator to search the logs in response to an abuse report, and could delay investigations. Accordingly, an international group of operators and public safety officials

approached the authors to identify a way to reduce this impact while improving abuse response.

The volume of CGN logging can be reduced by assigning port ranges instead of individual ports. Using this method, only the assignment of a new port range is logged. This may massively reduce logging volume. The log reduction may vary depending on the length of the assigned port range, whether the port range is static or dynamic, etc. This has been acknowledged in [RFC6269], which recommends source port logging at the server and/or destination logging at the CGN and [I-D.sivakumar-behave-nat-logging], which describes information to be logged at a NAT.

However, the existing solutions still poses an impact on operators and public safety officials for logging and searching. Instead, CGNs could be designed and/or configured to deterministically map internal addresses to {external address + port range} in such a way as to be able to algorithmically calculate the mapping. Only inputs and configuration of the algorithm need to be logged. This approach reduces both logging volume and subscriber identification times. In some cases, when full deterministic allocation is used, this approach can eliminate the need for translation logging.

This document describes a method for such CGN address mapping, combined with block port reservations, that significantly reduces the burden on operators while offering the ability to map a subscriber's inside IP address with an outside address and external port number observed on the Internet.

The activation of the proposed port range allocation scheme is compliant with BEHAVE requirements such as the support of APP.

2. Deterministic Port Ranges

While a subscriber uses thousands of connections per day, most subscribers use far fewer resources at any given time. When the compression ratio (see Appendix B of RFC6269 [RFC6269]) is low (e.g., the ratio of the number of subscribers to the number of public IPv4 addresses allocated to a CGN is closer to 10:1 than 1000:1), each subscriber could expect to have access to thousands of TCP/UDP ports at any given time. Thus, as an alternative to logging each connection, CGNs could deterministically map customer private addresses (received on the customer-facing interface of the CGN, a.k.a., internal side) to public addresses extended with port ranges (used on the Internet-facing interface of the CGN, a.k.a., external side). This algorithm allows an operator to identify a subscriber internal IP address when provided the public side IP and port number without having to examine the CGN translation logs. This prevents an

operator from having to transport and store massive amounts of session data from the CGN and then process it to identify a subscriber.

The algorithmic mapping can be expressed as:

(External IP Address, Port Range) = function 1 (Internal IP Address)

Internal IP Address = function 2 (External IP Address, Port Number)

The CGN SHOULD provide a method for administrators to test both mapping functions (e.g., enter an External IP Address + Port Number and receive the corresponding Internal IP Address).

Deterministic Port Range allocation requires configuration of the following variables:

- Inside IPv4/IPv6 address range (I);
- Outside IPv4 address range (O);
- Compression ratio (e.g. inside IP addresses I/outside IP addresses O) (C);
- Dynamic address pool factor (D), to be added to the compression ratio in order to create an overflow address pool;
- Maximum ports per user (M);
- Address assignment algorithm (A) (see below); and
- Reserved TCP/UDP port list (R)

Note: The inside address range (I) will be an IPv4 range in NAT444 operation (NAT444 [I-D.shirasaki-nat444]) and an IPv6 range in DS-Lite operation (DS-Lite [RFC6333]).

A subscriber is identified by an internal IPv4 address (e.g., NAT44) or an IPv6 prefix (e.g., DS-Lite or NAT64).

The algorithm may be generalized to L2-aware NAT [I-D.miles-behave-l2nat] but this requires the configuration of the Internal interface identifiers (e.g., MAC addresses).

The algorithm is not designed to retrieve an internal host among those sharing the same internal IP address (e.g., in a DS-Lite context, only an IPv6 address/prefix can be retrieved using the

algorithm while the internal IPv4 address used for the encapsulated IPv4 datagram is lost).

Several address assignment algorithms are possible. Using predefined algorithms, such as those that follow, simplifies the process of reversing the algorithm when needed. However, the CGN MAY support additional algorithms. Also, the CGN is not required to support all algorithms described below. Subscribers could be restricted to ports from a single IPv4 address, or could be allocated ports across all addresses in a pool, for example. The following algorithms and corresponding values of A are as follow:

0: Sequential (e.g. the first block goes to address 1, the second block to address 2, etc.)

1: Staggered (e.g. for every n between 0 and ((65536-R)/(C+D))-1 , address 1 receives ports n*C+R, address 2 receives ports (1+n)*C+R, etc.)

2: Round robin (e.g. the subscriber receives the same port number across a pool of external IP addresses. If the subscriber is to be assigned more ports than there are in the external IP pool, the subscriber receives the next highest port across the IP pool, and so on. Thus, if there are 10 IP addresses in a pool and a subscriber is assigned 1000 ports, the subscriber would receive a range such as ports 2000-2099 across all 10 external IP addresses).

3: Interlaced horizontally (e.g. each address receives every Cth port spread across a pool of external IP addresses).

4: Cryptographically random port assignment (Section 2.2 of RFC6431 [RFC6431]). If this algorithm is used, the Service Provider needs to retain the keying material and specific cryptographic function to support reversibility.

5: Vendor-specific. Other vendor-specific algorithms may also be supported.

The assigned range of ports MAY also be used when translating ICMP requests (when re-writing the Identifier field).

The CGN then reserves ports as follows:

1. The CGN removes reserved ports (R) from the port candidate list (e.g., 0-1023 for TCP and UDP). At a minimum, the CGN SHOULD remove system ports (RFC6335) [RFC6335] from the port candidate list reserved for deterministic assignment.

2. The CGN calculates the total compression ratio (C+D), and allocates 1/(C+D) of the available ports to each internal IP address. Specific port allocation is determined by the algorithm (A) configured on the CGN. Any remaining ports are allocated to the dynamic pool.

   Note: Setting D to 0 disables the dynamic pool. This option eliminates the need for per-subscriber logging at the expense of limiting the number of concurrent connections that 'power users' can initiate.

3. When a subscriber initiates a connection, the CGN creates a translation mapping between the subscriber's inside local IP address/port and the CGN outside global IP address/port. The CGN MUST use one of the ports allocated in step 2 for the translation as long as such ports are available. The CGN SHOULD allocate ports randomly within the port range assigned by the deterministic algorithm. This is to increase subscriber privacy. The CGN MUST use the preallocated port range from step 2 for Port Control Protocol (PCP, [RFC6887]) reservations as long as such ports are available. While the CGN maintains its mapping table, it need not generate a log entry for translation mappings created in this step.

4. If D>0, the CGN will have a pool of ports left for dynamic assignment. If a subscriber uses more than the range of ports allocated in step 2 (but fewer than the configured maximum ports M), the CGN assigns a block of ports from the dynamic assignment range for such a connection or for PCP reservations. The CGN MUST log dynamically assigned port blocks to facilitate subscriber-to-address mapping. The CGN SHOULD manage dynamic ports as described in [I-D.tsou-behave-natx4-log-reduction].

5. Configuration of reserved ports (e.g., system ports) is left to operator configuration.

Thus, the CGN will maintain translation mapping information for all connections within its internal translation tables; however, it only needs to externally log translations for dynamically-assigned ports.

2.1. IPv4 Port Utilization Efficiency

For Service Providers requiring an aggressive address sharing ratio, the use of the algorithmic mapping may impact the efficiency of the address sharing. A dynamic port range allocation assignment is more suitable in those cases.

2.2. Planning & Dimensioning

Unlike dynamic approaches, the use of the algorithmic mapping requires more effort from operational teams to tweak the algorithm (e.g., size of the port range, address sharing ratio, etc.). Dedicated alarms SHOULD be configured when some port utilization thresholds are fired so that the configuration can be refined.

The use of algorithmic mapping also affects geolocation. Changes to the inside and outside address ranges (e.g. due to growth, address allocation planning, etc.) would require external geolocation providers to recalibrate their mappings.

2.3. Deterministic CGN Example

To illustrate the use of deterministic NAT, let’s consider a simple example. The operator configures an inside address range (I) of 198.51.100.0/28 [RFC6598] and outside address (O) of 192.0.2.1. The dynamic address pool factor (D) is set to ’2’. Thus, the total compression ratio is 1:(14+2) = 1:16. Only the system ports (e.g. ports < 1024) are reserved (R) . This configuration causes the CGN to preallocate ((65536-1024)/16 =) 4032 TCP and 4032 UDP ports per inside IPv4 address. For the purposes of this example, let’s assume that they are allocated sequentially, where 198.51.100.1 maps to 192.0.2.1 ports 1024-5055, 198.51.100.2 maps to 192.0.2.1 ports 5056-9087, etc. The dynamic port range thus contains ports 57472-65535 (port allocation illustrated in the table below). Finally, the maximum ports/subscriber is set to 5040.

Inside Address / Pool	Outside Address & Port
Reserved	192.0.2.1:0-1023
198.51.100.1	192.0.2.1:1024-5055
198.51.100.2	192.0.2.1:5056-9087
198.51.100.3	192.0.2.1:9088-13119
198.51.100.4	192.0.2.1:13120-17151
198.51.100.5	192.0.2.1:17152-21183
198.51.100.6	192.0.2.1:21184-25215
198.51.100.7	192.0.2.1:25216-29247
198.51.100.8	192.0.2.1:29248-33279
198.51.100.9	192.0.2.1:33280-37311
198.51.100.10	192.0.2.1:37312-41343
198.51.100.11	192.0.2.1:41344-45375
198.51.100.12	192.0.2.1:45376-49407
198.51.100.13	192.0.2.1:49408-53439
198.51.100.14	192.0.2.1:53440-57471
Dynamic	192.0.2.1:57472-65535

When subscriber 1 using 198.51.100.1 initiates a low volume of connections (e.g. < 4032 concurrent connections), the CGN maps the outgoing source address/port to the preallocated range. These translation mappings are not logged.

Subscriber 2 concurrently uses more than the allocated 4032 ports (e.g. for peer-to-peer, mapping, video streaming, or other connection-intensive traffic types), the CGN allocates up to an additional 1008 ports using bulk port reservations. In this example, subscriber 2 uses outside ports 5056-9087, and then 100-port blocks between 58000-58999. Connections using ports 5056-9087 are not logged, while 10 log entries are created for ports 58000-58099, 58100-58199, 58200-58299, ..., 58900-58999.

In order to identify a subscriber behind a CGN (regardless of port allocation method), public safety agencies need to collect source address and port information from content provider log files. Thus, content providers are advised to log source address, source port, and timestamp for all log entries, per [RFC6302]. If a public safety agency collects such information from a content provider and reports abuse from 192.0.2.1, port 2001, the operator can reverse the mapping algorithm to determine that the internal IP address subscriber 1 has been assigned generated the traffic without consulting CGN logs (by correlating the internal IP address with DHCP/PPP lease connection records). If a second abuse report comes in for 192.0.2.1, port 58204, the operator will determine that port 58204 is within the dynamic pool range, consult the log file, correlate with connection

records, and determine that subscriber 2 generated the traffic (assuming that the public safety timestamp matches the operator timestamp. As noted in RFC6292 [RFC6292], accurate time-keeping (e.g., use of NTP or Simple NTP) is vital).

In this example, there are no log entries for the majority of subscribers, who only use pre-allocated ports. Only minimal logging would be needed for those few subscribers who exceed their pre-allocated ports and obtain extra bulk port assignments from the dynamic pool. Logging data for those users will include inside address, outside address, outside port range, and timestamp.

Note that in a production environment, operators are encouraged to consider [RFC6598] for assigning inside addresses.

3. Additional Logging Considerations

In order to be able to identify a subscriber based on observed external IPv4 address, port, and timestamp, an operator needs to know how the CGN was configured with regards to internal and external IP addresses, dynamic address pool factor, maximum ports per user, and reserved port range at any given time. Therefore, the CGN MUST generate a record any time such variables are changed. The CGN SHOULD generate a log message any time such variables are changed. The CGN MAY keep such a record in the form of a router configuration file. If the CGN does not generate a log message, it would be up to the operator to maintain version control of router config changes. Also, the CGN SHOULD generate such a log message once per day to facilitate quick identification of the relevant configuration in the event of an abuse notification.

Such a log message MUST, at minimum, include the timestamp, inside prefix I, inside mask, outside prefix O, outside mask, D, M, A, and reserved port list R; for example:

[Wed Oct 11 14:32:52
2000]:198.51.100.0:28:192.0.2.0:32:2:5040:0:1-1023,5004,5060.

3.1. Failover Considerations

Due to the deterministic nature of algorithmically-assigned translations, no additional logging is required during failover conditions provided that inside address ranges are unique within a given failover domain. Even when directed to a different CGN server, translations within the deterministic port range on either the primary or secondary server can be algorithmically reversed, provided the algorithm is known. Thus, if 198.51.100.1 port 3456 maps to 192.0.2.1 port 1000 on CGN 1 and 198.51.100.1 port 1000 on Failover

CGN 2, an operator can identify the subscriber based on outside source address and port information.

Similarly, assignments made from the dynamic overflow pool need to be logged as described above, whether translations are performed on the primary or failover CGN.

4. Impact on the IPv6 Transition

The solution described in this document is applicable to Carrier Grade NAT transition technologies (e.g. NAT444, DS-Lite, and NAT64). As discussed in [RFC7021], the authors acknowledge that native IPv6 will offer subscribers a better experience than CGN. However, many CPE devices only support IPv4. Likewise, as of October 2014, only approximately 5.2% of the top 1 million websites were available using IPv6. Accordingly, Deterministic CGN should in no way be understood as making CGN a replacement for IPv6 service; however, until such time as IPv6 content and devices are widely available, Deterministic CGN will provide operators with the ability to quickly respond to public safety requests without requiring excessive infrastructure, operations, and bandwidth to support per-connection logging.

5. Privacy Considerations

The algorithm described above makes it easier for Service Providers and public safety officials to identify the IP address of a subscriber through a CGN system. This is the equivalent level of privacy users could expect when they are assigned a public IP address and their traffic is not translated. However, this algorithm could be used by other actors on the Internet to map multiple transactions to a single subscriber, particularly if ports are distributed sequentially. While still preserving traceability, subscriber privacy can be increased by using one of the other values of the Address Assignment Algorithm (A), which would require interested parties to know more about the Service Provider’s CGN configuration to be able to tie multiple connections to a particular subscriber.

6. IANA Considerations

This document makes no request of IANA.

7. Security Considerations

The security considerations applicable to NAT operation for various protocols as documented in, for example, RFC 4787 [RFC4787] and RFC 5382 [RFC5382] also apply to this document.

Note that with the possible exception of cryptographically-based port allocations, attackers could reverse-engineer algorithmically-derived port allocations to either target a specific subscriber or to spoof traffic to make it appear to have been generated by a specific subscriber. However, this is exactly the same level of security that the subscriber would experience in the absence of CGN. CGN is not intended to provide additional security by obscurity.

8. Acknowledgements

The authors would like to thank the following people for their suggestions and feedback: Bobby Flaim, Lee Howard, Wes George, Jean-Francois Tremblay, Mohammed Boucadair, Alain Durand, David Miles, Andy Anchev, Victor Kuarsingh, Miguel Cros Cecilia, Fred Baker, Brian Carpenter, and Reinaldo Penno.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

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

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

[RFC6264] Jiang, S., Guo, D., and B. Carpenter, "An Incremental Carrier-Grade NAT (CGN) for IPv6 Transition", RFC 6264, June 2011.

[RFC6269] Ford, M., Boucadair, M., Durand, A., Levis, P., and P. Roberts, "Issues with IP Address Sharing", RFC 6269, June 2011.

9.2. Informative References

[I-D.miles-behave-l2nat] Miles, D. and M. Townsley, "Layer2-Aware NAT", draft-miles-behave-l2nat-00 (work in progress), March 2009.

[I-D.shirasaki-nat444]
Yamagata, I., Shirasaki, Y., Nakagawa, A., Yamaguchi, J., and H. Ashida, "NAT444", draft-shirasaki-nat444-06 (work in progress), July 2012.

[I-D.sivakumar-behave-nat-logging]
Sivakumar, S. and R. Penno, "IPFIX Information Elements for logging NAT Events", draft-sivakumar-behave-nat-logging-06 (work in progress), January 2013.

[I-D.tsou-behave-natx4-log-reduction]
Tsou, T., Li, W., Taylor, T., and J. Huang, "Port Management To Reduce Logging In Large-Scale NATs", draft-tsou-behave-natx4-log-reduction-04 (work in progress), July 2013.

[RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, April 2011.

[RFC6292] Hoffman, P., "Requirements for a Working Group Charter Tool", RFC 6292, June 2011.

[RFC6302] Durand, A., Gashinsky, I., Lee, D., and S. Sheppard, "Logging Recommendations for Internet-Facing Servers", BCP 162, RFC 6302, June 2011.

[RFC6333] Durand, A., Droms, R., Woodyatt, J., and Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion", RFC 6333, August 2011.

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, August 2011.

[RFC6431] Boucadair, M., Levis, P., Bajko, G., Savolainen, T., and T. Tsou, "Huawei Port Range Configuration Options for PPP IP Control Protocol (IPCP)", RFC 6431, November 2011.

[RFC6598] Weil, J., Kuarsingh, V., Donley, C., Liljenstolpe, C., and M. Azinger, "IANA-Reserved IPv4 Prefix for Shared Address Space", BCP 153, RFC 6598, April 2012.

[RFC6887] Wing, D., Cheshire, S., Boucadair, M., Penno, R., and P. Selkirk, "Port Control Protocol (PCP)", RFC 6887, April 2013.

[RFC7021] Donley, C., Howard, L., Kuarsingh, V., Berg, J., and J. Doshi, "Assessing the Impact of Carrier-Grade NAT on Network Applications", RFC 7021, September 2013.

Authors' Addresses

Chris Donley
CableLabs
858 Coal Creek Cir
Louisville, CO 80027
US

Email: c.donley@cablelabs.com

Chris Grundemann
Internet Society
Denver, CO
US

Email: cgrundemann@gmail.com

Vikas Sarawat
CableLabs
858 Coal Creek Cir
Louisville, CO 80027
US

Email: v.sarawat@cablelabs.com

Karthik Sundaresan
CableLabs
858 Coal Creek Cir
Louisville, CO 80027
US

Email: k.sundaresan@cablelabs.com

Olivier Vautrin
Juniper Networks
1194 N Mathilda Avenue
Sunnyvale, CA 94089
US

Email: olivier@juniper.net

Network Working Group
Internet-Draft
Intended status: Informational
Expires: October 4, 2013

C. Donley, Ed.
CableLabs
L. Howard
Time Warner Cable
V. Kuarsingh
Rogers Communications
J. Berg
CableLabs
J. Doshi
University of Colorado
April 2, 2013

Assessing the Impact of Carrier-Grade NAT on Network Applications
draft-donley-nat444-impacts-06

Abstract

NAT444 is an IPv4 extension technology being considered by Service Providers to continue offering IPv4 service to customers while transitioning to IPv6. This technology adds an extra Carrier-Grade NAT ("CGN") in the Service Provider network, often resulting in two NATs. CableLabs, Time Warner Cable, and Rogers Communications independently tested the impacts of NAT444 on many popular Internet services using a variety of test scenarios, network topologies, and vendor equipment. This document identifies areas where adding a second layer of NAT disrupts the communication channel for common Internet applications. This document was updated to also include Dual-Stack Lite impacts.

Table of Contents

1. Introduction

IANA, APNIC, and RIPE exhausted their IPv4 address space in 2011-2012. Current projections suggest that ARIN may exhaust its free pool of IPv4 addresses in 2013. IPv6 is the solution to the IPv4 depletion problem; however, the transition to IPv6 will not be completed prior to IPv4 exhaustion. NAT444 [I-D.shirasaki-nat444] and Dual-Stack Lite ([RFC6333]) are transition mechanisms that will allow Service Providers to multiplex customers behind a single IPv4 address, which will allow many legacy devices and applications some IPv4 connectivity. While both NAT444 and Dual-Stack Lite do provide basic IPv4 connectivity, they impact a number of advanced applications. This document describes suboptimal behaviors of NAT444 and DS-Lite in our test environments.

In July-August 2010, CableLabs, Time Warner Cable, and Rogers Communications tested the impact of NAT444 on common applications using Carrier Grade NAT (CGN) devices. This testing was focused on a wide array of real time usage scenarios designed to evaluate the user experience over the public Internet using NAT444, in both single ISP and dual ISP environments. The purpose of this testing was to identify applications where the technology either breaks or significantly impacts the user experience. The outcome of the testing revealed that applications such as video streaming, video gaming and peer-to-peer file sharing are impacted by NAT444.

From June - October 2011, CableLabs conducted additional testing of CGN technologies, including both NAT444 and Dual-Stack Lite. The testing focused on working with several vendors including A10, Alcatel-Lucent, and Juniper to optimize the performance of those applications that experienced negative impacts during earlier CGN testing and to expand the testing to DS-Lite.

Applications that were tested included but were not necessarily limited to the following:

1. Video/Audio streaming, e.g. Silverlight-based applications, Netflix, YouTube, Pandora 2.

2. Peer-to-peer applications, e.g. video gaming, uTorrent

3. On line gaming, e.g. Xbox

4. Large file transfers using File Transfer Protocol (FTP)

5. Session Initiation Protocol (SIP) calls via X-Lite, Skype

6. Social Networking, e.g. Facebook, Webkinz

7. Video chat, e.g. Skype

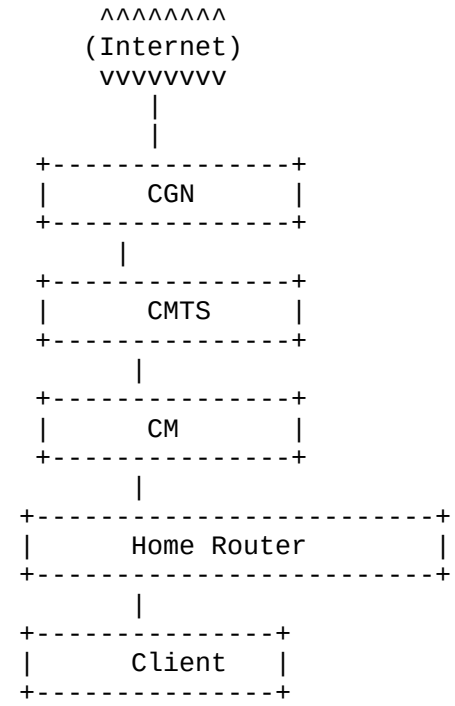
8. Web conferencing

2. Testing Scope

2.1. Test Cases

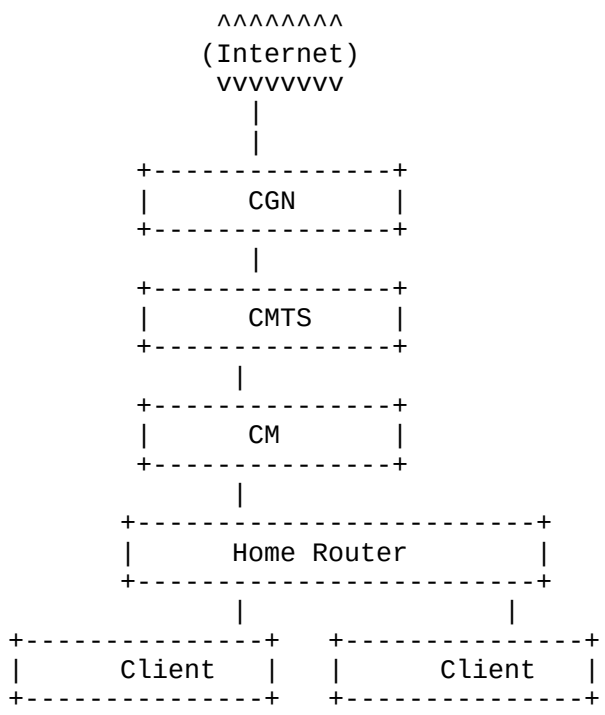
The diagrams below depict the general network architecture used for testing NAT444 and Dual Stack-Lite co-existence technologies at CableLabs.

2.1.1. Case1: Single Client, Single Home Network, Single Service Provider



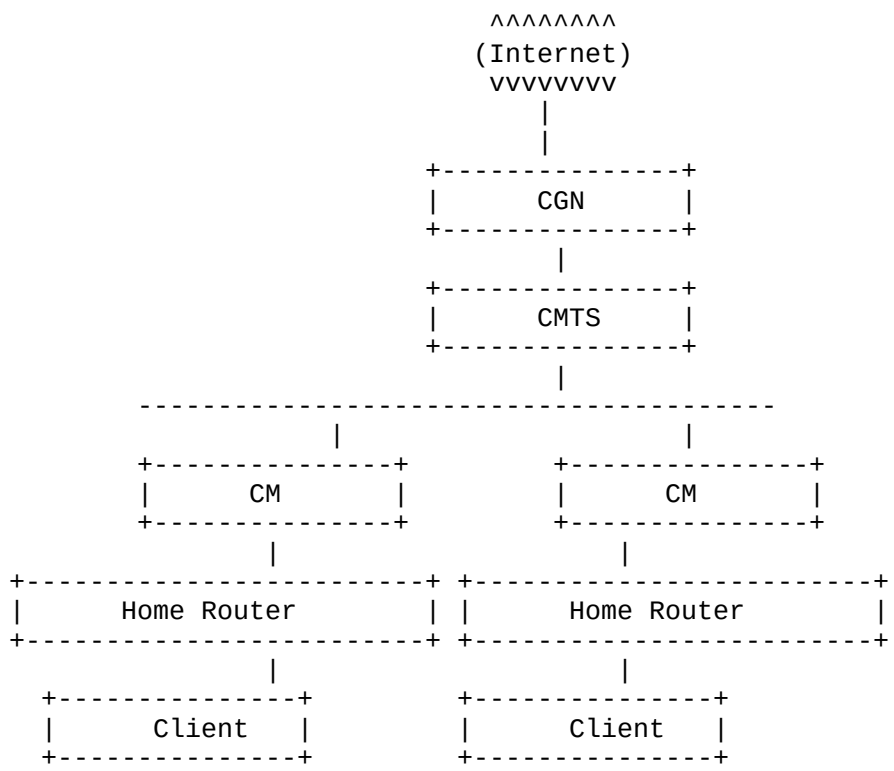
This is a typical case for a client accessing content on the Internet. For this case, we focused on basic web browsing, voice and video chat, instant messaging, video streaming (using YouTube, Google Videos , etc.), Torrent leeching and seeding, FTP, and gaming.

2.1.2. Case2: Two Clients, Single Home Network, Single Service Provider



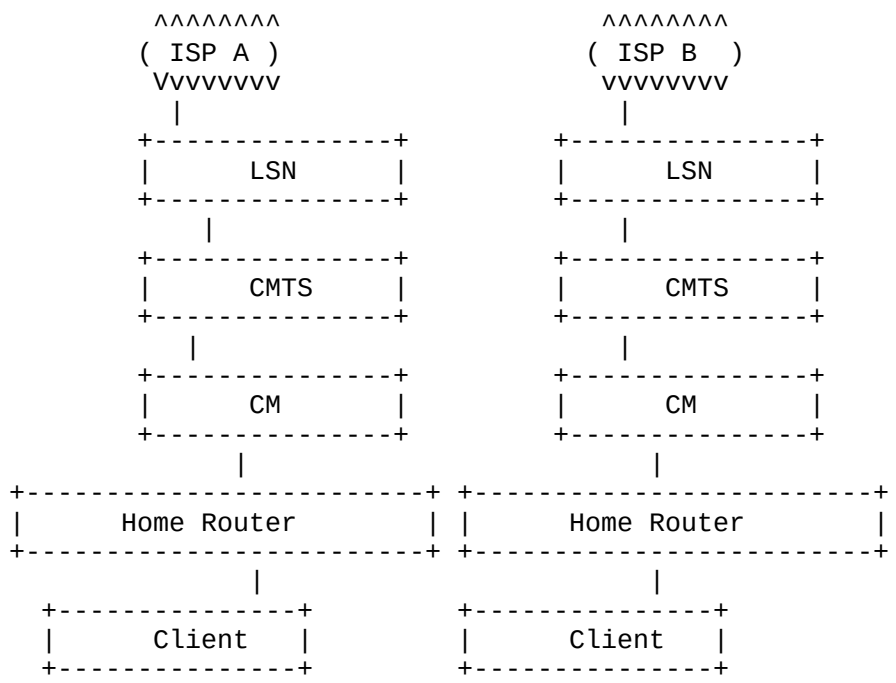
This is similar to Case 1, except that two clients are behind the same LSN and in the same home network. This test case was conducted to observe any change in speed in basic web browsing and video streaming.

2.1.3. Case3: Two Clients, Two Home Networks, Single Service Provider



In this scenario, the two clients are under the same LSN but behind two different gateways. This simulates connectivity between two residential subscribers on the same ISP. We tested peer-to-peer applications.

2.1.4. Case4: Two Clients, Two Home Networks, Two Service Providers Cross ISP



This test case is similar to Case 1 but with the addition of another identical ISP. This topology allows us to test traffic between two residential customers connected across the Internet. We focused on client-to-client applications such as IM and peer-to-peer.

2.2. General Test Environment

The lab environment was intended to emulate multiple service provider networks with a CGN deployed, and with connectivity to the public IPv4 or IPv6 internet (as dictated by the co-existence technology under test). This was accomplished by configuring a CGN behind multiple CMTSes and setting up multiple home networks for each ISP. Testing involved sending traffic to and from the public internet in both single and dual ISP environments, using both single and multiple home networks. The following equipment was used for testing:

- CGN
- CMTS

- IP sniffer
- RF sniffer
- Metrics tools (for network performance)
- CPE gateway devices
- Laptop or desktop computers (multiple OS used)
- Gaming consoles
- iPad or tablet devices
- other CE equipment, e.g. BluRay players supporting miscellaneous applications

One or more CPE gateway devices were configured in the home network. One or more host devices behind the gateways were also configured in order to test conditions such as multiple users on multiple home networks in the CGN architecture, both in single and dual ISP environments.

The scope of testing was honed down to the specific types of applications and network conditions that demonstrated a high probability of diminishing user experience based on prior testing. The following use cases were tested:

1. Video streaming over Netflix
2. Video streaming over YouTube
3. Video streaming over Joost
4. On line gaming with Xbox (one user)
5. Peer to Peer gaming with Xbox (two users)
6. Bit Torrent/uTorrent file seeding/leeching
7. Pandora internet radio
8. FTP server
9. Web conferencing (GTM, WebEx)
10. Social Networking - Facebook, Webkinz (chat, YouTube, file transfer)

11. Internet Archive - Video and Audio streaming; large file downloads

12. Video streaming using iClips

13. SIP Calls - X-Lite, Skype, PJSIP

14. MS Smooth Streaming (Silverlight)

15. Video chat - Skype, OoVoo

The following CPE devices were used for testing these applications on one or more home networks:

1. Windows 7, XP and Vista based laptops

2. MAC OS X laptop

3. iPad

4. Xbox gaming consoles

5. iPhone and Android smartphones

6. LG Blu-Ray player (test applications such as Netflix, Vudu, etc.)

7. Home routers - Netgear, Linksys, D-Link, Cisco, Apple

2.3. Test Metrics

Metrics data that were collected during the course of testing were related to throughput, latency, and jitter. These metrics were evaluated under three conditions:

1. Initial finding on the CGN configuration used for testing

2. Retest of the same test scenario with the CGN removed from the network

3. Retest with a new configuration (optimized) on the CGN (when possible)

In our testing, we found only slight differences with respect to latency or jitter when the CGN was in the network versus when it was not present in the network. It should be noted that we did not conduct any performance testing and metrics gathered were limited to single session scenarios. Also, bandwidth was not restricted on the DOCSIS network. Simulated homes shared a single DOCSIS upstream and

downstream channel.

Case	Avg Latency	Min Latency	Max Latency	RFC4689 Absolute Avg Jitter	Max Jitter
With CGN	240.32 us	233.77 us	428.40 us	1.86 us	191.22 us
Without CGN	211.88 us	190.39 us	402.69 us	0.07 us	176.16 us

CGN Performance

Note: Performance testing as defined by CableLabs includes load testing, induction of impairments on the network, etc. This type of testing was out of scope for CGN testing.

2.4. Test Scenarios Executed

The following test scenarios were executed using the aforementioned applications and test equipment:

1. Single ISP, Single Home Network with Single User

2. Single ISP, Two Home Networks With One User on Each Network

3. Dual ISPs, Single Home Network with Single User on each ISP

4. Dual ISPs, One Home Network With One User ISP-A; Two Home Networks with one user on each for ISP-B

These test scenarios were executed for both NAT444 and DS-Lite technologies.

2.5. General Test Methodologies

The CGN was configured for optimal setting for the specific test being executed for NAT444 or DS-Lite. Individual vendors provided validation of the configuration used for the co-existence technology under test prior to the start of testing. Some NAT444 testing used private [RFC1918] IPv4 space between the CGN and CPE router; other tests used public (non-[RFC1918]) IPv4 space between the CGN and CPE router. With the exception of 6to4 ([RFC3056]) traffic, we observed no difference in test results whether private or public address space was used. 6to4 failed when public space was used between the CGN and

CPE router was public, but CPE routers did not initiate 6to4 when private space was used.

CPE gateways and client devices were configured with IPv4 or IPv6 addresses using DHCP or manual configuration as required by each of the devices used in the test.

All devices were brought to operational state. Connectivity of CPE devices to provider network and public Internet were verified prior to start of each test.

IP sniffers and metrics tools were configured as required before starting tests. IP capture and metrics data was collected for all failed test scenarios. Sniffing was configured behind the home routers, north and south of the CMTS, and north and south of the CGN.

The test technician executed test scenarios listed above, for single and dual ISP environments, testing multiple users on multiple home networks, using the applications described above, where applicable to the each specific test scenario. Results checklists were compiled for all tests executed and for each combination of devices tested.

3. Observed CGN Impacts

CGN testing revealed that basic services such as e-mail and web browsing worked normally and as expected. However, there were some service affecting issues noted for applications that fall into two categories; dropped service and performance impacted service. In addition, for some specific applications in which the performance was impacted, throughput, latency and jitter measurements were taken. We observed that performance often differs from vendor to vendor and from test environment to test environment, and the results are somewhat difficult to predict. So as to not become a comparison between different vendor implementations, these results are presented in summary form. When issues were identified, we worked with the vendors involved to confirm the specific issues and explore workarounds. Except where noted, impacts to NAT444 and DS-Lite were similar.

In 2010 testing, we identified that IPv6 transition technologies such as 6to4 [RFC3056] and Teredo [RFC4380]) fail outright or are subject to severe service degradation. We did not repeat transition technology testing in 2011.

Note: While e-mail and web browsing operated as expected within our environment, there have been reports that anti-spam/anti-abuse measures limiting the number of connections from a single address can

cause problems in a CGN environment by improperly interpreting address sharing as too many connections from a single device. Care should be taken when deploying CGN to mitigate the impact of address sharing when configuring anti-spam/anti-abuse measures. See Section 3.4.

3.1. Dropped Services

Several peer-to-peer applications, specifically peer-to-peer gaming using Xbox and peer-to-peer SIP calls using the PJSIP client, failed in both the NAT444 and Dual-Stack Lite environments. Many CGN devices use "full cone" NAT so that once the CGN maps a port for outbound services, it will accept incoming connections to that port. However, some applications did not first send outgoing traffic and hence did not open an incoming port through the CGN. Other applications try to open a particular fixed port through the CGN; while service will work for a single subscriber behind the CGN, it fails when multiple subscribers try to use that port.

PJSIP and other SIP software worked when clients used a registration server to initiate calls, provided that the client inside the CGN initiated the traffic first and that only one SIP user was active behind a single IPv4 address at any given time. However, in our testing, we observed that when making a direct client-to-client SIP call across two home networks on a single ISP, or when calling from a single home network across dual ISPs, calls could neither be initiated nor received.

In the case of peer-to-peer gaming between two Xbox 360 users in different home networks on the same ISP, the game could not be connected between the two users. Both users shared an outside IP address, and tried to connect to the same port, causing a connection failure. There are some interesting nuances to this problem. In the case where two users are in the same home network and the scenario is through a single ISP, when the Xbox tries to register with the Xbox server, the server sees that both Xboxes are coming through the same public IP address and directs the devices to connect using their internal IP addresses. So, the connection ultimately gets established directly between both Xboxes via the home gateway, rather than the Xbox server. In the case where there are two Xbox users on two different home networks using a single ISP, and the CGN is configured with only one public IPv4 address, this scenario will not work because the route between the two users cannot be determined. However, if the CGN is configured with two public NAT IP addresses this scenario will work because now there is a unique IP address to communicate with. This is not an ideal solution, however, because it means that there is a one-to-one relationship between IP addresses in the public NAT and the number of Xbox users on each network.

Update: in December, 2011, Microsoft released an update for Xbox. While we did not conduct thorough testing using the new release, preliminary testing indicates that Xboxes that upgraded to the latest version can play head-to-head behind a CGN, at least for some games.

Other peer-to-peer applications that were noted to fail were seeding sessions initiated on Bittorent and uTorrent. In our test, torrent seeding was initiated on a client inside the CGN. Leeching was initiated using a client on the public Internet. It was observed that direct peer-to-peer seeding did not work. However, the torrent session typically redirected the leeching client to a proxy server, in which case the torrent session was set up successfully. Additionally, with the proxy in the network, re-seeding via additional leech clients worked as would be expected for a typical torrent session. Finally, uTorrent tries to use STUN to identify its outside address. In working with vendors, we learned that increasing the STUN timeout to 4 minutes improved uTorrent seeding performance behind a CGN, resulting in the ability for the uTorrent client to open a port and successfully seed content.

FTP sessions to servers located inside the home (e.g. behind two layers of NAT) failed. When the CGN was bypassed and traffic only needed to flow through one layer of NAT, clients were able to connect. Finally, multicast traffic was not forwarded through the CGN.

3.2. Performance Impacted Services

Large size file transfers and multiple video streaming sessions initiated on a single client on the same home network behind the CGN experienced reduced performance in our environment. We measured these variations in user experience against a baseline IPv4 environment where NAT is not deployed.

In our testing, we tried large file transfers from several FTP sites, as well as downloading sizable audio and video files (750MB - 1.4 GB) from the Internet Archive. We observed that when Dual-Stack Lite was implemented for some specific home router and client combinations, the transfer rate was markedly slower. For example, PC1 using one operating system behind the same home router as PC2 using a different operating system yielded a transfer rate of 120Kbps for PC1, versus 250Kbps for PC2. Our conclusion is that varying combinations of home routers and CE client devices may result in a user experience that is less than what the user would expect for typical applications. It is also difficult to predict which combinations of CPE routers and CE devices will produce a reduced experience for the user. We did not analyze the root cause of the divergence in performance across CE devices, as this was beyond the scope of our testing. However, as

this issue was specific to Dual-Stack Lite, we suspect that it is related to the MTU.

While video streaming sessions for a single user generally performed well, testing revealed that video streaming sessions such as Microsoft Smooth Streaming technology (i.e. Silverlight) or Netflix might also exhibit some service impacting behavior. In particular, this was observed on one older, yet popular and well-known CPE router where the first session was severely degraded when a second session was initiated in the same home network. Traffic from the first session ceased for 8 s once the second session was initiated. While we are tempted to write this off as a problematic home router, its popularity suggests that home router interactions may cause issues in NAT444 deployments (newer routers that support DS-Lite were not observed to experience this condition). Overall, longer buffering times for video sessions were noted for most client devices behind all types of home routers. However, once the initial buffering was complete, the video streams were consistently smooth. In addition, there were varying degrees as to how well multiple video sessions were displayed on various client devices across the CPE routers tested. Some video playback devices performed better than others.

3.3. Improvements since 2010

Since CableLabs completed initial CGN testing in 2010, there have been quantifiable improvements in performance over CGN since that time. These improvements may be categorized as follows:

- Content provider updates
- Application updates
- Improvements on the CGNs themselves

In terms of content provider updates, we have noted improvements in the overall performance of streaming applications in the CGN environment. Whereas applications such as streaming video were very problematic a year ago with regard to jitter and latency, our most recent testing revealed that there is less of an issue with these conditions, except in some cases when multiple video streaming sessions were initiated on the same client using specific types of home routers. Applications such as MS Smooth Streaming appear to have addressed these issues to some degree.

As far as application updates, use of STUN and/or proxy servers to offset some of the limitations of NAT and tunneling in the network are more evident as workarounds to the peer-to-peer issues. Applications appear to have incorporated other mechanisms for

delivering content faster, even if buffering times are somewhat slower and the content is not rendered as quickly.

CGN vendors have also upgraded their devices to mitigate several known issues with specific applications. With regard to addressing peer-to-peer SIP call applications, port reservations appear to be a workaround to the problem. However, this approach has limitations because of there are limited numbers of users that can have port reservations at any given time. For example, one CGN implementation allowed a port reservation to be made on port 5060 (default SIP port) but this was the only port that could be configured for the SIP client. This means that only one user can be granted the port reservation.

3.4. Additional CGN Challenges

There are other challenges that arise when using shared IPv4 address space, as with NAT444. Some of these challenges include:

- Loss of geolocation information - Often, translation zones will cross traditional geographic boundaries. Since the source addresses of packets traversing an LSN are set to the external address of the LSN, it is difficult for external entities to associate IP/Port information to specific locations/areas.

- Lawful Intercept/Abuse Response - Due to the nature of NAT444 address sharing, it will be hard to determine the customer/ endpoint responsible for initiating a specific IPv4 flow based on source IP address alone. Content providers, service providers, and law enforcement agencies will need to use new mechanisms (e.g., logging source port and timestamp in addition to source IP address) to potentially mitigate this new problem. This may impact the timely response to various identification requests. See [RFC6269].

- Antispoofing - Multiplexing users behind a single IP address can lead to situations where traffic from that address triggers antispoofing/DDoS protection mechanisms, resulting in unintentional loss of connectivity for some users. We have received reports of such antispoofing/DDoS mechanisms affecting email and web services in some instances, but did not experience them in our environment.

4. 2011 Summary of Results

4.1. NAT444

Test Scenario (per Test Plan)	Single ISP, Single HN, Single User	Single ISP, Two HN, Single User on Each	Dual ISP, One HN with One User on Each ISP	Dual ISP, One HN+One User on ISP-A, Two HN with One User on Each on ISP-B	Notes
Video streaming over Netflix	Pass	Pass	Pass	Pass	fails behind one router
Video streaming over YouTube	Pass	Pass	Pass	Pass	
Video streaming over Joost	Pass	Pass	Pass	Pass	
Online gaming with one user	Pass	Pass	Pass	NT	
Peer to Peer gaming with two users	Pass	Fail	Pass	NT	fails when both users NAT to same address
Bit Torrent uTorrent file seeding	Fail	Fail	Fail	Fail	
Bit Torrent uTorrent file leeching	Pass	Pass	Pass	Pass	

Pandora internet radio	Pass	Pass	Pass	Pass	
FTP server	Pass	Pass	Pass	Pass	
Web conferencing GTM	Pass	Pass	Pass	Pass	
Social Networking Facebook	Pass	Pass	Pass	Pass	
Social Networking Webkinz	Pass	Pass	Pass	Pass	
X-Lite for SIP calls with proxy	Pass	Pass	Pass	Pass	
X-Lite for SIP calls no proxy	Fail	Fail	Fail	Fail	
Skype text chat	Pass	Pass	Pass	Pass	
Skype video chat	Pass	Pass	Pass	Pass	
Oovoo	Pass	Pass	Pass	Pass	
MS Smooth streaming	Pass	Pass	Pass	Pass	
Internet Archive video streaming	Pass	Pass	Pass	Pass	
Internet Archive audio streaming	Pass	Pass	Pass	Pass	

Internet Archive file download	Pass	Pass	Pass	Pass	
Iclips	Pass	Pass	Pass	Pass	

NAT-444

4.2. DS-Lite

Test Scenario (per Test Plan)	DS-Lite Test Results	Duration of Test Performed	Description of Test Execution	General Observations/Notes
Video streaming over Netflix	Pass	15		
Video streaming over YouTube	Pass	10		
Video streaming over Joost	Pass	10		
On line gaming (one user)	Pass	15		
Peer to Peer gaming (two users)	Fail	NA	user inside HN1 playing game against user inside HN2	Users inside both HN are not able to connect. The error shown on console- "The game session is no longer available"

Bit Torrent/uTorrent file seeding	Fail	12	user on the internet is able to download file using proxy server and not peer-to-peer	
Bit Torrent/uTorrent file leeching	Pass	10		
Pandora internet radio	Pass	10		
FTP server	Pass	700 Mb		
Web conferencing (GTM)	Pass	10		
Social Networking - Facebook	Pass	NA		
Social Networking - Webkinz	Pass	NA		
X-Lite (for SIP calls) (proxy given)	Pass	10		
X-Lite (for SIP calls) (proxy not given)	Fail	NA		
Skype text chat	Pass	NA		

Skype video chat	Pass	20		
Oovoo	Pass	15		
MS Smooth streaming	Pass	10		
Internet Archive - video streaming	Pass	10		
Internet Archive - audio streaming	Pass	5		
Internet Archive - file download	Pass	80 Mb		
Iclips	Pass	10		

DSLite

5. 2010 Summary of Results

The tables below summarize results from 2010 NAT444 testing at CableLabs, Time Warner Cable, and Rogers Communications. They are included for comparison with 2011 results, documented above.

5.1. Case1: Single Client, Single Home Network, Single Service Provider

Test Case	Results	Notes
Web browsing	pass	
Email	pass	
FTP download	pass	performance degraded on very large downloads
Bittorrent leeching	pass	
Bittorrent seeding	fail	
Video streaming	pass	
Voice chat	pass	
Netflix streaming	pass	
Instant Messaging	pass	
Ping	pass	
Traceroute	pass	
Remote desktop	pass	
VPN	pass	
Xbox live	pass	
Xbox online	pass	Blocked by some LSNs.
Xbox network test	fail	Your NAT type is moderate. For best online experience you need an open NAT configuration. You should enable UPnP on the router.

Nintendo Wii	pass behind one LSN, fail behind another	
Playstation 3	pass	
Team Fortress 2	fail	pass behind one LSN, but performance degraded
Starcraft II	pass	
World of Warcraft	pass	
Call of Duty	pass	performance degraded behind one LSN
Slingcatcher	fail	
Netflix Party (Xbox)	fail	pass behind one LSN
Hulu	pass	performance degraded behind one LSN
AIM File Tranfer	pass	performance degraded
Webcam	fail	
6to4	fail	
Teredo	fail	

Case1

5.2. Case2: Two Clients, Single Home Network, Single Service Provider

Test Case	Results	Notes
Bittorrent leeching	pass	
Bittorrent seeding	fail	
Video streaming	fail	
Voice chat	pass	
Netflix streaming	pass	performance severely impacted, eventually failed
IM	pass	
Limewire leeching	pass	
Limewire seeding	fail	

Case2

5.3. Case3: Two Clients, Two Home Networks, Single Service Provider

Test Case	Results	Notes
Limewire leeching	pass	
Limewire seeding	fail	
Utorrent leeching	pass	
Utorrent seeding	fail	

Case3

5.4. Case4: Two Clients, Two Home Networks, Two Service Providers Cross ISP

Test Case	Results	Notes
Skype voice call	pass	
IM	pass	
FTP	fail	
Facebook chat	pass	
Skype video	pass	

Case4

6. CGN Mitigation

Our testing did not focus on mitigating the impact of Carrier Grade NAT, as described above. As such, mitigation is not the focus of this document. However, there are several approaches that could lessen the impacts described above.

Challenge	Potential Workaround(s)
Peer-to-peer	Use a proxy server; [I-D.ietf-pcp-base]
Gaming	[I-D.ietf-pcp-base]
Negative impact to geo-location services	Deploy CGN close to the edge of the network; use regional IP and port assignments.
Logging requirements for lawful intercept	Deterministic Logging [I-D.donley-behave-deterministic-cgn]; data compression [I-D.sivakumar-behave-nat-logging]; bulk port logging

CGN mitigation

Other mitigation techniques that are currently being researched, such as [I-D.tsou-stateless-nat44], may also improve performance.

7. IANA Considerations

This document has no IANA considerations.

8. Security Considerations

Security considerations are described in [RFC6264] and [RFC6269].

In general, since a CGN device shares a single IPv4 address with multiple subscribers, CGN devices may provide an attractive target for denial of service attacks. In addition, as described in [I-D.donley-behave-deterministic-cgn], abuse attribution is more challenging with CGN, and requires content providers to log IP address, source port, and time to correlate with service provider CGN logs. Also, if a CGN public IP address is added to a blacklist (e.g. for SPAM) or if a server limits the number of connections per IP address, it could negatively impact legitimate users.

9. Informative References

[I-D.donley-behave-deterministic-cgn]
Donley, C., Grundemann, C., Sarawat, V., Sundaresan, K., and O. Vautrin, "Deterministic Address Mapping to Reduce Logging in Carrier Grade NAT Deployments", draft-donley-behave-deterministic-cgn-05 (work in progress), January 2013.

[I-D.ietf-pcp-base]
Wing, D., Cheshire, S., Boucadair, M., Penno, R., and P. Selkirk, "Port Control Protocol (PCP)", draft-ietf-pcp-base-29 (work in progress), November 2012.

[I-D.shirasaki-nat444]
Yamagata, I., Shirasaki, Y., Nakagawa, A., Yamaguchi, J., and H. Ashida, "NAT444", draft-shirasaki-nat444-02 (work in progress), July 2010.

[I-D.sivakumar-behave-nat-logging]
Sivakumar, S. and R. Penno, "IPFIX Information Elements for logging NAT Events", draft-sivakumar-behave-nat-logging-06 (work in progress), January 2013.

[I-D.tsou-stateless-nat44]
Tsou, T., Liu, W., Perreault, S., Penno, R., and M. Chen, "Stateless IPv4 Network Address Translation",

draft-tsou-stateless-nat44-02 (work in progress), October 2012.

[RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.

[RFC3056] Carpenter, B. and K. Moore, "Connection of IPv6 Domains via IPv4 Clouds", RFC 3056, February 2001.

[RFC4380] Huitema, C., "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)", RFC 4380, February 2006.

[RFC4689] Poretsky, S., Perser, J., Erramilli, S., and S. Khurana, "Terminology for Benchmarking Network-layer Traffic Control Mechanisms", RFC 4689, October 2006.

[RFC6264] Jiang, S., Guo, D., and B. Carpenter, "An Incremental Carrier-Grade NAT (CGN) for IPv6 Transition", RFC 6264, June 2011.

[RFC6269] Ford, M., Boucadair, M., Durand, A., Levis, P., and P. Roberts, "Issues with IP Address Sharing", RFC 6269, June 2011.

[RFC6333] Durand, A., Droms, R., Woodyatt, J., and Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion", RFC 6333, August 2011.

Appendix A. Acknowledgements

Thanks to the following people for their testing, guidance, and feedback:

Paul Eldridge

Abishek Chandrasekaran

Vivek Ganti

Joey Padden

Lane Johnson

Authors' Addresses

Chris Donley (editor)
CableLabs
858 Coal Creek Circle
Louisville, CO 80027
USA

Email: c.donley@cablelabs.com

Lee Howard
Time Warner Cable
13241 Woodland Park Rd
Herndon, VA 20171
USA

Email: william.howard@twcable.com

Victor Kuarsingh
Rogers Communications
8200 Dixie Road
Brampton, ON L6T 0C1
Canada

Email: victor.kuarsingh@rci.rogers.com

John Berg
CableLabs
858 Coal Creek Circle
Louisville, CO 80027
USA

Email: j.berg@cablelabs.com

Jinesh Doshi
University of Colorado

Email: jinesh.doshi@colorado.edu

Internet Engineering Task Force
Internet-Draft
Updates: 4787 (if approved)
Intended status: BCP
Expires: June 9, 2013

S. Perreault, Ed.
Viagenie
I. Yamagata
S. Miyakawa
NTT Communications
A. Nakagawa
Japan Internet Exchange (JPIX)
H. Ashida
IS Consulting G.K.
December 6, 2012

Common requirements for Carrier Grade NATs (CGNs)
draft-ietf-behave-lsn-requirements-10

Abstract

This document defines common requirements for Carrier-Grade NAT (CGN). It updates RFC 4787.

Table of Contents

1. Introduction

With the shortage of IPv4 addresses, it is expected that more Internet Service Providers (ISPs) may want to provide a service where a public IPv4 address would be shared by many subscribers. Each subscriber is assigned a private address, and a Network Address Translator (NAT) [RFC2663] situated in the ISP's network translates between private and public addresses. When a second IPv4 NAT is located at the customer edge, this results in two layers of NAT.

This service can conceivably be offered alongside others, such as IPv6 services or regular IPv4 service assigning public addresses to subscribers. Some ISPs started offering such a service long before there was a shortage of IPv4 addresses, showing that there are driving forces other than the shortage of IPv4 addresses. One approach to CGN deployment is described in [RFC6264].

This document describes behavior that is required of those multi-subscriber NATs for interoperability. It is not an IETF endorsement of CGN or a real specification for CGN, but rather just a minimal set of requirements that will increase the likelihood of applications working across CGNs.

Because subscribers do not receive unique IPv4 addresses, Carrier Grade NATs introduce substantial limitations in communications between subscribers and with the rest of the Internet. In particular, it is considerably more involved to establish proxy functionality at the border between internal and external realms. Some applications may require substantial enhancements, while some others may not function at all in such an environment. Please see "Issues with IP Address Sharing" [RFC6269] for details.

This document builds upon previous works describing requirements for generic NATs [RFC4787][RFC5382][RFC5508]. These documents, and their updates if any, still apply in this context. What follows are additional requirements, to be satisfied on top of previous ones.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Readers are expected to be familiar with "NAT Behavioral Requirements for Unicast UDP" [RFC4787] and the terms defined there. The following additional term is used in this document:

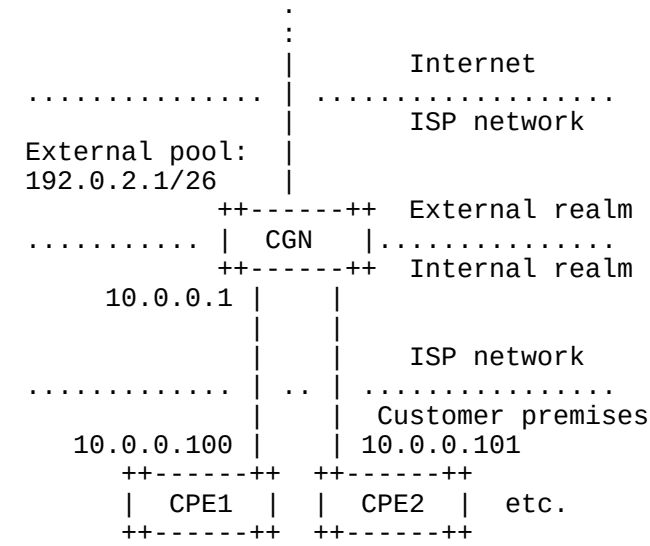
Carrier-Grade NAT (CGN): A NAT-based [RFC2663] logical function used to share the same IPv4 address among several subscribers. A CGN is not managed by the subscribers.

Note that the term "carrier-grade" has nothing to do with the quality of the NAT; that is left to discretion of implementers. Rather, it is to be understood as a topological qualifier: the NAT is placed in an ISP's network and translates the traffic of potentially many subscribers. Subscribers have limited or no control over the CGN, whereas they typically have full control over a NAT placed on their premises.

Note also that the CGN described in this document is IPv4-only. IPv6 address translation is not considered.

However, the scenario in which the IPv4-only CGN logical function is used may include IPv6 elements. For example, DS-Lite [RFC6333] uses an IPv4-only CGN logical function in a scenario making use of IPv6 encapsulation. Therefore, this document would also apply to the CGN part of DS-Lite.

Figure 1 summarizes a common network topology in which a CGN operates.



(IP addresses are only for example purposes)

Figure 1: CGN network topology

Another possible topology is one for hotspots, where there is no customer premise or customer-premises equipment (CPE), but where a CGN serves a bunch of customers who don't trust each other and hence fairness is an issue. One important difference with the previous topology is the absence of a second layer of NAT. This, however, has no impact on CGN requirements since they are driven by fairness and robustness in the service provided to customers, which applies in both cases.

3. Requirements for CGNs

What follows is a list of requirements for CGNs. They are in addition to those found in other documents such as [RFC4787], [RFC5382], and [RFC5508].

REQ-1: If a CGN forwards packets containing a given transport protocol, then it MUST fulfill that transport protocol's behavioral requirements. Current applicable documents are as follows:

A. "NAT Behavioral Requirements for Unicast UDP" [RFC4787]

B. "NAT Behavioral Requirements for TCP" [RFC5382]

C. "NAT Behavioral Requirements for ICMP" [RFC5508]

D. "NAT Behavioral Requirements for DCCP" [RFC5597]

Any future NAT behavioral requirements documents for IPv4 transport protocols will impose additional requirements for CGNs on top of those stated here.

Justification: It is crucial for CGNs to maximize the set of applications that can function properly across them. The IETF has documented the best current practices for UDP, TCP, ICMP, and DCCP.

REQ-2: A CGN MUST have a default "IP address pooling" behavior of "Paired" (as defined in [RFC4787] section 4.1). A CGN MAY provide a mechanism for administrators to change this behavior on an application protocol basis.

* When multiple overlapping internal IP address ranges share the same external IP address pool (e.g., DS-Lite [RFC6333]), the "IP address pooling" behavior applies to mappings between external IP addresses and internal subscribers rather than between external and internal IP

addresses.

Justification: This stronger form of REQ-2 from [RFC4787] is justified by the stronger need for not breaking applications that depend on the external address remaining constant.

Note that this requirement applies regardless of the transport protocol. In other words, a CGN must use the same external IP address mapping for all sessions associated with the same internal IP address, be they TCP, UDP, ICMP, something else, or a mix of different protocols.

The justification for allowing other behaviors is to allow the administrator to save external addresses and ports for application protocols that are known to work fine with other behaviors in practice. However, the default behavior MUST be "Paired".

REQ-3: The CGN function SHOULD NOT have any limitations on the size nor the contiguity of the external address pool. In particular, the CGN function MUST be configurable with contiguous or non-contiguous external IPv4 address ranges.

Justification: Given the increasing rarity of IPv4 addresses, it is becoming harder for an operator to provide large contiguous address pools to CGNs. Additionally, operational flexibility may require non-contiguous address pools for reasons such as differentiated services, routing management, etc.

The reason for having SHOULD instead of MUST is to account for limitations imposed by available resources as well as constraints imposed for security reasons.

REQ-4: A CGN MUST support limiting the number of external ports (or, equivalently, "identifiers" for ICMP) that are assigned per subscriber.

A. Per-subscriber limits MUST be configurable by the CGN administrator.

B. Per-subscriber limits MAY be configurable independently per transport protocol.

C. Additionally, it is RECOMMENDED that the CGN include administrator-adjustable thresholds to prevent a single subscriber from consuming excessive CPU resources from the CGN (e.g., rate limit the subscriber’s creation of new mappings).

Justification: A CGN can be considered a network resource that is shared by competing subscribers. Limiting the number of external ports assigned to each subscriber mitigates the DoS attack that a subscriber could launch against other subscribers through the CGN in order to get a larger share of the resource. It ensures fairness among subscribers. Limiting the rate of allocation mitigates a similar attack where the CPU is the resource being targeted instead of port numbers, however this requirement is not a MUST because it is very hard to explicitly call out all CPU-consuming events.

REQ-5: A CGN SHOULD support limiting the amount of state memory allocated per mapping and per subscriber. This may include limiting the number of sessions, the number of filters, etc., depending on the NAT implementation.

A. Limits SHOULD be configurable by the CGN administrator.

B. Additionally, it SHOULD be possible to limit the rate at which memory-consuming state elements are allocated.

Justification: A NAT needs to keep track of TCP sessions associated to each mapping. This state consumes resources for which, in the case of a CGN, subscribers may compete. It is necessary to ensure that each subscriber has access to a fair share of the CGN’s resources. Limiting the rate of allocation is intended to prevent CPU resource exhaustion. Item "B" is at the SHOULD level to account for the fact that means other than rate limiting may be used to attain the same goal.

REQ-6: It MUST be possible to administratively turn off translation for specific destination addresses and/or ports.

Justification: It is common for a CGN administrator to provide access for subscribers to servers installed in the ISP’s network in the external realm. When such a server is able to reach the internal realm via normal routing (which is entirely controlled by the ISP), translation is unneeded. In that case, the CGN may forward packets without modification, thus acting like a plain router. This may represent an important efficiency gain.

Figure 2 illustrates this use-case.

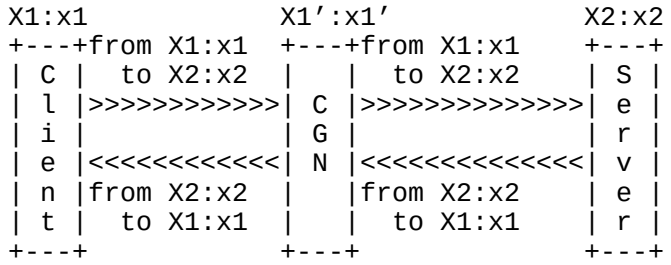


Figure 2: CGN pass-through

REQ-7: It is RECOMMENDED that a CGN use an "Endpoint-Independent Filtering" behavior (as defined in [RFC4787] section 5). If it is known that "Address-Dependent Filtering" does not cause the application-layer protocol to break (how to determine this is out of scope for this document), then it MAY be used instead.

Justification: This is a stronger form of REQ-8 from [RFC4787]. This is based on the observation that some games and peer-to-peer applications require EIF for the NAT traversal to work. In the context of a CGN it is important to minimize application breakage.

REQ-8: Once an external port is deallocated, it SHOULD NOT be reallocated to a new mapping until at least 120 seconds have passed, with the exceptions being:

A. If the CGN tracks TCP sessions (e.g., with a state machine, as in [RFC6146] section 3.5.2.2), TCP ports MAY be reused immediately.

B. If external ports are statically assigned to internal addresses (e.g., address X with port range 1000-1999 is assigned to subscriber A, 2000-2999 to subscriber B, etc.), and the assignment remains constant across state loss, then ports MAY be reused immediately.

C. If the allocated external ports used address-dependent or address-and-port-dependent filtering before state loss, they MAY be reused immediately.

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

Justification: 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.

The exceptions are for cases where reusing a port immediately does not create a possibility that packets would be redirected to the wrong peer. One can imagine other exceptions where mapping collisions are avoided, thus justifying the SHOULD level for this requirement.

The 120 seconds value corresponds to the Maximum Segment Lifetime (MSL) from [RFC0793].

Note that this requirement also applies to the case when a CGN loses state (due to a crash, reboot, failover to a cold standby, etc.). In that case, ports that were in use at the time of state loss SHOULD NOT be reallocated until at least 120 seconds have passed.

REQ-9: A CGN MUST implement a protocol giving subscribers explicit control over NAT mappings. That protocol SHOULD be the Port Control Protocol [I-D.ietf-pcp-base].

Justification: Allowing subscribers to manipulate the NAT state table with PCP greatly increases the likelihood that applications will function properly.

A study of PCP-less CGN impacts can be found in [I-D.donley-nat444-impacts]. Another study considering the effects of PCP on a peer-to-peer file sharing protocol can be found in [I-D.boucadair-pcp-bittorrent].

REQ-10: CGN implementers SHOULD make their equipment manageable. Standards-based management using standards such as "Definitions of Managed Objects for NAT" [RFC4008] is RECOMMENDED.

Justification: It is anticipated that CGNs will be primarily deployed in ISP networks where the need for management is critical. This requirement is at the SHOULD level to account for the fact that some CGN operators may not need management functionality.

Note also that there are efforts within the IETF toward creating a MIB tailored for CGNs (e.g., [I-D.ietf-behave-nat-mib]).

REQ-11: When a CGN is unable to create a dynamic mapping due to resource constraints or administrative restrictions (i.e., quotas):

A. it MUST drop the original packet;

B. it SHOULD send an ICMP Destination Unreachable message with code 1 (Host Unreachable) to the sender;

C. it SHOULD send a notification (e.g., SNMP trap) towards a management system (if configured to do so);

D. and it MUST NOT delete existing mappings in order to "make room" for the new one. (This only applies to normal CGN behavior, not to manual operator intervention.)

Justification: This is a slightly different form of REQ-8 from [RFC5508]. Code 1 is preferred to code 13 because it is listed as a "soft error" in [RFC1122], which is important because we don't want TCP stacks to abort the connection attempt in this case. See [RFC5461] for details on TCP's reaction to soft errors.

Sending ICMP errors and SNMP traps may be rate-limited for security reasons, which is why requirements B and C are SHOULDs, not a MUSTs.

Applications generally handle connection establishment failure better than established connection failure. This is why dropping the packet initiating the new connection is preferred over deleting existing mappings. See also the rationale in [RFC5508] section 6.

4. Logging

It may be necessary for CGN administrators to be able to identify a subscriber based on external IPv4 address, port, and timestamp in order to deal with abuse. When multiple subscribers share a single external address, the source address and port that are visible at the destination host have been translated from the ones originated by the subscriber.

In order to be able to do this, the CGN would need to log the following information for each mapping created (this list is for informational purposes only and does not constitute a requirement):

- transport protocol
- subscriber identifier (e.g., internal source address or tunnel endpoint identifier)
- external source address
- external source port
- timestamp

By "subscriber identifier" we mean information that uniquely identifies a subscriber. For example, in a traditional NAT scenario, the internal source address would be sufficient. In the case of DS-Lite, many subscribers share the same internal address and the subscriber identifier is the tunnel endpoint identifier (i.e., the B4's IPv6 address).

A disadvantage of logging mappings is that CGNs under heavy usage may produce large amounts of logs, which may require large storage volume.

REQ-12: A CGN SHOULD NOT log destination addresses or ports unless required to do so for administrative reasons.

Justification: Destination logging at the CGN creates privacy issues. Furthermore, readers should be aware of logging recommendations for Internet-facing servers [RFC6302]. With compliant servers, the destination address and port do not need to be logged by the CGN. This can help reduce the amount of logging.

This requirement is at the SHOULD level to account for the fact that there may be other reasons for logging destination addresses or ports. One such reason might be that the remote server is not following [RFC6302].

5. Port Allocation Scheme

A CGN's port allocation scheme is subject to three competing requirements:

REQ-13: A CGN's port allocation scheme SHOULD maximize port utilization.

Justification: External ports is one of the resources being shared by a CGN. Efficient management of that resource directly impacts the quality of a subscriber's Internet connection.

Some schemes are very efficient in their port utilization. In that sense, they have good scaling properties (nothing is wasted). Others will systematically waste ports.

REQ-14: A CGN's port allocation scheme SHOULD minimize log volume.

Justification: Huge log volumes can be problematic to CGN operators.

Some schemes create one log entry per mapping. Others allow multiple mappings to generate a single log entry, which sometimes can be expressed very compactly. With some schemes the logging frequency can approach that of DHCP servers.

REQ-15: A CGN's port allocation scheme SHOULD make it hard for attackers to guess port numbers.

Justification: Easily guessed port numbers put subscribers at risk of the attacks described in [RFC6056].

Some schemes provide very good security in that ports numbers are not easily guessed. Others provide poor security to subscribers

A CGN implementation's choice of port allocation scheme optimizes to satisfy one requirement at the expense of another. Therefore, these are soft requirements (SHOULD as opposed to MUST).

6. Deployment Considerations

Several issues are encountered when CGNs are used [RFC6269]. There is current work in the IETF toward alleviating some of these issues. For example, see [I-D.ietf-intarea-nat-reveal-analysis].

7. IANA Considerations

There are no IANA considerations.

8. Security Considerations

If a malicious subscriber can spoof another subscriber's CPE, it may cause a DoS to that subscriber by creating mappings up to the allowed limit. An ISP can prevent this with ingress filtering, as described

in [RFC2827].

This document recommends Endpoint-Independent Filtering (EIF) as the default filtering behavior for CGNs. EIF has security considerations which are discussed in [RFC4787].

NATs sometimes perform fragment reassembly. CGNs would do so at presumably high data rates. Therefore, the reader should be familiar with the potential security issues described in [RFC4963].

9. Acknowledgements

Thanks for the input and review by Alexey Melnikov, Arifumi Matsumoto, Barry Leiba, Benson Schliesser, Dai Kuwabara, Dan Wing, Dave Thaler, David Harrington, Francis Dupont, Jean-Francois Tremblay, Joe Touch, Lars Eggert, Kousuke Shishikura, Mohamed Boucadair, Martin Stiemerling, Meng Wei, Nejc Skoberne, Pete Resnick, Reinaldo Penno, Ron Bonica, Sam Hartman, Sean Turner, Senthil Sivakumar, Stephen Farrell, Stewart Bryant, Takanori Mizuguchi, Takeshi Tomochika, Tina Tsou, Tomohiro Fujisaki, Tomohiro Nishitani, Tomoya Yoshida, Wes George, Wesley Eddy, and Yasuhiro Shirasaki.

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4008] Rohit, R., Srisuresh, P., Raghunarayan, R., Pai, N., and C. Wang, "Definitions of Managed Objects for Network Address Translators (NAT)", RFC 4008, March 2005.

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

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

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

[RFC5597] Denis-Courmont, R., "Network Address Translation (NAT)

Behavioral Requirements for the Datagram Congestion Control Protocol", BCP 150, RFC 5597, September 2009.

[I-D.ietf-pcp-base]
Wing, D., Cheshire, S., Boucadair, M., Penno, R., and P. Selkirk, "Port Control Protocol (PCP)", draft-ietf-pcp-base-26 (work in progress), June 2012.

10.2. Informative Reference

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[RFC1122] Braden, R., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, October 1989.

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

[RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, May 2000.

[RFC4963] Heffner, J., Mathis, M., and B. Chandler, "IPv4 Reassembly Errors at High Data Rates", RFC 4963, July 2007.

[RFC5461] Gont, F., "TCP’s Reaction to Soft Errors", RFC 5461, February 2009.

[RFC6056] Larsen, M. and F. Gont, "Recommendations for Transport-Protocol Port Randomization", BCP 156, RFC 6056, January 2011.

[RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, April 2011.

[RFC6264] Jiang, S., Guo, D., and B. Carpenter, "An Incremental Carrier-Grade NAT (CGN) for IPv6 Transition", RFC 6264, June 2011.

[RFC6269] Ford, M., Boucadair, M., Durand, A., Levis, P., and P. Roberts, "Issues with IP Address Sharing", RFC 6269, June 2011.

[RFC6302] Durand, A., Gashinsky, I., Lee, D., and S. Sheppard, "Logging Recommendations for Internet-Facing Servers",

BCP 162, RFC 6302, June 2011.

[RFC6333] Durand, A., Droms, R., Woodyatt, J., and Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion", RFC 6333, August 2011.

[I-D.ietf-behave-nat-mib]
Perreault, S., Tsou, T., and S. Sivakumar, "Additional Managed Objects for Network Address Translators (NAT)", draft-ietf-behave-nat-mib-01 (work in progress), June 2012.

[I-D.ietf-intarea-nat-reveal-analysis]
Boucadair, M., Touch, J., Levis, P., and R. Penno, "Analysis of Solution Candidates to Reveal a Host Identifier (HOST_ID) in Shared Address Deployments", draft-ietf-intarea-nat-reveal-analysis-02 (work in progress), April 2012.

[I-D.donley-nat444-impacts]
Donley, C., Howard, L., Kuarsingh, V., Berg, J., and U. Colorado, "Assessing the Impact of Carrier-Grade NAT on Network Applications", draft-donley-nat444-impacts-04 (work in progress), May 2012.

[I-D.boucadair-pcp-bittorrent]
Boucadair, M., Zheng, T., Deng, X., and J. Queiroz, "Behavior of BitTorrent service in PCP-enabled networks with Address Sharing", draft-boucadair-pcp-bittorrent-00 (work in progress), May 2012.

Authors' Addresses

Simon Perreault (editor)
Viagenie
246 Aberdeen
Quebec, QC G1R 2E1
Canada

Phone: +1 418 656 9254
Email: simon.perreault@viagenie.ca
URI: http://www.viagenie.ca

Ikuhei Yamagata
NTT Communications Corporation
Gran Park Tower 17F, 3-4-1 Shibaura, Minato-ku
Tokyo 108-8118
Japan

Phone: +81 50 3812 4704
Email: ikuhei@nttv6.jp

Shin Miyakawa
NTT Communications Corporation
Gran Park Tower 17F, 3-4-1 Shibaura, Minato-ku
Tokyo 108-8118
Japan

Phone: +81 50 3812 4695
Email: miyakawa@nttv6.jp

Akira Nakagawa
Japan Internet Exchange Co., Ltd. (JPIX)
Otemachi Building 21F, 1-8-1 Otemachi, Chiyoda-ku
Tokyo 100-0004
Japan

Phone: +81 90 9242 2717
Email: a-nakagawa@jpix.ad.jp

Hiroyuki Ashida
IS Consulting G.K.
12-17 Odenma-cho Nihonbashi Chuo-ku
Tokyo 103-0011
Japan

Email: assie@hir.jp

Network Working Group
Internet-Draft
Obsoletes: 4008 (if approved)
Intended status: Standards Track
Expires: July 28, 2014

S. Perreault
Viagenie
T. Tsou
Huawei Technologies (USA)
S. Sivakumar
Cisco Systems

January 24, 2014

Definitions of Managed Objects for Network Address Translators (NAT) draft-ietf-behave-nat-mib-11

Abstract

This memo defines a portion of the Management Information Base (MIB) for devices implementing Network Address Translator (NAT) function. This MIB module may be used for monitoring of a device capable of NAT function.

This document obsoletes RFC 4008.

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 http://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 July 28, 2014.

Table of Contents

1. Introduction

This memo defines a portion of the Management Information Base (MIB) for devices implementing NAT function. This MIB module may be used for monitoring of a device capable of NAT function. Using it for configuration is deprecated. NAT types and their characteristics are defined in [RFC2663]. Traditional NAT function, in particular is defined in [RFC3022]. This MIB does not address the firewall functions and must not be used for configuring or monitoring these. Section 2 provides references to the SNMP management framework, which was used as the basis for the MIB module definition. Section 3 provides an overview of the MIB features. Lastly, Section 4 has the complete NAT MIB definition.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. The Internet-Standard Management Framework

For a detailed overview of the documents that describe the current Internet-Standard Management Framework, please refer to section 7 of RFC 3410 [RFC3410].

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. MIB objects are generally

accessed through the Simple Network Management Protocol (SNMP). Objects in the MIB are defined using the mechanisms defined in the Structure of Management Information (SMI). This memo specifies a MIB module that is compliant to the SMIv2, which is described in STD 58, RFC 2578 [RFC2578], STD 58, RFC 2579 [RFC2579] and STD 58, RFC 2580 [RFC2580].

3. Overview

3.1. Deprecated Features

All objects defined in [RFC4008] have been marked with "STATUS deprecated" for the following reasons:

Writability: Experience with NAT has shown that implementations vary tremendously. The NAT algorithms and data structures have little in common across devices, and this results in wildly incompatible configuration parameters. Therefore, few implementations were ever able to claim full compliance.

Lesson learned: the MIB should be read-only as much as possible.

Exposing configuration parameters: Even in read-only mode, many configuration parameters were exposed by [RFC4008] (e.g. timeouts). Since implementations vary wildly in their sets of configuration parameters, few implementations could claim even basic compliance.

Lesson learned: the NAT MIB’s purpose is not to expose configuration parameters.

Interfaces: Objects from [RFC4008] tie NAT state with interfaces (e.g. the interface table, the way map entries are grouped by interface). Many NAT implementations either never keep track of the interface or associate a mapping to a set of interfaces. Since interfaces are at the core of [RFC4008], many NAT devices were unable to have a proper implementation.

Lesson learned: NAT is a logical function that may be independent of interfaces. Do not tie NAT state with interfaces.

NAT service types: [RFC4008] used four categories of NAT service: basicNat, napt, bidirectionalNat, twiceNat. These are ill-defined and many implementations either use different categories or do not use categories at all.

Lesson learned: do not try to categorize NAT types.

Limited transport protocol set: The set of transport protocols was defined as: other, icmp, udp, tcp. Furthermore, the numeric values corresponding to those labels were arbitrary, without relation to the actual standard protocol numbers. This meant that NAT implementations were limited to those protocols and were unable to expose information about DCCP, SCTP, etc.

Lesson learned: use standard transport protocol numbers.

3.2. New Features

New features in this module are as follows:

Counters: Many new counters are introduced. Most of them are available in two variants: global and per-transport protocol.

Limits: A few limits on the quantity of state data stored by the NAT device. Some of them can trigger notifications.

Address+Port Pools: Pools of external addresses and ports are often used in enterprise and ISP settings. Pools are listed in a table, each with its range of addresses and ports. It is possible to inspect each pool's usage, to set limits, and to receive notifications when thresholds are crossed.

Address Mappings: NATs that have an "IP address pooling" behavior of "Paired" [RFC4787] maintain a mapping from internal address to external address. This module allows inspection of this mapping table.

Mapping table indexed by external 3-tuple: It is often necessary to determine the internal address that is mapped to a given external address and port. This MIB provides this table with an index to accomplish this efficiently, without having to iterate over all mappings.

Realms: See Section 3.3.

RFC 4787 terminology: Mapping table entries indicate the mapping behavior, the filtering behavior, and the address pooling behavior that were used to create the mapping.

Subscriber awareness: With the advent of CGN deployment, a set of subscriber specific counters, limits and parameters are added.

NAT instances: Multiple NAT instances may be managed by a single SNMP agent. All instance-specific objects (counters, limits, etc.) are indexed by NAT instance ID. In addition, NAT instances may be reliably identified using the natInstanceAlias object.

3.3. Realms

Current NAT devices commonly allow the internal and external parts of a mapping to come from different realms. The meaning of "realm" is implementation-dependent. On some implementations it can be equivalent to the name of a VPN Routing and Forwarding table (VRF). On others it is simply the numeric index of a virtual routing table. Note that this usage of "realm" is completely different from the one in [RFC4008].

This MIB allows the realm to be indicated where it makes sense. The format is an SnmpAdminString. On platforms that identify realms with integers, the string representation of the integer is used instead. The empty string has special meaning: it refers to the default realm.

Note that many MIBs implicitly support realms in one form or another by using SNMPv3 contexts. See for example the OSPFv2 MIB [RFC4750]. This method cannot be used for the NAT MIB because mapppings can belong to two realms simultaneously: the internal part can be in one realm while the external part is in another. In such cases the NAT function acts like a "wormhole" between two realms. Using contexts would implicitly impose the restriction that all objects would have to belong to the same realm.

4. Definitions

This MIB module IMPORTs objects from [RFC2578], [RFC2579], and [RFC4001].

```
NAT-MIB DEFINITIONS ::= BEGIN

IMPORTS
     MODULE-IDENTITY,
     OBJECT-TYPE,
     Integer32,
     Unsigned32,
     Gauge32,
     Counter64,
     TimeTicks,
     mib-2,
     NOTIFICATION-TYPE
             FROM SNMPv2-SMI
     TEXTUAL-CONVENTION,
```

```
        DisplayString,
        StorageType,
        RowStatus
                FROM SNMPv2-TC
        MODULE-COMPLIANCE,
        NOTIFICATION-GROUP,
        OBJECT-GROUP
                FROM SNMPv2-CONF
        ifIndex,
        ifCounterDiscontinuityGroup,
        InterfaceIndex
                FROM IF-MIB
        SnmpAdminString
                FROM SNMP-FRAMEWORK-MIB
        InetAddressType,
        InetAddress,
        InetAddressPrefixLength,
        InetPortNumber
                FROM INET-ADDRESS-MIB
        VPNIdOrZero
                FROM VPN-TC-STD-MIB;

   natMIB MODULE-IDENTITY
        LAST-UPDATED "201304260000Z"
   -- RFC Ed.: set to publication date
        ORGANIZATION
                "IETF Behavior Engineering for Hindrance Avoidance
                 (BEHAVE) Working Group"
        CONTACT-INFO
                "Working Group Email: behave@ietf.org

                 Simon Perreault
                 Viagenie
                 246 Aberdeen
                 Quebec, QC  G1R 2E1
                 Canada

                 Phone: +1 418 656 9254
                 Email: simon.perreault@viagenie.ca
                 URI:   http://viagenie.ca


                 Tina Tsou
                 Huawei Technologies (USA)
                 2330 Central Expressway
                 Santa Clara, CA  95050
                 USA
```

```
                Phone: +1 408 330 4424
                Email: tina.tsou.zouting@huawei.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"
        DESCRIPTION
                "This MIB module defines the generic managed objects
                for NAT.

                Copyright (C) The Internet Society (2013).  This
                version of this MIB module is part of RFC yyyy; see
                the RFC itself for full legal notices."
    -- RFC Ed.: replace yyyy with actual RFC number & remove this note"
        REVISION     "201304260000Z"
    -- RFC Ed.: set to publication date
        DESCRIPTION
                "Complete rewrite, published as RFC yyyy."
    -- RFC Ed.: replace yyyy with actual RFC number & set date"
        REVISION     "200503210000Z"  -- 21th March 2005
        DESCRIPTION
                "Initial version, published as RFC 4008."
        ::= { mib-2 123 }

    natMIBObjects OBJECT IDENTIFIER ::= { natMIB 1 }

    NatProtocolType ::= TEXTUAL-CONVENTION
          STATUS       deprecated
          DESCRIPTION
                  "A list of protocols that support the network
                   address translation.  Inclusion of the values is
                   not intended to imply that those protocols
                   need to be supported.  Any change in this
                   TEXTUAL-CONVENTION should also be reflected in
                   the definition of NatProtocolMap, which is a
                   BITS representation of this."
          SYNTAX   INTEGER {
                        none (1),  -- not specified
                        other (2), -- none of the following
                        icmp (3),
                        udp (4),
                        tcp (5)
```

```
                    }

   NatProtocolMap ::= TEXTUAL-CONVENTION
          STATUS       deprecated
          DESCRIPTION
                  "A bitmap of protocol identifiers that support
                   the network address translation.  Any change
                   in this TEXTUAL-CONVENTION should also be
                   reflected in the definition of NatProtocolType."
          SYNTAX   BITS {
                     other (0),
                     icmp (1),
                     udp (2),
                     tcp (3)
                   }

   NatAddrMapId ::= TEXTUAL-CONVENTION
          DISPLAY-HINT "d"
          STATUS deprecated
          DESCRIPTION
                  "A unique id that is assigned to each address map
                   by a NAT enabled device."
          SYNTAX   Unsigned32 (1..4294967295)

   NatBindIdOrZero ::= TEXTUAL-CONVENTION
          DISPLAY-HINT "d"
          STATUS deprecated
          DESCRIPTION
                  "A unique id that is assigned to each bind by
                   a NAT enabled device.  The bind id will be zero
                   in the case of a Symmetric NAT."
          SYNTAX   Unsigned32 (0..4294967295)

   NatBindId ::= TEXTUAL-CONVENTION
          DISPLAY-HINT "d"
          STATUS deprecated
          DESCRIPTION
                  "A unique id that is assigned to each bind by
                   a NAT enabled device."
          SYNTAX   Unsigned32 (1..4294967295)

   NatSessionId ::= TEXTUAL-CONVENTION
          DISPLAY-HINT "d"
          STATUS deprecated
          DESCRIPTION
                  "A unique id that is assigned to each session by
                   a NAT enabled device."
          SYNTAX   Unsigned32 (1..4294967295)
```

```
   NatBindMode ::= TEXTUAL-CONVENTION
          STATUS deprecated
          DESCRIPTION
                  "An indication of whether the bind is
                   an address bind or an address port bind."
          SYNTAX   INTEGER {
                        addressBind (1),
                        addressPortBind (2)
                   }

   NatAssociationType ::= TEXTUAL-CONVENTION
          STATUS deprecated
          DESCRIPTION
                  "An indication of whether the association is
                   static or dynamic."
          SYNTAX   INTEGER {
                        static (1),
                        dynamic (2)
                   }

   NatTranslationEntity ::= TEXTUAL-CONVENTION
          STATUS       deprecated
          DESCRIPTION
                  "An indication of a) the direction of a session for
                   which an address map entry, address bind or port
                   bind is applicable, and b) the entity (source or
                   destination) within the session that is subject to
                   translation."
          SYNTAX   BITS {
                     inboundSrcEndPoint (0),
                     outboundDstEndPoint(1),
                     inboundDstEndPoint (2),
                     outboundSrcEndPoint(3)
                   }


   --
   -- Default Values for the Bind and NAT Protocol Timers
   --

   natDefTimeouts OBJECT IDENTIFIER ::= { natMIBObjects 1 }

   natNotifCtrl OBJECT IDENTIFIER ::= { natMIBObjects 2 }

   --
   -- Address Bind and Port Bind related NAT configuration
   --
```

```
   natBindDefIdleTimeout OBJECT-TYPE
       SYNTAX     Unsigned32  (0..4294967295)
       UNITS      "seconds"
       MAX-ACCESS read-write
       STATUS     deprecated
       DESCRIPTION
               "The default Bind (Address Bind or Port Bind) idle
                timeout parameter.

                If the agent is capable of storing non-volatile
                configuration, then the value of this object must be
                restored after a re-initialization of the management
                system."
       DEFVAL { 0 }
       ::= { natDefTimeouts 1 }

   --
   -- UDP related NAT configuration
   --

   natUdpDefIdleTimeout OBJECT-TYPE
       SYNTAX     Unsigned32  (1..4294967295)
       UNITS      "seconds"
       MAX-ACCESS read-write
       STATUS     deprecated
       DESCRIPTION
               "The default UDP idle timeout parameter.

                If the agent is capable of storing non-volatile
                configuration, then the value of this object must be
                restored after a re-initialization of the management
                system."
       DEFVAL { 300 }
       ::= { natDefTimeouts 2 }

   --
   -- ICMP related NAT configuration
   --

   natIcmpDefIdleTimeout OBJECT-TYPE
       SYNTAX     Unsigned32  (1..4294967295)
       UNITS      "seconds"
       MAX-ACCESS read-write
       STATUS     deprecated
       DESCRIPTION
               "The default ICMP idle timeout parameter.

                If the agent is capable of storing non-volatile
```

```
                configuration, then the value of this object must be
                restored after a re-initialization of the management
                system."
        DEFVAL { 300 }
        ::= { natDefTimeouts 3 }

    --
    -- Other protocol parameters
    --

    natOtherDefIdleTimeout OBJECT-TYPE
        SYNTAX     Unsigned32  (1..4294967295)
        UNITS      "seconds"
        MAX-ACCESS read-write
        STATUS     deprecated
        DESCRIPTION
                "The default idle timeout parameter for protocols
                represented by the value other (2) in
                NatProtocolType.

                If the agent is capable of storing non-volatile
                configuration, then the value of this object must be
                restored after a re-initialization of the management
                system."
        DEFVAL { 60 }
        ::= { natDefTimeouts 4 }

    --
    -- TCP related NAT Timers
    --

    natTcpDefIdleTimeout OBJECT-TYPE
        SYNTAX     Unsigned32  (1..4294967295)
        UNITS      "seconds"
        MAX-ACCESS read-write
        STATUS     deprecated
        DESCRIPTION
                "The default time interval that a NAT session for an
                established TCP connection is allowed to remain
                valid without any activity on the TCP connection.

                If the agent is capable of storing non-volatile
                configuration, then the value of this object must be
                restored after a re-initialization of the management
                system."
        DEFVAL { 86400 }
        ::= { natDefTimeouts 5 }
```

```
   natTcpDefNegTimeout OBJECT-TYPE
       SYNTAX     Unsigned32  (1..4294967295)
       UNITS      "seconds"
       MAX-ACCESS read-write
       STATUS     deprecated
       DESCRIPTION
               "The default time interval that a NAT session for a TCP
                connection that is not in the established state
                is allowed to remain valid without any activity on
                the TCP connection.

                If the agent is capable of storing non-volatile
                configuration, then the value of this object must be
                restored after a re-initialization of the management
                system."
       DEFVAL { 60 }
       ::= { natDefTimeouts 6 }

   natNotifThrottlingInterval OBJECT-TYPE
       SYNTAX      Integer32 (0 | 5..3600)
       UNITS       "seconds"
       MAX-ACCESS  read-write
       STATUS      deprecated
       DESCRIPTION
               "This object controls the generation of the
                natPacketDiscard notification.

                If this object has a value of zero, then no
                natPacketDiscard notifications will be transmitted by
                the agent.

                If this object has a non-zero value, then the agent must
                not generate more than one natPacketDiscard
                'notification-event' in the indicated period, where a
                'notification-event' is the generation of a single
                notification PDU type to a list of notification
                destinations.  If additional NAT packets are discarded
                within the throttling period, then notification-events
                for these changes must be suppressed by the agent until
                the current throttling period expires.

                If natNotifThrottlingInterval notification generation
                is enabled, the suggested default throttling period is
                60 seconds, but generation of the natPacketDiscard
                notification should be disabled by default.

                If the agent is capable of storing non-volatile
                configuration, then the value of this object must be
```

```
                restored after a re-initialization of the management
                system.

                The actual transmission of notifications is controlled
                via the MIB modules in RFC 3413."
        DEFVAL { 0 }
        ::= { natNotifCtrl 1 }


    --
    -- The NAT Interface Table
    --

    natInterfaceTable OBJECT-TYPE
        SYNTAX      SEQUENCE OF NatInterfaceEntry
        MAX-ACCESS  not-accessible
        STATUS      deprecated
        DESCRIPTION
                "This table specifies the attributes for interfaces on a
                device supporting NAT function."
        ::= { natMIBObjects 3 }

    natInterfaceEntry OBJECT-TYPE
        SYNTAX      NatInterfaceEntry
        MAX-ACCESS  not-accessible
        STATUS      deprecated
        DESCRIPTION
                "Each entry in the natInterfaceTable holds a set of
                parameters for an interface, instantiated by
                ifIndex.  Therefore, the interface index must have been
                assigned, according to the applicable procedures,
                before it can be meaningfully used.
                Generally, this means that the interface must exist.

                When natStorageType is of type nonVolatile, however,
                this may reflect the configuration for an interface
                whose ifIndex has been assigned but for which the
                supporting implementation is not currently present."
        INDEX   { ifIndex }
        ::= { natInterfaceTable 1 }

    NatInterfaceEntry ::= SEQUENCE {
        natInterfaceRealm            INTEGER,
        natInterfaceServiceType      BITS,
        natInterfaceInTranslates     Counter64,
        natInterfaceOutTranslates    Counter64,
        natInterfaceDiscards         Counter64,
        natInterfaceStorageType      StorageType,
```

```
        natInterfaceRowStatus        RowStatus
    }

    natInterfaceRealm OBJECT-TYPE
        SYNTAX     INTEGER {
                       private (1),
                       public (2)
                   }
        MAX-ACCESS read-create
        STATUS     deprecated
        DESCRIPTION
                "This object identifies whether this interface is
                 connected to the private or the public realm."
        DEFVAL  { public }
        ::= { natInterfaceEntry 1 }

    natInterfaceServiceType OBJECT-TYPE
        SYNTAX  BITS {
                    basicNat (0),
                    napt (1),
                    bidirectionalNat (2),
                    twiceNat (3)
                }
        MAX-ACCESS  read-create
        STATUS      deprecated
        DESCRIPTION
                "An indication of the direction in which new sessions
                 are permitted and the extent of translation done within
                 the IP and transport headers."
        ::= { natInterfaceEntry 2 }

    natInterfaceInTranslates OBJECT-TYPE
        SYNTAX     Counter64
        MAX-ACCESS read-only
        STATUS     deprecated
        DESCRIPTION
                "Number of packets received on this interface that
                 were translated.
                 Discontinuities in the value of this counter can occur
                 at reinitialization of the management system and at
                 other times as indicated by the value of
                 ifCounterDiscontinuityTime on the relevant interface."
        ::= { natInterfaceEntry 3 }

    natInterfaceOutTranslates OBJECT-TYPE
        SYNTAX     Counter64
        MAX-ACCESS read-only
        STATUS     deprecated
```

```
        DESCRIPTION
                "Number of translated packets that were sent out this
                 interface.

                 Discontinuities in the value of this counter can occur
                 at reinitialization of the management system and at
                 other times as indicated by the value of
                 ifCounterDiscontinuityTime on the relevant interface."
        ::= { natInterfaceEntry 4 }

    natInterfaceDiscards OBJECT-TYPE
        SYNTAX     Counter64
        MAX-ACCESS read-only
        STATUS     deprecated
        DESCRIPTION
                "Number of packets that had to be rejected/dropped due to
                 a lack of resources for this interface.

                 Discontinuities in the value of this counter can occur
                 at reinitialization of the management system and at
                 other times as indicated by the value of
                 ifCounterDiscontinuityTime on the relevant interface."
         ::= { natInterfaceEntry 5 }

    natInterfaceStorageType OBJECT-TYPE
        SYNTAX      StorageType
        MAX-ACCESS  read-create
        STATUS      deprecated
        DESCRIPTION
                "The storage type for this conceptual row.
                 Conceptual rows having the value 'permanent'
                 need not allow write-access to any columnar objects
                 in the row."
        REFERENCE
                "Textual Conventions for SMIv2, Section 2."
        DEFVAL { nonVolatile }
        ::= { natInterfaceEntry 6 }

    natInterfaceRowStatus OBJECT-TYPE
        SYNTAX      RowStatus
        MAX-ACCESS  read-create
        STATUS      deprecated
        DESCRIPTION
                "The status of this conceptual row.

                 Until instances of all corresponding columns are
                 appropriately configured, the value of the
                 corresponding instance of the natInterfaceRowStatus
```

```
                column is 'notReady'.


                In particular, a newly created row cannot be made
                active until the corresponding instance of
                natInterfaceServiceType has been set.

                None of the objects in this row may be modified
                while the value of this object is active(1)."
        REFERENCE
                "Textual Conventions for SMIv2, Section 2."
        ::= { natInterfaceEntry 7 }

    --
    -- The Address Map Table
    --

    natAddrMapTable OBJECT-TYPE
        SYNTAX      SEQUENCE OF NatAddrMapEntry
        MAX-ACCESS  not-accessible
        STATUS      deprecated
        DESCRIPTION
                "This table lists address map parameters for NAT."
        ::= { natMIBObjects 4 }

    natAddrMapEntry OBJECT-TYPE
        SYNTAX      NatAddrMapEntry
        MAX-ACCESS  not-accessible
        STATUS      deprecated
        DESCRIPTION
                "This entry represents an address map to be used for
                 NAT and contributes to the dynamic and/or static
                 address mapping tables of the NAT device."
        INDEX   { ifIndex, natAddrMapIndex }
        ::= { natAddrMapTable 1 }

    NatAddrMapEntry ::= SEQUENCE {
        natAddrMapIndex                 NatAddrMapId,
        natAddrMapName                  SnmpAdminString,
        natAddrMapEntryType             NatAssociationType,
        natAddrMapTranslationEntity     NatTranslationEntity,
        natAddrMapLocalAddrType         InetAddressType,
        natAddrMapLocalAddrFrom         InetAddress,
        natAddrMapLocalAddrTo           InetAddress,
        natAddrMapLocalPortFrom         InetPortNumber,
        natAddrMapLocalPortTo           InetPortNumber,
        natAddrMapGlobalAddrType        InetAddressType,
        natAddrMapGlobalAddrFrom        InetAddress,
```

```
        natAddrMapGlobalAddrTo          InetAddress,
        natAddrMapGlobalPortFrom        InetPortNumber,
        natAddrMapGlobalPortTo          InetPortNumber,
        natAddrMapProtocol              NatProtocolMap,
        natAddrMapInTranslates          Counter64,
        natAddrMapOutTranslates         Counter64,
        natAddrMapDiscards              Counter64,
        natAddrMapAddrUsed              Gauge32,
        natAddrMapStorageType           StorageType,
        natAddrMapRowStatus             RowStatus
    }

    natAddrMapIndex  OBJECT-TYPE
        SYNTAX      NatAddrMapId
        MAX-ACCESS  not-accessible
        STATUS      deprecated
        DESCRIPTION
                "Along with ifIndex, this object uniquely
                 identifies an entry in the natAddrMapTable.
                 Address map entries are applied in the order
                 specified by natAddrMapIndex."
        ::= { natAddrMapEntry 1 }

    natAddrMapName OBJECT-TYPE
        SYNTAX      SnmpAdminString (SIZE(1..32))
        MAX-ACCESS  read-create
        STATUS      deprecated
        DESCRIPTION
                "Name identifying all map entries in the table associated
                 with the same interface.  All map entries with the same
                 ifIndex MUST have the same map name."
        ::= { natAddrMapEntry 2 }

    natAddrMapEntryType OBJECT-TYPE
        SYNTAX      NatAssociationType
        MAX-ACCESS  read-create
        STATUS      deprecated
        DESCRIPTION
                "This parameter can be used to set up static
                 or dynamic address maps."
        ::= { natAddrMapEntry 3 }

    natAddrMapTranslationEntity OBJECT-TYPE
        SYNTAX      NatTranslationEntity
        MAX-ACCESS  read-create
        STATUS      deprecated
        DESCRIPTION
                "The end-point entity (source or destination) in
```

```
                inbound or outbound sessions (i.e., first packets) that
                may be translated by an address map entry.

                Session direction (inbound or outbound) is
                derived from the direction of the first packet
                of a session traversing a NAT interface.
                NAT address (and Transport-ID) maps may be defined
                to effect inbound or outbound sessions.

                Traditionally, address maps for Basic NAT and NAPT are
                configured on a public interface for outbound sessions,
                effecting translation of source end-point.  The value of
                this object must be set to outboundSrcEndPoint for
                those interfaces.

                Alternately, if address maps for Basic NAT and NAPT were
                to be configured on a private interface, the desired
                value for this object for the map entries
                would be inboundSrcEndPoint (i.e., effecting translation
                of source end-point for inbound sessions).

                If TwiceNAT were to be configured on a private
                interface, the desired value for this object for the map
                entries would be a bitmask of inboundSrcEndPoint and
                inboundDstEndPoint."
        ::= { natAddrMapEntry 4 }

    natAddrMapLocalAddrType OBJECT-TYPE
        SYNTAX      InetAddressType
        MAX-ACCESS  read-create
        STATUS      deprecated
        DESCRIPTION
                "This object specifies the address type used for
                 natAddrMapLocalAddrFrom and natAddrMapLocalAddrTo."
        ::= { natAddrMapEntry 5 }

    natAddrMapLocalAddrFrom OBJECT-TYPE
        SYNTAX      InetAddress
        MAX-ACCESS  read-create
        STATUS      deprecated
        DESCRIPTION
                "This object specifies the first IP address of the range
                 of IP addresses mapped by this translation entry.  The
                 value of this object must be less than or equal to the
                 value of the natAddrMapLocalAddrTo object.

                 The type of this address is determined by the value of
                 the natAddrMapLocalAddrType object."
```

```
        ::= { natAddrMapEntry 6 }

    natAddrMapLocalAddrTo OBJECT-TYPE
        SYNTAX      InetAddress
        MAX-ACCESS  read-create
        STATUS      deprecated
        DESCRIPTION
                "This object specifies the last IP address of the range
                 of IP addresses mapped by this translation entry.  If
                 only a single address is being mapped, the value of this
                 object is equal to the value of natAddrMapLocalAddrFrom.
                 For a static NAT, the number of addresses in the range
                 defined by natAddrMapLocalAddrFrom and
                 natAddrMapLocalAddrTo must be equal to the number of
                 addresses in the range defined by
                 natAddrMapGlobalAddrFrom and natAddrMapGlobalAddrTo.
                 The value of this object must be greater than or equal
                 to the value of the natAddrMapLocalAddrFrom object.

                 The type of this address is determined by the value of
                 the natAddrMapLocalAddrType object."
        ::= { natAddrMapEntry 7 }

    natAddrMapLocalPortFrom OBJECT-TYPE
        SYNTAX      InetPortNumber
        MAX-ACCESS  read-create
        STATUS      deprecated
        DESCRIPTION
                "If this conceptual row describes a Basic NAT address
                 mapping, then the value of this object must be zero.  If
                 this conceptual row describes NAPT, then the value of
                 this object specifies the first port number in the range
                 of ports being mapped.

                 The value of this object must be less than or equal to
                 the value of the natAddrMapLocalPortTo object.  If the
                 translation specifies a single port, then the value of
                 this object is equal to the value of
                 natAddrMapLocalPortTo."
        DEFVAL { 0 }
        ::= { natAddrMapEntry 8 }

    natAddrMapLocalPortTo OBJECT-TYPE
        SYNTAX      InetPortNumber
        MAX-ACCESS  read-create
        STATUS      deprecated
        DESCRIPTION
                "If this conceptual row describes a Basic NAT address
```

```
                mapping, then the value of this object must be zero.  If
                this conceptual row describes NAPT, then the value of
                this object specifies the last port number in the range
                of ports being mapped.

                The value of this object must be greater than or equal
                to the value of the natAddrMapLocalPortFrom object.  If
                the translation specifies a single port, then the value
                of this object is equal to the value of
                natAddrMapLocalPortFrom."
        DEFVAL { 0 }
        ::= { natAddrMapEntry 9 }

    natAddrMapGlobalAddrType OBJECT-TYPE
        SYNTAX      InetAddressType
        MAX-ACCESS  read-create
        STATUS      deprecated
        DESCRIPTION
                "This object specifies the address type used for
                 natAddrMapGlobalAddrFrom and natAddrMapGlobalAddrTo."
        ::= { natAddrMapEntry 10 }

    natAddrMapGlobalAddrFrom OBJECT-TYPE
        SYNTAX      InetAddress
        MAX-ACCESS  read-create
        STATUS      deprecated
        DESCRIPTION
                "This object specifies the first IP address of the range
                 of IP addresses being mapped to.  The value of this
                 object must be less than or equal to the value of the
                 natAddrMapGlobalAddrTo object.

                 The type of this address is determined by the value of
                 the natAddrMapGlobalAddrType object."
        ::= { natAddrMapEntry 11 }

    natAddrMapGlobalAddrTo OBJECT-TYPE
        SYNTAX      InetAddress
        MAX-ACCESS  read-create
        STATUS      deprecated
        DESCRIPTION
                "This object specifies the last IP address of the range
                 of IP addresses being mapped to.  If only a single
                 address is being mapped to, the value of this object is
                 equal to the value of natAddrMapGlobalAddrFrom.  For a
                 static NAT, the number of addresses in the range defined
                 by natAddrMapGlobalAddrFrom and natAddrMapGlobalAddrTo
                 must be equal to the number of addresses in the range
```

```
                defined by natAddrMapLocalAddrFrom and
                natAddrMapLocalAddrTo.  The value of this object must be
                greater than or equal to the value of the
                natAddrMapGlobalAddrFrom object.

                The type of this address is determined by the value of
                the natAddrMapGlobalAddrType object."
        ::= { natAddrMapEntry 12 }

    natAddrMapGlobalPortFrom OBJECT-TYPE
        SYNTAX      InetPortNumber
        MAX-ACCESS  read-create
        STATUS      deprecated
        DESCRIPTION
                "If this conceptual row describes a Basic NAT address
                 mapping, then the value of this object must be zero.  If
                 this conceptual row describes NAPT, then the value of
                 this object specifies the first port number in the range
                 of ports being mapped to.


                 The value of this object must be less than or equal to
                 the value of the natAddrMapGlobalPortTo object.  If the
                 translation specifies a single port, then the value of
                 this object is equal to the value
                 natAddrMapGlobalPortTo."
        DEFVAL { 0 }
        ::= { natAddrMapEntry 13 }

    natAddrMapGlobalPortTo OBJECT-TYPE
        SYNTAX      InetPortNumber
        MAX-ACCESS  read-create
        STATUS      deprecated
        DESCRIPTION
                "If this conceptual row describes a Basic NAT address
                 mapping, then the value of this object must be zero.  If
                 this conceptual row describes NAPT, then the value of
                 this object specifies the last port number in the range
                 of ports being mapped to.

                 The value of this object must be greater than or equal
                 to the value of the natAddrMapGlobalPortFrom object.  If
                 the translation specifies a single port, then the value
                 of this object is equal to the value of
                 natAddrMapGlobalPortFrom."
        DEFVAL { 0 }
        ::= { natAddrMapEntry 14 }
```

```
   natAddrMapProtocol OBJECT-TYPE
       SYNTAX      NatProtocolMap
       MAX-ACCESS  read-create
       STATUS      deprecated
       DESCRIPTION
               "This object specifies a bitmap of protocol identifiers."
       ::= { natAddrMapEntry 15 }

   natAddrMapInTranslates OBJECT-TYPE
       SYNTAX     Counter64
       MAX-ACCESS read-only
       STATUS     deprecated
       DESCRIPTION
               "The number of inbound packets pertaining to this address
                map entry that were translated.

                Discontinuities in the value of this counter can occur
                at reinitialization of the management system and at
                other times, as indicated by the value of
                ifCounterDiscontinuityTime on the relevant interface."
       ::= { natAddrMapEntry 16 }

   natAddrMapOutTranslates OBJECT-TYPE
       SYNTAX     Counter64
       MAX-ACCESS read-only
       STATUS     deprecated
       DESCRIPTION
               "The number of outbound packets pertaining to this
                address map entry that were translated.

                Discontinuities in the value of this counter can occur
                at reinitialization of the management system and at
                other times, as indicated by the value of
                ifCounterDiscontinuityTime on the relevant interface."
       ::= { natAddrMapEntry 17 }

   natAddrMapDiscards OBJECT-TYPE
       SYNTAX     Counter64
       MAX-ACCESS read-only
       STATUS     deprecated
       DESCRIPTION
               "The number of packets pertaining to this address map
                entry that were dropped due to lack of addresses in the
                address pool identified by this address map.  The value
                of this object must always be zero in case of static
                address map.

                Discontinuities in the value of this counter can occur
```

```
                at reinitialization of the management system and at
                other times, as indicated by the value of
                ifCounterDiscontinuityTime on the relevant interface."
        ::= { natAddrMapEntry 18 }

    natAddrMapAddrUsed OBJECT-TYPE
        SYNTAX     Gauge32
        MAX-ACCESS read-only
        STATUS     deprecated
        DESCRIPTION
                "The number of addresses pertaining to this address map
                that are currently being used from the NAT pool.
                The value of this object must always be zero in the case
                of a static address map."
        ::= { natAddrMapEntry 19 }

    natAddrMapStorageType OBJECT-TYPE
        SYNTAX      StorageType
        MAX-ACCESS  read-create
        STATUS      deprecated
        DESCRIPTION
                "The storage type for this conceptual row.
                Conceptual rows having the value 'permanent'
                need not allow write-access to any columnar objects
                in the row."
        REFERENCE
                "Textual Conventions for SMIv2, Section 2."
        DEFVAL { nonVolatile }
        ::= { natAddrMapEntry 20 }

    natAddrMapRowStatus OBJECT-TYPE
        SYNTAX      RowStatus
        MAX-ACCESS  read-create
        STATUS      deprecated
        DESCRIPTION
                "The status of this conceptual row.

                Until instances of all corresponding columns are
                appropriately configured, the value of the
                corresponding instance of the natAddrMapRowStatus
                column is 'notReady'.

                None of the objects in this row may be modified
                while the value of this object is active(1)."
        REFERENCE
                "Textual Conventions for SMIv2, Section 2."
        ::= { natAddrMapEntry 21 }
```

```
   --
   -- Address Bind section
   --

   natAddrBindNumberOfEntries OBJECT-TYPE
       SYNTAX     Gauge32
       MAX-ACCESS read-only
       STATUS     deprecated
       DESCRIPTION
               "This object maintains a count of the number of entries
                that currently exist in the natAddrBindTable."
       ::= { natMIBObjects 5 }

   --
   -- The NAT Address BIND Table
   --

   natAddrBindTable OBJECT-TYPE
       SYNTAX     SEQUENCE OF NatAddrBindEntry
       MAX-ACCESS not-accessible
       STATUS     deprecated
       DESCRIPTION
               "This table holds information about the currently
                active NAT BINDs."
       ::= { natMIBObjects 6 }

   natAddrBindEntry OBJECT-TYPE
       SYNTAX     NatAddrBindEntry
       MAX-ACCESS not-accessible
       STATUS     deprecated
       DESCRIPTION
               "Each entry in this table holds information about
                an active address BIND.  These entries are lost
                upon agent restart.

                This row has indexing which may create variables with
                more than 128 subidentifiers.  Implementers of this
                table must be careful not to create entries that would
                result in OIDs which exceed the 128 subidentifier limit.
                Otherwise, the information cannot be accessed using
                SNMPv1, SNMPv2c or SNMPv3."

       INDEX   { ifIndex,
                 natAddrBindLocalAddrType,
                 natAddrBindLocalAddr }
       ::= { natAddrBindTable 1 }

   NatAddrBindEntry ::= SEQUENCE {
```

```
        natAddrBindLocalAddrType        InetAddressType,
        natAddrBindLocalAddr            InetAddress,
        natAddrBindGlobalAddrType       InetAddressType,
        natAddrBindGlobalAddr           InetAddress,
        natAddrBindId                   NatBindId,
        natAddrBindTranslationEntity    NatTranslationEntity,
        natAddrBindType                 NatAssociationType,
        natAddrBindMapIndex             NatAddrMapId,
        natAddrBindSessions             Gauge32,
        natAddrBindMaxIdleTime          TimeTicks,
        natAddrBindCurrentIdleTime      TimeTicks,
        natAddrBindInTranslates         Counter64,
        natAddrBindOutTranslates        Counter64
    }

    natAddrBindLocalAddrType OBJECT-TYPE
        SYNTAX      InetAddressType
        MAX-ACCESS  not-accessible
        STATUS      deprecated
        DESCRIPTION
                "This object specifies the address type used for
                 natAddrBindLocalAddr."
        ::= { natAddrBindEntry 1 }

    natAddrBindLocalAddr OBJECT-TYPE
        SYNTAX     InetAddress (SIZE (4|16))
        MAX-ACCESS not-accessible
        STATUS     deprecated
        DESCRIPTION
                "This object represents the private-realm specific
                 network layer address, which maps to the public-realm
                 address represented by natAddrBindGlobalAddr.

                 The type of this address is determined by the value of
                 the natAddrBindLocalAddrType object."
       ::= { natAddrBindEntry 2 }

    natAddrBindGlobalAddrType OBJECT-TYPE
        SYNTAX      InetAddressType
        MAX-ACCESS  read-only
        STATUS      deprecated
        DESCRIPTION
                "This object specifies the address type used for
                 natAddrBindGlobalAddr."
        ::= { natAddrBindEntry 3 }

    natAddrBindGlobalAddr OBJECT-TYPE
        SYNTAX     InetAddress
```

```
        MAX-ACCESS read-only
        STATUS     deprecated
        DESCRIPTION
                "This object represents the public-realm network layer
                 address that maps to the private-realm network layer
                 address represented by natAddrBindLocalAddr.

                 The type of this address is determined by the value of
                 the natAddrBindGlobalAddrType object."
        ::= { natAddrBindEntry 4 }

    natAddrBindId OBJECT-TYPE
        SYNTAX     NatBindId
        MAX-ACCESS read-only
        STATUS     deprecated
        DESCRIPTION
                "This object represents a bind id that is dynamically
                 assigned to each bind by a NAT enabled device.  Each
                 bind is represented by a bind id that is
                 unique across both, the natAddrBindTable and the
                 natAddrPortBindTable."
        ::= { natAddrBindEntry 5 }

    natAddrBindTranslationEntity OBJECT-TYPE
        SYNTAX     NatTranslationEntity
        MAX-ACCESS read-only
        STATUS     deprecated
        DESCRIPTION
                "This object represents the direction of sessions
                 for which this bind is applicable and the endpoint
                 entity (source or destination) within the sessions that
                 is subject to translation using the BIND.

                 Orientation of the bind can be a superset of
                 translationEntity of the address map entry which
                 forms the basis for this bind.

                 For example, if the translationEntity of an
                 address map entry is outboundSrcEndPoint, the
                 translationEntity of a bind derived from this
                 map entry may either be outboundSrcEndPoint or
                 it may be bidirectional (a bitmask of
                 outboundSrcEndPoint and inboundDstEndPoint)."
        ::= { natAddrBindEntry 6 }

    natAddrBindType OBJECT-TYPE
        SYNTAX     NatAssociationType
        MAX-ACCESS read-only
```

```
        STATUS     deprecated
        DESCRIPTION
                "This object indicates whether the bind is static or
                 dynamic."
        ::= { natAddrBindEntry 7 }

    natAddrBindMapIndex OBJECT-TYPE
        SYNTAX     NatAddrMapId
        MAX-ACCESS read-only
        STATUS     deprecated
        DESCRIPTION
                "This object is a pointer to the natAddrMapTable entry
                 (and the parameters of that entry) which was used in
                 creating this BIND.  This object, in conjunction with
                 the ifIndex (which identifies a unique addrMapName)
                 points to a unique entry in the natAddrMapTable."
        ::= { natAddrBindEntry 8 }

    natAddrBindSessions OBJECT-TYPE
        SYNTAX     Gauge32
        MAX-ACCESS read-only
        STATUS     deprecated
        DESCRIPTION
                "Number of sessions currently using this BIND."
        ::= { natAddrBindEntry 9 }

    natAddrBindMaxIdleTime OBJECT-TYPE
        SYNTAX     TimeTicks
        MAX-ACCESS read-only
        STATUS     deprecated
        DESCRIPTION
                "This object indicates the maximum time for
                 which this bind can be idle with no sessions
                 attached to it.

                 The value of this object is of relevance only for
                 dynamic NAT."
        ::= { natAddrBindEntry 10 }

    natAddrBindCurrentIdleTime OBJECT-TYPE
        SYNTAX     TimeTicks
        MAX-ACCESS read-only
        STATUS     deprecated
        DESCRIPTION
                "At any given instance, this object indicates the
                 time that this bind has been idle without any sessions
                 attached to it.
```

```
                The value of this object is of relevance only for
                dynamic NAT."
        ::= { natAddrBindEntry 11 }

    natAddrBindInTranslates OBJECT-TYPE
        SYNTAX     Counter64
        MAX-ACCESS read-only
        STATUS     deprecated
        DESCRIPTION
                "The number of inbound packets that were successfully
                translated by using this bind entry.

                Discontinuities in the value of this counter can occur
                at reinitialization of the management system and at
                other times, as indicated by the value of
                ifCounterDiscontinuityTime on the relevant interface."
        ::= { natAddrBindEntry 12 }

    natAddrBindOutTranslates OBJECT-TYPE
        SYNTAX     Counter64
        MAX-ACCESS read-only
        STATUS     deprecated
        DESCRIPTION
                "The number of outbound packets that were successfully
                translated using this bind entry.

                Discontinuities in the value of this counter can occur
                at reinitialization of the management system and at
                other times as indicated by the value of
                ifCounterDiscontinuityTime on the relevant interface."
        ::= { natAddrBindEntry 13 }

    --
    -- Address Port Bind section
    --

    natAddrPortBindNumberOfEntries OBJECT-TYPE
        SYNTAX     Gauge32
        MAX-ACCESS read-only
        STATUS     deprecated
        DESCRIPTION
                "This object maintains a count of the number of entries
                that currently exist in the natAddrPortBindTable."
        ::= { natMIBObjects 7 }

    --
    -- The NAT Address Port Bind Table
    --
```

```
   natAddrPortBindTable OBJECT-TYPE
       SYNTAX     SEQUENCE OF NatAddrPortBindEntry
       MAX-ACCESS not-accessible
       STATUS     deprecated
       DESCRIPTION
               "This table holds information about the currently
                active NAPT BINDs."
       ::= { natMIBObjects 8 }

   natAddrPortBindEntry OBJECT-TYPE
       SYNTAX     NatAddrPortBindEntry
       MAX-ACCESS not-accessible
       STATUS     deprecated
       DESCRIPTION
               "Each entry in the this table holds information
                about a NAPT bind that is currently active.
                These entries are lost upon agent restart.

                This row has indexing which may create variables with
                more than 128 subidentifiers.  Implementers of this
                table must be careful not to create entries which would
                result in OIDs that exceed the 128 subidentifier limit.
                Otherwise, the information cannot be accessed using
                SNMPv1, SNMPv2c or SNMPv3."
       INDEX   { ifIndex, natAddrPortBindLocalAddrType,
                 natAddrPortBindLocalAddr, natAddrPortBindLocalPort,
                 natAddrPortBindProtocol }
       ::= { natAddrPortBindTable 1 }

   NatAddrPortBindEntry ::= SEQUENCE {
       natAddrPortBindLocalAddrType        InetAddressType,
       natAddrPortBindLocalAddr            InetAddress,
       natAddrPortBindLocalPort            InetPortNumber,
       natAddrPortBindProtocol             NatProtocolType,
       natAddrPortBindGlobalAddrType       InetAddressType,
       natAddrPortBindGlobalAddr           InetAddress,
       natAddrPortBindGlobalPort           InetPortNumber,
       natAddrPortBindId                   NatBindId,
       natAddrPortBindTranslationEntity    NatTranslationEntity,
       natAddrPortBindType                 NatAssociationType,
       natAddrPortBindMapIndex             NatAddrMapId,
       natAddrPortBindSessions             Gauge32,
       natAddrPortBindMaxIdleTime          TimeTicks,
       natAddrPortBindCurrentIdleTime      TimeTicks,
       natAddrPortBindInTranslates         Counter64,
       natAddrPortBindOutTranslates        Counter64
   }
```

```
   natAddrPortBindLocalAddrType OBJECT-TYPE
       SYNTAX      InetAddressType
       MAX-ACCESS  not-accessible
       STATUS      deprecated
       DESCRIPTION
               "This object specifies the address type used for
                natAddrPortBindLocalAddr."
       ::= { natAddrPortBindEntry 1 }

   natAddrPortBindLocalAddr OBJECT-TYPE
       SYNTAX     InetAddress (SIZE(4|16))
       MAX-ACCESS not-accessible
       STATUS     deprecated
       DESCRIPTION
               "This object represents the private-realm specific
                network layer address which, in conjunction with
                natAddrPortBindLocalPort, maps to the public-realm
                network layer address and transport id represented by
                natAddrPortBindGlobalAddr and natAddrPortBindGlobalPort
                respectively.


                The type of this address is determined by the value of
                the natAddrPortBindLocalAddrType object."
       ::= { natAddrPortBindEntry 2 }

   natAddrPortBindLocalPort OBJECT-TYPE
       SYNTAX     InetPortNumber
       MAX-ACCESS not-accessible
       STATUS     deprecated
       DESCRIPTION
               "For a protocol value TCP or UDP, this object represents
                the private-realm specific port number.  On the other
                hand, for ICMP a bind is created only for query/response
                type ICMP messages such as ICMP echo, Timestamp, and
                Information request messages, and this object represents
                the private-realm specific identifier in the ICMP
                message, as defined in RFC 792 for ICMPv4 and in RFC
                2463 for ICMPv6.

                This object, together with natAddrPortBindProtocol,
                natAddrPortBindLocalAddrType, and
                natAddrPortBindLocalAddr, constitutes a session endpoint
                in the private realm.  A bind entry binds a private
                realm specific endpoint to a public realm specific
                endpoint, as represented by the tuple of
                (natAddrPortBindGlobalPort, natAddrPortBindProtocol,
                natAddrPortBindGlobalAddrType, and
```

```
                natAddrPortBindGlobalAddr)."
       ::= { natAddrPortBindEntry 3 }

   natAddrPortBindProtocol OBJECT-TYPE
       SYNTAX      NatProtocolType
       MAX-ACCESS  not-accessible
       STATUS      deprecated
       DESCRIPTION
               "This object specifies a protocol identifier.  If the
                value of this object is none(1), then this bind entry
                applies to all IP traffic.  Any other value of this
                object specifies the class of IP traffic to which this
                BIND applies."
       ::= { natAddrPortBindEntry 4 }

   natAddrPortBindGlobalAddrType OBJECT-TYPE
       SYNTAX      InetAddressType
       MAX-ACCESS  read-only
       STATUS      deprecated
       DESCRIPTION
               "This object specifies the address type used for
                natAddrPortBindGlobalAddr."
       ::= { natAddrPortBindEntry 5 }

   natAddrPortBindGlobalAddr OBJECT-TYPE
       SYNTAX     InetAddress
       MAX-ACCESS read-only
       STATUS     deprecated
       DESCRIPTION
               "This object represents the public-realm specific network
                layer address that, in conjunction with
                natAddrPortBindGlobalPort, maps to the private-realm

                network layer address and transport id represented by
                natAddrPortBindLocalAddr and natAddrPortBindLocalPort,
                respectively.

                The type of this address is determined by the value of
                the natAddrPortBindGlobalAddrType object."
       ::= { natAddrPortBindEntry 6 }

   natAddrPortBindGlobalPort OBJECT-TYPE
       SYNTAX     InetPortNumber
       MAX-ACCESS read-only
       STATUS     deprecated
       DESCRIPTION
               "For a protocol value TCP or UDP, this object represents
                the public-realm specific port number.  On the other
```

```
                hand, for ICMP a bind is created only for query/response
                type ICMP messages such as ICMP echo, Timestamp, and
                Information request messages, and this object represents
                the public-realm specific identifier in the ICMP
                message, as defined in RFC 792 for ICMPv4 and in RFC
                2463 for ICMPv6.

                This object, together with natAddrPortBindProtocol,
                natAddrPortBindGlobalAddrType, and
                natAddrPortBindGlobalAddr, constitutes a session
                endpoint in the public realm.  A bind entry binds a
                public realm specific endpoint to a private realm
                specific endpoint, as represented by the tuple of
                (natAddrPortBindLocalPort, natAddrPortBindProtocol,
                natAddrPortBindLocalAddrType, and
                natAddrPortBindLocalAddr)."
        ::= { natAddrPortBindEntry 7 }

    natAddrPortBindId OBJECT-TYPE
        SYNTAX     NatBindId
        MAX-ACCESS read-only
        STATUS     deprecated
        DESCRIPTION
                "This object represents a bind id that is dynamically
                 assigned to each bind by a NAT enabled device.  Each
                 bind is represented by a unique bind id across both
                 the natAddrBindTable and the natAddrPortBindTable."
        ::= { natAddrPortBindEntry 8 }

    natAddrPortBindTranslationEntity OBJECT-TYPE
        SYNTAX     NatTranslationEntity
        MAX-ACCESS read-only
        STATUS     deprecated
        DESCRIPTION
                "This object represents the direction of sessions
                 for which this bind is applicable and the entity
                 (source or destination) within the sessions that is
                 subject to translation with the BIND.

                 Orientation of the bind can be a superset of the
                 translationEntity of the address map entry that
                 forms the basis for this bind.

                 For example, if the translationEntity of an
                 address map entry is outboundSrcEndPoint, the
                 translationEntity of a bind derived from this
                 map entry may either be outboundSrcEndPoint or
                 may be bidirectional (a bitmask of
```

```
                outboundSrcEndPoint and inboundDstEndPoint)."
        ::= { natAddrPortBindEntry 9 }

    natAddrPortBindType OBJECT-TYPE
        SYNTAX     NatAssociationType
        MAX-ACCESS read-only
        STATUS     deprecated
        DESCRIPTION
                "This object indicates whether the bind is static or
                 dynamic."
        ::= { natAddrPortBindEntry 10 }

    natAddrPortBindMapIndex OBJECT-TYPE
        SYNTAX     NatAddrMapId
        MAX-ACCESS read-only
        STATUS     deprecated
        DESCRIPTION
                "This object is a pointer to the natAddrMapTable entry
                 (and the parameters of that entry) used in
                 creating this BIND.  This object, in conjunction with
                 the ifIndex (which identifies a unique addrMapName),
                 points to a unique entry in the natAddrMapTable."
        ::= { natAddrPortBindEntry 11 }

    natAddrPortBindSessions OBJECT-TYPE
        SYNTAX     Gauge32
        MAX-ACCESS read-only
        STATUS     deprecated
        DESCRIPTION
                "Number of sessions currently using this BIND."
        ::= { natAddrPortBindEntry 12 }

    natAddrPortBindMaxIdleTime OBJECT-TYPE
        SYNTAX     TimeTicks
        MAX-ACCESS read-only
        STATUS     deprecated

        DESCRIPTION
                "This object indicates the maximum time for
                 which this bind can be idle without any sessions
                 attached to it.
                 The value of this object is of relevance
                 only for dynamic NAT."
        ::= { natAddrPortBindEntry 13 }

    natAddrPortBindCurrentIdleTime OBJECT-TYPE
        SYNTAX     TimeTicks
        MAX-ACCESS read-only
```

```
        STATUS     deprecated
        DESCRIPTION
                "At any given instance, this object indicates the
                 time that this bind has been idle without any sessions
                 attached to it.

                 The value of this object is of relevance
                 only for dynamic NAT."
        ::= { natAddrPortBindEntry 14 }

    natAddrPortBindInTranslates OBJECT-TYPE
        SYNTAX     Counter64
        MAX-ACCESS read-only
        STATUS     deprecated
        DESCRIPTION
                "The number of inbound packets that were translated as
                 per this bind entry.

                 Discontinuities in the value of this counter can occur
                 at reinitialization of the management system and at
                 other times, as indicated by the value of
                 ifCounterDiscontinuityTime on the relevant interface."
        ::= { natAddrPortBindEntry 15 }

    natAddrPortBindOutTranslates OBJECT-TYPE
        SYNTAX     Counter64
        MAX-ACCESS read-only
        STATUS     deprecated
        DESCRIPTION
                "The number of outbound packets that were translated as
                 per this bind entry.

                 Discontinuities in the value of this counter can occur
                 at reinitialization of the management system and at
                 other times, as indicated by the value of
                 ifCounterDiscontinuityTime on the relevant interface."
        ::= { natAddrPortBindEntry 16 }

    --
    -- The Session Table
    --

    natSessionTable OBJECT-TYPE
        SYNTAX     SEQUENCE OF NatSessionEntry
        MAX-ACCESS not-accessible
        STATUS     deprecated
        DESCRIPTION
                "The (conceptual) table containing one entry for each
```

```
                NAT session currently active on this NAT device."
        ::= { natMIBObjects 9 }

    natSessionEntry OBJECT-TYPE
        SYNTAX     NatSessionEntry
        MAX-ACCESS not-accessible
        STATUS     deprecated
        DESCRIPTION
                "An entry (conceptual row) containing information
                 about an active NAT session on this NAT device.
                 These entries are lost upon agent restart."
        INDEX   { ifIndex, natSessionIndex }
        ::= { natSessionTable 1 }

    NatSessionEntry ::= SEQUENCE {
        natSessionIndex                         NatSessionId,
        natSessionPrivateSrcEPBindId            NatBindIdOrZero,
        natSessionPrivateSrcEPBindMode          NatBindMode,
        natSessionPrivateDstEPBindId            NatBindIdOrZero,
        natSessionPrivateDstEPBindMode          NatBindMode,
        natSessionDirection                     INTEGER,
        natSessionUpTime                        TimeTicks,
        natSessionAddrMapIndex                  NatAddrMapId,
        natSessionProtocolType                  NatProtocolType,
        natSessionPrivateAddrType               InetAddressType,
        natSessionPrivateSrcAddr                InetAddress,
        natSessionPrivateSrcPort                InetPortNumber,
        natSessionPrivateDstAddr                InetAddress,
        natSessionPrivateDstPort                InetPortNumber,
        natSessionPublicAddrType                InetAddressType,
        natSessionPublicSrcAddr                 InetAddress,
        natSessionPublicSrcPort                 InetPortNumber,
        natSessionPublicDstAddr                 InetAddress,
        natSessionPublicDstPort                 InetPortNumber,
        natSessionMaxIdleTime                   TimeTicks,
        natSessionCurrentIdleTime               TimeTicks,
        natSessionInTranslates                  Counter64,
        natSessionOutTranslates                 Counter64
    }

    natSessionIndex OBJECT-TYPE
        SYNTAX     NatSessionId
        MAX-ACCESS not-accessible
        STATUS     deprecated
        DESCRIPTION
                "The session ID for this NAT session."
        ::= { natSessionEntry 1 }
```

```
   natSessionPrivateSrcEPBindId OBJECT-TYPE
       SYNTAX     NatBindIdOrZero
       MAX-ACCESS read-only
       STATUS     deprecated
       DESCRIPTION
               "The bind id associated between private and public
                source end points.  In the case of Symmetric-NAT,
                this should be set to zero."
       ::= { natSessionEntry 2 }

   natSessionPrivateSrcEPBindMode OBJECT-TYPE
       SYNTAX     NatBindMode
       MAX-ACCESS read-only
       STATUS     deprecated
       DESCRIPTION
               "This object indicates whether the bind indicated
                by the object natSessionPrivateSrcEPBindId
                is an address bind or an address port bind."
       ::= { natSessionEntry 3 }

   natSessionPrivateDstEPBindId OBJECT-TYPE
       SYNTAX     NatBindIdOrZero
       MAX-ACCESS read-only
       STATUS     deprecated
       DESCRIPTION
               "The bind id associated between private and public
                destination end points."
       ::= { natSessionEntry 4 }

   natSessionPrivateDstEPBindMode OBJECT-TYPE
       SYNTAX     NatBindMode
       MAX-ACCESS read-only
       STATUS     deprecated
       DESCRIPTION
               "This object indicates whether the bind indicated
                by the object natSessionPrivateDstEPBindId
                is an address bind or an address port bind."
       ::= { natSessionEntry 5 }

   natSessionDirection OBJECT-TYPE
       SYNTAX     INTEGER {
                      inbound (1),
                      outbound (2)
                  }

       MAX-ACCESS read-only
       STATUS     deprecated
       DESCRIPTION
```

```
                "The direction of this session with respect to the
                 local network.  'inbound' indicates that this session
                 was initiated from the public network into the private
                 network.  'outbound' indicates that this session was
                 initiated from the private network into the public
                 network."
        ::= { natSessionEntry 6 }

    natSessionUpTime OBJECT-TYPE
        SYNTAX     TimeTicks
        MAX-ACCESS read-only
        STATUS     deprecated
        DESCRIPTION
                "The up time of this session in one-hundredths of a
                 second."
        ::= { natSessionEntry 7 }

    natSessionAddrMapIndex OBJECT-TYPE
        SYNTAX     NatAddrMapId
        MAX-ACCESS read-only
        STATUS     deprecated
        DESCRIPTION
                "This object is a pointer to the natAddrMapTable entry
                 (and the parameters of that entry) used in
                 creating this session.  This object, in conjunction with
                 the ifIndex (which identifies a unique addrMapName),
                 points to a unique entry in the natAddrMapTable."
        ::= { natSessionEntry 8 }

    natSessionProtocolType OBJECT-TYPE
        SYNTAX     NatProtocolType
        MAX-ACCESS read-only
        STATUS     deprecated
        DESCRIPTION
                "The protocol type of this session."
        ::= { natSessionEntry 9 }

    natSessionPrivateAddrType OBJECT-TYPE
        SYNTAX      InetAddressType
        MAX-ACCESS  read-only
        STATUS      deprecated
        DESCRIPTION
                "This object specifies the address type used for
                 natSessionPrivateSrcAddr and natSessionPrivateDstAddr."
        ::= { natSessionEntry 10 }

    natSessionPrivateSrcAddr OBJECT-TYPE
        SYNTAX     InetAddress
```

```
        MAX-ACCESS read-only
        STATUS     deprecated
        DESCRIPTION
                "The source IP address of the session endpoint that
                 lies in the private network.

                 The value of this object must be zero only when the
                 natSessionPrivateSrcEPBindId object has a zero value.
                 When the value of this object is zero, the NAT session
                 lookup will match any IP address to this field.

                 The type of this address is determined by the value of
                 the natSessionPrivateAddrType object."
        ::= { natSessionEntry 11 }

    natSessionPrivateSrcPort OBJECT-TYPE
        SYNTAX     InetPortNumber
        MAX-ACCESS read-only
        STATUS     deprecated
        DESCRIPTION
                "When the value of protocol is TCP or UDP, this object
                 represents the source port in the first packet of
                 session while in private-realm.  On the other hand, when
                 the protocol is ICMP, a NAT session is created only for
                 query/response type ICMP messages such as ICMP echo,
                 Timestamp, and Information request messages, and this
                 object represents the private-realm specific identifier
                 in the ICMP message, as defined in RFC 792 for ICMPv4
                 and in RFC 2463 for ICMPv6.

                 The value of this object must be zero when the
                 natSessionPrivateSrcEPBindId object has zero value
                 and value of natSessionPrivateSrcEPBindMode is
                 addressPortBind(2).  In such a case, the NAT session
                 lookup will match any port number to this field.

                 The value of this object must be zero when the object
                 is not a representative field (SrcPort, DstPort, or
                 ICMP identifier) of the session tuple in either the
                 public realm or the private realm."
        ::= { natSessionEntry 12 }

    natSessionPrivateDstAddr OBJECT-TYPE
        SYNTAX     InetAddress
        MAX-ACCESS read-only
        STATUS     deprecated
        DESCRIPTION
                "The destination IP address of the session endpoint that
```

```
                lies in the private network.

                The value of this object must be zero when the
                natSessionPrivateDstEPBindId object has a zero value.
                In such a scenario, the NAT session lookup will match
                any IP address to this field.

                The type of this address is determined by the value of
                the natSessionPrivateAddrType object."
        ::= { natSessionEntry 13 }

    natSessionPrivateDstPort OBJECT-TYPE
        SYNTAX     InetPortNumber
        MAX-ACCESS read-only
        STATUS     deprecated
        DESCRIPTION
                "When the value of protocol is TCP or UDP, this object
                represents the destination port in the first packet
                of session while in private-realm.  On the other hand,
                when the protocol is ICMP, this object is not relevant
                and should be set to zero.

                The value of this object must be zero when the
                natSessionPrivateDstEPBindId object has a zero
                value and natSessionPrivateDstEPBindMode is set to
                addressPortBind(2).  In such a case, the NAT session
                lookup will match any port number to this field.

                The value of this object must be zero when the object
                is not a representative field (SrcPort, DstPort, or
                ICMP identifier) of the session tuple in either the
                public realm or the private realm."
        ::= { natSessionEntry 14 }

    natSessionPublicAddrType OBJECT-TYPE
        SYNTAX      InetAddressType
        MAX-ACCESS  read-only
        STATUS      deprecated
        DESCRIPTION
                "This object specifies the address type used for
                natSessionPublicSrcAddr and natSessionPublicDstAddr."
        ::= { natSessionEntry 15 }

    natSessionPublicSrcAddr OBJECT-TYPE
        SYNTAX     InetAddress
        MAX-ACCESS read-only
        STATUS     deprecated
        DESCRIPTION
```

```
                "The source IP address of the session endpoint that
                 lies in the public network.

                 The value of this object must be zero when the
                 natSessionPrivateSrcEPBindId object has a zero value.
                 In such a scenario, the NAT session lookup will match
                 any IP address to this field.

                 The type of this address is determined by the value of
                 the natSessionPublicAddrType object."
        ::= { natSessionEntry 16 }

    natSessionPublicSrcPort OBJECT-TYPE
        SYNTAX     InetPortNumber
        MAX-ACCESS read-only
        STATUS     deprecated
        DESCRIPTION
                "When the value of protocol is TCP or UDP, this object
                 represents the source port in the first packet of
                 session while in public-realm.  On the other hand, when
                 protocol is ICMP, a NAT session is created only for
                 query/response type ICMP messages such as ICMP echo,
                 Timestamp, and Information request messages, and this
                 object represents the public-realm specific identifier
                 in the ICMP message, as defined in RFC 792 for ICMPv4
                 and in RFC 2463 for ICMPv6.

                 The value of this object must be zero when the
                 natSessionPrivateSrcEPBindId object has a zero value
                 and natSessionPrivateSrcEPBindMode is set to
                 addressPortBind(2).  In such a scenario, the NAT
                 session lookup will match any port number to this
                 field.

                 The value of this object must be zero when the object
                 is not a representative field (SrcPort, DstPort or
                 ICMP identifier) of the session tuple in either the
                 public realm or the private realm."
        ::= { natSessionEntry 17 }

    natSessionPublicDstAddr OBJECT-TYPE
        SYNTAX     InetAddress
        MAX-ACCESS read-only
        STATUS     deprecated
        DESCRIPTION
                "The destination IP address of the session endpoint that
                 lies in the public network.
```

```
                The value of this object must be non-zero when the
                natSessionPrivateDstEPBindId object has a non-zero
                value.  If the value of this object and the
                corresponding natSessionPrivateDstEPBindId object value
                is zero, then the NAT session lookup will match any IP
                address to this field.

                The type of this address is determined by the value of
                the natSessionPublicAddrType object."
        ::= { natSessionEntry 18 }

    natSessionPublicDstPort OBJECT-TYPE
        SYNTAX     InetPortNumber
        MAX-ACCESS read-only
        STATUS     deprecated
        DESCRIPTION
                "When the value of protocol is TCP or UDP, this object
                represents the destination port in the first packet of
                session while in public-realm.  On the other hand, when
                the protocol is ICMP, this object is not relevant for
                translation and should be zero.

                The value of this object must be zero when the
                natSessionPrivateDstEPBindId object has a zero value
                and natSessionPrivateDstEPBindMode is
                addressPortBind(2).  In such a scenario, the NAT
                session lookup will match any port number to this
                field.

                The value of this object must be zero when the object
                is not a representative field (SrcPort, DstPort, or
                ICMP identifier) of the session tuple in either the
                public realm or the private realm."
        ::= { natSessionEntry 19 }

    natSessionMaxIdleTime OBJECT-TYPE
        SYNTAX     TimeTicks
        MAX-ACCESS read-only
        STATUS     deprecated
        DESCRIPTION
                "The max time for which this session can be idle
                without detecting a packet."
        ::= { natSessionEntry 20 }

    natSessionCurrentIdleTime OBJECT-TYPE
        SYNTAX     TimeTicks
        MAX-ACCESS read-only
        STATUS     deprecated
```

```
        DESCRIPTION
                "The time since a packet belonging to this session was
                last detected."
        ::= { natSessionEntry 21 }

    natSessionInTranslates OBJECT-TYPE
        SYNTAX     Counter64
        MAX-ACCESS read-only
        STATUS     deprecated
        DESCRIPTION
                "The number of inbound packets that were translated for
                 this session.

                 Discontinuities in the value of this counter can occur
                 at reinitialization of the management system and at
                 other times, as indicated by the value of
                 ifCounterDiscontinuityTime on the relevant interface."
        ::= { natSessionEntry 22 }

    natSessionOutTranslates OBJECT-TYPE
        SYNTAX     Counter64
        MAX-ACCESS read-only
        STATUS     deprecated
        DESCRIPTION
                "The number of outbound packets that were translated for
                 this session.

                 Discontinuities in the value of this counter can occur
                 at reinitialization of the management system and at
                 other times, as indicated by the value of
                 ifCounterDiscontinuityTime on the relevant interface."
        ::= { natSessionEntry 23 }

    --
    -- The Protocol table
    --

    natProtocolTable OBJECT-TYPE
        SYNTAX     SEQUENCE OF NatProtocolEntry
        MAX-ACCESS not-accessible
        STATUS     deprecated
        DESCRIPTION
                "The (conceptual) table containing per protocol NAT
                 statistics."
        ::= { natMIBObjects 10 }

    natProtocolEntry OBJECT-TYPE
        SYNTAX     NatProtocolEntry
```

```
        MAX-ACCESS not-accessible
        STATUS     deprecated
        DESCRIPTION
                "An entry (conceptual row) containing NAT statistics
                 pertaining to a particular protocol."
        INDEX   { natProtocol }
        ::= { natProtocolTable 1 }

    NatProtocolEntry ::= SEQUENCE {
        natProtocol                 NatProtocolType,
        natProtocolInTranslates     Counter64,
        natProtocolOutTranslates    Counter64,
        natProtocolDiscards         Counter64
    }

    natProtocol    OBJECT-TYPE
        SYNTAX     NatProtocolType
        MAX-ACCESS not-accessible
        STATUS     deprecated
        DESCRIPTION
                "This object represents the protocol pertaining to which
                 parameters are reported."
        ::= { natProtocolEntry 1 }

    natProtocolInTranslates OBJECT-TYPE
        SYNTAX     Counter64
        MAX-ACCESS read-only
        STATUS     deprecated
        DESCRIPTION
                "The number of inbound packets pertaining to the protocol
                 identified by natProtocol that underwent NAT.

                 Discontinuities in the value of this counter can occur
                 at reinitialization of the management system and at
                 other times, as indicated by the value of
                 ifCounterDiscontinuityTime on the relevant interface."
        ::= { natProtocolEntry 2 }

    natProtocolOutTranslates OBJECT-TYPE
        SYNTAX     Counter64
        MAX-ACCESS read-only
        STATUS     deprecated
        DESCRIPTION
                "The number of outbound packets pertaining to the
                 protocol identified by natProtocol that underwent NAT.

                 Discontinuities in the value of this counter can occur
                 at reinitialization of the management system and at
```

```
                other times, as indicated by the value of
                ifCounterDiscontinuityTime on the relevant interface."
        ::= { natProtocolEntry 3 }

    natProtocolDiscards OBJECT-TYPE
        SYNTAX     Counter64
        MAX-ACCESS read-only
        STATUS     deprecated
        DESCRIPTION
                "The number of packets pertaining to the protocol
                identified by natProtocol that had to be
                rejected/dropped due to lack of resources.  These
                rejections could be due to session timeout, resource
                unavailability, lack of address space, etc.

                Discontinuities in the value of this counter can occur
                at reinitialization of the management system and at
                other times, as indicated by the value of
                ifCounterDiscontinuityTime on the relevant interface."
         ::= { natProtocolEntry 4 }

    --
    -- Notifications section
    --

    natMIBNotifications OBJECT IDENTIFIER ::= { natMIB 0 }

    --
    -- Notifications
    --

    natPacketDiscard NOTIFICATION-TYPE
        OBJECTS { ifIndex }
        STATUS  deprecated
        DESCRIPTION
                "This notification is generated when IP packets are
                discarded by the NAT function; e.g., due to lack of
                mapping space when NAT is out of addresses or ports.

                Note that the generation of natPacketDiscard
                notifications is throttled by the agent, as specified
                by the 'natNotifThrottlingInterval' object."
        ::= { natMIBNotifications 1 }



    --
    -- Conformance information.
```

```
   --

   natMIBConformance OBJECT IDENTIFIER ::= { natMIB 2 }

   natMIBGroups      OBJECT IDENTIFIER ::= { natMIBConformance 1 }
   natMIBCompliances OBJECT IDENTIFIER ::= { natMIBConformance 2 }

   --
   -- Units of conformance
   --

   natConfigGroup OBJECT-GROUP
       OBJECTS { natInterfaceRealm,
                 natInterfaceServiceType,
                 natInterfaceStorageType,
                 natInterfaceRowStatus,
                 natAddrMapName,
                 natAddrMapEntryType,
                 natAddrMapTranslationEntity,
                 natAddrMapLocalAddrType,
                 natAddrMapLocalAddrFrom,
                 natAddrMapLocalAddrTo,
                 natAddrMapLocalPortFrom,
                 natAddrMapLocalPortTo,
                 natAddrMapGlobalAddrType,
                 natAddrMapGlobalAddrFrom,
                 natAddrMapGlobalAddrTo,
                 natAddrMapGlobalPortFrom,
                 natAddrMapGlobalPortTo,
                 natAddrMapProtocol,
                 natAddrMapStorageType,
                 natAddrMapRowStatus,
                 natBindDefIdleTimeout,
                 natUdpDefIdleTimeout,
                 natIcmpDefIdleTimeout,
                 natOtherDefIdleTimeout,
                 natTcpDefIdleTimeout,
                 natTcpDefNegTimeout,
                 natNotifThrottlingInterval }
       STATUS  deprecated
       DESCRIPTION
               "A collection of configuration-related information
                required to support management of devices supporting
                NAT."
       ::= { natMIBGroups 1 }

   natTranslationGroup OBJECT-GROUP
       OBJECTS { natAddrBindNumberOfEntries,
```

```
                natAddrBindGlobalAddrType,
                natAddrBindGlobalAddr,
                natAddrBindId,
                natAddrBindTranslationEntity,
                natAddrBindType,
                natAddrBindMapIndex,
                natAddrBindSessions,
                natAddrBindMaxIdleTime,
                natAddrBindCurrentIdleTime,
                natAddrBindInTranslates,
                natAddrBindOutTranslates,
                natAddrPortBindNumberOfEntries,
                natAddrPortBindGlobalAddrType,
                natAddrPortBindGlobalAddr,
                natAddrPortBindGlobalPort,
                natAddrPortBindId,
                natAddrPortBindTranslationEntity,
                natAddrPortBindType,
                natAddrPortBindMapIndex,
                natAddrPortBindSessions,
                natAddrPortBindMaxIdleTime,
                natAddrPortBindCurrentIdleTime,
                natAddrPortBindInTranslates,
                natAddrPortBindOutTranslates,
                natSessionPrivateSrcEPBindId,
                natSessionPrivateSrcEPBindMode,
                natSessionPrivateDstEPBindId,
                natSessionPrivateDstEPBindMode,
                natSessionDirection,
                natSessionUpTime,
                natSessionAddrMapIndex,
                natSessionProtocolType,
                natSessionPrivateAddrType,
                natSessionPrivateSrcAddr,
                natSessionPrivateSrcPort,
                natSessionPrivateDstAddr,
                natSessionPrivateDstPort,
                natSessionPublicAddrType,
                natSessionPublicSrcAddr,
                natSessionPublicSrcPort,
                natSessionPublicDstAddr,
                natSessionPublicDstPort,
                natSessionMaxIdleTime,
                natSessionCurrentIdleTime,
                natSessionInTranslates,
                natSessionOutTranslates }
        STATUS  deprecated
```

```
        DESCRIPTION
                "A collection of BIND-related objects required to support
                 management of devices supporting NAT."
        ::= { natMIBGroups 2 }

    natStatsInterfaceGroup OBJECT-GROUP
        OBJECTS { natInterfaceInTranslates,
                  natInterfaceOutTranslates,
                  natInterfaceDiscards }
        STATUS  deprecated
        DESCRIPTION
                "A collection of NAT statistics associated with the
                 interface on which NAT is configured, to aid
                 troubleshooting/monitoring of the NAT operation."
        ::= { natMIBGroups 3 }

    natStatsProtocolGroup OBJECT-GROUP
        OBJECTS { natProtocolInTranslates,
                  natProtocolOutTranslates,
                  natProtocolDiscards }
        STATUS  deprecated
        DESCRIPTION
                "A collection of protocol specific NAT statistics,
                 to aid troubleshooting/monitoring of NAT operation."
        ::= { natMIBGroups 4 }

    natStatsAddrMapGroup OBJECT-GROUP
        OBJECTS { natAddrMapInTranslates,
                  natAddrMapOutTranslates,
                  natAddrMapDiscards,
                  natAddrMapAddrUsed }
        STATUS  deprecated
        DESCRIPTION
                "A collection of address map specific NAT statistics,
                 to aid troubleshooting/monitoring of NAT operation."
        ::= { natMIBGroups 5 }

    natMIBNotificationGroup NOTIFICATION-GROUP
        NOTIFICATIONS { natPacketDiscard }
        STATUS        deprecated
        DESCRIPTION
                "A collection of notifications generated by
                devices supporting this MIB."
        ::= { natMIBGroups 6 }


    --
    -- Compliance statements
```

```
   --

   natMIBFullCompliance MODULE-COMPLIANCE
       STATUS  deprecated
       DESCRIPTION
               "When this MIB is implemented with support for
                read-create, then such an implementation can claim
                full compliance.  Such devices can then be both
                monitored and configured with this MIB.

                The following index objects cannot be added as OBJECT
                clauses but nevertheless have the compliance
                requirements:
                    "
                -- OBJECT  natAddrBindLocalAddrType
                -- SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
                -- DESCRIPTION
                --         "An implementation is required to support
                --          global IPv4 and/or IPv6 addresses, depending
                --          on its support for IPv4 and IPv6."

                -- OBJECT  natAddrBindLocalAddr
                -- SYNTAX  InetAddress (SIZE(4|16))
                -- DESCRIPTION
                --         "An implementation is required to support
                --          global IPv4 and/or IPv6 addresses, depending
                --          on its support for IPv4 and IPv6."

                -- OBJECT  natAddrPortBindLocalAddrType
                -- SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
                -- DESCRIPTION
                --         "An implementation is required to support
                --          global IPv4 and/or IPv6 addresses, depending
                --          on its support for IPv4 and IPv6."

                -- OBJECT  natAddrPortBindLocalAddr
                -- SYNTAX  InetAddress (SIZE(4|16))
                -- DESCRIPTION
                --         "An implementation is required to support
                --          global IPv4 and/or IPv6 addresses, depending
                --          on its support for IPv4 and IPv6."

       MODULE IF-MIB -- The interfaces MIB, RFC2863
         MANDATORY-GROUPS {
           ifCounterDiscontinuityGroup
         }

       MODULE  -- this module
```

```
        MANDATORY-GROUPS { natConfigGroup, natTranslationGroup,
                           natStatsInterfaceGroup }

        GROUP       natStatsProtocolGroup
        DESCRIPTION
                 "This group is optional."
        GROUP       natStatsAddrMapGroup
        DESCRIPTION
                 "This group is optional."
        GROUP       natMIBNotificationGroup
        DESCRIPTION
                 "This group is optional."

        OBJECT  natAddrMapLocalAddrType
        SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
        DESCRIPTION
                "An implementation is required to support global IPv4
                 and/or IPv6 addresses, depending on its support
                 for IPv4 and IPv6."

        OBJECT  natAddrMapLocalAddrFrom
        SYNTAX  InetAddress (SIZE(4|16))
        DESCRIPTION
                "An implementation is required to support global IPv4
                 and/or IPv6 addresses, depending on its support
                 for IPv4 and IPv6."

        OBJECT  natAddrMapLocalAddrTo
        SYNTAX  InetAddress (SIZE(4|16))
        DESCRIPTION
                "An implementation is required to support global IPv4
                 and/or IPv6 addresses, depending on its support
                 for IPv4 and IPv6."

        OBJECT  natAddrMapGlobalAddrType
        SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
        DESCRIPTION
                "An implementation is required to support global IPv4
                 and/or IPv6 addresses, depending on its support
                 for IPv4 and IPv6."

        OBJECT  natAddrMapGlobalAddrFrom
        SYNTAX  InetAddress (SIZE(4|16))
        DESCRIPTION
                "An implementation is required to support global IPv4
                 and/or IPv6 addresses, depending on its support
                 for IPv4 and IPv6."
```

```
        OBJECT  natAddrMapGlobalAddrTo
        SYNTAX  InetAddress (SIZE(4|16))
        DESCRIPTION
                "An implementation is required to support global IPv4
                 and/or IPv6 addresses, depending on its support
                 for IPv4 and IPv6."

        OBJECT  natAddrBindGlobalAddrType
        SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
        DESCRIPTION
                "An implementation is required to support global IPv4
                 and/or IPv6 addresses, depending on its support
                 for IPv4 and IPv6."

        OBJECT  natAddrBindGlobalAddr
        SYNTAX  InetAddress (SIZE(4|16))
        DESCRIPTION
                "An implementation is required to support global IPv4
                 and/or IPv6 addresses, depending on its support
                 for IPv4 and IPv6."

        OBJECT  natAddrPortBindGlobalAddrType
        SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
        DESCRIPTION
                "An implementation is required to support global IPv4
                 and/or IPv6 addresses, depending on its support
                 for IPv4 and IPv6."

        OBJECT  natAddrPortBindGlobalAddr
        SYNTAX  InetAddress (SIZE(4|16))
        DESCRIPTION
                "An implementation is required to support global IPv4
                 and/or IPv6 addresses, depending on its support
                 for IPv4 and IPv6."

        OBJECT  natSessionPrivateAddrType
        SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
        DESCRIPTION
                "An implementation is required to support global IPv4
                 and/or IPv6 addresses, depending on its support
                 for IPv4 and IPv6."

        OBJECT  natSessionPrivateSrcAddr
        SYNTAX  InetAddress (SIZE(4|16))
        DESCRIPTION
                "An implementation is required to support global IPv4
                 and/or IPv6 addresses, depending on its support
                 for IPv4 and IPv6."
```

```
          OBJECT  natSessionPrivateDstAddr
          SYNTAX  InetAddress (SIZE(4|16))
          DESCRIPTION
                  "An implementation is required to support global IPv4
                   and/or IPv6 addresses, depending on its support
                   for IPv4 and IPv6."

          OBJECT  natSessionPublicAddrType
          SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
          DESCRIPTION
                  "An implementation is required to support global IPv4
                   and/or IPv6 addresses, depending on its support
                   for IPv4 and IPv6."

          OBJECT  natSessionPublicSrcAddr
          SYNTAX  InetAddress (SIZE(4|16))
          DESCRIPTION
                  "An implementation is required to support global IPv4
                   and/or IPv6 addresses, depending on its support
                   for IPv4 and IPv6."

          OBJECT  natSessionPublicDstAddr
          SYNTAX  InetAddress (SIZE(4|16))
          DESCRIPTION
                  "An implementation is required to support global IPv4
                   and/or IPv6 addresses, depending on its support
                   for IPv4 and IPv6."

        ::= { natMIBCompliances 1 }

    natMIBReadOnlyCompliance MODULE-COMPLIANCE
        STATUS  deprecated
        DESCRIPTION
                "When this MIB is implemented without support for
                 read-create (i.e., in read-only mode), then such an
                 implementation can claim read-only compliance.
                 Such a device can then be monitored but cannot be
                 configured with this MIB.

                 The following index objects cannot be added as OBJECT
                 clauses but nevertheless have the compliance
                 requirements:
                 "
                 -- OBJECT  natAddrBindLocalAddrType
                 -- SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
                 -- DESCRIPTION
                 --         "An implementation is required to support
                 --          global IPv4 and/or IPv6 addresses, depending
```

```
                --          on its support for IPv4 and IPv6."

                -- OBJECT  natAddrBindLocalAddr
                -- SYNTAX  InetAddress (SIZE(4|16))

                -- DESCRIPTION
                --          "An implementation is required to support
                --           global IPv4 and/or IPv6 addresses, depending
                --           on its support for IPv4 and IPv6."

                -- OBJECT  natAddrPortBindLocalAddrType
                -- SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
                -- DESCRIPTION
                --          "An implementation is required to support
                --           global IPv4 and/or IPv6 addresses, depending
                --           on its support for IPv4 and IPv6."
                -- OBJECT  natAddrPortBindLocalAddr
                -- SYNTAX  InetAddress (SIZE(4|16))
                -- DESCRIPTION
                --          "An implementation is required to support
                --           global IPv4 and/or IPv6 addresses, depending
                --           on its support for IPv4 and IPv6."

        MODULE IF-MIB -- The interfaces MIB, RFC2863
          MANDATORY-GROUPS {
            ifCounterDiscontinuityGroup
          }

        MODULE  -- this module
          MANDATORY-GROUPS { natConfigGroup, natTranslationGroup,
                             natStatsInterfaceGroup }

          GROUP       natStatsProtocolGroup
          DESCRIPTION
                   "This group is optional."
          GROUP       natStatsAddrMapGroup
          DESCRIPTION
                   "This group is optional."
          GROUP       natMIBNotificationGroup
          DESCRIPTION
                   "This group is optional."
          OBJECT natInterfaceRowStatus
          SYNTAX RowStatus { active(1) }
          MIN-ACCESS   read-only
          DESCRIPTION
                  "Write access is not required, and active is the only
                   status that needs to be supported."
```

```
        OBJECT  natAddrMapLocalAddrType
        SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
        MIN-ACCESS   read-only
        DESCRIPTION
                "Write access is not required.  An implementation is
                 required to support global IPv4 and/or IPv6 addresses,
                 depending on its support for IPv4 and IPv6."

        OBJECT  natAddrMapLocalAddrFrom
        SYNTAX  InetAddress (SIZE(4|16))
        MIN-ACCESS   read-only
        DESCRIPTION
                "Write access is not required.  An implementation is
                 required to support global IPv4 and/or IPv6 addresses,
                 depending on its support for IPv4 and IPv6."

        OBJECT  natAddrMapLocalAddrTo
        SYNTAX  InetAddress (SIZE(4|16))
        MIN-ACCESS   read-only
        DESCRIPTION
                "Write access is not required.  An implementation is
                 required to support global IPv4 and/or IPv6 addresses,
                 depending on its support for IPv4 and IPv6."

        OBJECT  natAddrMapGlobalAddrType
        SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
        MIN-ACCESS   read-only
        DESCRIPTION
                "Write access is not required.  An implementation is
                 required to support global IPv4 and/or IPv6 addresses,
                 depending on its support for IPv4 and IPv6."

        OBJECT  natAddrMapGlobalAddrFrom
        SYNTAX  InetAddress (SIZE(4|16))
        MIN-ACCESS   read-only
        DESCRIPTION
                "Write access is not required.  An implementation is
                 required to support global IPv4 and/or IPv6 addresses,
                 depending on its support for IPv4 and IPv6."

        OBJECT  natAddrMapGlobalAddrTo
        SYNTAX  InetAddress (SIZE(4|16))
        MIN-ACCESS   read-only
        DESCRIPTION
                "Write access is not required.  An implementation is
                 required to support global IPv4 and/or IPv6 addresses,
                 depending on its support for IPv4 and IPv6."
```

```
        OBJECT natAddrMapRowStatus
        SYNTAX RowStatus { active(1) }
        MIN-ACCESS   read-only
        DESCRIPTION
                "Write access is not required, and active is the only
                 status that needs to be supported."

        OBJECT  natAddrBindGlobalAddrType
        SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
        DESCRIPTION
                "An implementation is required to support global IPv4
                 and/or IPv6 addresses, depending on its support for
                 IPv4 and IPv6."

        OBJECT  natAddrBindGlobalAddr
        SYNTAX  InetAddress (SIZE(4|16))
        DESCRIPTION
                "An implementation is required to support global IPv4
                 and/or IPv6 addresses, depending on its support for
                 IPv4 and IPv6."

        OBJECT  natAddrPortBindGlobalAddrType
        SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
        DESCRIPTION
                "An implementation is required to support global IPv4
                 and/or IPv6 addresses, depending on its support for
                 IPv4 and IPv6."

        OBJECT  natAddrPortBindGlobalAddr
        SYNTAX  InetAddress (SIZE(4|16))
        DESCRIPTION
                "An implementation is required to support global IPv4
                 and/or IPv6 addresses, depending on its support for
                 IPv4 and IPv6."

        OBJECT  natSessionPrivateAddrType
        SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
        DESCRIPTION
                "An implementation is required to support global IPv4
                 and/or IPv6 addresses, depending on its support for
                 IPv4 and IPv6."

        OBJECT  natSessionPrivateSrcAddr
        SYNTAX  InetAddress (SIZE(4|16))
        DESCRIPTION
                "An implementation is required to support global IPv4
                 and/or IPv6 addresses, depending on its support for
                 IPv4 and IPv6."
```

```
        OBJECT  natSessionPrivateDstAddr
        SYNTAX  InetAddress (SIZE(4|16))
        DESCRIPTION
                "An implementation is required to support global IPv4
                 and/or IPv6 addresses, depending on its support for
                 IPv4 and IPv6."

        OBJECT  natSessionPublicAddrType
        SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
        DESCRIPTION
                "An implementation is required to support global IPv4
                 and/or IPv6 addresses, depending on its support for
                 IPv4 and IPv6."

        OBJECT  natSessionPublicSrcAddr
        SYNTAX  InetAddress (SIZE(4|16))
        DESCRIPTION
                "An implementation is required to support global IPv4
                 and/or IPv6 addresses, depending on its support for
                 IPv4 and IPv6."

        OBJECT  natSessionPublicDstAddr
        SYNTAX  InetAddress (SIZE(4|16))
        DESCRIPTION
                "An implementation is required to support global IPv4
                 and/or IPv6 addresses, depending on its support for
                 IPv4 and IPv6."

      ::= { natMIBCompliances 2 }


   --=================================================================
   -- END OF DEPRECATED OBJECTS. CURRENT OBJECTS FOLLOW.


   -- textual conventions

   ProtocolNumber ::= TEXTUAL-CONVENTION
       DISPLAY-HINT "d"
       STATUS current
       DESCRIPTION
           "A transport protocol number, from the 'protocol-numbers'
            IANA registry."
       SYNTAX Unsigned32 (0..255)

   NatPoolId ::= TEXTUAL-CONVENTION
       DISPLAY-HINT "d"
       STATUS current
```

```
        DESCRIPTION
            "A unique ID that is assigned to each pool."
        SYNTAX Unsigned32 (1..4294967295)

    NatBehaviorType ::= TEXTUAL-CONVENTION
        STATUS current
        DESCRIPTION
            "Behavior type as described in [RFC4787] sections 4.1 and 5."
        SYNTAX INTEGER {
            endpointIndependent (0),
            addressDependent (1),
            addressAndPortDependent (2)
        }

    NatPoolingType ::= TEXTUAL-CONVENTION
        STATUS current
        DESCRIPTION
            "Pooling type as described in [RFC4787] sections 4.1."
        SYNTAX INTEGER {
            arbitrary (0),
            paired (1)
        }

    VlanIndexOrZero ::= TEXTUAL-CONVENTION
        DISPLAY-HINT "d"
        STATUS      current
        DESCRIPTION
            "A value used to index per-VLAN tables: a value of 4095 is
             not permitted.  A value of 0 indicates no index is present.
             If the value is between 1 and 4094 inclusive, it represents
             an IEEE 802.1Q VLAN-ID with global scope within a given
             bridged domain (see VlanId textual convention in
             [RFC4363]).  If the value is greater than 4095, then it
             represents a VLAN with scope local to the particular agent,
             i.e., one without a global VLAN-ID assigned to it. Such
             VLANs are outside the scope of IEEE 802.1Q, but it is
             convenient to be able to manage them in the same way using
             this MIB."
        SYNTAX      Unsigned32

    SubscriberIndex ::= TEXTUAL-CONVENTION
        DISPLAY-HINT "d"
        STATUS current
        DESCRIPTION
            "A unique ID that is assigned to each subscriber."
        SYNTAX Unsigned32 (1..4294967295)

    SubscriberIdentifierType ::= TEXTUAL-CONVENTION
```

```
        STATUS      current
        DESCRIPTION
            "Type of additional classifying information used by the NAT
             to identify the subscriber from an incoming packet, when
             the packet source address is not sufficient to do so
             unambiguously.

             null(0)

                No additional information is needed.

             interfaces(1)

                A set of one or more ingress interface indexes specified
                by the [RFC2863] InterfaceIndex textual convention.

             vlan(2)

                An ingress VLAN index using the VlanIndexOrZero textual
                convention, which is the [RFC4363] VlanIndex textual
                convention modified for local use in this MIB.

             vpn(3)

                An ingress layer 3 VPN identifier using the [RFC4265]
                VPNIdOrZero textual convention.

             ipencaps(4)

                Incoming source address of an encapsulating IPv4 or IPv6
                tunnel (e.g., IPv6 as used in DS-Lite, [RFC6333]) as
                defined by the InetAddressType and InetAddress textual
                conventions.

             other(5)

                The implementation supports other classifiers and/or
                combinations of classifier types. In the latter case the
                implementation MUST specify the semantics of the
                combination ('OR' or 'AND')."

        SYNTAX       INTEGER {
                         null(0),
                         interfaces(1),
                         vlan(2),
                         vpn(3),
                         ipencaps(4),
                         other(5)
```

```
                    }

   SubsInterfaceIdRowIndex ::= TEXTUAL-CONVENTION
       DISPLAY-HINT "d"
       STATUS current
       DESCRIPTION
           "A unique ID that is assigned to each row in the
            natSubsInterfaceIdentifierTable."
       SYNTAX Unsigned32 (1..4294967295)


   -- notifications

   natNotifPoolWatermarkLow NOTIFICATION-TYPE
       OBJECTS { natPoolWatermarkLow }
       STATUS current
       DESCRIPTION
           "This notification is generated when a pool's usage
            percentage becomes lower than or equal to the specified
            threshold. The threshold is specified by the
            natPoolWatermarkLow object"
       ::= { natMIBNotifications 2 }

   natNotifPoolWatermarkHigh NOTIFICATION-TYPE
       OBJECTS { natPoolWatermarkHigh }
       STATUS current
       DESCRIPTION
           "This notification is generated when a pool's usage
            percentage becomes greater than or equal to the specified
            threshold. The threshold is specified by the
            natPoolWatermarkHigh object"
       ::= { natMIBNotifications 3 }

   natNotifMappings NOTIFICATION-TYPE
       OBJECTS { natMappingCreations, natMappingRemovals }
       STATUS current
       DESCRIPTION
           "This notification is generated when the number of active
            mappings exceeds the value of natMappingsNotifyThreshold."
       ::= { natMIBNotifications 4 }

   natNotifAddrMappings NOTIFICATION-TYPE
       OBJECTS { natAddressMappingCreations, natAddressMappingRemovals }
       STATUS current
       DESCRIPTION
           "This notification is generated when the number of active
            address mappings exceeds the value of
            natAddrMapNotifyThreshold."
```

```
        ::= { natMIBNotifications 5 }

    natNotifSubscriberMappings NOTIFICATION-TYPE
        OBJECTS { natSubscriberMappingCreations,
                  natSubscriberMappingRemovals }
        STATUS current
        DESCRIPTION
            "This notification is generated when the number of active
             mappings exceeds the value of natSubscriberMapNotifyThresh,
             unless natSubscriberMapNotifyThresh is zero.."
        ::= { natMIBNotifications 6 }


    -- instance table

    natInstanceTable OBJECT-TYPE
        SYNTAX SEQUENCE OF NatInstanceEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "Table of NAT instances."
        ::= { natMIBObjects 11 }

    natInstanceEntry OBJECT-TYPE
        SYNTAX NatInstanceEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "Objects related to a single NAT instance."
        INDEX { natInstanceIndex }
        ::= { natInstanceTable 1 }

    NatInstanceEntry ::=
        SEQUENCE {
            natInstanceIndex Unsigned32,
            natInstanceAlias DisplayString
        }

    natInstanceIndex OBJECT-TYPE
        SYNTAX Unsigned32
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "NAT instance index. Semantics of this number are
             implementation-specific. This object is used as an index for
             many tables defined below."
        ::= { natInstanceEntry 1 }
```

```
   natInstanceAlias OBJECT-TYPE
       SYNTAX DisplayString (SIZE (0..64))
       MAX-ACCESS read-write
       STATUS current
       DESCRIPTION
           "This object is an 'alias' name for the NAT instance as
            specified by a network manager, and provides a non-volatile
            'handle' for the instance.

            On the first instantiation of a NAT instance, the value of
            natInstanceAlias associated with that instance is the
            zero-length string.  As and when a value is written into an
            instance of natInstanceAlias through a network management
            set operation, then the agent must retain the supplied value
            in this obejct instance associated with the same interface
            for as long as that NAT instance remains instantiated,
            including across all re-initializations/reboots of the
            network management system, including those which result in a
            change of the interface's natInstanceIndex value.

            An example of the value which a network manager might store
            in this object for a NAT instance is the name/identifier of
            the interface that brings in internal traffic for this NAT
            instance or the name of the VRF for internal traffic.

            An agent may choose to provide read-only access if the agent
            itself assigns an identifier for the NAT instance. An agent
            which supports write access to this object is required to
            keep the value in non-valite storage, but it may limit the
            length of new values depending on how much storage is
            already occupied by the current values for other
            NAT instances."
       ::= { natInstanceEntry 2 }


   -- counters

   natCounters OBJECT IDENTIFIER ::= { natMIBObjects 12 }

   natCountersTable OBJECT-TYPE
       SYNTAX SEQUENCE OF NatCountersEntry
       MAX-ACCESS not-accessible
       STATUS current
       DESCRIPTION
           "Table of counters of a NAT instance. The counters are global
            across L4 protocols."
       ::= { natCounters 1 }
```

```
   natCountersEntry OBJECT-TYPE
       SYNTAX NatCountersEntry
       MAX-ACCESS not-accessible
       STATUS current
       DESCRIPTION
           "Counters related to a single NAT instance."
       INDEX { natInstanceIndex }
       ::= { natCountersTable 1 }

   NatCountersEntry ::=
       SEQUENCE {
           natTranslations             Counter64,
           natOutOfPortErrors          Counter64,
           natResourceErrors           Counter64,
           natQuotaDrops               Counter64,
           natMappingCreations         Counter64,
           natMappingRemovals          Counter64,
           natAddressMappingCreations  Counter64,
           natAddressMappingRemovals   Counter64
       }

   natTranslations OBJECT-TYPE
       SYNTAX Counter64
       MAX-ACCESS read-only
       STATUS current
       DESCRIPTION
           "The number of packets translated."
       ::= { natCountersEntry 1 }

   natOutOfPortErrors OBJECT-TYPE
       SYNTAX Counter64
       MAX-ACCESS read-only
       STATUS current
       DESCRIPTION
           "The number of packets not translated because no external
            port was available, excluding quota limitations."
       ::= { natCountersEntry 2 }

   natResourceErrors OBJECT-TYPE
       SYNTAX Counter64
       MAX-ACCESS read-only
       STATUS current
       DESCRIPTION
           "The number of packets not translated because of resource
            constraints (excluding out-of-ports error and quota drops)."
       ::= { natCountersEntry 3 }

   natQuotaDrops OBJECT-TYPE
```

```
        SYNTAX Counter64
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The number of incoming packets not translated because of
             quota limitations.  Quotas include absolute limits as well
             as limits on rate of allocation."
        ::= { natCountersEntry 4 }

    natMappingCreations OBJECT-TYPE
        SYNTAX Counter64
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Number of mapping creations. This includes static mappings."
        ::= { natCountersEntry 5 }

    natMappingRemovals OBJECT-TYPE
        SYNTAX Counter64
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Number of mapping removals. This includes static mappings."
        ::= { natCountersEntry 6 }

    natAddressMappingCreations OBJECT-TYPE
        SYNTAX Counter64
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Number of address mapping creations. This includes static
             mappings."
        ::= { natCountersEntry 7 }

    natAddressMappingRemovals OBJECT-TYPE
        SYNTAX Counter64
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Number of address mapping removals. This includes static
             mappings.

             The number of active mappings is equal to
             natAddressMappingCreations - natAddressMappingRemovals."
        ::= { natCountersEntry 8 }

    natL4ProtocolTable OBJECT-TYPE
        SYNTAX SEQUENCE OF NatL4ProtocolEntry
```

```
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "Table of protocols with per-protocol counters."
        ::= { natCounters 2 }

    natL4ProtocolEntry OBJECT-TYPE
        SYNTAX NatL4ProtocolEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "Per-protocol counters."
        INDEX { natInstanceIndex, natL4ProtocolNumber }
        ::= { natL4ProtocolTable 1 }

    NatL4ProtocolEntry ::=
        SEQUENCE {
            natL4ProtocolNumber            ProtocolNumber,
            natL4ProtocolTranslations      Counter64,
            natL4ProtocolOutOfPortErrors   Counter64,
            natL4ProtocolResourceErrors    Counter64,
            natL4ProtocolQuotaDrops        Counter64,
            natL4ProtocolMappingCreations  Counter64,
            natL4ProtocolMappingRemovals   Counter64
        }

    natL4ProtocolNumber OBJECT-TYPE
        SYNTAX ProtocolNumber
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "Counters in this conceptual row apply to packets using the
             transport protocol identified by this object's value."
        ::= { natL4ProtocolEntry 1 }

    natL4ProtocolTranslations  OBJECT-TYPE
        SYNTAX Counter64
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The number of packets translated."
        ::= { natL4ProtocolEntry 2 }

    natL4ProtocolOutOfPortErrors OBJECT-TYPE
        SYNTAX Counter64
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
```

```
            "The number of packets not translated because no external
             port was available."
        ::= { natL4ProtocolEntry 3 }

    natL4ProtocolResourceErrors OBJECT-TYPE
        SYNTAX Counter64
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The number of packets not translated because of resource
             constraints (excluding out-of-ports errors and quota
             drops)."
        ::= { natL4ProtocolEntry 4 }

    natL4ProtocolQuotaDrops OBJECT-TYPE
        SYNTAX Counter64
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The number of incoming packets not translated because of
             exceeded quotas. Quotas include absolute limits as well as
             limits on rate of allocation."
        ::= { natL4ProtocolEntry 5 }

    natL4ProtocolMappingCreations OBJECT-TYPE
        SYNTAX Counter64
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Number of mapping creations. This includes static mappings."
        ::= { natL4ProtocolEntry 6 }

    natL4ProtocolMappingRemovals OBJECT-TYPE
        SYNTAX Counter64
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Number of mapping removals. This includes static mappings.

             The number of active mappings is equal to
             natL4ProtocolMappingCreations -
             natL4ProtocolMappingRemovals."
        ::= { natL4ProtocolEntry 7 }


    -- limits

    natLimitsTable OBJECT-TYPE
```

```
        SYNTAX SEQUENCE OF NatLimitsEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "Table of limits for a NAT instance."
        ::= { natMIBObjects 13 }

    natLimitsEntry OBJECT-TYPE
        SYNTAX NatLimitsEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "Limit related to a single NAT instance."
        INDEX { natInstanceIndex }
        ::= { natLimitsTable 1 }

    NatLimitsEntry ::=
        SEQUENCE {
            natLimitMappings            Unsigned32,
            natMappingsNotifyThreshold  Unsigned32,
            natLimitAddressMappings     Unsigned32,
            natAddrMapNotifyThreshold   Unsigned32,
            natLimitFragments           Unsigned32,
            natLimitSubscribers         Unsigned32
        }

    natLimitMappings OBJECT-TYPE
        SYNTAX Unsigned32
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
            "Global limit on the total number of mappings. Zero means
             unlimited."
        ::= { natLimitsEntry 1 }

    natMappingsNotifyThreshold OBJECT-TYPE
        SYNTAX Unsigned32
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
            "See natNotifMappings."
        ::= { natLimitsEntry 2 }

    natLimitAddressMappings OBJECT-TYPE
        SYNTAX Unsigned32
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
```

```
            "Global limit on the total number of internal-to-external
             address mappings.  Zero means unlimited.

             This limit is only applicable to NATs that have an 'IP
             address pooling' behavior of 'Paired' [RFC4787]."
        ::= { natLimitsEntry 3 }

    natAddrMapNotifyThreshold OBJECT-TYPE
        SYNTAX Unsigned32
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
            "See natNotifAddrMappings."
        ::= { natLimitsEntry 4 }

    natLimitFragments OBJECT-TYPE
        SYNTAX Unsigned32
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
            "Global limit on the total number of fragments pending
             reassembly.  Zero means unlimited.

             This limit is only applicable to NATs having 'Receive
             Fragments Out of Order' behavior [RFC4787]."
        ::= { natLimitsEntry 5 }

    natLimitSubscribers OBJECT-TYPE
        SYNTAX Unsigned32
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
            "Global limit on the number of subscribers with active
             mappings.  Zero means unlimited."
        ::= { natLimitsEntry 6 }


    -- pools

    natPoolObjects OBJECT IDENTIFIER ::= { natMIBObjects 14 }

    natPoolTable OBJECT-TYPE
        SYNTAX SEQUENCE OF NatPoolEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "Table of pools."
        ::= { natPoolObjects 1 }
```

```
   natPoolEntry OBJECT-TYPE
       SYNTAX NatPoolEntry
       MAX-ACCESS not-accessible
       STATUS current
       DESCRIPTION
           "Entry in the table of pools."
       INDEX { natInstanceIndex, natPoolIndex }
       ::= { natPoolTable 1 }

   NatPoolEntry ::=
       SEQUENCE {
           natPoolIndex         NatPoolId,
           natPoolRealm         SnmpAdminString,
           natPoolWatermarkLow  Integer32,
           natPoolWatermarkHigh Integer32,
           natPoolPortMin       InetPortNumber,
           natPoolPortMax       InetPortNumber
       }

   natPoolIndex OBJECT-TYPE
       SYNTAX NatPoolId
       MAX-ACCESS not-accessible
       STATUS current
       DESCRIPTION
           "Index of an address pool."
       ::= { natPoolEntry 1 }

   natPoolRealm OBJECT-TYPE
       SYNTAX SnmpAdminString (SIZE (0..32))
       MAX-ACCESS read-only
       STATUS current
       DESCRIPTION
           "Realm to which this pool's addresses belong."
       ::= { natPoolEntry 2 }

   natPoolWatermarkLow OBJECT-TYPE
       SYNTAX Integer32 (-1|0..100)
       MAX-ACCESS read-create
       STATUS current
       DESCRIPTION
           "Low watermark on a pool's usage, in percentage of the total
            number of ports available. If set to -1, the watermark is
            disabled. Otherwise when the usage percentage becomes lower
            than or equal to natPoolWatermarkLow, a notification is
            sent. The NAT may also start behaving in low usage mode
            (this is implementation-defined).

            The pool's current usage percentage can be computed by
```

```
            summing (natPoolRangeAllocations -
            natPoolRangeDeallocations) over all address ranges
            belonging to this pool, then dividing by the total number of
            IP addresses in this pool and by the size of the port range
            in this pool (natPoolPortMax - natPoolPortMin + 1)."
        ::= { natPoolEntry 3 }

    natPoolWatermarkHigh OBJECT-TYPE
        SYNTAX Integer32 (-1|0..100)
        MAX-ACCESS read-create
        STATUS current
        DESCRIPTION
            "High watermark on a pool's usage, in percentage of the total
             number of ports available. If set to -1, the watermark is
             disabled. Otherwise, when the usage percentage becomes
             higher than or equal to natPoolWatermarkHigh, a notification
             is sent.  The NAT may also start behaving in high usage mode
             (this is implementation-defined)."
        ::= { natPoolEntry 4 }

    natPoolPortMin OBJECT-TYPE
        SYNTAX InetPortNumber
        MAX-ACCESS read-create
        STATUS current
        DESCRIPTION
            "Minimal port number to be allocated in this pool."
        ::= { natPoolEntry 5 }

    natPoolPortMax OBJECT-TYPE
        SYNTAX InetPortNumber
        MAX-ACCESS read-create
        STATUS current
        DESCRIPTION
            "Maximal port number to be allocated in this pool."
        ::= { natPoolEntry 6 }


    natPoolRangeTable OBJECT-TYPE
        SYNTAX SEQUENCE OF NatPoolRangeEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "This table contains address ranges used by pool entries."
        ::= { natPoolObjects 2 }

    natPoolRangeEntry OBJECT-TYPE
        SYNTAX NatPoolRangeEntry
        MAX-ACCESS not-accessible
```

```
        STATUS current
        DESCRIPTION
            "NAT pool address range."
        INDEX { natInstanceIndex, natPoolRangePoolIndex }
        ::= { natPoolRangeTable 1 }

    NatPoolRangeEntry ::=
        SEQUENCE {
            natPoolRangePoolIndex        NatPoolId,
            natPoolRangeType             InetAddressType,
            natPoolRangeBegin            InetAddress,
            natPoolRangeEnd              InetAddress,
            natPoolRangeAllocations      Counter64,
            natPoolRangeDeallocations    Counter64
        }

    natPoolRangePoolIndex OBJECT-TYPE
        SYNTAX NatPoolId
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "Index of the address pool to which this address range
             belongs.  See natPoolIndex."
        ::= { natPoolRangeEntry 1 }

    natPoolRangeType OBJECT-TYPE
        SYNTAX InetAddressType
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The address type of natPoolRangeBegin and
             natPoolRangeEnd."
        ::= { natPoolRangeEntry 2 }

    natPoolRangeBegin OBJECT-TYPE
        SYNTAX InetAddress
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Lowest address included in this range."
        ::= { natPoolRangeEntry 3 }

    natPoolRangeEnd OBJECT-TYPE
        SYNTAX InetAddress
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Highest address included in this range."
```

```
        ::= { natPoolRangeEntry 4 }

    natPoolRangeAllocations OBJECT-TYPE
        SYNTAX Counter64
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Number of ports that have been allocated on the addresses in
             this range."
        ::= { natPoolRangeEntry 5 }

    natPoolRangeDeallocations OBJECT-TYPE
        SYNTAX Counter64
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Number of ports that have been allocated and then
             deallocated on the addresses in this range.

             The number of ports currently allocated on the addresses in
             this range can be computed by subtracting
             natPoolRangeDeallocations from natPoolRangeAllocations."
        ::= { natPoolRangeEntry 6 }


    -- indexed mapping tables

    natMapObjects OBJECT IDENTIFIER ::= { natMIBObjects 15 }

    natMapIntAddrTable OBJECT-TYPE
        SYNTAX SEQUENCE OF NatMapIntAddrEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "Table of mappings from internal to external address.

             This table is only applicable to NATs that have an 'IP
             address pooling' behavior of 'Paired' [RFC4787]."
        ::= { natMapObjects 1 }

    natMapIntAddrEntry OBJECT-TYPE
        SYNTAX NatMapIntAddrEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "Mapping from internal to external address."
        INDEX { natInstanceIndex,
                natMapIntAddrIntRealm,
```

```
                natMapIntAddrIntType,
                natMapIntAddrInt }
        ::= { natMapIntAddrTable 1 }

    NatMapIntAddrEntry ::=
        SEQUENCE {
            natMapIntAddrIntRealm   SnmpAdminString,
            natMapIntAddrExtRealm   SnmpAdminString,
            natMapIntAddrIntType    InetAddressType,
            natMapIntAddrInt        InetAddress,
            natMapIntAddrExtType    InetAddressType,
            natMapIntAddrExt        InetAddress,
            natMapIntAddrSubsIndex  Unsigned32
        }

    natMapIntAddrIntRealm OBJECT-TYPE
        SYNTAX SnmpAdminString (SIZE(0..32))
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "Realm to which natMapIntAddrInt belongs."
        ::= { natMapIntAddrEntry 1 }

    natMapIntAddrExtRealm OBJECT-TYPE
        SYNTAX SnmpAdminString
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Realm to which natMapIntAddrExt belongs."
        ::= { natMapIntAddrEntry 2 }

    natMapIntAddrIntType OBJECT-TYPE
        SYNTAX InetAddressType
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "Address type for natMapIntAddrInt."
        ::= { natMapIntAddrEntry 3 }

    natMapIntAddrInt OBJECT-TYPE
        SYNTAX InetAddress (SIZE (4|16))
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "Internal address."
        ::= { natMapIntAddrEntry 4 }

    natMapIntAddrExtType OBJECT-TYPE
```

```
        SYNTAX InetAddressType
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Address type for natMapIntAddrExt."
        ::= { natMapIntAddrEntry 5 }

    natMapIntAddrExt OBJECT-TYPE
        SYNTAX InetAddress
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "External address."
        ::= { natMapIntAddrEntry 6 }

    natMapIntAddrSubsIndex OBJECT-TYPE
        SYNTAX Unsigned32 (0|1..4294967295)
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Subscriber to which this address mapping applies, or zero if
             it applies to all subscribers."
        ::= { natMapIntAddrEntry 7 }

    natMappingTable OBJECT-TYPE
        SYNTAX SEQUENCE OF NatMappingEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "Table of mappings indexed by external 3-tuple."
        ::= { natMapObjects 2 }

    natMappingEntry OBJECT-TYPE
        SYNTAX NatMappingEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "A single NAT mapping."
        INDEX { natInstanceIndex,
                natMappingProto,
                natMappingExtRealm,
                natMappingExtAddressType,
                natMappingExtAddress,
                natMappingExtPort }
        ::= { natMappingTable 1 }

    NatMappingEntry ::=
        SEQUENCE {
```

```
            natMappingProto          ProtocolNumber,
            natMappingExtRealm       SnmpAdminString,
            natMappingExtAddressType InetAddressType,
            natMappingExtAddress     InetAddress,
            natMappingExtPort        InetPortNumber,
            natMappingIntRealm       SnmpAdminString,
            natMappingIntAddressType InetAddressType,
            natMappingIntAddress     InetAddress,
            natMappingIntPort        InetPortNumber,
            natMappingPool           Unsigned32,
            natMappingMapBehavior    NatBehaviorType,
            natMappingFilterBehavior NatBehaviorType,
            natMappingAddressPooling NatPoolingType,
            natMappingSubsIndex      SubscriberIndex
        }

    natMappingProto OBJECT-TYPE
        SYNTAX ProtocolNumber
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "The mapping's transport protocol number."
        ::= { natMappingEntry 1 }

    natMappingExtRealm OBJECT-TYPE
        SYNTAX SnmpAdminString (SIZE(0..32))
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "The realm to which natMappingExtAddress belongs."
        ::= { natMappingEntry 2 }

    natMappingExtAddressType OBJECT-TYPE
        SYNTAX InetAddressType
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "Type of the mapping's external address."
        ::= { natMappingEntry 3 }

    natMappingExtAddress OBJECT-TYPE
        SYNTAX InetAddress (SIZE (4|16))
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "The mapping's external address. If this is the undefined
             address, all external addresses are mapped to the internal
             address."
```

```
        ::= { natMappingEntry 4 }

    natMappingExtPort OBJECT-TYPE
        SYNTAX InetPortNumber
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "The mapping's external port number. If this is zero, all
             external ports are mapped to the internal port."
        ::= { natMappingEntry 5 }

    natMappingIntRealm OBJECT-TYPE
        SYNTAX SnmpAdminString
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The realm to which natMappingIntAddress belongs."
        ::= { natMappingEntry 6 }

    natMappingIntAddressType OBJECT-TYPE
        SYNTAX InetAddressType
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Type of the mapping's internal address."
        ::= { natMappingEntry 7 }

    natMappingIntAddress OBJECT-TYPE
        SYNTAX InetAddress
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The mapping's internal address. If this is the undefined
             address, addresses are not translated."
        ::= { natMappingEntry 8 }

    natMappingIntPort OBJECT-TYPE
        SYNTAX InetPortNumber
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The mapping's internal port number. If this is zero, ports
             are not translated."
        ::= { natMappingEntry 9 }

    natMappingPool OBJECT-TYPE
        SYNTAX Unsigned32 (0|1..4294967295)
        MAX-ACCESS read-only
```

```
        STATUS current
        DESCRIPTION
            "Index of the pool that contains this mapping's external
             address and port. If zero, no pool is associated with this
             mapping."
        ::= { natMappingEntry 10 }

    natMappingMapBehavior OBJECT-TYPE
        SYNTAX NatBehaviorType
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Mapping behavior as described in [RFC4787] section 4.1."
        ::= { natMappingEntry 11 }

    natMappingFilterBehavior OBJECT-TYPE
        SYNTAX NatBehaviorType
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Filtering behavior as described in [RFC4787] section 5."
        ::= { natMappingEntry 12 }

    natMappingAddressPooling OBJECT-TYPE
        SYNTAX NatPoolingType
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Type of address pooling behavior that was used to create
             this mapping."
        ::= { natMappingEntry 13 }

    natMappingSubsIndex OBJECT-TYPE
        SYNTAX SubscriberIndex
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Subscriber using this mapping."
        ::= { natMappingEntry 14 }


    -- subscribers

    natSubscribers OBJECT IDENTIFIER ::= { natMIBObjects 16 }

    natSubscribersTable OBJECT-TYPE
        SYNTAX SEQUENCE OF NatSubscribersEntry
        MAX-ACCESS not-accessible
```

```
        STATUS current
        DESCRIPTION
            "Table of CGN subscribers."
        ::= { natSubscribers 1 }

    natSubscribersEntry OBJECT-TYPE
        SYNTAX NatSubscribersEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "Each entry describes a single CGN subscriber or a host
             served by a managed enterprise NAT."
        INDEX { natInstanceIndex,
                natSubscriberIndex }
        ::= { natSubscribersTable 1 }

    NatSubscribersEntry ::=
        SEQUENCE {
            natSubscriberIndex              SubscriberIndex,
            natSubscriberIdentifierType     SubscriberIdentifierType,
            natSubscriberIntPrefixType      InetAddressType,
            natSubscriberIntPrefix          InetAddress,
            natSubscriberIntPrefixLength    InetAddressPrefixLength,
            natSubscriberRealm              SnmpAdminString,
            natSubscriberTranslations       Counter64,
            natSubscriberOutOfPortErrors    Counter64,
            natSubscriberResourceErrors     Counter64,
            natSubscriberQuotaDrops         Counter64,
            natSubscriberMappingCreations   Counter64,
            natSubscriberMappingRemovals    Counter64,
            natSubscriberLimitMappings      Unsigned32,
            natSubscriberMapNotifyThresh    Unsigned32,
            natSubscriberVlanIdentifier     VlanIndexOrZero,
            natSubscriberVpnIdentifier      VPNIdOrZero,
            natSubscriberIPEncapsIdType     InetAddressType,
            natSubscriberIPEncapsIdAddr     InetAddress
        }

    natSubscriberIndex OBJECT-TYPE
        SYNTAX SubscriberIndex
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "Index of the subscriber or host."
        ::= { natSubscribersEntry 1 }

    natSubscriberIdentifierType OBJECT-TYPE
        SYNTAX SubscriberIdentifierType
```

```
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Type of additional information needed to identify the
             subscriber or host from incoming packets, when the packet
             source address does not do so unambiguously.

             The implementation MUST ensure that the type and the
             identifier value provided are synchronized, as follows.
             Unused identifier values MUST be zero or equivalent.

             Type           Identifier object

             null(0)        None.
             interfaces(1)  natSubsInterfaceIdentifierTable
             vlan(2)        natSubscriberVlanIdentifier
             vpn(3)         natSubscriberVpnIdentifier
             ipencaps(4)    natSubscriberIPEncapsIdType and
                            natSubscriberIPEncapsIdAddr
             other(5)       As specified by the implementation"
        ::= { natSubscribersEntry 2 }

    natSubscriberIntPrefixType OBJECT-TYPE
        SYNTAX InetAddressType
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Subscriber's internal prefix type."
        ::= { natSubscribersEntry 3 }

    natSubscriberIntPrefix OBJECT-TYPE
        SYNTAX InetAddress
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Prefix assigned to a subscriber's CPE."
        ::= { natSubscribersEntry 4 }

    natSubscriberIntPrefixLength OBJECT-TYPE
        SYNTAX InetAddressPrefixLength
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Length of the prefix assigned to a subscriber's CPE, in
             bits.  In case a single address is assigned, this will be 32
             for IPv4 and 128 for IPv6."
        ::= { natSubscribersEntry 5 }
```

```
   natSubscriberRealm OBJECT-TYPE
       SYNTAX SnmpAdminString
       MAX-ACCESS read-only
       STATUS current
       DESCRIPTION
           "The realm to which this subscriber belongs."
       ::= { natSubscribersEntry 6 }

   natSubscriberTranslations OBJECT-TYPE
       SYNTAX Counter64
       MAX-ACCESS read-only
       STATUS current
       DESCRIPTION
           "The number of translated packets received from or sent to
            this subscriber."
       ::= { natSubscribersEntry 7 }

   natSubscriberOutOfPortErrors OBJECT-TYPE
       SYNTAX Counter64
       MAX-ACCESS read-only
       STATUS current
       DESCRIPTION
           "The number of packets received from this subscriber not
            translated because no external port was available, excluding
            quota limitations."
       ::= { natSubscribersEntry 8 }

   natSubscriberResourceErrors OBJECT-TYPE
       SYNTAX Counter64
       MAX-ACCESS read-only
       STATUS current
       DESCRIPTION
           "The number of packets received from this subscriber not
            translated because of resource constraints (excluding
            out-of-port errors and quota drops)."
       ::= { natSubscribersEntry 9 }

   natSubscriberQuotaDrops OBJECT-TYPE
       SYNTAX Counter64
       MAX-ACCESS read-only
       STATUS current
       DESCRIPTION
           "The number of incoming packets received from or destined to
            this subscriber not translated because of quota limitations.
            Quotas include absolute limits as well as limits on the rate
            of allocation."
       ::= { natSubscribersEntry 10 }
```

```
   natSubscriberMappingCreations OBJECT-TYPE
       SYNTAX Counter64
       MAX-ACCESS read-only
       STATUS current
       DESCRIPTION
           "Number of mappings created by or for this subscriber."
       ::= { natSubscribersEntry 11 }

   natSubscriberMappingRemovals OBJECT-TYPE
       SYNTAX Counter64
       MAX-ACCESS read-only
       STATUS current
       DESCRIPTION
           "Number of mappings removed by or for this subscriber."
       ::= { natSubscribersEntry 12 }

   natSubscriberLimitMappings OBJECT-TYPE
       SYNTAX Unsigned32
       MAX-ACCESS read-write
       STATUS current
       DESCRIPTION
           "Limit on the number of active mappings created by or for
            this subscriber. Zero means unlimited."
       ::= { natSubscribersEntry 13 }

   natSubscriberMapNotifyThresh OBJECT-TYPE
       SYNTAX Unsigned32
       MAX-ACCESS read-write
       STATUS current
       DESCRIPTION
           "See natNotifSubscriberMappings."
       ::= { natSubscribersEntry 14 }

   natSubscriberVlanIdentifier OBJECT-TYPE
       SYNTAX VlanIndexOrZero
       MAX-ACCESS read-only
       STATUS current
       DESCRIPTION
           "When non-zero, VLAN index used to identify subscriber in
            combination with packet source address."
       ::= { natSubscribersEntry 15 }

   natSubscriberVpnIdentifier OBJECT-TYPE
       SYNTAX VPNIdOrZero
       MAX-ACCESS read-only
       STATUS current
       DESCRIPTION
           "When non-zero, VPN identifier used to identify subscriber
```

```
            in combination with packet source address."
        ::= { natSubscribersEntry 16 }

    natSubscriberIPEncapsIdType OBJECT-TYPE
        SYNTAX InetAddressType
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "When not unknown(0), type of address of encapsulating IP
             ingress tunnel."
        ::= { natSubscribersEntry 17 }

    natSubscriberIPEncapsIdAddr OBJECT-TYPE
        SYNTAX InetAddress
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Source address in outer header of packets incoming via IP
             tunnel, used to identify subscriber in combination with
             inner packet source address."
        ::= { natSubscribersEntry 18 }

    natSubsInterfaceIdentifierTable OBJECT-TYPE
        SYNTAX SEQUENCE OF NatSubsInterfaceIdentifierEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "Table of interface indexes. If non-empty, used along with
             packet source address to identify the subscriber sending
             the packet. 'OR' semantics if multiple interface indexes
             are present."
        ::= { natSubscribers 2 }

    natSubsInterfaceIdentifierEntry OBJECT-TYPE
        SYNTAX NatSubsInterfaceIdentifierEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "Each entry provides a single interface index."
        INDEX { natInstanceIndex,
                natSubsInterfaceIdSubsIndex,
                natSubsInterfaceIdRowIndex }
        ::= { natSubsInterfaceIdentifierTable 1 }

    NatSubsInterfaceIdentifierEntry ::=
        SEQUENCE {
            natSubsInterfaceIdSubsIndex      SubscriberIndex,
            natSubsInterfaceIdRowIndex       SubsInterfaceIdRowIndex,
```

```
            natSubsInterfaceIndex           InterfaceIndex
        }

    natSubsInterfaceIdSubsIndex OBJECT-TYPE
        SYNTAX SubscriberIndex
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "Index of the subscriber to which this conceptual table is
             related."
        ::= { natSubsInterfaceIdentifierEntry 1 }

    natSubsInterfaceIdRowIndex OBJECT-TYPE
        SYNTAX SubsInterfaceIdRowIndex
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "Row index."
        ::= { natSubsInterfaceIdentifierEntry 2 }

    natSubsInterfaceIndex OBJECT-TYPE
        SYNTAX InterfaceIndex
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Interface index of an ingress interface through which
             packets from this subscriber may flow."
        ::= { natSubsInterfaceIdentifierEntry 3 }


    -- object groups

    natGroupStatelessObjects OBJECT-GROUP
        OBJECTS { natInstanceAlias,
                  natTranslations,
                  natResourceErrors,
                  natQuotaDrops,
                  natMappingCreations,
                  natMappingRemovals,
                  natL4ProtocolTranslations ,
                  natL4ProtocolResourceErrors,
                  natL4ProtocolQuotaDrops,
                  natL4ProtocolMappingCreations,
                  natL4ProtocolMappingRemovals,
                  natMappingIntRealm,
                  natMappingIntAddressType,
                  natMappingIntAddress,
                  natMappingIntPort,
```

```
                natMappingPool,
                natMappingMapBehavior,
                natMappingFilterBehavior }
        STATUS current
        DESCRIPTION
            "Basic counters, limits, and thresholds that do not require
             stateful NAT. That is, they apply to both stateless and
             stateful NATs.

             For this MIB's purposes, stateless NATs are defined as NATs
             that do not create mappings dynamically (either implicitly
             or explicitly using, for instance, the Port Control
             Protocol).  Their mappings are created statically by the NAT
             administrator."
        ::= { natMIBGroups 7 }

    natGroupStatefulObjects OBJECT-GROUP
        OBJECTS { natOutOfPortErrors,
                  natL4ProtocolOutOfPortErrors,
                  natLimitMappings,
                  natMappingsNotifyThreshold,
                  natPoolRealm,
                  natPoolWatermarkLow,
                  natPoolWatermarkHigh,
                  natPoolPortMin,
                  natPoolPortMax,
                  natPoolRangeType,
                  natPoolRangeBegin,
                  natPoolRangeEnd,
                  natPoolRangeAllocations,
                  natPoolRangeDeallocations,
                  natMappingAddressPooling }
        STATUS current
        DESCRIPTION
            "Basic counters, limits, and thresholds that require stateful
             NAT."
        ::= { natMIBGroups 8 }

    natGroupAddrMapObjects OBJECT-GROUP
        OBJECTS { natAddressMappingCreations,
                  natAddressMappingRemovals,
                  natLimitAddressMappings,
                  natAddrMapNotifyThreshold,
                  natMapIntAddrExtRealm,
                  natMapIntAddrExtType,
                  natMapIntAddrExt }
        STATUS current
        DESCRIPTION
```

```
            "Objects that require 'Paired IP address pooling' behavior
             [RFC4787]."
        ::= { natMIBGroups 9 }

    natGroupFragmentObjects OBJECT-GROUP
        OBJECTS { natLimitFragments }
        STATUS current
        DESCRIPTION
            "Objects that require 'Receive Fragments Out of Order'
             behavior [RFC4787]."
        ::= { natMIBGroups 10 }

    natGroupBasicNotifications NOTIFICATION-GROUP
        NOTIFICATIONS { natNotifPoolWatermarkLow,
                        natNotifPoolWatermarkHigh,
                        natNotifMappings }
        STATUS current
        DESCRIPTION
            "Basic notifications."
        ::= { natMIBGroups 11 }

    natGroupAddrMapNotifications NOTIFICATION-GROUP
        NOTIFICATIONS { natNotifAddrMappings }
        STATUS current
        DESCRIPTION
            "Notifications about address mappings."
        ::= { natMIBGroups 12 }

    natGroupSubscriberObjects OBJECT-GROUP
        OBJECTS { natMapIntAddrSubsIndex,
                  natMappingSubsIndex,
                  natSubscriberIdentifierType,
                  natSubscriberIntPrefixType,
                  natSubscriberIntPrefix,
                  natSubscriberIntPrefixLength,
                  natSubscriberRealm,
                  natSubscriberTranslations,
                  natSubscriberOutOfPortErrors,
                  natSubscriberResourceErrors,
                  natSubscriberQuotaDrops,
                  natSubscriberMappingCreations,
                  natSubscriberMappingRemovals,
                  natSubscriberLimitMappings,
                  natSubscriberVlanIdentifier,
                  natSubscriberVpnIdentifier,
                  natSubscriberIPEncapsIdType,
                  natSubscriberIPEncapsIdAddr,
                  natSubsInterfaceIndex,
```

```
                natLimitSubscribers,
                natSubscriberMapNotifyThresh }
        STATUS current
        DESCRIPTION
            "Per-subscriber counters, limits, and thresholds."
        ::= { natMIBGroups 13 }

    natGroupSubscriberNotifications NOTIFICATION-GROUP
        NOTIFICATIONS { natNotifSubscriberMappings }
        STATUS current
        DESCRIPTION
            "Subscriber notifications."
        ::= { natMIBGroups 14 }


    -- compliance statements

    natBasicStatelessCompliance MODULE-COMPLIANCE
        STATUS current
        DESCRIPTION
            "Basic stateless compliance with this MIB is attained when
             the objects contained in the mandatory groups are
             implemented."
        MODULE  -- this module
            MANDATORY-GROUPS { natGroupStatelessObjects }

            OBJECT      natInstanceAlias
            MIN-ACCESS  read-only
            DESCRIPTION
                "Write access is not required."

        ::= { natMIBCompliances 3 }

    natBasicStatefulCompliance MODULE-COMPLIANCE
        STATUS current
        DESCRIPTION
            "Basic stateful compliance with this MIB is attained when the
             objects contained in the mandatory groups are implemented."
        MODULE  -- this module
            MANDATORY-GROUPS { natGroupStatelessObjects,
                               natGroupStatefulObjects,
                               natGroupBasicNotifications }
        ::= { natMIBCompliances 4 }

    natAddrMapCompliance MODULE-COMPLIANCE
        STATUS current
        DESCRIPTION
            "NATs that have 'Paired IP address pooling' behavior
```

```
            [RFC4787] and implement the objects in this group can claim
            this level of compliance."
        MODULE  -- this module
            MANDATORY-GROUPS { natGroupStatelessObjects,
                               natGroupStatefulObjects,
                               natGroupBasicNotifications,
                               natGroupAddrMapObjects,
                               natGroupAddrMapNotifications }
        ::= { natMIBCompliances 5 }

    natFragmentsCompliance MODULE-COMPLIANCE
        STATUS current
        DESCRIPTION
            "NATs that have 'Receive Fragments Out of Order' behavior
            [RFC4787] and implement the objects in this group can claim
            this level of compliance."
        MODULE  -- this module
            MANDATORY-GROUPS { natGroupStatelessObjects,
                               natGroupStatefulObjects,
                               natGroupBasicNotifications,
                               natGroupFragmentObjects }
        ::= { natMIBCompliances 6 }

    natCGNCompliance MODULE-COMPLIANCE
        STATUS current
        DESCRIPTION
            "NATs that have 'Paired IP address pooling' and 'Receive
            Fragments Out of Order' behavior [RFC4787] and implement the
            objects in this group can claim this level of compliance.

            This level of compliance is to be expected of a CGN
            compliant with [RFC6888]."
        MODULE  -- this module
            MANDATORY-GROUPS { natGroupStatelessObjects,
                               natGroupStatefulObjects,
                               natGroupBasicNotifications,
                               natGroupAddrMapObjects,
                               natGroupAddrMapNotifications,
                               natGroupFragmentObjects,
                               natGroupSubscriberObjects,
                               natGroupSubscriberNotifications }
        ::= { natMIBCompliances 7 }


    END
```

5. Security Considerations

There are a number of management objects defined in this MIB module with a MAX-ACCESS clause of read-write and/or read-create. Such objects may be considered sensitive or vulnerable in some network environments. The support for SET operations in a non-secure environment without proper protection can have a negative effect on network operations. These are the tables and objects and their sensitivity/vulnerability:

Limits: An attacker setting a very low or very high limit can easily cause a denial-of-service situation.

* natLimitMappings
* natLimitAddressMappings
* natLimitFragments
* natLimitSubscribers
* natSubscriberLimitMappings

Notification thresholds: An attacker setting an arbitrarily low treshold can cause many useless notifications to be generated. Setting an arbitrarily high threshold can effectively disable notifications, which could be used to hide another attack.

* natMappingsNotifyThreshold
* natAddrMapNotifyThreshold
* natSubscriberMapNotifyThresh

Some of the readable objects in this MIB module (i.e., objects with a MAX-ACCESS other than not-accessible) may be considered sensitive or vulnerable in some network environments. It is thus important to control even GET and/or NOTIFY access to these objects and possibly to even encrypt the values of these objects when sending them over the network via SNMP. These are the tables and objects and their sensitivity/vulnerability:

Objects that reveal host identities: Various objects can reveal the identity of private hosts that are engaged in a session with external end nodes. A curious outsider could monitor these to assess the number of private hosts being supported by the NAT device. Further, a disgruntled former employee of an enterprise could use the information to break into specific private hosts by

intercepting the existing sessions or originating new sessions into the host.

* natMapIntAddrType
* natMapIntAddrInt
* natMapIntAddrExt
* natMappingIntRealm
* natMappingIntAddressType
* natMappingIntAddress
* natMappingIntPort
* natMappingMapBehavior
* natMappingFilterBehavior
* natMappingAddressPooling
* natSubscriberIntPrefixType
* natSubscriberIntPrefix
* natSubscriberIntPrefixLength

Other objects that reveal NAT state: Other managed objects in this MIB may contain information that may be sensitive from a business perspective, in that they may represent NAT state information.

* natCntAddressMappings
* natCntProtocolMappings
* natPoolUsage
* natPoolRangeAllocatedPorts
* natSubscriberCntMappings

There are no objects that are sensitive in their own right, such as passwords or monetary amounts.

SNMP versions prior to SNMPv3 did not include adequate security. Even if the network itself is secure (for example by using IPsec),

there is no control as to who on the secure network is allowed to access and GET/SET (read/change/create/delete) the objects in this MIB module.

Implementations SHOULD provide the security features described by the SNMPv3 framework (see [RFC3410]), and implementations claiming compliance to the SNMPv3 standard MUST include full support for authentication and privacy via the User-based Security Model (USM) [RFC3414] with the AES cipher algorithm [RFC3826]. Implementations MAY also provide support for the Transport Security Model (TSM) [RFC5591] in combination with a secure transport such as SSH [RFC5592] or TLS/DTLS [RFC6353].

Further, deployment of SNMP versions prior to SNMPv3 is NOT RECOMMENDED. Instead, it is RECOMMENDED to deploy SNMPv3 and to enable cryptographic security. It is then a customer/operator responsibility to ensure that the SNMP entity giving access to an instance of this MIB module is properly configured to give access to the objects only to those principals (users) that have legitimate rights to indeed GET or SET (change/create/delete) them.

6. IANA Considerations

IANA has assigned object identifier 123 to the natMIB module, with prefix iso.org.dod.internet.mgmt.mib-2 in the Network Management Parameters registry [SMI-NUMBERS].

No IANA actions are required by this document.

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2578] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.

[RFC2579] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Textual Conventions for SMIv2", STD 58, RFC 2579, April 1999.

[RFC2580] McCloghrie, K., Perkins, D., and J. Schoenwaelder, "Conformance Statements for SMIv2", STD 58, RFC 2580, April 1999.

[RFC2863] McCloghrie, K. and F. Kastenholz, "The Interfaces Group MIB", RFC 2863, June 2000.

[RFC3414] Blumenthal, U. and B. Wijnen, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)", STD 62, RFC 3414, December 2002.

[RFC3826] Blumenthal, U., Maino, F., and K. McCloghrie, "The Advanced Encryption Standard (AES) Cipher Algorithm in the SNMP User-based Security Model", RFC 3826, June 2004.

[RFC4001] Daniele, M., Haberman, B., Routhier, S., and J. Schoenwaelder, "Textual Conventions for Internet Network Addresses", RFC 4001, February 2005.

[RFC4265] Schliesser, B. and T. Nadeau, "Definition of Textual Conventions for Virtual Private Network (VPN) Management", RFC 4265, November 2005.

[RFC4363] Levi, D. and D. Harrington, "Definitions of Managed Objects for Bridges with Traffic Classes, Multicast Filtering, and Virtual LAN Extensions", RFC 4363, January 2006.

[RFC4750] Joyal, D., Galecki, P., Giacalone, S., Coltun, R., and F. Baker, "OSPF Version 2 Management Information Base", RFC 4750, December 2006.

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

[RFC5591] Harrington, D. and W. Hardaker, "Transport Security Model for the Simple Network Management Protocol (SNMP)", RFC 5591, June 2009.

[RFC5592] Harrington, D., Salowey, J., and W. Hardaker, "Secure Shell Transport Model for the Simple Network Management Protocol (SNMP)", RFC 5592, June 2009.

[RFC6353] Hardaker, W., "Transport Layer Security (TLS) Transport Model for the Simple Network Management Protocol (SNMP)", RFC 6353, July 2011.

7.2. Informative References

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

[RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January 2001.

[RFC3410] Case, J., Mundy, R., Partain, D., and B. Stewart, "Introduction and Applicability Statements for Internet-Standard Management Framework", RFC 3410, December 2002.

[RFC4008] Rohit, R., Srisuresh, P., Raghunarayan, R., Pai, N., and C. Wang, "Definitions of Managed Objects for Network Address Translators (NAT)", RFC 4008, March 2005.

[RFC6333] Durand, A., Droms, R., Woodyatt, J., and Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion", RFC 6333, August 2011.

[RFC6674] Brockners, F., Gundavelli, S., Speicher, S., and D. Ward, "Gateway-Initiated Dual-Stack Lite Deployment", RFC 6674, July 2012.

[RFC6888] Perreault, S., Yamagata, I., Miyakawa, S., Nakagawa, A., and H. Ashida, "Common Requirements for Carrier-Grade NATs (CGNs)", BCP 127, RFC 6888, April 2013.

[SMI-NUMBERS] , "Network Management Parameters registry at IANA", , <http://www.iana.org/assignments/smi-numbers>.

Authors' Addresses

Simon Perreault
Viagenie
246 Aberdeen
Quebec, QC G1R 2E1
Canada

Phone: +1 418 656 9254
Email: simon.perreault@viagenie.ca
URI: http://viagenie.ca

Tina Tsou
Huawei Technologies (USA)
2330 Central Expressway
Santa Clara, CA 95050
USA

Phone: +1 408 330 4424
Email: tina.tsou.zouting@huawei.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

Behave WG
Internet-Draft
Intended status: Standards Track
Expires: October 06, 2013

T. Savolainen
Nokia
J. Korhonen
Nokia Siemens Networks
D. Wing
Cisco Systems
April 04, 2013

Discovery of the IPv6 Prefix Used for IPv6 Address Synthesis
draft-ietf-behave-nat64-discovery-heuristic-17.txt

Abstract

This document describes a method for detecting the presence of DNS64 and for learning the IPv6 prefix used for protocol translation on an access network. The method depends on the existence of a well-known IPv4-only fully qualified domain name "ipv4only.arpa". The information learned enables nodes to perform local IPv6 address synthesis and to potentially avoid NAT64 on dual-stack and multi-interface deployments.

Table of Contents

1. Introduction

As part of the transition to IPv6, NAT64 [RFC6146] and DNS64 [RFC6147] technologies will be utilized by some access networks to provide IPv4 connectivity for IPv6-only nodes [RFC6144]. DNS64 utilizes IPv6 address synthesis to create local IPv6 addresses for peers having only IPv4 addresses, hence allowing DNS-using IPv6-only nodes to communicate with IPv4-only peers.

However, DNS64 cannot serve applications not using DNS, such as those receiving IPv4 address literals as referrals. Such applications could nevertheless be able to work through NAT64, provided they are able to create locally valid IPv6 addresses that would be translated to the peers' IPv4 addresses.

Additionally, DNS64 is not able to do IPv6 address synthesis for nodes running validating DNSSEC-enabled DNS resolvers, but instead the synthesis must be done by the nodes themselves. In order to perform IPv6 synthesis, nodes have to learn the IPv6 prefix(es) used on the access network for protocol translation. A prefix, which may be a Network Specific Prefix (NSP) or Well-Known Prefix (WKP) [RFC6052], is referred to in this document as Pref64::/n [RFC6146].

This document describes a best-effort method for applications and nodes to learn the information required to perform local IPv6 address synthesis. The IPv6 address synthesis procedure itself is out-of-scope of this document. An example application is a browser encountering IPv4 address literals in an IPv6-only access network. Another example is a node running a validating security-aware DNS resolver in an IPv6-only access network.

The knowledge of IPv6 address synthesis taking place may also be useful if DNS64 and NAT64 are used in dual-stack enabled access networks or if a node is multi-interfaced [RFC6418]. In such cases, nodes may choose to prefer IPv4 or an alternative network interface in order to avoid traversal through protocol translators.

It is important to note that use of this approach will not result in a system that is as robust, secure, and well-behaved as an all-IPv6 system would be. Hence it is highly recommended to upgrade nodes' destinations to IPv6 and utilize the described method only as a transition solution.

2. Requirements and Terminology

2.1. Requirements

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2.2. Terminology

NAT64 FQDN: a fully qualified domain name (FQDN) for a NAT64 protocol translator.

Pref64::/n: a IPv6 prefix used for IPv6 address synthesis [RFC6146].

Pref64::WKA: an IPv6 address consisting of Pref64::/n and WKA at any of the locations allowed by RFC 6052 [RFC6052].

Secure Channel: a communication channel a node has between itself and a DNS64 server protecting DNS protocol related messages from interception and tampering. The channel can be, for example, IPsec-based virtual private network (VPN) tunnel or a link layer utilizing data encryption technologies.

Well-Known IPv4-only Name (WKN): the fully qualified domain name, "ipv4only.arpa", well-known to have only A record(s).

Well-Known IPv4 Address (WKA): an IPv4 address that is well-known and present in an A record for the well-known name. Two well-known IPv4 addresses are defined for Pref64::/n discovery purposes: 192.0.0.170 and 192.0.0.171.

3. Node Behavior

A node requiring information about the presence (or absence) of NAT64, and one or more Pref64::/n used for protocol translation SHALL send a DNS query for AAAA resource records of the Well-Known IPv4-only Name (WKN) "ipv4only.arpa". The node MAY perform the DNS query in both IPv6-only and dual-stack access networks.

When sending a DNS AAAA resource record query for the WKN, a node MUST set the "Checking Disabled (CD)" bit to zero [RFC4035], as otherwise the DNS64 server will not perform IPv6 address synthesis (Section 3 of [RFC6147]) and hence would not reveal the Pref64::/n used for protocol translation.

A DNS reply with one or more AAAA resource records indicates that the access network is utilizing IPv6 address synthesis. In some scenarios captive portals, or NXDOMAIN and NODATA hijacking, performed by the access network may result in a false positive. One method to detect such hijacking is to query a fully qualified domain name that is known to be invalid (and normally return an empty response or an error response) and see if it returns a valid resource record. However, as long as the hijacked domain does not result in AAAA resource record responses that contain well-known IPv4 address in any location defined by RFC6052, the response will not disturb the Pref64::/n learning procedure.

A node MUST look through all of the received AAAA resource records to collect one or more Pref64::/n. The Pref64::/n list might include the Well-Known Prefix 64:ff9b::/96 [RFC6052] or one or more Network-Specific Prefixes. In the case of NSPs, the node SHALL determine the used address format by searching the received IPv6 addresses for the

WKN's well-known IPv4 addresses. The node SHALL assume the well-known IPv4 addresses might be found at the locations specified by [RFC6052] section 2.2. The node MUST check on octet boundaries to ensure a 32-bit well-known IPv4 address value is present only once in an IPv6 address. In case another instance of the value is found inside the IPv6 address, the node SHALL repeat the search with the other well-known IPv4 address.

If only one Pref64::/n was present in the DNS response: a node SHALL use that Pref64::/n for both local synthesis and for detecting synthesis done by the DNS64 server on the network.

If more than one Pref64::/n was present in the DNS response: a node SHOULD use all of them when determining whether other received IPv6 addresses are synthetic. The node MUST use all learned Pref64::/n when performing local IPv6 address synthesis, and use the prefixes in the order received from the DNS64 server. That is, when the node is providing a list of locally synthesized IPv6 addresses to upper layers, IPv6 addresses MUST be synthesized by using all discovered Pref64::/n in the received order.

If the well-known IPv4 addresses are not found within the standard locations, it indicates that the network is not using a standard address format or unexpected IPv4 addresses were used in the AAAA resource record synthesis. In either case, the Pref64::/n cannot be determined and the heuristic procedure has failed. Developers can over time learn of IPv6 translated address formats that are extensions or alternatives to the standard formats. Developers MAY at that point add additional steps to the described discovery procedure. The additional steps are outside the scope of the present document.

In case a node does not receive a positive DNS reply to the AAAA resource record query, the node MAY perform a DNS A resource record query for the well-known name. If the node receives a positive reply to the DNS A resource record query it means the used recursive DNS server is not a DNS64 server.

In the case of a negative response (NXDOMAIN, NODATA) or a DNS query timeout: a DNS64 server is not available on the access network, the access network filtered out the well-known query, or something went wrong in the DNS resolution. All unsuccessful cases result in a node being unable to perform local IPv6 address synthesis. In the case of timeout, the node SHOULD retransmit the DNS query like any other DNS query the node makes [RFC1035]. In the case of a negative response (NXDOMAIN, NODATA), the node MUST obey the Time-To-Live [RFC1035] of the response before resending the AAAA resource record query. The node MAY monitor for DNS replies with IPv6 addresses constructed from

the WKP, in which case if any are observed the node SHOULD use the WKP as if it were learned during the query for the well-known name.

To save Internet resources if possible, a node should perform Pref64::/n discovery only when needed (e.g., when local synthesis is required, a new network interface is connected to a new network, and so forth). The node SHALL cache the replies it receives during the Pref64::/n discovery procedure and it SHOULD repeat the discovery process ten seconds before the Time-To-Live of the Well-Known Name's synthetic AAAA resource record expires.

3.1. Validation of Discovered Pref64::/n

If a node is using an insecure channel between itself and a DNS64 server, or the DNS64 server is untrusted, it is possible for an attacker to influence the node's Pref64::/n discovery procedures. This may result in denial-of-service, redirection, man-in-the-middle, or other attacks.

To mitigate against attacks, the node SHOULD communicate with a trusted DNS64 server over a secure channel, or use DNSSEC. NAT64 operators SHOULD provide facilities for validating discovery of Pref64::/n via a secure channel and/or DNSSEC protection.

It is important to understand that DNSSEC only validates that the discovered Pref64::/n is the one that belongs to a domain used by NAT64 FQDN. Importantly, the DNSSEC validation does not tell if the node is at the network where the Pref64::/n is intended to be used. Furthermore, DNSSEC validation cannot be utilized in the case of WKP.

3.1.1. DNSSEC Requirements for the Network

If the operator has chosen to support nodes performing validation of discovered Pref64::/n with DNSSEC, the operator of the NAT64 device MUST perform the following configurations.

1. Have one or more fully qualified domain names for the NAT64 translator entities (later referred as NAT64 FQDN). In the case of more than one Pref64::/n being used in a network, e.g., for load-balancing purposes, it is for network administrators to decide whether a single NAT64's fully qualified domain name maps to more than one Pref64::/n, or whether there will be a dedicated NAT64 FQDN per Pref64::/n.

2. Each NAT64 FQDN MUST have one or more DNS AAAA resource records containing Pref64::WKA (Pref64::/n combined with WKA).

3. Each Pref64::WKA MUST have a PTR resource record that points to the corresponding NAT64 FQDN.

4. Sign the NAT64 FQDNs' AAAA and A resource records with DNSSEC.

3.1.2. DNSSEC Requirements for the Node

A node SHOULD prefer a secure channel to talk to a DNS64 server, whenever possible. In addition, a node that implements a DNSSEC validating resolver MAY use the following procedure to validate discovery of the Pref64::/n.

1. Heuristically find Pref64::/n candidates by making a AAAA resource record query for "ipv4only.arpa" by following the procedure in Section 3. This will result in IPv6 addresses consisting of Pref64::/n combined with WKA, i.e., Pref64::WKA. For each Pref64::/n that the node wishes to validate, the node performs the following steps.

2. Send a DNS PTR resource record query for the IPv6 address of the translator (for "ip6.arpa"), using the Pref64::WKA learned in step 1. CNAME and DNAME results should be followed according to the rules in RFC 1034 [RFC1034], RFC 1034 [RFC1035], and RFC 6672 [RFC6672]. The ultimate response will include one or more NAT64 FQDNs.

3. The node SHOULD compare the domains of learned NAT64 FQDNs to a list of the node's trusted domains and choose a NAT64 FQDN that matches. The means for a node to learn the trusted domains is implementation-specific. If the node has no trust for the domain, the discovery procedure is not secure and the remaining steps described below MUST NOT be performed.

4. Send a DNS AAAA resource record query for the NAT64 FQDN.

5. Verify the DNS AAAA resource record contains Pref64::WKA addresses received at the step 1. It is possible that the NAT64 FQDN has multiple AAAA records, in which case the node MUST check if any of the addresses match the ones obtained in step 1. The node MUST ignore other responses and not use them for local IPv6 address synthesis.

6. Perform DNSSEC validation of the DNS AAAA response.

After the node has successfully performed the above five steps, the node can consider Pref64::/n validated.

3.2. Connectivity Check

After learning a Pref64::/n, the node SHOULD perform a connectivity check to ensure the learned Pref64::/n is functional. It could be non-functional for a variety of reasons -- the discovery failed to work as expected, the IPv6 path to the NAT64 is down, the NAT64 is down, or the IPv4 path beyond the NAT64 is down.

There are two main approaches to determine if the learned Pref64::/n is functional. The first approach is to perform a dedicated connectivity check. The second approach is to simply attempt to use the learned Pref64::/n. Each approach has some trade-offs (e.g., additional network traffic or possible user-noticeable delay), and implementations should carefully weigh which approach is appropriate for their application and the network.

The node SHOULD use an implementation-specific connectivity check server and a protocol of the implementation's choice, but if that is not possible, a node MAY do a PTR resource record query of the Pref64::WKA to get a NAT64 FQDN. The node then does an A resource query of the NAT64 FQDN, which will return zero or more A resource records pointing to connectivity check servers used by the network operator. A negative response to the PTR or A resource query means there are no connectivity check servers available. A network operator that provides NAT64 services for a mix of nodes with and without implementation-specific connectivity check servers SHOULD assist nodes in their connectivity checks by mapping each NAT64 FQDN to one or more DNS A resource records with IPv4 address(es) pointing to connectivity check server(s). The Pref64::/n -based connectivity check approach works only with NSPs, as it is not possible to register A records for each different domain using WKP. The network operator MUST disable ICMPv6 rate limiting for connectivity check messages.

In case of multiple connectivity check servers being available for use, the node chooses the first one, preferring implementation-specific servers.

The connectivity check protocol used with implementation-specific connectivity check servers is implementation-specific.

The connectivity check protocol used with connectivity check servers pointed to by the NAT64 FQDN's A resource records is ICMPv6 [RFC4443]. The node performing a connectivity check against these servers SHALL send an ICMPv6 Echo Request to an IPv6 address synthesized by combining discovered Pref64::/n with an IPv4 address of the server as specified in [RFC6052]. This will test the IPv6 path to the NAT64, the NAT64's operation, and the IPv4 path all the

way to the connectivity check server. If no response is received for the ICMPv6 Echo Request, the node SHALL send another ICMPv6 Echo Request, a second later. If still no response is received, the node SHALL send a third ICMPv6 Echo Request two seconds later. If an ICMPv6 Echo Response is received, the node knows the IPv6 path to the connectivity check server is functioning normally. If, after the three transmissions and three seconds since the last ICMPv6 Echo Request, no response is received, the node learns this Pref64::/n might not be functioning, and the node MAY choose a different Pref64::/n (if available), choose to alert the user, or proceed anyway assuming the failure is temporary or with the connectivity check itself. After all, the ICMPv6 is by design unreliable and failure to receive ICMPv6 responses may not indicate anything other than network failure to transport ICMPv6 messages through.

If no separate connectivity check is performed before local IPv6 address synthesis, a node MAY monitor success of connection attempts performed with locally synthesized IPv6 addresses. Based on success of these connections, and based on possible ICMPv6 error messages received (such as Destination Unreachable messages), the node MAY cease to perform local address synthesis and MAY restart the Pref64::/n discovery procedures.

3.2.1. No Connectivity Checks Against ipv4only.arpa

Clients MUST NOT send a connectivity check to an address returned by the ipv4only.arpa query. This is because, by design, no server will be operated on the Internet at that address as such. Similarly, network operators MUST NOT operate a server on that address. The reason this address isn’t used for connectivity checks is that operators who neglect to operate a connectivity check server will allow that traffic towards the Internet where it will be dropped and cause a false negative connectivity check with the client (that is, the NAT64 is working fine, but the connectivity check fails because a server is not operating at "ipv4only.arpa" on the Internet and a server is not operated by the NAT64 operator). Instead, for the connectivity check, an additional DNS resource record is looked up and used for the connectivity check. This ensures that packets don’t unnecessarily leak to the Internet and reduces the chance of a false negative connectivity check.

3.3. Alternative Fully Qualified Domain Names

Some applications, operating systems, devices, or networks may find it advantageous to operate their own DNS infrastructure to perform a function similar to "ipv4only.arpa", but using a different resource record. The primary advantage is to ensure availability of the DNS infrastructure and ensure the proper configuration of the DNS record

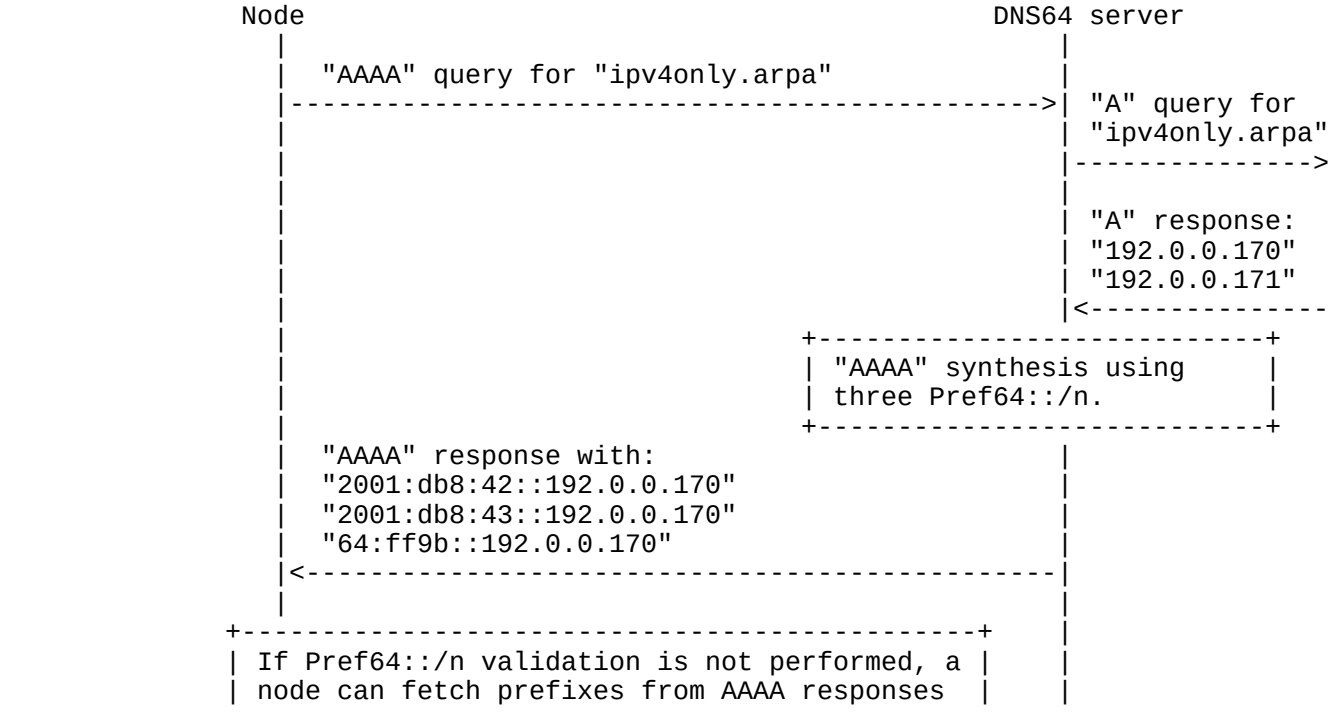
itself. For example, a company named Example might have their application query "ipv4only.example.com". Other than the different DNS resource record being queried, the rest of the operations are anticipated to be identical to the steps described in this document.

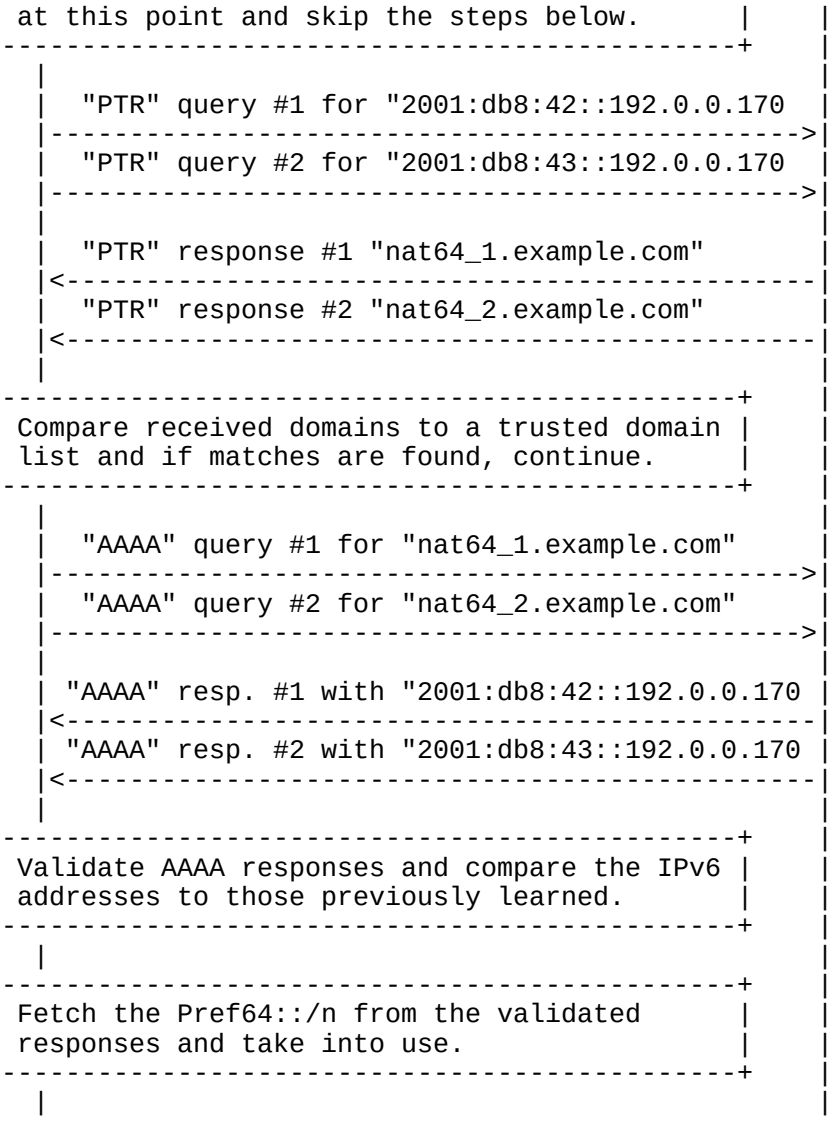
3.4. Message Flow Illustration

The figure below gives an example illustration of a message flow in the case of prefix discovery utilizing Pref64::/n validation. The figure shows also a step where the procedure ends if no Pref64::/n validation is performed.

In this example, three Pref64::/n are provided by the DNS64 server. The first Pref64::/n is using an NSP, in this example "2001:db8:42::/96". The second Pref64::/n is using an NSP, in this example "2001:db8:43::/96". The third Pref64::/n is using the WKP. Hence, when the Pref64::/n are combined with the WKA to form Pref64::WKA, the synthetic IPv6 addresses returned by DNS64 server are "2001:db8:42::192.0.0.170", "2001:db8:43::192.0.0.170", and "64:ff9b::192.0.0.170". The DNS64 server could also return synthetic addresses containing the IPv4 address 192.0.0.171.

The validation is not done for the WKP, see Section 3.1.





Pref64::/n discovery procedure

4. Operational Considerations for Hosting the IPv4-Only Well-Known Name

The authoritative name server for the well-known name SHALL have DNS record Time-To-Live (TTL) set to at least 60 minutes in order to improve effectiveness of DNS caching. The exact TTL value will be determined and tuned based on operational experiences.

The domain serving the well-known name MUST be signed with DNSSEC. See also section 7.

5. Operational Considerations for DNS64 Operator

A network operator of a DNS64 server can guide nodes utilizing heuristic discovery procedures by managing the responses a DNS64 server provides.

If the network operator would like nodes to utilize multiple Pref64::/n, the operator needs to configure DNS64 servers to respond with multiple synthetic AAAA records. As per Section 3 the nodes can then use them all.

There are no guarantees on which of the Pref64::/n nodes will end up using. If the operator wants nodes to specifically use a certain Pref64::/n or periodically change the Pref64::/n they use, for example for load balancing reasons, the only guaranteed method is to make DNS64 servers return only a single synthetic AAAA resource record, and have the Time-To-Live of that synthetic record such that the node repeats the Pref64::/n discovery when required.

Besides choosing how many Pref64::/n to respond and what Time-To-Live to use, DNS64 servers MUST NOT interfere with or perform other special procedures for the queries related to the well-known name.

5.1. Mapping of IPv4 Address Ranges to IPv6 Prefixes

RFC 6147 [RFC6147] allows DNS64 implementations to be able to map specific IPv4 address ranges to separate Pref64::/n. That allows handling of special use IPv4 addresses [RFC5735]. The example setup where this might be used is illustrated in Figure 1. The NAT64 "A" is used when accessing IPv4-only servers in the datacenter, and the NAT64 "B" is used for Internet access.

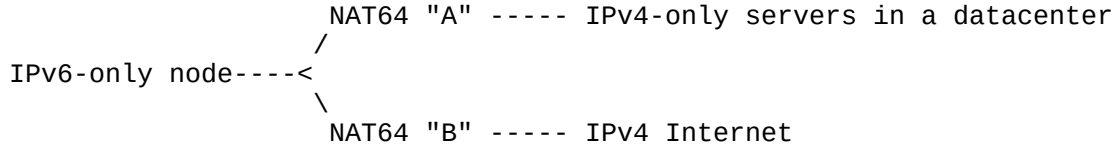


Figure 1: NAT64s with IPv4 Address Ranges

The heuristic discovery method described herein does not support learning of the possible rules used by a DNS64 server for mapping specific IPv4 address ranges to separate Pref64::/n. Therefore, nodes will use the same discovered Pref64::/n to synthesize IPv6 addresses from any IPv4 address. This can cause issues for routing

and connectivity establishment procedures. The operator of the NAT64 and the DNS64 ought to take this into account in the network design.

The network operators can help IPv6-only nodes by ensuring the nodes do not have to work with IPv4 address literals for which special mapping rules are used. That is, the IPv4-only servers addressed from the special IPv4 address ranges ought to have signed AAAA records, which allows IPv6-only nodes to avoid local address synthesis. If the IPv6-only nodes are not using DNSSEC, then it is enough if the network's DNS64 server returns synthetic AAAA resource records pointing to IPv4-only servers. Avoiding the need for IPv6-only nodes to perform address synthesis for IPv4 addresses belonging to special ranges is the best approach to assist nodes.

If the IPv6-only nodes have no other choice than using IPv4-address literals belonging to special IPv4 address ranges, and the IPv6-only node will perform local synthesis by using the discovered Pref64::/n, then the network ought to ensure with routing that the packets are delivered to the correct NAT64. For example, a router in the path from an IPv6-only host to NAT64s can forward the IPv6 packets to the correct NAT64 as illustrated in Figure 2. The routing could be based on the last 32-bits of the IPv6 address, but the network operator can also use some other IPv6 address format allowed by RFC 6052 [RFC6052], if it simplifies routing setup. This setup requires additional logic on the NAT64 providing connectivity to special IPv4 address ranges: it needs to be able to translate packets it receives that are using the Pref64::/n used with Internet connections.

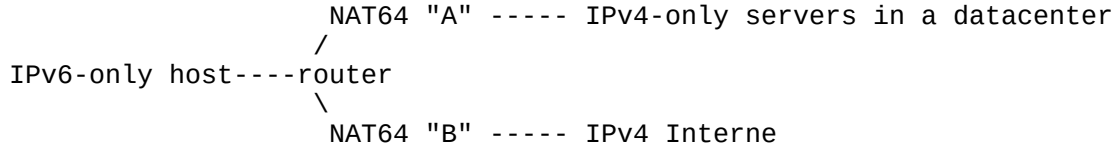


Figure 2: NAT64s with Assisting Router

6. Exit Strategy

A day will come when this tool is no longer needed. A node SHOULD implement a configuration knob for disabling the Pref64::/n discovery feature.

7. Security Considerations

The security considerations follow closely those of RFC 6147 [RFC6147]. The possible attacks are very similar in the case where an attacker controls a DNS64 server and returns tampered IPv6 addresses to a node and in the case where an attacker causes the node

to use tampered Pref64::/n for local address synthesis. DNSSEC cannot be used to validate responses created by a DNS64 server the node has no trust relationship with. Hence this document does not change the big picture for untrusted network scenarios. If an attacker alters the Pref64::/n used by a DNS64 server or a node, the traffic generated by the node will be delivered to an altered destination. This can result in either a denial-of-service (DoS) attack (if the resulting IPv6 addresses are not assigned to any device), a flooding attack (if the resulting IPv6 addresses are assigned to devices that do not wish to receive the traffic), or an eavesdropping attack (in case the altered NSP is routed through the attacker).

Even though well-known name's DNS A resource record would not necessarily need to be protected with DNSSEC, as both the name and IPv4 addresses well-known, for the DNS AAAA resource record queries DNSSEC protection is required. Without DNSSEC, fake positive AAAA responses could cause hosts to erroneously detect Pref64::/n, thus allowing attacker to inject malicious Pref64::/n for hosts' synthesis procedures. A signed ipv4only.arpa allows validating DNS64 servers (see [RFC6147] Section 3 case 5, for example) to detect malicious AAAA resource records. Therefore, the zone serving the well-known name has to be protected with DNSSEC.

For Pref64::/n discovery validation, the access network SHOULD sign the NAT64 translator's fully qualified domain name. A node SHOULD use the algorithm described in Section 3.1 to validate each discovered Pref64::/n.

The procedure of Section 3.1.2 requires a node using DNSSEC to validate discovery of Pref64::/n to have a list of trusted domains. In the case of not having matching domain at the step 3 of Section 3.1.2 an implementation might be tempted to ask for a user to add a received domain as trusted - temporarily or permanently. The history has shown that average users are unable to properly handle such queries, and tend to answer positively without thinking in an attempt to get quickly forward. Therefore, unless the DNSSEC-using implementation has a way to dynamically and reliably add trusted domains, it is better to fail the Pref64::/n discovery procedure.

Lastly, the best mitigation action against Pref64::/n discovery attacks is to add IPv6 support for nodes' destinations and hence reduce the need to perform local IPv6 address synthesis.

8. IANA Considerations

8.1. Domain Name Reservation Considerations

According to procedures described in [RFC3172] and [I-D.cheshire-dnsext-special-names] this document requests IANA to delegate a new second level domain in the .ARPA zone for the well-known domain name "ipv4only.arpa". The intention is that there will not be any further delegation of names below the ipv4only.arpa domain. The administrative and operational management of this zone is to be undertaken by IANA. The answers to seven questions of [I-D.cheshire-dnsext-special-names] are as follows:

1. No, although this is a domain delegated under the .arpa infrastructural identifier top level domain."

2. Yes. Any application attempting to perform NAT64 discovery will query the name.

3. Yes, to the extent the API or library is affected by NAT64.

4. No.

5. No.

6. This name has effects for operators of NAT64/DNS64, but otherwise is just another delegated .arpa domain.

7. The registry for .arpa is held at IANA, and only IANA needs to take action here.

8.2. IPv4 Address Allocation Considerations

The well-known name needs to map to two different global IPv4 addresses, which are to be allocated as described in [RFC5736] and [I-D.bonica-special-purpose]. The addresses are to be taken from the IANA IPv4 Special Purpose Address Registry [RFC5736], and 192.0.0.170 and 192.0.0.171 are to be assigned to this document with parameters shown below:

Attribute	Value
Address Block	192.0.0.170/32
	192.0.0.171/32
Name	NAT64/DNS64 Discovery
RFC	RFC TBD Section 2.2.
Allocation Date	February 2013
Termination Date	N/A
Source	False
Destination	False
Forwardable	False

| Global | False |
| Reserved-by-protocol | True |

The Record for IPv4 address allocation for IPv4 Special Purpose Address Registry

The following two records should be added to .arpa to the name servers run by ICANN/IANA:

ipv4only IN A 192.0.0.170

ipv4only IN A 192.0.0.171

8.3. IAB Statement Regarding This .arpa Request

With the publication of this document, the IAB approves of the delegation of "ipv4only" in the .arpa domain. Under [RFC3172], the IAB is requesting IANA to delegate and provision ipv4only.arpa as written in this specification. However, the IAB does not take any architectural or technical position about this specification.

9. Acknowledgements

Authors would like to thank Dmitry Anipko, Cameron Byrne, Aaron Yi Ding Christian Huitema, Washam Fan, Peter Koch, Stephan Lagerholm, Zhenqiang Li, Simon Perreault, Marc Petit-Huguenin, Andrew Sullivan, and Dave Thaler, for significant improvement ideas and comments.

10. References

10.1. Normative References

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, March 2005.

[RFC4443] Conta, A., Deering, S., and M. Gupta, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, March 2006.

[RFC6052] Bao, C., Huitema, C., Bagnulo, M., Boucadair, M., and X. Li, "IPv6 Addressing of IPv4/IPv6 Translators", RFC 6052, October 2010.

[RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, April 2011.

[RFC6147] Bagnulo, M., Sullivan, A., Matthews, P., and I. van Beijnum, "DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers", RFC 6147, April 2011.

[RFC6672] Rose, S. and W. Wijngaards, "DNAME Redirection in the DNS", RFC 6672, June 2012.

10.2. Informative References

[I-D.bonica-special-purpose] Cotton, M., Vegoda, L., Bonica, R., and B. Haberman, "Special-Purpose IP Address Registries", draft-bonica-special-purpose-07 (work in progress), January 2013.

[I-D.cheshire-dnsext-special-names] Cheshire, S. and M. Krochmal, "Special-Use Domain Names", draft-cheshire-dnsext-special-names-03 (work in progress), September 2012.

[RFC3172] Huston, G., "Management Guidelines & Operational Requirements for the Address and Routing Parameter Area Domain ("arpa")", BCP 52, RFC 3172, September 2001.

[RFC5735] Cotton, M. and L. Vegoda, "Special Use IPv4 Addresses", BCP 153, RFC 5735, January 2010.

[RFC5736] Huston, G., Cotton, M., and L. Vegoda, "IANA IPv4 Special Purpose Address Registry", RFC 5736, January 2010.

[RFC6144] Baker, F., Li, X., Bao, C., and K. Yin, "Framework for IPv4/IPv6 Translation", RFC 6144, April 2011.

[RFC6418] Blanchet, M. and P. Seite, "Multiple Interfaces and Provisioning Domains Problem Statement", RFC 6418, November 2011.

Appendix A. Example of DNS Record Configuration

The following BIND-style examples illustrate how A and AAAA records could be configured by a NAT64 operator.

The examples use Pref64::/n of 2001:db8::/96, both WKAs, and the example.com domain.

The PTR record for reverse queries (Section 3.1.1 bullet 3):

```
$ORIGIN A.A.0.0.0.0.0.C\
.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.IP6.ARPA.
@         IN      SOA   ns1.example.com. hostmaster.example.com. (
                        2003080800 12h 15m 3w 2h)
          IN      NS    ns.example.com.

          IN      PTR   nat64.example.com.

$ORIGIN B.A.0.0.0.0.0.C\
.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.IP6.ARPA.
@         IN      SOA   ns1.example.com. hostmaster.example.com. (
                        2003080800 12h 15m 3w 2h)
          IN      NS    ns.example.com.

          IN      PTR   nat64.example.com.
```

If example.com does not use DNSSEC, the following configuration file could be used. Please note that nat64.example.com has both an AAAA record with the Pref64::/n and an A record for the connectivity check (Section 3.1.1 bullet 2).

```
example.com.  IN SOA  ns.example.com. hostmaster.example.com. (
                      2002050501 ; serial
                      100        ; refresh (1 minute 40 seconds)
                      200        ; retry (3 minutes 20 seconds)
                      604800     ; expire (1 week)
                      100        ; minimum (1 minute 40 seconds)
                      )

example.com.  IN NS  ns.example.com.

nat64.example.com.
              IN AAAA  2001:db8:0:0:0:0:C000:00AA
              IN AAAA  2001:db8:0:0:0:0:C000:00AB
              IN A  192.0.2.1
```

To DNSSEC sign the records, the owner of the example.com zone would have RRSIG records for both the AAAA and A records for nat64.example.com. As a normal DNSSEC requirement, the zone and its parent also need to be signed.

Appendix B. About the IPv4 Address for the Well-Known Name

The IPv4 addresses for the well-known name cannot be non-global IPv4 addresses as listed in the Section 3 of [RFC5735]. Otherwise DNS64 servers might not perform AAAA record synthesis when the well-known prefix is used, as stated in Section 3.1 of [RFC6052]. However, the addresses do not have to be routable or allocated to any real node, as no communications will be initiated to these IPv4 address.

Allocation of at least two IPv4 addresses improves the heuristics in cases where the bit pattern of the primary IPv4 address appears more than once in the synthetic IPv6 address (i.e., the NSP prefix contains the same bit pattern as the IPv4 address).

If no well-known IPv4 addresses would be statically allocated for this method, the heuristic would require sending of an additional A query to learn the IPv4 addresses that would be then searched from inside of the received IPv6 address.

Authors' Addresses

Teemu Savolainen
Nokia
Hermiankatu 12 D
FI-33720 Tampere
Finland

Email: teemu.savolainen@nokia.com

Jouni Korhonen
Nokia Siemens Networks
Linnoitustie 6
FI-02600 Espoo
Finland

Email: jouni.nospam@gmail.com

Dan Wing
Cisco Systems
170 West Tasman Drive
San Jose, California 95134
USA

Email: dwing@cisco.com

OPSAWG
Internet-Draft
Intended status: Informational
Expires: August 23, 2012

V. Kuarsingh, Ed.
J. Cianfarani
Rogers Communications

February 20, 2012

CGN Deployment with MPLS/VPNs
draft-kuarsingh-lsn-deployment-06

Abstract

This document specifies a framework to integrate a Network Address Translation layer into an operator's network to function as a Carrier Grade NAT (also known as CGN or Large Scale NAT). CGN is a concept also described in [I-D.ietf-behave-lsn-requirements] and describes the model as a dual layer translation model. Although operators may wish to deploy IPv6 to strategically overcome IPv4 exhaustion, near term needs may not be satisfied with an IPv6 deployment alone. This document provides a practical integration model which allows CGN to be integrated into the network meeting the connectivity needs of the customer while being mindful of not disrupting existing services and meeting the technical challenges that CGN brings. The model includes the use of MPLS/VPNs defined in [RFC4364] as a tool to achieve this goal. This document does not intend to defend the merits of CGN.

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 http://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 August 23, 2012.

Copyright Notice

Table of Contents

1. Introduction

Operators are faced with near term IPv4 address exhaustion challenges. Many operators may not have a sufficient amount of IPv4 addresses in the future to satisfy the needs of their growing customer base. This challenge may also be present before or during an active transition to IPv6 somewhat complicating the overall problem space.

To face this challenge, operators may need to deploy CGN (Carrier Grade NAT) as described in [I-D.ietf-behave-lsn-requirements] to help extend the connectivity matrix once IPv4 addresses run out in the network. CGN's addition to the network requires integration in an often running state environment with working IPv4 and/or IPv6 services.

The addition of the CGN introduces an operator controlled and administered translation layer which needs to be added in a manner which does not overly disrupt existing services. This addition may also include interworking in a dual stack environment where the IPv4 path requires translation.

This document shows how MPLS/VPNs as described in [RFC4364] can be used to integrate the CGN infrastructure solving key problems faced by the operator. This model has also been tested and validated in real production network models and allows fluid operation with existing IPv4 and IPv6 services.

2. Motivation

The selection of CGN may be made by an operator based on a number of factors. The overall driver may be the depletion of IPv4 address pools which leaves little to no addresses for IPv4 service growth. IPv6 is considered the strategic answer, but it's applicability and usefulness in many networks is limited by the current access network and consumer home network. These environments often are filled with IPv4-Only equipment which may not be upgradable to IPv6.

The ability to replace IPv4-Only equipment may be out of the control of the operator, and even when it's in the administrative control; it poses both cost and technical challenges as operators build out massive programs for equipment retirement or upgrade. Theses issues leave an operator in a precarious position which may lead to the decision to deploy CGN. Other address IPv4 sharing options do exist which are more architecturally desirable, but the practical and workable approach in many cases is a CGN deployment using NAT444.

If the operator as has chosen to deploy CGN, they should this in a manner as not to negatively impact the existing IPv4 or IPv6 customer base. This will include solving a number of challenges since customers who's connections require translation will have network routing and flow needs which are different from legacy IPv4 connections.

The solution will also need to work in a dual stack environment where other options such as DS-Lite [RFC6333] are not yet viable. Even technologies like 6RD [RFC5969] still require an IPv4 connectivity path to service the customer endpoint. The solution will need to address basic Internet connectivity, on-net service offerings, back office management, billing, policy and security models already in place within the operator's network. CGN will often integrate quite readily with the aforementioned requirements where as other transition mechanism may not due to the requirements to support IPv6 as the base protocol for IPv4 connectivity.

3. CGN Network Deployment Requirements

If a service provider is considering a CGN deployment with a provider NAT44 function, there are a number of basic requirements which are of importance. Preliminary requirements may require the following from the incoming CGN system architecture:

- Support distributed (sparse) and centralized (dense) deployment models;

- Allow co-existence with traditional IPv4 based deployments, which provide global scoped IPs to CPEs;

- Provide a framework for CGN by-pass supporting non-translated flows between endpoints within a provider's network;

- Provide routing framework which allows the segmentation of routing control and forwarding paths between CGN and non-CGN mediated flows;

- Provide flexibility for operators to modify their deployments over time as translation demands change (connections, bandwidth, translation realms/zones and other vectors);

- Flexibility should include integration options for common access technologies such as DSL (BRAS), DOCSIS (CMTS), Mobile (GGSN/PGW/ ASN-GW), and Ethernet access;

- Support deployment modes that allow for IPv4 address overlap within the operator's network (between various translation realms or zones);

- Allow for evolution to future dual-stack and IPv4/IPv6 transition deployment modes;

- Transactional logging and export capabilities to support auxiliary functions including abuse mitigation;

- Support for stateful connection synchronization between translation instances/elements (redundancy);

- Support for CGN Shared Space [I-D.weil-shared-transition-space-request] deployment modes if applicable;

- Allows for the enablement of CGN functionality (if required) while still minimizing costs and customer impact to the best extend possible;

Other requirements may be assessed on a operator-by-operator basis, but those listed above should be considered for any given deployment architecture.

3.1. Centralized versus Distributed Deployment

Centralized deployments of CGN (longer proximity to end user and/or higher densities of subscribers/connections to CGN instances) differ from distributed deployments of CGN (closer proximity to end user and/or lower densities of subscribers/connections to CGN instances). Service providers will likely deploy CGN translation points more centrally during initial phases. Early deployments will likely see light loading on these new systems since legacy IPv4 services will continue to operate with most endpoints using globally unique IPv4 addresses. Exceptional cases which may drive heavy usage in initial stages may include operators who already translate most IPv4 traffic and will migrate to a CGN implementation from legacy firewalls; or a green field deployment which may see quick growth in the number of new IPv4 endpoints which require Internet connectivity.

Over time, most providers will likely need to expand and possibly distribute the translation points as demand for the CGN system increases. The extent of the expansion of the CGN infrastructure will depend on factors such as growth in the number of IPv4 endpoints, status of IPv6 content on the Internet and the overall progress globally to an IPv6-dominate Internet (reducing the demand for IPv4 connectivity).

3.2. CGN and Traditional IPv4 Service Co-existence

Newer CGN serviced endpoints will exist alongside endpoints served by traditional IPv4 global IPs. Providers will need to rationalize these environments since both have distinct forwarding needs. Traditional IPv4 services will likely require (or be best served) direct forwarding towards Internet peering points while CGN mediated flows require access to a translator. CGN and non-CGN mediated flows post two fundamentally different forwarding needs.

The new CGN environments should not negatively impact the existing IPv4 service base by forcing all traffic to translation enabled network points since many flows do not require translation and this would reduce performance of the existing flows. This would also require massive scaling of the CGN which is a cost and efficiency concern as well.

Traffic flow and forwarding efficiency is considered important since networks are under considerable demand to deliver more and more bandwidth without the luxury of needless inefficiencies which can be introduced with CGN.

3.3. CGN By-Pass

The CGN environment is only needed for flows with translation requirements. Many flows which remain in a service provider environment, do not require translation. Such services include operator offered DNS Services, DHCP Services, NTP Services, Web Caching, Mail, News and other services which are local to the operator’s network.

The operator may want to leverage opportunities to offer third parties a platform to also provide services without translation. CGN By-pass can be accomplished in many ways, but a simplistic, deterministic and scalable model is preferred.

3.4. Routing Plane Separation

Many operators will want to engineer traffic separately for CGN flows versus flows which are part of the more traditional IPv4 environment. Many times the routing of these two major flow types differ, therefore route separation may be required.

Routing plane separation also allows the operator to utilize other addressing techniques, which may not be feasible on a single routing plane. Such examples include the use of overlapping private address space [RFC1918] or use of other IPv4 space which may overlap globally within the operator’s network.

3.5. Flexible Deployment Options

Service providers operate complex routing environments and offer a variety of IPv4 based services. Many operator environments utilize distributed peering infrastructures for transit and peering and these may span large geographical areas and regions. A CGN solution should offer the operator an ability to place CGN translation points at various points within their network.

The CGN deployment should also be flexible enough to change over time as demand for translation services increase. In turn, the deployment will need to then adapt as translation demand decreases caused by the transition of flows to IPv6. Translation points should be able to be placed and moved with as little re-engineering effort as possible minimizing the risks to the customer base.

Depending on hardware capabilities, security practices and IPv4 address availability, the translation environments my need to be segmented and/or scaled over time to meet organic IPv4 demand growth. Operators will want to seek deployment models which are conducive to meeting these goals as well.

3.6. IPv4 Overlap Space

IP address overlap for CGN translation realms may be required if insufficient IPv4 addresses are available within the service provider environment to assign internally unique IPs to the CGN customer base . The CGN deployment should provide mechanisms to manage IPv4 overlap if required.

3.7. Transactional Logging for LSN Systems

CGNs may require transactional logging since the source IP and related transport protocol information is not easily visible to external hosts and system.

If needed, the CGN systems should be able to generate logs which identify 'internal' host parameters (i.e. IP/Port) and associated them to external translated parameters imposed by the translator. The logged information should be stored on the CGN hardware and/or exported to an external system for processing. Operators may need to keep track of this information (securely) to meet regulatory and/or legal obligations. Further information can be found in [I-D.ietf-behave-lsn-requirements] with respect to CGN logging requirements (Logging Section).

3.8. Additional CGN Requirements

The CGN platform will also need to meet the needs of additional requirements such as Bulk Port Allocation and other CGN device specific functions. These additional requirements are captured within [I-D.ietf-behave-lsn-requirements].

4. MPLS/VPN based CGN Framework

The MPLS/VPN [RFC4364] framework for CGN segregates the 'pre-translated' realms within the service provider space into layer-3 MPLS/VPNs. The operator can deploy a single realm for all CGN based flows, or can deploy multiple realms based on translation demand and other factors such as geographical proximity. A realm in this model refers to a 'VPN' which shares a unique RD/RT combination, routing plane and forwarding behaviours.

The MPLS/VPN infrastructure provides control plane and forwarding separation for the traditional IPv4 service environment and CGN environment(s). The separation allows for routing information (such as default routes) to be propagated separately for CGN and non-CGN based customer flows. Traffic can be efficiently routed to the Internet for normal flows, and routed directly to translators for CGN mediated flows. Although many operators may run a "default-route-free" core, IPv4 flows which require translation must obviously be routed first to a translator, so a default route is acceptable for the pre-translated realms.

The physical location of the VRF Termination point for a MPLS/VPN enabled CGN can vary and be located anywhere within the operator's network. This model fully virtualizes the translation service from the base IPv4 forwarding environment which will likely carrying Internet bound traffic. The base IPv4 environment can continue to service traditional IPv4 customer flows plus post translated CGN flows.

Figure 1 provides a view of the basic model. The Access node provides CPE access to either the CGN VRF or the Global Routing Table, depending on whether the customer receives a private or public IP. Translator mediated traffic follows an MPLS LSP which can be setup dynamically and can span one hop, or many hops (with no need for complex routing policies). Traffic is then forwarded to the translator (shown below) which can be an external appliance or integrated into the VRF Termination (Provider Edge) router. Once traffic is translated, it is forwarded to the global routing table for general Internet forwarding. The Global Routing table can also be a separate VRF (Internet Access VPN/VRF) should the provider

choose to implement their Internet based services in that fashion. The translation services are effectively overlaid onto the network, but are maintained within a separate forwarding and control plane.

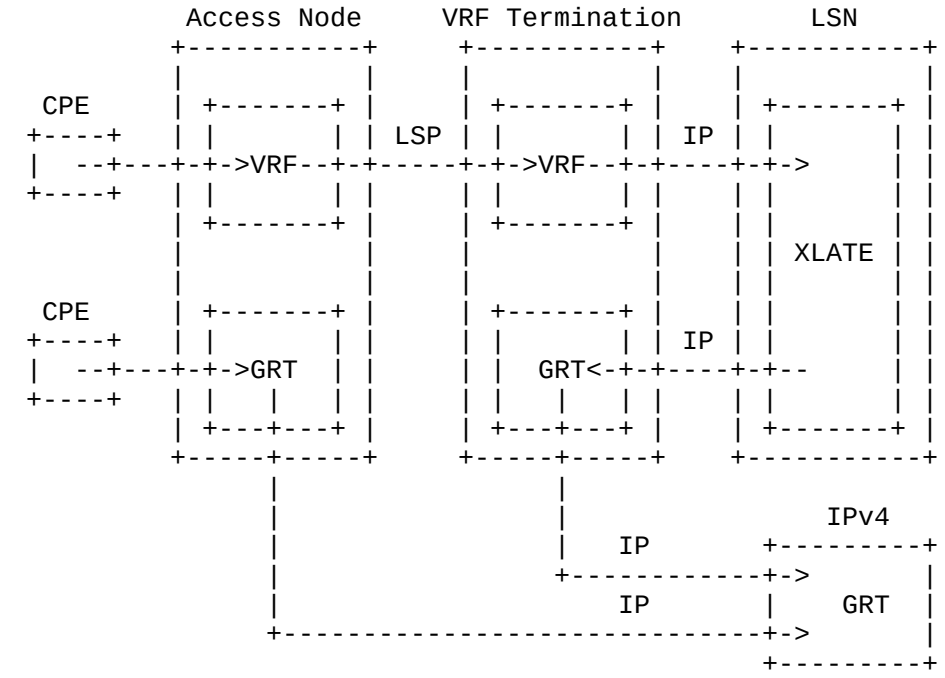


Figure 1: Basic MPLS/VPN CGN Model

If more then one VRF (translation realm) is used within the operator's network, each VPN instance can manage CGN flows independently for the respective realm. Various redundancy models can be used within this architecture to support failover from one physical CGN hardware instance to another. If state information needs to be passed or maintained between hardware instances, the vendor would need to enable this feature in a suitable manner.

4.1. Service Separation

The MPLS/VPN CGN framework supports route separation. The traditional IPv4 flows can be separated at the access node (Initial Layer 3 service point) from those which require translation. This type of service separation is possible on common technologies used for Internet access within many operator networks. Service separation can be accomplished on common access technology including

those used for DOCSIS (CMTS), Ethernet Access, DSL (BRAS), and Mobile Access (GGSN/ASN-GW) architectures.

4.2. Internal Service Delivery

Internal services can be delivered directly to the privately addressed endpoint within the CGN domain without translation. This can be accomplished using direct route exchange (import/export) between the CGN VRFs and the Services VRFs. The previous statement assumes the provider puts key services into a VRF for simple route exchange. This model allows the provider to maintain separate forwarding rules for translated flows, which require a pass through the translator to reach external network entities, versus those flows which need to access internal services. This operational detail can be advantageous for a number of reasons.

First, the provider can reduce the load on the translator since internal services do not need to be factored into the scaling of the CGN hardware. Secondly, more direct forwarding paths can be maintained providing better network efficiency. Thirdly, geographic locations of the translators and the services infrastructure can be deployed in a location in an independent manner. Additionally, the operator can allow CGN subject endpoints to be accessible via an untranslated path reducing the complexities of provider initiated management flows. This last point is of key interest since NAT removes transparency to the end device in normal cases.

Figure 2 below shows how internal services are provided untranslated since flows are sent directly from the access node to the services node/VRF via an MPLS LSP. This traffic is not forwarded to the CGN translator and therefore is not subject to problematic behaviours related to NAT. The services VRF contains routing information which can be "imported" into the access node VRF and the CGN VRF routing information can be "imported" into the Services VRF.

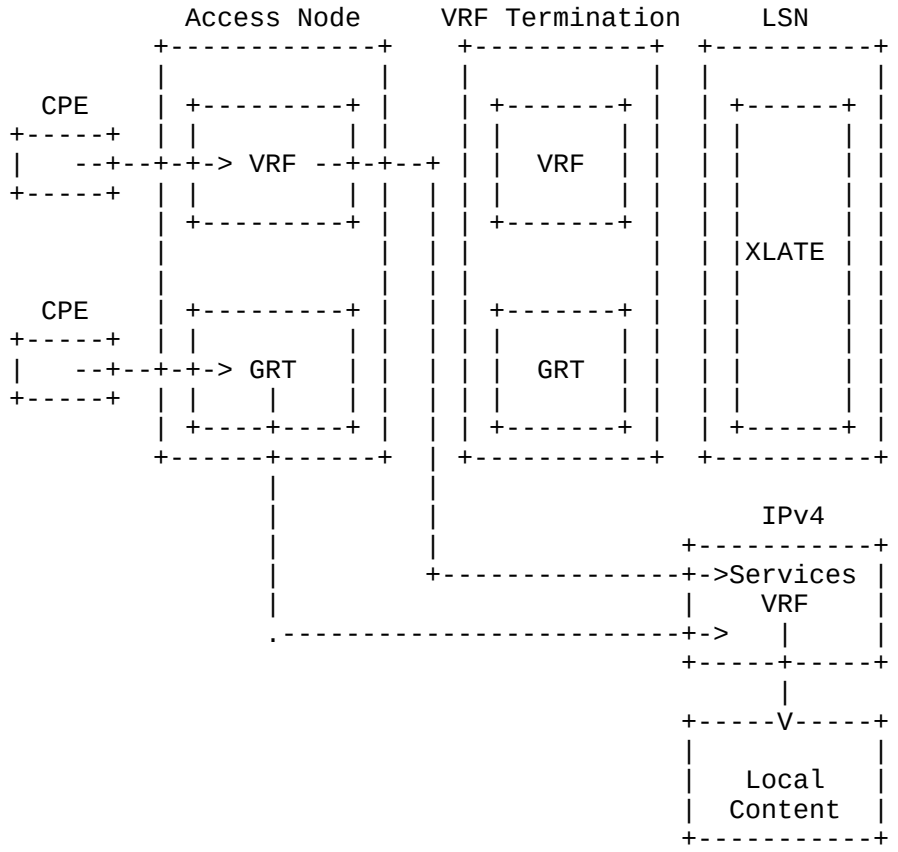


Figure 2: Internal Services and CGN By-Pass

This demonstrates the ability to offer CGN By-Pass in a simple and deterministic manner without the need of policy based routing or traffic engineering.

4.2.1. Dual Stack Operation

The MPLS/VPN CGN model can also be used in conjunction with IPv4/IPv6 dual stack service modes. Since many providers will use CGNs on an interim basis while IPv6 matures within the global Internet or due to technical constraints, a dual stack option is of strategic importance. Operators can offer this dual stack service for both traditional IPv4 (global IP) endpoints and CGN mediated endpoints.

Operators can separate the IP flows for IPv4 and IPv6 traffic, or use other routing techniques to move IPv6 based flows towards the GRT

(Global Routing Table or Instance) while allowing IPv4 flows to remain within the IPv4 CGN VRF for translator services.

The Figure 3 below shows how IPv4 translation services can be provided alongside IPv6 based services. The model shown allows the provider to enable CGN to manage IPv4 flows (translated) and IPv6 flows are routed without translation efficiently towards the Internet. Once again, forwarding of flows to the translator does not impact IPv6 flows which do not require this service.

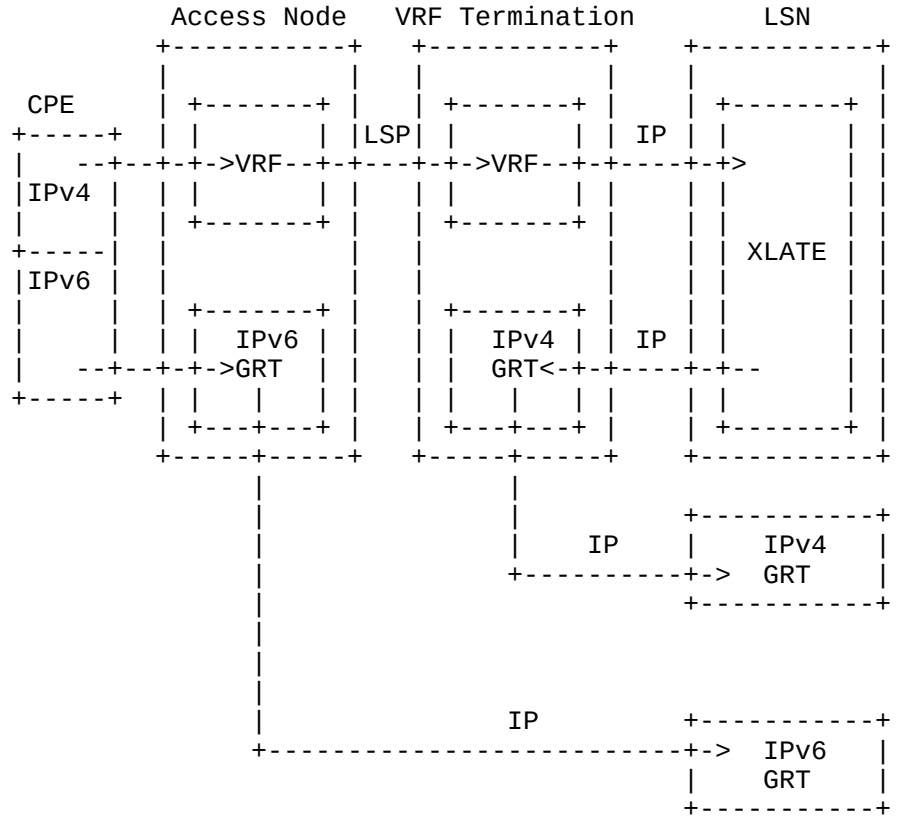


Figure 3: CGN with IPv6 Dual Stack Operation

4.3. Deployment Flexibility

The CGN translator services can be moved, separated or segmented (new translation realms) without the need to change the overall translation design. Since dynamic LSPs are used to forward traffic

from the access nodes to the translation points, the physical location of the VRF termination points can vary and be changed easily.

This type of flexibility allows the service provider to initially deploy more centralized translation services based on relatively low loading factors, and distribute the translation points over time to improve network traffic efficiencies and support higher translation load.

Although traffic engineered paths are not required within the MPLS/VPN deployment model, nothing precludes an operator from using technologies like MPLS with Traffic Engineering [RFC3031]. Additional routing mechanisms can be used as desired by the provider and can be seen as independent. There is no specific need to diversify the existing infrastructure in most cases.

4.4. Comparison of MPLS/VPN Option versus other CGN Attachment Options

Other integration architecture options exist which can attach CGN based service flows to a translator instance. Alternate options which can be used to attach such services include:

- IEEE 802.1Q for direct attachment to a next hop translator;

- Policy Based Routing (Static) to direct translation bound traffic to a network based translator;

- Traffic Engineering or;

- Multiple Routing Topologies

4.4.1. IEEE 802.1Q

IEEE 802.1Q can be used to associate separated traffic from the access node to the next hop router’s CGN instance. This technology option may limit the CGN placement to the next hop router unless a second technology option is paired with it to extend connectivity deeper in the network.

This option is most effective if CGN instances are placed directly upstream of the access node. Distributed CGN instance placement is not likely an initial stage of the CGN deployment due to cost and demand factors.

4.4.2. Policy Based Routing

Policy Based Routing (PBR) provides another option to direct CGN mediated flows to a translator. PBR options, although possible, are difficult to maintain (static policy) and must be configured throughout the network with considerable maintenance overhead.

More centralized deployments may be difficult or too onerous to deploy using Policy Based Routing methods. Policy Based Routing would not achieve route separation (unless used with others options), and may add complexities to the providers' routing environment.

4.4.3. Traffic Engineering

Traffic Engineering can also be used to direct traffic from an access node towards a translator. Traffic Engineering, like MPLS-TE, may be difficult to setup and maintain. Traffic Engineering provides additional benefits if used with MPLS by adding potentials for faster path re-convergence. Traffic Engineering paths would need to be updated and redefined overtime as CGN translation points are augmented or moved.

4.4.4. Multiple Routing Topologies

Multiple routing topologies can be used to direct CGN based flows to translators. This option would achieve the same basic goal as the MPLS/VPN option but with additional implementation overhead and platform configuration complexity. Since operator based translation is expected to have an unknown lifecycle, and may see various degrees of demand (dependant on operator IPv4 Global space availability and shift of traffic to IPv6), it may be too large of an undertaking for the provider to enabled this as their primary option for CGN.

5. Experiences

6. Basic Integration and Requirements Support

The MPLS/VPN CGN environment has been successfully integrated into real network environments utilizing existing network service delivery mechanisms. It solves many issues related to provider based translation environments, while still subject to problematic behaviours inherent within NAT.

Key issues which are solved or managed with the MPLS/VPN option include:

- Centralized and Distributed Deployment model support

- Routing Plane Separation for CGN flows versus traditional IPv4 flows

- Flexible Translation Point Design (can relocate translators and split translation zones easily)

- Low maintenance overhead (dynamic routing environment with little maintenance of separate routing infrastructure other then management of MPLS/VPNs)

- CGN By-pass options (for internal and third party services which exist within the provider domain)

- IPv4 Translation Realm overlap support (can reuse IP addresses between zones with some impact to extranet service model)

- Simple failover techniques can be implemented with redundant translators, such as using a second default route

7. Performance

The MPLS/VPN CGN model was observed to support basic functions which are typically used by customers within an operator environment. Examples of successful operation include:

- Traditional Web (HTTP) Surfing (client initiated)

- Internet Video Streaming

- HTTP Based Client Connections

- High Connection Count sites (i.e. Google Maps)

- Email Transaction Support (POP, IMAP, SMTP)

- Instant Messaging Support (Online Status, File transfers, text chat)

- ICMP Operation (client initiated Echo, Traceroute)

- Peer to Peer application support (download)

- DNS (based on services extranet option, but was problematic when passed through a translator)

CGNs are still subject to problematic connectivity even within the MPLS/VPN technology approach. Problems which arise, or are not inherently addressed in this model include:

- Inward services from the Internet to the CPE
- Web session tracking
- Restricting usage and/or access based on source IP
- Abuse mitigation (masquerade of potential offenders)
- Increased network or server IDS false positives
- Increased customer risk for session hijacking
- Exceeding firewall TCP/UDP limits
- Customer identification (external site)
- Poor source based load balancing
- Customer usage tracking / Ad insertion
- Other applications or operations may be negatively impacted

8. IANA Considerations

There are not specific IANA considerations known at this time with the architecture described herein. Should a provide choose to use non-assigned IP address space within their translation realms, then considerations may apply.

9. Security Considerations

The same security considerations would typically exist for CGN deployments when compared with traditional IPv4 based services. With the MPLS/VPN model, the operator would want to consider security issues related to offering IP services over MPLS.

If a provider plans to operate the pre-translation realm (CPE towards translator IPv4 zone) as a non-public like network, then additional security measures may be needed to secure this environment. It is however the position in this document that CGN realms are public domains which utilize non-Internet routable IP addresses for endpoint addressing.

10. Conclusions

The MPLS/VPN delivery method for a CGN deployment is an effective and scalable way to deliver mass translation services. The architecture avoids the complex requirements of traffic engineering and policy based routing when combining these new service flows to existing IPv4 operation. This is advantageous since the NAT44/CGN environments should be introduced with as little impact as possible and these environments are expected to change over time.

The MPLS/VPN based CGN architecture solves many of this issues related to deploying this technology in existing operator networks.

11. Acknowledgements

Thanks to the following people for their participating in integrating and testing the CGN environment: Chris Metz, Syd Alam, Richard Lawson, John E Spence.

Additional thanks for the following people for the guidance on IPv6 transition considerations: John Jason Brzozowski, Chris Donley, Jason Weil, Lee Howard, Jean-Francois Tremblay

12. References

12.1. Normative References

[I-D.ietf-behave-lsn-requirements]
Perreault, S., Yamagata, I., Miyakawa, S., Nakagawa, A., and H. Ashida, "Common requirements for Carrier Grade NATs (CGNs)", draft-ietf-behave-lsn-requirements-05 (work in progress), November 2011.

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, February 2006.

12.2. Informative References

[I-D.weil-shared-transition-space-request]
Weil, J., Kuarsingh, V., Donley, C., Liljenstolpe, C., and M. Azinger, "IANA Reserved IPv4 Prefix for Shared Address Space", draft-weil-shared-transition-space-request-15 (work in progress), February 2012.

[RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and E. Lear, "Address Allocation for Private Internets",

BCP 5, RFC 1918, February 1996.

[RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, January 2001.

[RFC5969] Townsley, W. and O. Troan, "IPv6 Rapid Deployment on IPv4 Infrastructures (6rd) -- Protocol Specification", RFC 5969, August 2010.

[RFC6333] Durand, A., Droms, R., Woodyatt, J., and Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion", RFC 6333, August 2011.

Authors' Addresses

Victor Kuarsingh (editor)
Rogers Communications
8200 Dixie Road
Brampton, Ontario L6T 0C1
Canada

Email: victor.kuarsingh@gmail.com
URI: http://www.rogers.com

John Cianfarani
Rogers Communications
8200 Dixie Road
Brampton, Ontario L6T 0C1
Canada

Email: john.cianfarani@rci.rogers.com
URI: http://www.rogers.com

Port Control Protocol
Internet-Draft
Intended status: BCP
Expires: July 11, 2013

R. Penno
Cisco
S. Perreault
Viagenie
S. Kamiset
Consultant
M. Boucadair
France Telecom
K. Naito
NTT

January 07, 2013

Network Address Translation (NAT) Behavioral Requirements Updates draft-penno-behave-rfc4787-5382-5508-bis-04

Abstract

This document clarifies and updates several requirements of RFC4787, RFC5382 and RFC5508 based on operational and development experience. The focus of this document is NAPT44.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

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 http://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 July 11, 2013.

Copyright Notice

Table of Contents

1. Terminology

The reader should be familiar with all terms defined in RFC2663 [RFC2663],RFC4787 [RFC4787],RFC5382 [RFC5382],RFC5508 [RFC5508]

2. Introduction

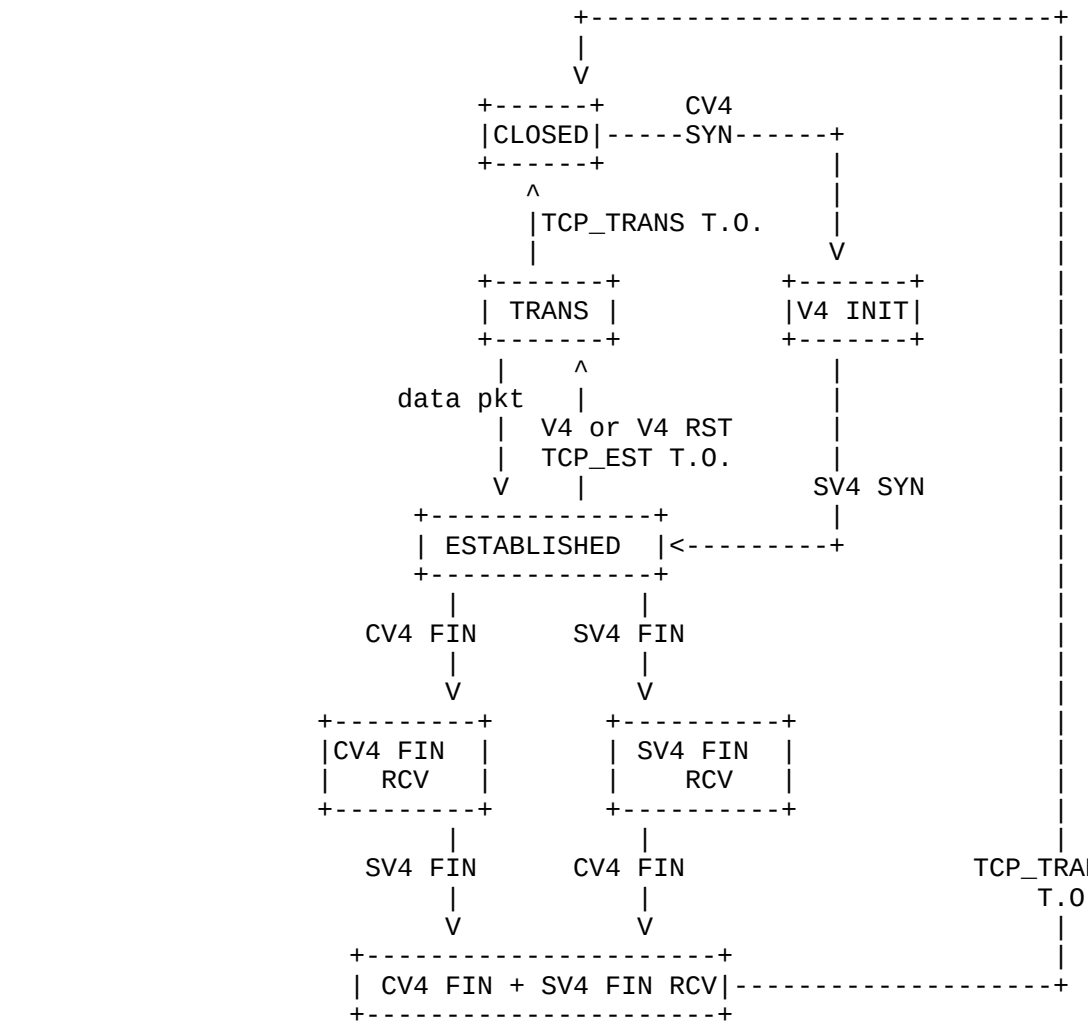
[RFC4787], [RFC5382] and [RFC5508] greatly advanced NAT interoperability and conformance. But with widespread deployment and evolution of NAT more development and operational experience was acquired some areas of the original documents need further clarification or updates. This documents provides such clarifications and updates.

2.1. Scope

This document focuses solely on NAPT44 and its goal is to clarify, fill gaps or update requirements of [RFC4787], [RFC5382] and [RFC5508]. It is out of the scope of this document the creation of completely new requirements not associated with the documents cited above. New requirements would be better served elsewhere and if they are CGN specific in [I-D.ietf-behave-lsn-requirements]

3. TCP Session Tracking

[RFC5382] specifies TCP timers associated with various connection states but does not specify the TCP state machine a NAPT44 should use as a basis to apply such timers. The TCP state machine below, adapted from [RFC6146], provides guidance on how TCP session tracking could be implemented - it is non-normative.



(postamble)

3.1. TCP Transitory Connection Idle-Timeout

[RFC5382]:REQ-5 The transitory connection idle-timeout is defined as the minimum time a TCP connection in the partially open or closing phases must remain idle before the NAT considers the associated session a candidate for removal. But the document does not clearly states if these can be configured separately. This document clarifies that a NAT device SHOULD provide different knobs for configuring the open and closing idle timeouts. This document further acknowledges that most TCP flows are very short (less than 10 seconds) [FLOWRATE][TCPWILD] and therefore a partially open timeout

of 4 minutes might be excessive if security is a concern. Therefore it MAY be configured to be less than 4 minutes in such cases. There also may be a case that timeout of 4 minutes might be excessive. The case and the solution are written below.

3.1.1. Port resources limited case

After IPv4 addresses run out, IPv4 address resources will be further restricted site-by-site. If global IPv4 address are shared between several clients, assignable port resources at each client will be limited.

NAT is a tool that is widely used to deal with this IPv4 address shortage problem. However, the demand for resources to provide Internet access to users and devices will continue to increase. IPv6 is a fundamental solution to this problem, but the deployment of IPv6 will take time.

In some cases, e.g. browsing a dynamic web page for a map service, a lot of sessions are used by the browser, and a number of ports are eaten up in a short time. What is worse is that when a NAT is between a PC and a server, TIME_WAIT state of each TCP connection is kept for certain period, typically for four minutes, which consumes port resources. Therefore, new connections cannot be established.

This problem is caused or worsened by the following behavior.

> TIME_WAIT state assigned for a TCP connection remains active for 2MSL after the last ACK to the last FIN is transferred.

To reuse resources effectively, reducing TIME_WAIT without making any bad effect is important. To reduce TIME_WAIT, [RFC6191] is proposed for clients and remote hosts. To prevent bad effects, there is a PAWS mechanism, which prevent the old duplicate problem. We propose mechanisms adopting to NAT, to change the TIME_WAIT behavior that make it possible to save addresses and ports resources.

3.1.1.1. RFC6191 Reducing the TIME-WAIT State Using TCP Timestamps

[RFC6191] defines a mechanism for reducing the TIME_WAIT state using TCP timestamps and sequence numbers. When a connection request is received with a four-tuple that is in the TIME-WAIT state, the connection request may be accepted if the sequence number or the timestamp of the incoming SYN segment is greater than the last sequence number seen on the previous incarnation of the connection

3.1.1.2. TCP TIME_WAIT

The TCP TIME_WAIT state is described in [RFC0793]. The TCP TIME_WAIT state needs to be kept for 2MSL before a connection is CLOSED, for the reasons below.

1: In the event that packets from a session are delayed in the in-between network, and delivered to the end relatively later, we should prevent the packets from being transferred and interpreted as a packet that belongs to a new session.

2: If the remote TCP has not received the acknowledgment of its connection termination request, it will re-send the FIN packet several times.

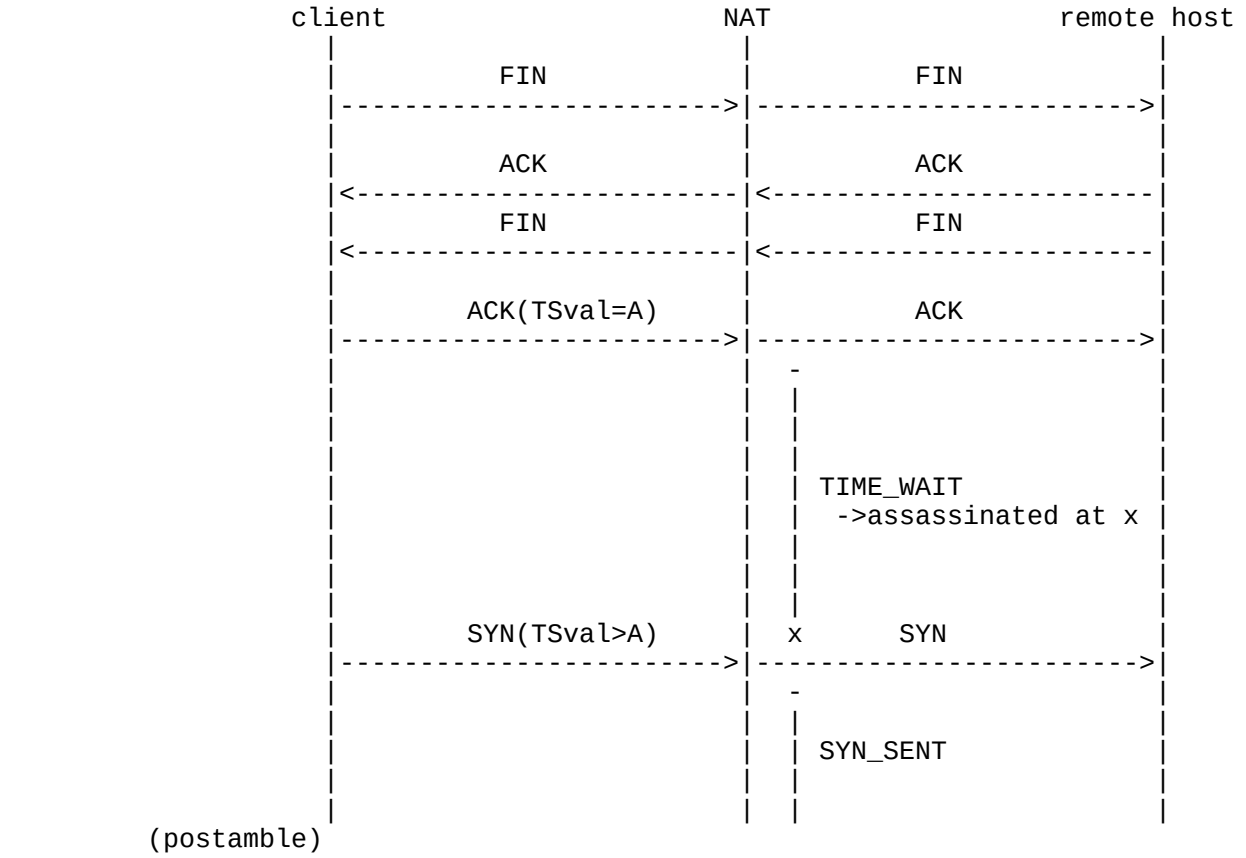
These points are important for the TCP to work without problems.

3.1.1.3. Protect Against Wrapped Sequence numbers (PAWS)

The TCP sequence number wraps frequently especially in a high bandwidth session. PAWS is used to prevent old duplicate packets that occurred in a previous session from being transferred to the new session whose valid TCP sequence numbers happen to overlap with the old duplicate packets. This is implemented by introducing TCP timestamp option, and checking the timestamp option value of each packet. PAWS is described in [RFC1323].

3.1.2. Proposal: Apply RFC6191 and PAWS to NAT

This section proposes to apply [RFC6191] mechanism at NAT. This mechanism MAY be adopted for both clients' and remote hosts' TCP active close.



(postamble)

Also, PAWS works to discard old duplicate packets at NAT. A packet can be discarded as an old duplicate if it is received with a timestamp or sequence number value less than a value recently received on the connection.

To make these mechanisms work, we should concern the case that there are several clients with nonsuccessive timestamp or sequence number values are connected to a NAT device (i.e. not monotonically increasing among clients). Two mechanisms to solve this mechanism and applying [RFC6191] and PAWS to NAT are described below. These mechanisms are optional.

3.1.2.1. Rewrite timestamp and sequence number values at NAT

Rewrite timestamp and sequence number values of outgoings packets at NAT to be monotonically increasing. This can be done by adopting following mechanisms at NAT.

A: Store the newest rewritten value of timestamp and sequence number as the "max value at the time".

B: NAT rewrite timestamp and sequence number values of incoming packets to be monotonically increasing.

When packets come back as replies from remote hosts, NAT rewrite again the timestamp and sequence number values to be the original values. This can be done by adopting following mechanisms at NAT.

C: Store the values of original timestamp and sequence number of packets, and rewritten values of those.

3.1.2.2. Split an assignable number of port space to each client

Adopt following mechanisms at NAT.

A: Choose clients that can be assigned ports.

B: Split assignable port numbers between clients.

Packets from other clients which are not chosen by these mechanisms are rejected at NAT, unless there is unassigned port left.

3.1.2.3. Resend the last ACK to the resended FIN

We should concern another case to make RFC6191 work at NAT. In case the remote TCP could not receive the acknowledgment of its connection termination request, NAT, on behalf of clients. resends the last ACK packet when it receives an FIN packet of the previous connection, and when the state of the previous connection is deleted from the NAT. This mechanism MAY be used when clients starts closing process, and the remote host could not receive the last ACK.

3.1.2.4. Remote host behavior of several implementations

To solve the port shortage problem on the client side, the behavior of remote host should be compliant to [RFC6191] or the mechanism written in 4.2.2.13 of [RFC1122], since NAT may reuse the same 5 tuple for a new connection.We have investigated behaviors of OSes (e.g., Linux, FreeBSD, Windows, MacOS), and found that they implemented the server side behavior of the above two.

3.2. TCP RST

[RFC5382] leaves the handling of TCP RST packets unspecified. This document does not try standardize such behavior but clarifies based on operational experience that a NAT that receives a TCP RST for an active mapping and performs session tracking MAY immediately delete the sessions and remove any state associated with it. If the NAT device that performs TCP session tracking receives a TCP RST for the first session that created a mapping, it MAY remove the session and the mapping immediately.

4. Port Overlapping behavior

There may be another solution to the address resource restricted environment written in 3.1.1. Also NAT are required to be maped endpoint-independent in [RFC4787] and [RFC5382] REQ-1, the mechanism below MAY be one optional implement to NAT.

If destination addresses and ports are different for outgoing connections started by local clients, NAT MAY assign the same external port as the source ports for the connections. The port overlapping mechanism manages mappings between external packets and internal packets by looking at and storing the 5-tuple (protocol, source address, source port, destination address, destination port) of them. This enables concurrent use of a single NAT external port for multiple transport sessions, which enables NAT to work correctly in IP address resource limited network.

Discussions:

[RFC4787]and[FC5382] requires "endpoint-independent mapping" at NAT, and port overlapping NAT cannot meet the requirement. This mechanism can degrade the transparency of NAT in that its mapping mechanism is endpoint-dependent and makes NAT traversal harder. However, if a NAT adopts endpoint-independent mapping together with endpoint-dependent filtering, then the actual behavior of the NAT will be the same as port overlapping NAT. It should also be noted that a lot of existing NAT devices(e.g., SEIL, FITELnet Series) adopted this port overlapping mechanism.

A: Reference URL for SEIL -> www.seil.jp

B: Reference URL for FITELnet -> www.furukawa.co.jp/fitelnet

The netfilter, which is a popular packet filtering mechanism for

Linux, also adopts port overlapping behavior.

5. Address Pooling Paired (APP)

[RFC4787]: REQ-2 [RFC5382]:ND Address Pooling Paired behavior for NAT is recommended in previous documents but behavior when a public IPv4 run out of ports is left undefined. This document clarifies that if APP is enabled new sessions from a subscriber that already has a mapping associated with a public IP that ran out of ports SHOULD be dropped. The administrator MAY provide a knob that allows a NAT device to starting using ports from another public IP when the one that anchored the APP mapping ran out of ports. This is trade-off between subscriber service continuity and APP strict enforcement. (NE: It is sometimes referred as 'soft-APP')

6. EIF Security

[RFC4787]:REQ-8 and [RFC5382]:REQ-3 End-point independent filtering could potentially result in security attacks from the public realm. In order to handle this, when possible there MUST be strict filtering checks in the inbound direction. A knob SHOULD be provided to limit the number of inbound sessions and a knob SHOULD be provided to enable or disable EIF on a per application basis. This is specially important in the case of Mobile networks where such attacks can consume radio resources and count against the user quota.

7. EIF Protocol Independence

[RFC4787]:REQ-8 and[RFC5382]: REQ-3 Current RFCs do not specify whether EIF mappings are protocol independent. In other words, if a outbound TCP SYN creates a mapping it is left undefined whether inbound UDP packets create sessions and are forwarded. EIF mappings SHOULD be protocol independent in order allow inbound packets for protocols that multiplex TCP and UDP over the same IP: port through the NAT and maintain compatibility with stateful NAT64 RFC6146 [RFC6146]. But the administrator MAY provide a configuration knob to make it protocol dependent.

8. EIF Mapping Refresh

[RFC4787]: REQ-6 [RFC5382]: ND The NAT mapping Refresh direction MAY have a "NAT Inbound refresh behavior" of "True" but it does not clarifies how this applies to EIF mappings. The issue in question is whether inbound packets that match an EIF mapping but do not create a

new session due to a security policy should refresh the mapping timer. This document clarifies that even when a NAT device has a inbound refresh behavior of TRUE, that such packets SHOULD NOT refresh the mapping. Otherwise a simple attack of a packet every 2 minutes can keep the mapping indefinitely.

8.1. Outbound Mapping Refresh and Error Packets

In the case of NAT outbound refresh behavior there might be certain types of packets that should not refresh the mapping. For example, if the mapping is kept alive by ICMP Error or TCP RST outbound packets sent as response to inbound packets, these SHOULD NOT refresh the mapping.

9. EIM Protocol Independence

[RFC4787] [RFC5382]: REQ-1 Current RFCs do not specify whether EIM are protocol independent. In other words, if a outbound TCP SYN creates a mapping it is left undefined whether outbound UDP can reuse such mapping and create session. On the other hand, Stateful NAT64 [RFC6146] clearly specifies three binding information bases (TCP, UDP, ICMP). This document clarifies that EIM mappings SHOULD be protocol dependent . A knob MAY be provided in order allow protocols that multiplex TCP and UDP over the same source IP and port to use a single mapping.

10. Port Parity

A NAT devices MAY disable port parity preservation for dynamic mappings. Nevertheless, A NAT SHOULD support means to explicitly request to preserve port parity (e.g., [I-D.boucadair-pcp-rtp-rtcp]).

11. Port Randomization

A NAT SHOULD follow the recommendations specified in Section 4 of [RFC6056] especially: "A NAPT that does not implement port preservation [RFC4787] [RFC5382] SHOULD obfuscate selection of the ephemeral port of a packet when it is changed during translation of that packet. A NAPT that does implement port preservation SHOULD obfuscate the ephemeral port of a packet only if the port must be changed as a result of the port being already in use for some other session. A NAPT that performs parity preservation and that must change the ephemeral port during translation of a packet SHOULD obfuscate the ephemeral ports. The algorithms described in this document could be easily adapted such that the parity is preserved

(i.e., force the lowest order bit of the resulting port number to 0 or 1 according to whether even or odd parity is desired)."

12. IP Identification (IP ID)

A NAT SHOULD handle the Identification field of translated IPv4 packets as specified in Section 9 of [I-D.ietf-intarea-ipv4-id-update].

13. ICMP Query Mappings Timeout

Section 3.1 of [RFC5508] says that ICMP Query Mappings are to be maintained by NAT device. However, RFC doesn't discuss about the Query Mapping timeout values. Section 3.2 of that RFC only discusses about ICMP Query Session Timeouts. ICMP Query Mappings MAY be deleted once the last the session using the mapping is deleted.

14. Hairpinning Support for ICMP Packets

[RFC5508]:REQ-7 This requirement specifies that NAT devices enforcing Basic NAT MUST support traversal of hairpinned ICMP Query sessions. This implicitly means that address mappings from external address to internal address (similar to Endpoint Independent Filters) MUST be maintained to allow inbound ICMP Query sessions. If an ICMP Query is received on an external address, NAT device can then translate to an internal IP. [RFC5508]:REQ-7 This requirement specifies that all NAT devices (i.e., Basic NAT as well as NAPT devices) MUST support the traversal of hairpinned ICMP Error messages. This too requires NAT devices to maintain address mappings from external IP address to internal IP address in addition to the ICMP Query Mappings described in section 3.1 of that RFC.

15. IANA Considerations

TBD

16. Security Considerations

In the case of EIF mappings due to high risk of resource crunch, a NAT device MAY provide a knob to limit the number of inbound sessions spawned from a EIF mapping.

[TCP-Security] contains a detailed discussion of the security

implications of TCP Timestamps and of different timestamp generation algorithms.

17. Acknowledgements

Thanks to Dan Wing, Suresh Kumar, Mayuresh Bakshi, Rajesh Mohan and Senthil Sivamular for review and discussions

18. References

18.1. Normative References

[I-D.ietf-pcp-base] Wing, D., Cheshire, S., Boucadair, M., Penno, R., and P. Selkirk, "Port Control Protocol (PCP)", draft-ietf-pcp-base-29 (work in progress), November 2012.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[RFC1122] Braden, R., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, October 1989.

[RFC1323] Jacobson, V., Braden, B., and D. Borman, "TCP Extensions for High Performance", RFC 1323, May 1992.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

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

[RFC3605] Huitema, C., "Real Time Control Protocol (RTCP) attribute in Session Description Protocol (SDP)", RFC 3605, October 2003.

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

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

[RFC5508] Srisuresh, P., Ford, B., Sivakumar, S., and S. Guha, "NAT

Behavioral Requirements for ICMP", BCP 148, RFC 5508, April 2009.

[RFC6056] Larsen, M. and F. Gont, "Recommendations for Transport-Protocol Port Randomization", BCP 156, RFC 6056, January 2011.

[RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, April 2011.

[RFC6191] Gont, F., "Reducing the TIME-WAIT State Using TCP Timestamps", BCP 159, RFC 6191, April 2011.

18.2. Informative References

[FLOWRATE]
Zhang, Y., Breslau, L., Paxson, V., and S. Shenker, "On the Characteristics and Origins of Internet Flow Rates".

[I-D.boucadair-pcp-rtp-rtcp]
Boucadair, M. and S. Sivakumar, "Reserving N and N+1 Ports with PCP", draft-boucadair-pcp-rtp-rtcp-05 (work in progress), October 2012.

[I-D.ietf-behave-lsn-requirements]
Perreault, S., Yamagata, I., Miyakawa, S., Nakagawa, A., and H. Ashida, "Common requirements for Carrier Grade NATs (CGNs)", draft-ietf-behave-lsn-requirements-10 (work in progress), December 2012.

[I-D.naito-nat-resource-optimizing-extension]
Kengo, K. and A. Matsumoto, "NAT TIME_WAIT reduction", draft-naito-nat-resource-optimizing-extension-02 (work in progress), July 2012.

[TCPWILD] Qian, F., Subhabrata, S., Spatscheck, O., Morley Mao, Z., and W. Willinger, "TCP Revisited: A Fresh Look at TCP in the Wild".

Authors' Addresses

Reinaldo Penno
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, California 95134
USA

Email: repenno@cisco.com

Simon Perreault
Viagenie
2875 boul. Laurier, suite D2-630
Quebec, QC G1V 2M2
Canada

Email: simon.perreault@viagenie.ca

Sarat Kamiset
Consultant
California

Phone:
Fax:

Mohamed Boucadair
France Telecom
Rennes,   35000
France

Email: mohamed.boucadair@orange.com

Kengo Naito
NTT
Tokyo
Japan

Email: kengo@lab.ntt.co.jp

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: September 12, 2012

R. Penno
A. Durand
Juniper Networks
A. Clauberg
Deutsche Telekom AG
L. Hoffmann
Bouygues Telecom

March 11, 2012

Stateless DS-Lite
draft-penno-softwire-sdnat-02

Abstract

This memo define a simple stateless and deterministic mode of operating a carrier-grade NAT as a backward compatible evolution of DS-Lite.

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 http://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 September 12, 2012.

Copyright Notice

Table of Contents

1. Introduction

DS-Lite [RFC6333],is a solution to deal with the IPv4 exhaustion problem once an IPv6 access network is deployed. It enables unmodified IPv4 application to access the IPv4 Internet over the IPv6 access network. In the DS-Lite architecture, global IPv4 addresses are shared among subscribers in the AFTR, acting as a Carrier-Grade NAT (CGN).

[I-D.ietf-softwire-public-4over6] extends the original DS-Lite model to offer a mode where the NAT function is performed in the CPE. This simplifies the AFTR operation as it does not have to perform the NAT function anymore, however, the flip side is that the address sharing function among subscribers was no longer available. [I-D.cui-softwire-b4-translated-ds-lite] introduces port restrictions, but does not completely specifies how the CPE acquires the information about its IPv4 address and its port range. More importantly, that draft does not explain how this solution can be deployed in a regular DS-Lite environment. This memo addresses these issues and clarifies the operation model.

Other approaches like variations of 4rd allows also for a full stateless operation of the decapsulation device. By introducing a strong coupling between the IPv6 address and the derived IPv4 address, they get rid of the per-subscriber state on the decapsulation devices. The approach take here argues that such per-subscriber state is not an issue as it is easily replicated among all decapsulation devices. Eliminating the strong coupling between IPv6 and IPv4 derived addresses, the approach presented here enables service providers a greater flexibility on how their limited pool of IPv4 addresses is managed. It also provide greater freedom on how IPv6 addresses are allocated, as sequential allocation is no longer a pre-requisite.

The approach presented here is stateless and deterministic. It is stateless is NAT bindings are maintained on the CPE, not on the AFTR. It is deterministic as no logs are required on the AFTR to identify which subscriber is using an external Ipv4 address and port.

The stateless DS-Lite architecture has the following characteristics:

- Backward compatible with DS-Lite. A mix of regular DS-Lite CPE and stateless DS-Lite CPEs can interoperate with a stateless DS-Lite AFTR.

- Zero log: Because the AFTR relies only on a per-subscriber mapping table that is reversible, the ISP does not need to keep any NAT binding logs.

- Stateless AFTR: There is no per-session state on the AFTRs. By leveraging this stateless and deterministic mode of operation, an ISP can deploy any number of AFTRs to provide redundancy and scalability at low cost. Because there is no per-flow state to maintain, AFTR can implement the functionality in hardware and perform it at high speed with low latency.

- Flexibility of operation: The ISP can add or remove addresses from the NAT pool without having to renumber the access network.

- Leverage IPv6: This stateless DS-Lite model leverage the IPv6 access network deployed by the ISPs.

2. Stateless DS-Lite CPE

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

A Stateless DS-Lite CPE operates in similar fashion than a regular DS-Lite CPE, where the NAT function is re-introduced in CPE with a modification on how ports are managed.

2.1. Learning external IPv4 address

A stateless DS-Lite CPE MUST implement the DHCPv4 client relay option defined in [I-D.ietf-dhc-dhcpv4-over-ipv6] to learn is external IPv4 address. Other mechanism, such as manual configuration or TR69, MAY be implemented.

2.2. Learning external port range

A stateless DS-Lite CPE MUST implement the ICMP "port restricted" option defined later in this memo.

At boot time and later at intervals of 1h +/- a random number of seconds between 0 and 900), the stateless DS-Lite CPE MUST send packets with source port 0, source IPv4 address of the B4 element, destination IPv4 address 192.0.0.1 (the AFTR well-known IPv4 address) destination port 0, for each of the supported transport protocols (usually TCP and UDP). This will trigger an ICMP "port restricted" message from the AFTR.

After validating the content of the "ICMP port restricted" message, the stateless DS-Lite CPE MUST configure its port pool with it. If existing connections were using source ports outside of that range, the stateless DS-Lite CPE MUST terminate them.

2.3. Stateless DS-Lite CPE operation

The stateless DS-Lite CPE performs IPv4 NAPT from the internal RFC1918 addresses to the IPv4 address configured on the WAN interface, restricting its available ports to the range obtained as described above.

2.4. Host-based Stateless DS-Lite

Any host initiating directly a DS-Lite IPv4 over IPv6 tunnel can benefit from this techniques by implementing a 'virtual' stateless DS-Lite CPE function within its IP stack.

3. Stateless AFTR

3.1. Anycast IPv6 address for Stateless AFTR

All stateless AFTRs associated to a domain (or group of subscribers) will be configured with the same IPv6 address on the interface facing IPv6 subscribers. A route for that IPv6 address will be anycasted within the access network.

3.2. Stateless AFTR IPv4 address pool

All stateless AFTRs associated to a domain (or group of subscribers) MUST be configured with the same pool of global IPv4 addresses.

Routes to the pool of global IPv4 addresses configured on the stateless AFTRs will be anycasted by the relevant AFTRs within the ISP routing domain.

3.3. Stateless AFTR per-subscriber mapping table

Stateless AFTRs associated to a domain (or group of subscribers) MUST be configured with the same per-subscriber mapping table, associating the IPv6 address of the subscriber CPE to the external IPv4 address and port range provisioned for this subscriber.

Because the association IPv6 address --- IPv4 address + port range is not tied to a mathematical formula, the ISP maintains all flexibility to allocate independently IPv6 address and IPv4 addresses. In particular, IPv6 addresses do not have to be allocated sequentially and IPv4 resources can be modified freely.

IPv6 address	IPv4 address	port-range
2001:db8::1	1.2.3.4	1000-1999
2001:db8::5	1.2.3.4	2000-2999
2001:db8::a:1	1.2.3.4	3000-3999

Figure 1: Per-subscriber mapping table example

This per-subscriber mapping table can be implemented in various ways which details are out of scope for this memo. In its simplest form, it can be a static file that is replicated out-of-band on the AFTRs. In a more elaborated way, this table can be dynamically built using radius queries to a subscriber database.

3.4. Stateless AFTR decapsulation rules

Upstream IPv4 over IPv6 traffic will be decapsulated by the AFTR. The AFTR MUST check the outer IPv6 source address belongs to an identified subscriber and drop the traffic if not. The AFTR MUST then check the inner IPv4 header to make sure the IPv4 source address and ports are valid according to the per-subscriber mapping table.

If the inner IPv4 source address does not match the entry in the per-subscriber mapping table, the packet MUST be discarded and an ICMP 'administratively prohibited' message MAY be returned.

If the IPv4 source port number falls outside of the range allocated to the subscriber, the AFTR MUST discard the datagram and MUST send back an ICMP "port restricted" message to the IPv6 source address of the packet.

Fragmentation and reassembly is treated as in DS-Lite [RFC6333].

3.5. Stateless AFTR encapsulation rules

Downstream traffic is validated using the per-subscriber mapping table. Traffic that falls outside of the IPv4 address/port range entries in that table MUST be discarded. Validated traffic is then encapsulated in IPv6 and forwarded to the associated IPv6 address.

Fragmentation and reassembly is treated as in DS-Lite [RFC6333].

3.6. Redundancy and fail over

Because there is no per-flow state, upstream and downstream traffic can use any stateless AFTR.

3.7. SD-AFTR stateless domain

Using the DHCPv6 DS-Lite tunnel-end-point option, groups of subscribers and can be associated to a different stateless AFTR domain. That can allow for differentiated level of services, e.g. number of ports per customer device, QoS, bandwidth, value added services,...

4. Backward compatibility with DS-Lite

A number of service providers are, or are in the process of, deploying DS-Lite in their network. They are interested in evolving their design toward a stateless model. Backward compatibility is a critical issue, as, from an operational perspective, it is difficult to get all CPEs evolve at the same time.

So AFTRs have to be ready to service CPEs that are pure DS-Lite, some that are implementing only DHCPv4 over IPv6 and handle the NAT on the full IPv4 address themselves and some that also implement port restrictions via the ICMP message described here. For this reason, a AFTR operating in backward compatibility mode MAY decide to re-NAT upstream packets which source port number do not fall into the predefined range instead of simply dropping the packets.

The operating model is the following:

- Stateless DS-Lite: for CPEs that pre-NAT and pre-shape the source port space into the range assigned to the subscriber: decapsulate, check per-subscriber mapping, forward.

- B4-translated DS-Lite: for CPEs that performs NAT before encapsulation and are allocated a full IPv4 address: decapsulate, check per-subscriber mapping, forward.

- Re-shaper DS-Lite: for CPEs that pre NAT but fail to restrict the source ports: decapsulate, check per-subscriber mapping, re-NAT statefully the packets into the restricted port range, mark range as ’stateful’, forward.

- Regular DS-Lite: for regular DS-Lite CPEs that do not pre-NAT: decapsulate, NAT statefully, forward.

In such a backward compatibility mode, the AFTR is only operating statelessly for the stateless DS-Lite CPEs. It needs to maintain per-flow state for the regular DS-Lite CPEs and the non-ICMP port restricted compliant CPEs. In this legacy mode where per-flow state is required, the simple anycast-based fail-over mechanism is no longer available.

5. ICMP port restricted message

Note: this section may end-up being a separate Internet draft.

5.1. Introduction

In the framework of A+P RFC 6346 [RFC6346], sources may be restricted to use only a subset of the port range of a transport protocol associated with an IPv4 address. When that source transmit a packet with a source outside of the pre-authorized range, the upstream NAT will drop the packet and use the ICMP message defined here to inform the source of the actual port range allocated.

This memo defines such ICMP messages for TCP and UDP and leaves the definition of the ICMP option for other transport protocol for future work.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

5.2. Source port restricted ICMP

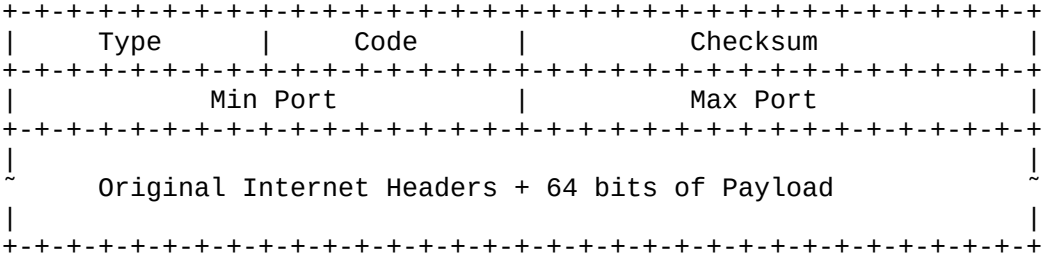


Figure 2: Source Port Restricted ICMP

Type: TBD for Source Port Restricted

Checksum: The checksum is the 16-bit ones’s complement of the one’s complement sum of the ICMP message starting with the ICMP Type. For

computing the checksum , the checksum field should be zero. This checksum may be replaced in the future.

Code: 6 for TCP, 17 for UDP

Min Port: The lowest port number allocated for that source.

Max Port: The highest port number allocated for that source.

5.3. Host behavior

A host receiving an ICMP type TBD message for a given transport protocol SHOULD NOT send packets sourced by the IP address(es) corresponding to the interface that received that ICMP message with source ports outside of the range specified for the given transport protocol.

Packets sourced with port numbers outside of the restricted range MAY be dropped or NATed upstream to fit within the restricted range.

A host MUST NOT take port restriction information applying to a given IP address and transport protocol and applies it to other IP addresses on other interfaces and/or other transport protocols.

If Min Port = 0 and Max Port = 65535, it indicates that the entire port range for the given transport protocol is available. If such 'full range' messages are received for all transport protocols, the host can take this as an indication that its IP address is probably not shared with other devices.

In order to mitigate possible man in the middle attacks, a host MUST discard ICMP type TBD messages if the associated port range (Max Port - Min Port) is lower than 64.

6. IANA Considerations

IANA is to allocated a code point for this ICMP message type.

7. Security Considerations

This ICMP message type has the same security properties as other ICMP messages such as Redirect or Destination Unreachable. A man-in-the-middle attack can be mounted to create a DOS attack on the source. Ingress filtering on network boundary can mitigate such attacks. However, in case such filtering measures are not enough, the additional provision that a host MUST discard such ICMP message with

a port range smaller than 64 can mitigate even further such attacks.

As described in [RFC6269], with any fixed size address sharing techniques, port randomization is achieved with a smaller entropy.

Recommendations listed in [RFC6302] applies.

8. References

8.1. Normative references

[I-D.ietf-dhc-dhcpv4-over-ipv6]
Lemon, T., Cui, Y., Wu, P., and J. Wu, "DHCPv4 over IPv6 Transport", draft-ietf-dhc-dhcpv4-over-ipv6-00 (work in progress), November 2011.

[RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, September 1981.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC6333] Durand, A., Droms, R., Woodyatt, J., and Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion", RFC 6333, August 2011.

8.2. Informative references

[I-D.cui-softwire-b4-translated-ds-lite]
Boucadair, M., Sun, Q., Tsou, T., Lee, Y., and Y. Cui, "Lightweight 4over6: An Extension to DS-Lite Architecture", draft-cui-softwire-b4-translated-ds-lite-05 (work in progress), February 2012.

[I-D.ietf-pcp-base]
Cheshire, S., Boucadair, M., Selkirk, P., Wing, D., and R. Penno, "Port Control Protocol (PCP)", draft-ietf-pcp-base-23 (work in progress), February 2012.

[I-D.ietf-softwire-public-4over6]
Cui, Y., Wu, J., Wu, P., Metz, C., Vautrin, O., and Y. Lee, "Public IPv4 over Access IPv6 Network", draft-ietf-softwire-public-4over6-00 (work in progress), September 2011.

[RFC6269] Ford, M., Boucadair, M., Durand, A., Levis, P., and P. Roberts, "Issues with IP Address Sharing", RFC 6269,

June 2011.

[RFC6302] Durand, A., Gashinsky, I., Lee, D., and S. Sheppard, "Logging Recommendations for Internet-Facing Servers", BCP 162, RFC 6302, June 2011.

[RFC6346] Bush, R., "The Address plus Port (A+P) Approach to the IPv4 Address Shortage", RFC 6346, August 2011.

Authors' Addresses

Reinaldo Penno
Juniper Networks
1194 North Mathilda Avenue
Sunnyvale, CA 94089-1206
USA

Email: rpenno@juniper.net

Alain Durand
Juniper Networks
1194 North Mathilda Avenue
Sunnyvale, CA 94089-1206
USA

Email: adurand@juniper.net

Alex Clauberg
Deutsche Telekom AG
GTN-FM4
Landgrabenweg 151
Bonn, CA 53227
Germany

Email: axel.clauberg@telekom.de

Lionel Hoffmann
Bouygues Telecom
TECHNOPOLE
13/15 Avenue du Marechal Juin
Meudon 92360
France

Email: lhoffman@bouyguestelecom.fr

Network Working Group
Internet-Draft
Obsoletes: 4008 (if approved)
Intended status: Standards Track
Expires: March 8, 2012

S. Perreault
Viagenie
T. Tsou
Huawei Technologies
September 5, 2011

Definitions of Managed Objects for Network Address Translators (NAT)
draft-perreault-opsawg-natmib-bis-00

Abstract

This memo defines a portion of the Management Information Base (MIB) for devices implementing Network Address Translator (NAT) function. This MIB module may be used for configuration as well as monitoring of a device capable of NAT function. This memo is a revision of the previous NAT-MIB [RFC4008] to take into account new types of NAT.

Status of This Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on March 8, 2012.

Copyright Notice

Table of Contents

1. Introduction

This memo defines a portion of the Management Information Base (MIB) for devices implementing NAT function. This MIB module may be used for configuration and monitoring of a device capable of NAT function. NAT types and their characteristics are defined in [RFC2663]. Traditional NAT function, in particular is defined in [RFC3022]. This MIB does not address the firewall functions and must not be used for configuring or monitoring these. Section 3 provides references to the SNMP management framework, which was used as the basis for the MIB module definition. Section 4 describes the terms used throughout the document. Section 5 provides an overview of the key objects, their inter-relationship, and how the MIB module may be used to configure and monitor a NAT device. Lastly, Section 6 has the complete NAT MIB definition.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Changes from RFC4008

TODO: Move this section to an appendix after initial reviews.

- Address pools can now be shared between multiple interfaces. This change makes this MIB applicable to DS-Lite's AFTR [RFC6333]. See [draft-schoenw-behave-nat-mib-bis-00] for rationale.

- TODO: Merge CGN stuff from draft-jpdionne-behave-cgn-mib.

- TODO: Merge NAT64 stuff from draft-jpdionne-behave-nat64-mib.

- TODO: Update to RFC 4787 terminology for describing NAT behavior.

- TODO: Support protocols other than UDP and TCP.

- TODO: Add support to limit and/or throttle binding allocations.

- TODO: Clarify existing notifications (e.g., natPacketDiscard) and add any additional notifications that may be needed for binding limits / binding throttling.

- TODO: Are we missing anything for PCP support? (time-limited static entries)

- TODO: Include (for example in an appendix) a description plus examples how the revised NAT-MIB can be used by NAT64 implementations, CGNs, and DS- Lite implementations.

3. The Internet-Standard Management Framework

For a detailed overview of the documents that describe the current Internet-Standard Management Framework, please refer to section 7 of [RFC3410].

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. MIB objects are generally accessed through the Simple Network Management Protocol (SNMP). Objects in the MIB are defined using the mechanisms defined in the Structure of Management Information (SMI). This memo specifies a MIB module that is compliant to the SMIv2, which is described in STD 58, RFC 2578 [RFC2578], STD 58, RFC 2579 [RFC2579] and STD 58, RFC 2580 [RFC2580].

4. Terminology

[To be Reviewed]

Definitions for a majority of the terms used throughout the document may be found in [RFC2663]. Additional terms that further classify NAPT implementations are defined in [RFC3489]. Listed below are terms used in this document.

Address realm - An address realm is a realm of unique network addresses that are routable within the realm. For example, an enterprise address realm could be constituted of private IP addresses in the ranges specified in [RFC1918], which are routable within the enterprise, but not across the Internet. A public realm is constituted of globally unique network addresses.

Symmetric NAT - Symmetric NAT, as defined in [RFC3489], is a variation of Network Address Port Translator (NAPT). Symmetric NAT does not use port bind for translation across all sessions originating from the same private host. Instead, it assigns a new public port to each new session, irrespective of whether the new session used the same private end-point as before.

Bind or Binding - Several variations of the term ’Bind’ (or ’Binding’) are used throughout the document. Address Bind (or Address Binding) is a tuple of (Private IP address, Public IP Address) used for translating an IP address end-point in IP packets. Port Bind (or, Port Binding, or Address Port Bind, or Address Port Binding) is a tuple of (transport protocol, Private IP address, Private port, Public IP Address, Public port) used for translating a port end-point tuple of (transport protocol, IP address, port). Bind is used to refer to either Address Bind or Port Bind. Bind Mode identifies whether a bind is Address Bind or Port Bind.

NAT Session - A NAT session is an association between a session as seen in the private realm and a session as seen in the public realm, by virtue of NAT translation. If a session in the private realm were to be represented as (PrivateSrcAddr, PrivateDstAddr, TransportProtocol, PrivateSrcPort, PrivateDstPort) and the same session in the public realm were to be represented as (PublicSrcAddr, PublicDstAddr, TransportProtocol, PublicSrcPort, PublicDstPort), the NAT session will provide the translation glue between the two session representations. NAT sessions in the document are restricted to sessions based on TCP and UDP only. In the future, NAT sessions may be extended to be based on other transport protocols such as SCTP, UDP-lite and DCCP.

The terms 'local' and 'private' are used interchangeably throughout the document when referring to private networks, IP addresses, and ports. Likewise, the terms 'global' and 'public' are used interchangeably when referring to public networks, IP addresses, and ports.

5. Overview

NAT MIB is configurable on a per-interface basis and depends in several parts on the IF-MIB [RFC2863].

NAT MIB requires that an interface for which NAT is configured be connected to either a private or a public realm. The realm association of the interface plays an important role in the definition of address maps for the interface. An address map entry identifies the orientation of the session (inbound or outbound to the interface) for which the entry may be used for NAT translation. The address map entry also identifies the end-point of the session that must be subject to translation. An SNMP Textual-Convention 'NatTranslationEntity' is defined to capture this important characteristic that combines session orientation and applicable session endpoint for translation.

An address map may consist of static or dynamic entries. NAT creates static binds from a static address map entry. Each static bind has a direct one-to-one relationship with a static address map entry. NAT creates dynamic binds from a dynamic address map entry upon seeing the first packet of a new session.

The following subsections define the key objects used in NAT MIB, their inter-relationship, and how to configure a NAT device using the MIB module.

5.1. natInterfaceTable

[To be reviewed]

natInterfaceTable is defined in the MIB module to configure interface specific realm type and the NAT services enabled for the interface. natInterfaceTable is indexed by ifIndex and also includes interface specific NAT statistics.

The first step for an operator in configuring a NAT device is determining the interface over which NAT service is to be configured. When NAT service is operational, translated packets traverse the NAT device by ingressing on a private interface and egressing on a public interface or vice versa. An operator may configure the NAT service on either the public interface or the private interface in the traversal path.

As the next step, the operator must identify the NAT service(s) desired for the interface. The operator may configure one or more NAT services on the same interface. The MIB module identifies four types of NAT services: Basic NAT, NAPT, twice NAT and bidirectional NAT. These are NAT varieties as defined in [RFC2663]. Note that [RFC3489] further classifies NAPT implementations based on the behavior exhibited by the NAPT devices from different vendors. However, the MIB module does not explicitly distinguish between the NAPT implementations. NAPT implementations may be distinguished between one another by monitoring the BIND and NAT Session objects generated by the NAT device as described in section Section 5.6.

5.2. natAddrMapTable

[To be reviewed]

natAddrMapTable is defined in the MIB module to configure address maps on a per-interface basis. natAddrMapTable is indexed by the tuple of (ifIndex, natAddrMapIndex). The same table is also used to collect Statistics for the address map entries. Address maps are key to NAT configuration. An operator may configure one or more address map entries per interface. NAT looks up address map entries in the order in which they are defined to determine the translation function at the start of each new session traversing the interface. An address map may consist of static or dynamic entries. A static address map entry has a direct one-to-one relationship with binds. NAT will dynamically create binds from a dynamic address map entry.

The operator must be careful in selecting address map entries for an interface based on the interface realm-type and the type of NAT service desired. The operator can be amiss in the selection of

address map entries when not paying attention to the associated interface characteristics defined in natInterfaceTable (described in section 4.1). For example, say the operator wishes to configure a NAPT map entry on an interface of a NAT device. If the operator chooses to configure the NAPT map entry on a public interface (i.e., interface realm-type is public), the operator should set the TranslationEntity of the NAPT address map entry to be outboundSrcEndPoint. On the other hand, if the operator chooses to configure the NAPT map entry on a private interface (i.e., interface realm-type is private), the operator should set the TranslationEntity of the NAPT address map entry to be InboundSrcEndPoint.

5.3. Default Timeouts, Protocol Table, and Other Scalars

[To be reviewed]

DefTimeouts is defined in the MIB module to configure idle Bind timeout and IP protocol specific idle NAT session timeouts. The timeouts defined are global to the system and are not interface specific.

Protocol specific statistics are maintained in natProtocolTable, which is indexed by the protocol type.

The scalars natAddrBindNumberOfEntries and natAddrPortBindNumberOfEntries hold the number of entries that currently exist in the Address Bind and the Address Port Bind tables, respectively.

The generation of natPacketDiscard notifications can be configured by using the natNotifThrottlingInterval scalar MIB object.

5.4. natAddrBindTable and natAddrPortBindTable

[To be reviewed]

Two Bind tables, natAddrBindTable and natAddrPortBindTable, are defined to hold the bind entries. Entries are derived from the address map table and are not configurable. natAddrBindTable contains Address Binds, and natAddrPortBindTable contains Address Port Binds. natAddrBindTable is indexed by the tuple of (ifIndex, LocalAddrType, LocalAddr). natAddrPortBindTable is indexed by the tuple of (ifIndex, LocalAddrType, LocalAddr, LocalPort, Protocol). These tables also maintain bind specific statistics. A Symmetric NAT will have no entries in the Bind tables.

5.5. natSessionTable

[To be reviewed]

natSessionTable is defined to hold NAT session entries. NAT session entries are derived from NAT Binds (except in the case of Symmetric NAT) and are not configurable.

The NAT session provides the necessary translation glue between two session representations of the same end-to-end session; that is, a session as seen in the private realm and in the public realm. Session orientation (inbound or outbound) is determined from the orientation of the first packet traversing the NAT interface. Address map entries and bind entries on the interface determine whether a session is subject to NAT translation. One or both endpoints of a session may be subject to translation.

With the exception of symmetric NAT, all other NAT functions use end-point specific bind to perform individual end-point translations. Multiple NAT sessions would use the same bind as long as they share the same endpoint. Symmetric NAT does not retain a consistent port bind across multiple sessions using the same endpoint. For this reason, the bind identifier for a NAT session in symmetric NAT is set to zero. natSessionTable is indexed by the tuple of (ifIndex, natSessionIndex). Statistics for NAT sessions are also maintained in the same table.

5.6. RFC 3489 NAPT Variations, NAT Session and Bind Tables

[To be reviewed, translate to new terminology]

[RFC3489] defines four variations of NAPT - Full Cone, Restricted Cone, Port Restricted Cone, and Symmetric NAT. These can be differentiated in the NAT MIB based on different values for the objects in the session and the bind tables, as indicated below.

In a Port Restricted Cone NAT, NAT Session objects will contain a non-zero PrivateSrcEPBindId object. Further, all address and port objects within a NAT session will have non-zero values (i.e., no wildcard matches).

An Address Restricted Cone NAT may have been implemented in the same way as a Port Restricted Cone NAT, except that the UDP NAT Sessions may use ANY match on PrivateDstPort and PublicDstPort objects; i.e., PrivateDstPort and PublicDstPort objects within a NAT session may be set to zero.

A Full Cone NAT may have also been implemented in the same way as a

Port Restricted Cone NAT, except that the UDP NAT Sessions may use ANY match on PrivateDstAddr, PrivateDstPort, PublicDstAddr, and PublicDstPort objects. Within a NAT Session, all four of these objects may be set to zero. Alternately, all address and port objects within a NAT Session may have non-zero values, yet the TranslationEntity of the PrivateSrcEPBindId for the NAT Sessions may be set bi-directionally, i.e., as a bit mask of (outboundSrcEndPoint and inboundDstEndPoint) or (inboundSrcEndPoint and outboundDstEndPoint), depending on the interface realm type. Lastly, a Symmetric NAT does not maintain Port Bindings. As such, the NAT Session objects will have the PrivateSrcEPBindId set to zero.

5.7. Notifications

[To be reviewed]

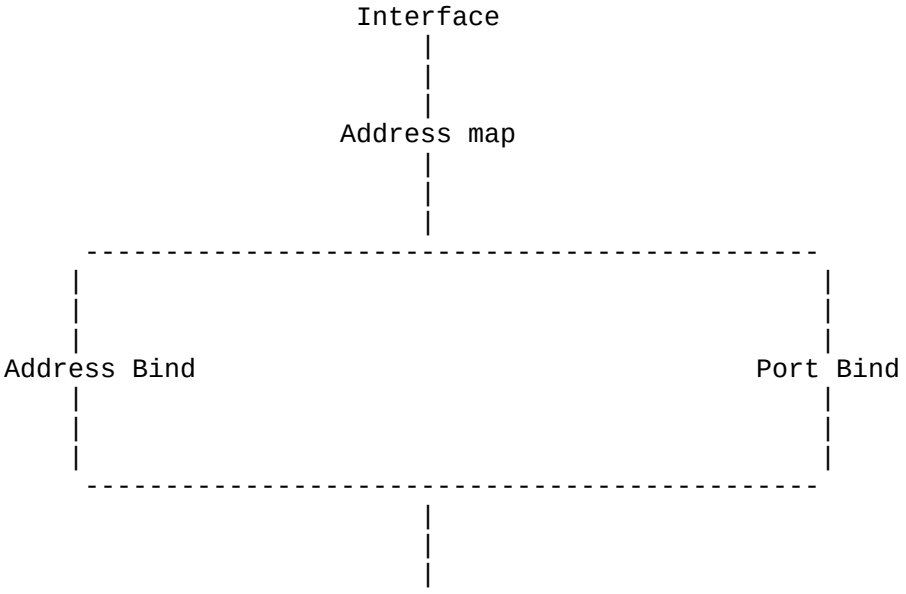
natPacketDiscard notifies the end user/manager of packets being discarded due to lack of address mappings.

[Port exhaustion, CGN-MIB?]

5.8. Notifications

[To be reviewed]

The association between the various NAT tables can be represented as follows:



NAT Session

All NAT functions, with the exception of Symmetric NAT, use Bind(s) to provide the glue necessary for a NAT Session. natSessionPrivateSrcEPBindId and natSessionPrivateDstEPBindId objects represent the endpoint Binds used by NAT Sessions.

5.9. Configuration via the MIB

[To be reviewed]

Section 5.1, and Section 5.2 and part of Section 5.3 refer to objects that are configurable on a NAT device. NAT derives Address Bind and Address Port Bind entries from the Address Map table. Hence, an Address Bind or an Address Port Bind entry must not exist without an associated entry in the Address Map table.

Further, NAT derives NAT session entries from NAT Binds, except in the case of symmetric NAT, which derives translation parameters for a NAT session directly from an address map entry. Hence, with the exception of Symmetric NAT, a NAT session entry must not exist in the NAT Session table without a corresponding bind.

A Management station may use the following steps to configure entries in the NAT-MIB:

- Create an entry in the natInterfaceTable specifying the value of ifIndex as the interface index of the interface on which NAT is being configured. Specify appropriate values, as applicable, for the other objects (e.g., natInterfaceRealm, natInterfaceServiceType) in the table (refer to Section 5.1).

- Create one or more address map entries sequentially in reduced order of priority in the natAddrMapTable, specifying the value of ifIndex to be the same for all entries. The ifIndex specified would be the same as that specified for natInterfaceTable (refer to Section 5.2).

- Configure the maximum permitted idle time duration for BINDs and TCP, UDP, and ICMP protocol sessions by setting the relevant scalars in natDefTimeouts object (refer to Section 5.3).

5.10. Relationship to Interface MIB

[To be reviewed, relationship to other MIB?]

The natInterfaceTable specifies the NAT configuration attributes on each interface. The concept of "interface" is as defined by

InterfaceIndex/ifIndex of the IETF Interfaces MIB [RFC2863].

6. Definitions

This MIB module IMPORTs objects from [RFC2578], [RFC2579], [RFC2580], [RFC2863], [RFC3411], and [RFC4001]. It also refers to information in [RFC0792], [RFC2463], and [RFC3413].

```
NAT-MIB DEFINITIONS ::= BEGIN

IMPORTS
     MODULE-IDENTITY,
     OBJECT-TYPE,
     Integer32,
     Unsigned32,
     Gauge32,
     Counter64,
     TimeTicks,
     mib-2,
     NOTIFICATION-TYPE
             FROM SNMPv2-SMI
     TEXTUAL-CONVENTION,
     StorageType,
     RowStatus
             FROM SNMPv2-TC
     MODULE-COMPLIANCE,
     NOTIFICATION-GROUP,
     OBJECT-GROUP
             FROM SNMPv2-CONF
     ifIndex,
     ifCounterDiscontinuityGroup
             FROM IF-MIB
     SnmpAdminString
             FROM SNMP-FRAMEWORK-MIB
     InetAddressType,
     InetAddress,
     InetPortNumber
             FROM INET-ADDRESS-MIB;

natMIB MODULE-IDENTITY
     LAST-UPDATED "YYYYMMDDhhmmZ"
     ORGANIZATION "IETF Transport Area"
     CONTACT-INFO
              "
               Simon Perreault
               Viagenie
               2875 boul. Laurier, suite D2-630
               Quebec
               Canada
```

```
               Phone: +1-418-656-9254
               EMail: simon.perreault@viagenie.ca

               Tina Tsou
               Huawei Technologies
               2330 Central Expressway
               Santa Clara
               USA
               Phone: +1-408-330-4424
               EMail: tena@huawei.com
              "
     DESCRIPTION
             "This MIB module defines the generic managed objects
              for NAT.

              Copyright (C) The Internet Society (YYYY).  This version
              of this MIB module is part of RFC XXXX;  see the RFC
              itself for full legal notices."
     REVISION     "200503210000Z"  -- 21th March 2005
     DESCRIPTION
             "Initial version, published as RFC 4008."
     REVISION     "YYYYMMDDhhmmZ"
     DESCRIPTION
             "Second version, published as RFC XXXX."

     ::= { mib-2 123 }

natMIBObjects OBJECT IDENTIFIER ::= { natMIB 1 }

NatProtocolType ::= TEXTUAL-CONVENTION
       STATUS       current
       DESCRIPTION
               "A list of protocols that support the network
                address translation.  Inclusion of the values is
                not intended to imply that those protocols
                need to be supported.  Any change in this
                TEXTUAL-CONVENTION should also be reflected in
                the definition of NatProtocolMap, which is a
                BITS representation of this."
       SYNTAX   INTEGER {
                    none (1),  -- not specified
                    other (2), -- none of the following
                    icmp (3),
                    udp (4),
                    tcp (5)
                  }

NatProtocolMap ::= TEXTUAL-CONVENTION
```

```
        STATUS       current
        DESCRIPTION
                "A bitmap of protocol identifiers that support
                 the network address translation.  Any change
                 in this TEXTUAL-CONVENTION should also be
                 reflected in the definition of NatProtocolType."
        SYNTAX   BITS {
                   other (0),
                   icmp (1),
                   udp (2),
                   tcp (3)
                 }

NatAddrMapId ::= TEXTUAL-CONVENTION
        DISPLAY-HINT "d"
        STATUS current
        DESCRIPTION
                "A unique id that is assigned to each address map
                 by a NAT enabled device."
        SYNTAX   Unsigned32 (1..4294967295)

NatSharedAddrMapId ::= TEXTUAL-CONVENTION
        DISPLAY-HINT "d"
        STATUS current
        DESCRIPTION
                "A unique id that is assigned to each shared address
                 map by a NAT enabled device."
        SYNTAX   Unsigned32 (1..4294967295)

NatBindIdOrZero ::= TEXTUAL-CONVENTION
        DISPLAY-HINT "d"
        STATUS current
        DESCRIPTION
                "A unique id that is assigned to each bind by
                 a NAT enabled device.  The bind id will be zero
                 in the case of a Symmetric NAT."
        SYNTAX   Unsigned32 (0..4294967295)

NatBindId ::= TEXTUAL-CONVENTION
        DISPLAY-HINT "d"
        STATUS current
        DESCRIPTION
                "A unique id that is assigned to each bind by
                 a NAT enabled device."
        SYNTAX   Unsigned32 (1..4294967295)

NatSessionId ::= TEXTUAL-CONVENTION
        DISPLAY-HINT "d"
```

```
       STATUS current
       DESCRIPTION
               "A unique id that is assigned to each session by
                a NAT enabled device."
       SYNTAX   Unsigned32 (1..4294967295)

NatBindMode ::= TEXTUAL-CONVENTION
       STATUS current
       DESCRIPTION
               "An indication of whether the bind is
                an address bind or an address port bind."
       SYNTAX   INTEGER {
                    addressBind (1),
                    addressPortBind (2)
                }

NatAssociationType ::= TEXTUAL-CONVENTION
       STATUS current
       DESCRIPTION
               "An indication of whether the association is
                static or dynamic."
       SYNTAX   INTEGER {
                    static (1),
                    dynamic (2)
                }

NatTranslationEntity ::= TEXTUAL-CONVENTION
       STATUS       current
       DESCRIPTION
               "An indication of a) the direction of a session for
                which an address map entry, address bind or port
                bind is applicable, and b) the entity (source or
                destination) within the session that is subject to
                translation."
       SYNTAX   BITS {
                  inboundSrcEndPoint (0),
                  outboundDstEndPoint(1),
                  inboundDstEndPoint (2),
                  outboundSrcEndPoint(3)
                }

--
-- Default Values for the Bind and NAT Protocol Timers
--

natDefTimeouts OBJECT IDENTIFIER ::= { natMIBObjects 1 }

natNotifCtrl OBJECT IDENTIFIER ::= { natMIBObjects 2 }
```

```
--
-- Address Bind and Port Bind related NAT configuration
--

natBindDefIdleTimeout OBJECT-TYPE
    SYNTAX     Unsigned32  (0..4294967295)
    UNITS      "seconds"
    MAX-ACCESS read-write
    STATUS     current
    DESCRIPTION
            "The default Bind (Address Bind or Port Bind) idle
             timeout parameter.

             If the agent is capable of storing non-volatile
             configuration, then the value of this object must be
             restored after a re-initialization of the management
             system."
    DEFVAL { 0 }
    ::= { natDefTimeouts 1 }

--
-- UDP related NAT configuration
--

natUdpDefIdleTimeout OBJECT-TYPE
    SYNTAX     Unsigned32  (1..4294967295)
    UNITS      "seconds"
    MAX-ACCESS read-write
    STATUS     current
    DESCRIPTION
            "The default UDP idle timeout parameter.

             If the agent is capable of storing non-volatile
             configuration, then the value of this object must be
             restored after a re-initialization of the management
             system."
    DEFVAL { 300 }
    ::= { natDefTimeouts 2 }

--
-- ICMP related NAT configuration
--

natIcmpDefIdleTimeout OBJECT-TYPE
    SYNTAX     Unsigned32  (1..4294967295)
    UNITS      "seconds"
    MAX-ACCESS read-write
    STATUS     current
```

```
    DESCRIPTION
            "The default ICMP idle timeout parameter.

             If the agent is capable of storing non-volatile
             configuration, then the value of this object must be
             restored after a re-initialization of the management
             system."
    DEFVAL { 300 }
    ::= { natDefTimeouts 3 }

--
-- Other protocol parameters
--

natOtherDefIdleTimeout OBJECT-TYPE
    SYNTAX     Unsigned32  (1..4294967295)
    UNITS      "seconds"
    MAX-ACCESS read-write
    STATUS     current
    DESCRIPTION
            "The default idle timeout parameter for protocols
             represented by the value other (2) in
             NatProtocolType.

             If the agent is capable of storing non-volatile
             configuration, then the value of this object must be
             restored after a re-initialization of the management
             system."
    DEFVAL { 60 }
    ::= { natDefTimeouts 4 }

--
-- TCP related NAT Timers
--

natTcpDefIdleTimeout OBJECT-TYPE
    SYNTAX     Unsigned32  (1..4294967295)
    UNITS      "seconds"
    MAX-ACCESS read-write
    STATUS     current
    DESCRIPTION
            "The default time interval that a NAT session for an
             established TCP connection is allowed to remain
             valid without any activity on the TCP connection.

             If the agent is capable of storing non-volatile
             configuration, then the value of this object must be
             restored after a re-initialization of the management
```

```
             system."
    DEFVAL { 86400 }
    ::= { natDefTimeouts 5 }

natTcpDefNegTimeout OBJECT-TYPE
    SYNTAX     Unsigned32  (1..4294967295)
    UNITS      "seconds"
    MAX-ACCESS read-write
    STATUS     current
    DESCRIPTION
            "The default time interval that a NAT session for a TCP
             connection that is not in the established state
             is allowed to remain valid without any activity on
             the TCP connection.

             If the agent is capable of storing non-volatile
             configuration, then the value of this object must be
             restored after a re-initialization of the management
             system."
    DEFVAL { 60 }
    ::= { natDefTimeouts 6 }

natNotifThrottlingInterval OBJECT-TYPE
    SYNTAX      Integer32 (0 | 5..3600)
    UNITS       "seconds"
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
            "This object controls the generation of the
             natPacketDiscard notification.

             If this object has a value of zero, then no
             natPacketDiscard notifications will be transmitted by the
             agent.

             If this object has a non-zero value, then the agent must
             not generate more than one natPacketDiscard
             'notification-event' in the indicated period, where a
             'notification-event' is the generation of a single
             notification PDU type to a list of notification
             destinations.  If additional NAT packets are discarded
             within the throttling period, then notification-events
             for these changes must be suppressed by the agent until
             the current throttling period expires.

             If natNotifThrottlingInterval notification generation
             is enabled, the suggested default throttling period is
             60 seconds, but generation of the natPacketDiscard
```

```
             notification should be disabled by default.

             If the agent is capable of storing non-volatile
             configuration, then the value of this object must be
             restored after a re-initialization of the management
             system.

             The actual transmission of notifications is controlled
             via the MIB modules in RFC 3413."
    DEFVAL { 0 }
    ::= { natNotifCtrl 1 }

--
-- The NAT Interface Table
--

natInterfaceTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF NatInterfaceEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "This table specifies the attributes for interfaces on a
             device supporting NAT function."
    ::= { natMIBObjects 3 }

natInterfaceEntry OBJECT-TYPE
    SYNTAX      NatInterfaceEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "Each entry in the natInterfaceTable holds a set of
             parameters for an interface, instantiated by
             ifIndex.  Therefore, the interface index must have been
             assigned, according to the applicable procedures,
             before it can be meaningfully used.
             Generally, this means that the interface must exist.

             When natStorageType is of type nonVolatile, however,
             this may reflect the configuration for an interface whose
             ifIndex has been assigned but for which the supporting
             implementation is not currently present."
    INDEX   { ifIndex }
    ::= { natInterfaceTable 1 }

NatInterfaceEntry ::= SEQUENCE {
    natInterfaceRealm               INTEGER,
    natInterfaceServiceType         BITS,
    natInterfaceInTranslates        Counter64,
```

```
    natInterfaceOutTranslates       Counter64,
    natInterfaceDiscards            Counter64,
    natInterfaceStorageType         StorageType,
    natInterfaceRowStatus           RowStatus,
    natInterfaceSharedAddrMapIndex  NatSharedAddrMapId
}

natInterfaceRealm OBJECT-TYPE
    SYNTAX     INTEGER {
                   private (1),
                   public (2)
               }
    MAX-ACCESS read-create
    STATUS     current
    DESCRIPTION
            "This object identifies whether this interface is
             connected to the private or the public realm."
    DEFVAL  { public }
    ::= { natInterfaceEntry 1 }

natInterfaceServiceType OBJECT-TYPE
    SYNTAX  BITS {
                basicNat (0),
                napt (1),
                bidirectionalNat (2),
                twiceNat (3)
            }
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "An indication of the direction in which new sessions
             are permitted and the extent of translation done within
             the IP and transport headers."
    ::= { natInterfaceEntry 2 }

natInterfaceInTranslates OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "Number of packets received on this interface that
             were translated.
             Discontinuities in the value of this counter can occur at
             reinitialization of the management system and at other
             times as indicated by the value of
             ifCounterDiscontinuityTime on the relevant interface."
    ::= { natInterfaceEntry 3 }
```

```
natInterfaceOutTranslates OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "Number of translated packets that were sent out this
             interface.

             Discontinuities in the value of this counter can occur at
             reinitialization of the management system and at other
             times as indicated by the value of
             ifCounterDiscontinuityTime on the relevant interface."
    ::= { natInterfaceEntry 4 }

natInterfaceDiscards OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "Number of packets that had to be rejected/dropped due to
             a lack of resources for this interface.

             Discontinuities in the value of this counter can occur at
             reinitialization of the management system and at other
             times as indicated by the value of
             ifCounterDiscontinuityTime on the relevant interface."
     ::= { natInterfaceEntry 5 }

natInterfaceStorageType OBJECT-TYPE
    SYNTAX      StorageType
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "The storage type for this conceptual row.
             Conceptual rows having the value 'permanent'
             need not allow write-access to any columnar objects
             in the row."
    REFERENCE
            "Textual Conventions for SMIv2, Section 2."
    DEFVAL { nonVolatile }
    ::= { natInterfaceEntry 6 }

natInterfaceRowStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "The status of this conceptual row.
```

```
             Until instances of all corresponding columns are
             appropriately configured, the value of the
             corresponding instance of the natInterfaceRowStatus
             column is 'notReady'.


             In particular, a newly created row cannot be made
             active until the corresponding instance of
             natInterfaceServiceType has been set.

             None of the objects in this row may be modified
             while the value of this object is active(1)."
    REFERENCE
            "Textual Conventions for SMIv2, Section 2."
    ::= { natInterfaceEntry 7 }

natInterfaceSharedAddrMapIndex OBJECT-TYPE
    SYNTAX      NatSharedAddrMapId
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "Link to a NatSharedAddrMapEntry.  If NULL,
             it is expected that there exist at least one
             NatAddrMapEntry pointing to this interface entry."
    ::= { natInterfaceEntry 8 }


--
-- The Address Map Table
--

natAddrMapTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF NatAddrMapEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "This table lists address map parameters for NAT."
    ::= { natMIBObjects 4 }

natAddrMapEntry OBJECT-TYPE
    SYNTAX      NatAddrMapEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "This entry represents an address map to be used for
             NAT and contributes to the dynamic and/or static
             address mapping tables of the NAT device."
    INDEX   { ifIndex, natAddrMapIndex }
```

```
    ::= { natAddrMapTable 1 }

NatAddrMapEntry ::= SEQUENCE {
    natAddrMapIndex                 NatAddrMapId,
    natAddrMapName                  SnmpAdminString,
    natAddrMapEntryType             NatAssociationType,
    natAddrMapTranslationEntity     NatTranslationEntity,
    natAddrMapLocalAddrType         InetAddressType,
    natAddrMapLocalAddrFrom         InetAddress,
    natAddrMapLocalAddrTo           InetAddress,
    natAddrMapLocalPortFrom         InetPortNumber,
    natAddrMapLocalPortTo           InetPortNumber,
    natAddrMapGlobalAddrType        InetAddressType,
    natAddrMapGlobalAddrFrom        InetAddress,
    natAddrMapGlobalAddrTo          InetAddress,
    natAddrMapGlobalPortFrom        InetPortNumber,
    natAddrMapGlobalPortTo          InetPortNumber,
    natAddrMapProtocol              NatProtocolMap,
    natAddrMapInTranslates          Counter64,
    natAddrMapOutTranslates         Counter64,
    natAddrMapDiscards              Counter64,
    natAddrMapAddrUsed              Gauge32,
    natAddrMapStorageType           StorageType,
    natAddrMapRowStatus             RowStatus
}

natAddrMapIndex  OBJECT-TYPE
    SYNTAX      NatAddrMapId
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "Along with ifIndex, this object uniquely
             identifies an entry in the natAddrMapTable.
             Address map entries are applied in the order
             specified by natAddrMapIndex."
    ::= { natAddrMapEntry 1 }

natAddrMapName OBJECT-TYPE
    SYNTAX      SnmpAdminString (SIZE(1..32))
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "Name identifying all map entries in the table associated
             with the same interface.  All map entries with the same
             ifIndex MUST have the same map name."
    ::= { natAddrMapEntry 2 }

natAddrMapEntryType OBJECT-TYPE
```

```
    SYNTAX      NatAssociationType
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "This parameter can be used to set up static
             or dynamic address maps."
    ::= { natAddrMapEntry 3 }

natAddrMapTranslationEntity OBJECT-TYPE
    SYNTAX      NatTranslationEntity
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "The end-point entity (source or destination) in
             inbound or outbound sessions (i.e., first packets) that
             may be translated by an address map entry.

             Session direction (inbound or outbound) is
             derived from the direction of the first packet
             of a session traversing a NAT interface.
             NAT address (and Transport-ID) maps may be defined
             to effect inbound or outbound sessions.

             Traditionally, address maps for Basic NAT and NAPT are
             configured on a public interface for outbound sessions,
             effecting translation of source end-point.  The value of
             this object must be set to outboundSrcEndPoint for
             those interfaces.

             Alternately, if address maps for Basic NAT and NAPT were
             to be configured on a private interface, the desired
             value for this object for the map entries
             would be inboundSrcEndPoint (i.e., effecting translation
             of source end-point for inbound sessions).

             If TwiceNAT were to be configured on a private interface,
             the desired value for this object for the map entries
             would be a bitmask of inboundSrcEndPoint and
             inboundDstEndPoint."
    ::= { natAddrMapEntry 4 }

natAddrMapLocalAddrType OBJECT-TYPE
    SYNTAX      InetAddressType
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "This object specifies the address type used for
             natAddrMapLocalAddrFrom and natAddrMapLocalAddrTo."
```

```
    ::= { natAddrMapEntry 5 }

natAddrMapLocalAddrFrom OBJECT-TYPE
    SYNTAX      InetAddress
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "This object specifies the first IP address of the range
             of IP addresses mapped by this translation entry.  The
             value of this object must be less than or equal to the
             value of the natAddrMapLocalAddrTo object.

             The type of this address is determined by the value of
             the natAddrMapLocalAddrType object."
    ::= { natAddrMapEntry 6 }

natAddrMapLocalAddrTo OBJECT-TYPE
    SYNTAX      InetAddress
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "This object specifies the last IP address of the range of
             IP addresses mapped by this translation entry.  If only
             a single address is being mapped, the value of this object
             is equal to the value of natAddrMapLocalAddrFrom.  For a
             static NAT, the number of addresses in the range defined
             by natAddrMapLocalAddrFrom and natAddrMapLocalAddrTo must
             be equal to the number of addresses in the range defined by
             natAddrMapGlobalAddrFrom and natAddrMapGlobalAddrTo.
             The value of this object must be greater than or equal to
             the value of the natAddrMapLocalAddrFrom object.

             The type of this address is determined by the value of
             the natAddrMapLocalAddrType object."
    ::= { natAddrMapEntry 7 }

natAddrMapLocalPortFrom OBJECT-TYPE
    SYNTAX      InetPortNumber
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "If this conceptual row describes a Basic NAT address
             mapping, then the value of this object must be zero.  If
             this conceptual row describes NAPT, then the value of
             this object specifies the first port number in the range
             of ports being mapped.

             The value of this object must be less than or equal to the
```

```
             value of the natAddrMapLocalPortTo object.  If the
             translation specifies a single port, then the value of this
             object is equal to the value of natAddrMapLocalPortTo."
    DEFVAL { 0 }
    ::= { natAddrMapEntry 8 }

natAddrMapLocalPortTo OBJECT-TYPE
    SYNTAX      InetPortNumber
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "If this conceptual row describes a Basic NAT address
             mapping, then the value of this object must be zero.  If
             this conceptual row describes NAPT, then the value of
             this object specifies the last port number in the range
             of ports being mapped.

             The value of this object must be greater than or equal to
             the value of the natAddrMapLocalPortFrom object.  If the
             translation specifies a single port, then the value of this
             object is equal to the value of natAddrMapLocalPortFrom."
    DEFVAL { 0 }
    ::= { natAddrMapEntry 9 }

natAddrMapGlobalAddrType OBJECT-TYPE
    SYNTAX      InetAddressType
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "This object specifies the address type used for
             natAddrMapGlobalAddrFrom and natAddrMapGlobalAddrTo."
    ::= { natAddrMapEntry 10 }

natAddrMapGlobalAddrFrom OBJECT-TYPE
    SYNTAX      InetAddress
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "This object specifies the first IP address of the range of
             IP addresses being mapped to.  The value of this object
             must be less than or equal to the value of the
             natAddrMapGlobalAddrTo object.

             The type of this address is determined by the value of
             the natAddrMapGlobalAddrType object."
    ::= { natAddrMapEntry 11 }

natAddrMapGlobalAddrTo OBJECT-TYPE
```

```
    SYNTAX      InetAddress
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "This object specifies the last IP address of the range of
             IP addresses being mapped to.  If only a single address is
             being mapped to, the value of this object is equal to the
             value of natAddrMapGlobalAddrFrom.  For a static NAT, the
             number of addresses in the range defined by
             natAddrMapGlobalAddrFrom and natAddrMapGlobalAddrTo must be
             equal to the number of addresses in the range defined by
             natAddrMapLocalAddrFrom and natAddrMapLocalAddrTo.
             The value of this object must be greater than or equal to
             the value of the natAddrMapGlobalAddrFrom object.

             The type of this address is determined by the value of
             the natAddrMapGlobalAddrType object."
    ::= { natAddrMapEntry 12 }

natAddrMapGlobalPortFrom OBJECT-TYPE
    SYNTAX      InetPortNumber
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "If this conceptual row describes a Basic NAT address
             mapping, then the value of this object must be zero.  If
             this conceptual row describes NAPT, then the value of
             this object specifies the first port number in the range
             of ports being mapped to.


             The value of this object must be less than or equal to the
             value of the natAddrMapGlobalPortTo object.  If the
             translation specifies a single port, then the value of this
             object is equal to the value natAddrMapGlobalPortTo."
    DEFVAL { 0 }
    ::= { natAddrMapEntry 13 }

natAddrMapGlobalPortTo OBJECT-TYPE
    SYNTAX      InetPortNumber
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "If this conceptual row describes a Basic NAT address
             mapping, then the value of this object must be zero.  If
             this conceptual row describes NAPT, then the value of this
             object specifies the last port number in the range of
             ports being mapped to.
```

```
             The value of this object must be greater than or equal to
             the value of the natAddrMapGlobalPortFrom object.  If the
             translation specifies a single port, then the value of this
             object is equal to the value of natAddrMapGlobalPortFrom."
    DEFVAL { 0 }
    ::= { natAddrMapEntry 14 }

natAddrMapProtocol OBJECT-TYPE
    SYNTAX      NatProtocolMap
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "This object specifies a bitmap of protocol identifiers."
    ::= { natAddrMapEntry 15 }

natAddrMapInTranslates OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The number of inbound packets pertaining to this address
             map entry that were translated.

             Discontinuities in the value of this counter can occur at
             reinitialization of the management system and at other
             times, as indicated by the value of
             ifCounterDiscontinuityTime on the relevant interface."
    ::= { natAddrMapEntry 16 }

natAddrMapOutTranslates OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The number of outbound packets pertaining to this
             address map entry that were translated.

             Discontinuities in the value of this counter can occur at
             reinitialization of the management system and at other
             times, as indicated by the value of
             ifCounterDiscontinuityTime on the relevant interface."
    ::= { natAddrMapEntry 17 }

natAddrMapDiscards OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
```

```
            "The number of packets pertaining to this address map
             entry that were dropped due to lack of addresses in the
             address pool identified by this address map.  The value of
             this object must always be zero in case of static
             address map.

             Discontinuities in the value of this counter can occur at
             reinitialization of the management system and at other
             times, as indicated by the value of
             ifCounterDiscontinuityTime on the relevant interface."
    ::= { natAddrMapEntry 18 }

natAddrMapAddrUsed OBJECT-TYPE
    SYNTAX     Gauge32
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The number of addresses pertaining to this address map
             that are currently being used from the NAT pool.
             The value of this object must always be zero in the case
             of a static address map."
    ::= { natAddrMapEntry 19 }

natAddrMapStorageType OBJECT-TYPE
    SYNTAX      StorageType
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "The storage type for this conceptual row.
             Conceptual rows having the value 'permanent'
             need not allow write-access to any columnar objects
             in the row."
    REFERENCE
            "Textual Conventions for SMIv2, Section 2."
    DEFVAL { nonVolatile }
    ::= { natAddrMapEntry 20 }

natAddrMapRowStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "The status of this conceptual row.

             Until instances of all corresponding columns are
             appropriately configured, the value of the
             corresponding instance of the natAddrMapRowStatus
             column is 'notReady'.
```

```
             None of the objects in this row may be modified
             while the value of this object is active(1)."
    REFERENCE
            "Textual Conventions for SMIv2, Section 2."
    ::= { natAddrMapEntry 21 }


--
-- Address Bind section
--

natAddrBindNumberOfEntries OBJECT-TYPE
    SYNTAX     Gauge32
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "This object maintains a count of the number of entries
             that currently exist in the natAddrBindTable."
    ::= { natMIBObjects 5 }

--
-- The NAT Address BIND Table
--

natAddrBindTable OBJECT-TYPE
    SYNTAX     SEQUENCE OF NatAddrBindEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
            "This table holds information about the currently
             active NAT BINDs."
    ::= { natMIBObjects 6 }

natAddrBindEntry OBJECT-TYPE
    SYNTAX     NatAddrBindEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
            "Each entry in this table holds information about
             an active address BIND.  These entries are lost
             upon agent restart.

             This row has indexing which may create variables with
             more than 128 subidentifiers.  Implementers of this table
             must be careful not to create entries that would result
             in OIDs which exceed the 128 subidentifier limit.
             Otherwise, the information cannot be accessed using
             SNMPv1, SNMPv2c or SNMPv3."
```

```
    INDEX   { ifIndex, natAddrBindLocalAddrType, natAddrBindLocalAddr }
    ::= { natAddrBindTable 1 }

NatAddrBindEntry ::= SEQUENCE {
    natAddrBindLocalAddrType        InetAddressType,
    natAddrBindLocalAddr            InetAddress,
    natAddrBindGlobalAddrType       InetAddressType,
    natAddrBindGlobalAddr           InetAddress,
    natAddrBindId                   NatBindId,
    natAddrBindTranslationEntity    NatTranslationEntity,
    natAddrBindType                 NatAssociationType,
    natAddrBindMapIndex             NatAddrMapId,
    natAddrBindSessions             Gauge32,
    natAddrBindMaxIdleTime          TimeTicks,
    natAddrBindCurrentIdleTime      TimeTicks,
    natAddrBindInTranslates         Counter64,
    natAddrBindOutTranslates        Counter64
}

natAddrBindLocalAddrType OBJECT-TYPE
    SYNTAX      InetAddressType
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "This object specifies the address type used for
             natAddrBindLocalAddr."
    ::= { natAddrBindEntry 1 }

natAddrBindLocalAddr OBJECT-TYPE
    SYNTAX     InetAddress
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
            "This object represents the private-realm specific network
             layer address, which maps to the public-realm address
             represented by natAddrBindGlobalAddr.

             The type of this address is determined by the value of
             the natAddrBindLocalAddrType object."
   ::= { natAddrBindEntry 2 }

natAddrBindGlobalAddrType OBJECT-TYPE
    SYNTAX      InetAddressType
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
            "This object specifies the address type used for
             natAddrBindGlobalAddr."
```

```
    ::= { natAddrBindEntry 3 }

natAddrBindGlobalAddr OBJECT-TYPE
    SYNTAX     InetAddress
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "This object represents the public-realm network layer
             address that maps to the private-realm network layer
             address represented by natAddrBindLocalAddr.

             The type of this address is determined by the value of
             the natAddrBindGlobalAddrType object."
    ::= { natAddrBindEntry 4 }

natAddrBindId OBJECT-TYPE
    SYNTAX     NatBindId
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "This object represents a bind id that is dynamically
             assigned to each bind by a NAT enabled device.  Each
             bind is represented by a bind id that is
             unique across both, the natAddrBindTable and the
             natAddrPortBindTable."
    ::= { natAddrBindEntry 5 }

natAddrBindTranslationEntity OBJECT-TYPE
    SYNTAX     NatTranslationEntity
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "This object represents the direction of sessions
             for which this bind is applicable and the endpoint entity
             (source or destination) within the sessions that is
             subject to translation using the BIND.

             Orientation of the bind can be a superset of
             translationEntity of the address map entry which
             forms the basis for this bind.

             For example, if the translationEntity of an
             address map entry is outboundSrcEndPoint, the
             translationEntity of a bind derived from this
             map entry may either be outboundSrcEndPoint or
             it may be bidirectional (a bitmask of
             outboundSrcEndPoint and inboundDstEndPoint)."
    ::= { natAddrBindEntry 6 }
```

```
natAddrBindType OBJECT-TYPE
    SYNTAX     NatAssociationType
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "This object indicates whether the bind is static or
             dynamic."
    ::= { natAddrBindEntry 7 }

natAddrBindMapIndex OBJECT-TYPE
    SYNTAX     NatAddrMapId
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "This object is a pointer to the natAddrMapTable entry
             (and the parameters of that entry) which was used in
             creating this BIND.  This object, in conjunction with the
             ifIndex (which identifies a unique addrMapName) points to
             a unique entry in the natAddrMapTable."
    ::= { natAddrBindEntry 8 }

natAddrBindSessions OBJECT-TYPE
    SYNTAX     Gauge32
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "Number of sessions currently using this BIND."
    ::= { natAddrBindEntry 9 }

natAddrBindMaxIdleTime OBJECT-TYPE
    SYNTAX     TimeTicks
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "This object indicates the maximum time for
             which this bind can be idle with no sessions
             attached to it.

             The value of this object is of relevance only for
             dynamic NAT."
    ::= { natAddrBindEntry 10 }

natAddrBindCurrentIdleTime OBJECT-TYPE
    SYNTAX     TimeTicks
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "At any given instance, this object indicates the
```

```
              time that this bind has been idle without any sessions
              attached to it.

              The value of this object is of relevance only for
              dynamic NAT."
    ::= { natAddrBindEntry 11 }

natAddrBindInTranslates OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The number of inbound packets that were successfully
             translated by using this bind entry.

             Discontinuities in the value of this counter can occur at
             reinitialization of the management system and at other
             times, as indicated by the value of
             ifCounterDiscontinuityTime on the relevant interface."
    ::= { natAddrBindEntry 12 }

natAddrBindOutTranslates OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The number of outbound packets that were successfully
             translated using this bind entry.

             Discontinuities in the value of this counter can occur at
             reinitialization of the management system and at other
             times as indicated by the value of
             ifCounterDiscontinuityTime on the relevant interface."
    ::= { natAddrBindEntry 13 }

--
-- Address Port Bind section
--

natAddrPortBindNumberOfEntries OBJECT-TYPE
    SYNTAX     Gauge32
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "This object maintains a count of the number of entries
             that currently exist in the natAddrPortBindTable."
    ::= { natMIBObjects 7 }
```

```
--
-- The NAT Address Port Bind Table
--

natAddrPortBindTable OBJECT-TYPE
    SYNTAX     SEQUENCE OF NatAddrPortBindEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
            "This table holds information about the currently
             active NAPT BINDs."
    ::= { natMIBObjects 8 }

natAddrPortBindEntry OBJECT-TYPE
    SYNTAX     NatAddrPortBindEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
            "Each entry in the this table holds information
             about a NAPT bind that is currently active.
             These entries are lost upon agent restart.

             This row has indexing which may create variables with
             more than 128 subidentifiers.  Implementers of this table
             must be careful not to create entries which would result
             in OIDs that exceed the 128 subidentifier limit.
             Otherwise, the information cannot be accessed using
             SNMPv1, SNMPv2c or SNMPv3."
    INDEX   { ifIndex, natAddrPortBindLocalAddrType,
              natAddrPortBindLocalAddr, natAddrPortBindLocalPort,
              natAddrPortBindProtocol }
    ::= { natAddrPortBindTable 1 }

NatAddrPortBindEntry ::= SEQUENCE {
    natAddrPortBindLocalAddrType        InetAddressType,
    natAddrPortBindLocalAddr            InetAddress,
    natAddrPortBindLocalPort            InetPortNumber,
    natAddrPortBindProtocol             NatProtocolType,
    natAddrPortBindGlobalAddrType       InetAddressType,
    natAddrPortBindGlobalAddr           InetAddress,
    natAddrPortBindGlobalPort           InetPortNumber,
    natAddrPortBindId                   NatBindId,
    natAddrPortBindTranslationEntity    NatTranslationEntity,
    natAddrPortBindType                 NatAssociationType,
    natAddrPortBindMapIndex             NatAddrMapId,
    natAddrPortBindSessions             Gauge32,
    natAddrPortBindMaxIdleTime          TimeTicks,
    natAddrPortBindCurrentIdleTime      TimeTicks,
```

```
    natAddrPortBindInTranslates         Counter64,
    natAddrPortBindOutTranslates        Counter64
}

natAddrPortBindLocalAddrType OBJECT-TYPE
    SYNTAX      InetAddressType
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "This object specifies the address type used for
             natAddrPortBindLocalAddr."
    ::= { natAddrPortBindEntry 1 }

natAddrPortBindLocalAddr OBJECT-TYPE
    SYNTAX     InetAddress
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
            "This object represents the private-realm specific network
             layer address which, in conjunction with
             natAddrPortBindLocalPort, maps to the public-realm
             network layer address and transport id represented by
             natAddrPortBindGlobalAddr and natAddrPortBindGlobalPort
             respectively.


             The type of this address is determined by the value of
             the natAddrPortBindLocalAddrType object."
    ::= { natAddrPortBindEntry 2 }

natAddrPortBindLocalPort OBJECT-TYPE
    SYNTAX     InetPortNumber
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
            "For a protocol value TCP or UDP, this object represents
             the private-realm specific port number.  On the other
             hand, for ICMP a bind is created only for query/response
             type ICMP messages such as ICMP echo, Timestamp, and
             Information request messages, and this object represents
             the private-realm specific identifier in the ICMP
             message, as defined in RFC 792 for ICMPv4 and in RFC
             2463 for ICMPv6.

             This object, together with natAddrPortBindProtocol,
             natAddrPortBindLocalAddrType, and natAddrPortBindLocalAddr,
             constitutes a session endpoint in the private realm.  A
             bind entry binds a private realm specific endpoint to a
```

```
             public realm specific endpoint, as represented by the
             tuple of (natAddrPortBindGlobalPort,
             natAddrPortBindProtocol, natAddrPortBindGlobalAddrType,
             and natAddrPortBindGlobalAddr)."
   ::= { natAddrPortBindEntry 3 }

natAddrPortBindProtocol OBJECT-TYPE
    SYNTAX      NatProtocolType
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "This object specifies a protocol identifier.  If the
             value of this object is none(1), then this bind entry
             applies to all IP traffic.  Any other value of this object
             specifies the class of IP traffic to which this BIND
             applies."
    ::= { natAddrPortBindEntry 4 }

natAddrPortBindGlobalAddrType OBJECT-TYPE
    SYNTAX      InetAddressType
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
            "This object specifies the address type used for
             natAddrPortBindGlobalAddr."
    ::= { natAddrPortBindEntry 5 }

natAddrPortBindGlobalAddr OBJECT-TYPE
    SYNTAX     InetAddress
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "This object represents the public-realm specific network
             layer address that, in conjunction with
             natAddrPortBindGlobalPort, maps to the private-realm

             network layer address and transport id represented by
             natAddrPortBindLocalAddr and natAddrPortBindLocalPort,
             respectively.

             The type of this address is determined by the value of
             the natAddrPortBindGlobalAddrType object."
    ::= { natAddrPortBindEntry 6 }

natAddrPortBindGlobalPort OBJECT-TYPE
    SYNTAX     InetPortNumber
    MAX-ACCESS read-only
    STATUS     current
```

```
    DESCRIPTION
            "For a protocol value TCP or UDP, this object represents
             the public-realm specific port number.  On the other
             hand, for ICMP a bind is created only for query/response
             type ICMP messages such as ICMP echo, Timestamp, and
             Information request messages, and this object represents
             the public-realm specific identifier in the ICMP message,
             as defined in RFC 792 for ICMPv4 and in RFC 2463 for
             ICMPv6.

             This object, together with natAddrPortBindProtocol,
             natAddrPortBindGlobalAddrType, and
             natAddrPortBindGlobalAddr, constitutes a session endpoint
             in the public realm.  A bind entry binds a public realm
             specific endpoint to a private realm specific endpoint,
             as represented by the tuple of
              (natAddrPortBindLocalPort, natAddrPortBindProtocol,
               natAddrPortBindLocalAddrType, and
               natAddrPortBindLocalAddr)."
    ::= { natAddrPortBindEntry 7 }

natAddrPortBindId OBJECT-TYPE
    SYNTAX     NatBindId
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "This object represents a bind id that is dynamically
             assigned to each bind by a NAT enabled device.  Each
             bind is represented by a unique bind id across both
             the natAddrBindTable and the natAddrPortBindTable."
    ::= { natAddrPortBindEntry 8 }

natAddrPortBindTranslationEntity OBJECT-TYPE
    SYNTAX     NatTranslationEntity
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "This object represents the direction of sessions
             for which this bind is applicable and the entity
             (source or destination) within the sessions that is
             subject to translation with the BIND.

             Orientation of the bind can be a superset of the
             translationEntity of the address map entry that
             forms the basis for this bind.

             For example, if the translationEntity of an
             address map entry is outboundSrcEndPoint, the
```

```
             translationEntity of a bind derived from this
             map entry may either be outboundSrcEndPoint or
             may be bidirectional (a bitmask of
             outboundSrcEndPoint and inboundDstEndPoint)."
    ::= { natAddrPortBindEntry 9 }

natAddrPortBindType OBJECT-TYPE
    SYNTAX     NatAssociationType
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "This object indicates whether the bind is static or
             dynamic."
    ::= { natAddrPortBindEntry 10 }

natAddrPortBindMapIndex OBJECT-TYPE
    SYNTAX     NatAddrMapId
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "This object is a pointer to the natAddrMapTable entry
             (and the parameters of that entry) used in
             creating this BIND.  This object, in conjunction with the
             ifIndex (which identifies a unique addrMapName), points
             to a unique entry in the natAddrMapTable."
    ::= { natAddrPortBindEntry 11 }

natAddrPortBindSessions OBJECT-TYPE
    SYNTAX     Gauge32
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "Number of sessions currently using this BIND."
    ::= { natAddrPortBindEntry 12 }

natAddrPortBindMaxIdleTime OBJECT-TYPE
    SYNTAX     TimeTicks
    MAX-ACCESS read-only
    STATUS     current

    DESCRIPTION
            "This object indicates the maximum time for
             which this bind can be idle without any sessions
             attached to it.
             The value of this object is of relevance
             only for dynamic NAT."
    ::= { natAddrPortBindEntry 13 }
```

```
natAddrPortBindCurrentIdleTime OBJECT-TYPE
    SYNTAX     TimeTicks
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "At any given instance, this object indicates the
             time that this bind has been idle without any sessions
             attached to it.

             The value of this object is of relevance
             only for dynamic NAT."
    ::= { natAddrPortBindEntry 14 }

natAddrPortBindInTranslates OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The number of inbound packets that were translated as per
             this bind entry.

             Discontinuities in the value of this counter can occur at
             reinitialization of the management system and at other
             times, as indicated by the value of
             ifCounterDiscontinuityTime on the relevant interface."
    ::= { natAddrPortBindEntry 15 }

natAddrPortBindOutTranslates OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The number of outbound packets that were translated as per
             this bind entry.

             Discontinuities in the value of this counter can occur at
             reinitialization of the management system and at other
             times, as indicated by the value of
             ifCounterDiscontinuityTime on the relevant interface."
    ::= { natAddrPortBindEntry 16 }

--
-- The Session Table
--

natSessionTable OBJECT-TYPE
    SYNTAX     SEQUENCE OF NatSessionEntry
    MAX-ACCESS not-accessible
```

```
    STATUS     current
    DESCRIPTION
            "The (conceptual) table containing one entry for each
             NAT session currently active on this NAT device."
    ::= { natMIBObjects 9 }

natSessionEntry OBJECT-TYPE
    SYNTAX     NatSessionEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
            "An entry (conceptual row) containing information
             about an active NAT session on this NAT device.
             These entries are lost upon agent restart."
    INDEX   { ifIndex, natSessionIndex }
    ::= { natSessionTable 1 }

NatSessionEntry ::= SEQUENCE {
    natSessionIndex                     NatSessionId,
    natSessionPrivateSrcEPBindId        NatBindIdOrZero,
    natSessionPrivateSrcEPBindMode      NatBindMode,
    natSessionPrivateDstEPBindId        NatBindIdOrZero,
    natSessionPrivateDstEPBindMode      NatBindMode,
    natSessionDirection                 INTEGER,
    natSessionUpTime                    TimeTicks,
    natSessionAddrMapIndex              NatAddrMapId,
    natSessionProtocolType              NatProtocolType,
    natSessionPrivateAddrType           InetAddressType,
    natSessionPrivateSrcAddr            InetAddress,
    natSessionPrivateSrcPort            InetPortNumber,
    natSessionPrivateDstAddr            InetAddress,
    natSessionPrivateDstPort            InetPortNumber,
    natSessionPublicAddrType            InetAddressType,
    natSessionPublicSrcAddr             InetAddress,
    natSessionPublicSrcPort             InetPortNumber,
    natSessionPublicDstAddr             InetAddress,
    natSessionPublicDstPort             InetPortNumber,
    natSessionMaxIdleTime               TimeTicks,
    natSessionCurrentIdleTime           TimeTicks,
    natSessionInTranslates              Counter64,
    natSessionOutTranslates             Counter64
}

natSessionIndex OBJECT-TYPE
    SYNTAX     NatSessionId
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
```

```
            "The session ID for this NAT session."
    ::= { natSessionEntry 1 }

natSessionPrivateSrcEPBindId OBJECT-TYPE
    SYNTAX     NatBindIdOrZero
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The bind id associated between private and public
             source end points.  In the case of Symmetric-NAT,
             this should be set to zero."
    ::= { natSessionEntry 2 }

natSessionPrivateSrcEPBindMode OBJECT-TYPE
    SYNTAX     NatBindMode
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "This object indicates whether the bind indicated
             by the object natSessionPrivateSrcEPBindId
             is an address bind or an address port bind."
    ::= { natSessionEntry 3 }

natSessionPrivateDstEPBindId OBJECT-TYPE
    SYNTAX     NatBindIdOrZero
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The bind id associated between private and public
             destination end points."
    ::= { natSessionEntry 4 }

natSessionPrivateDstEPBindMode OBJECT-TYPE
    SYNTAX     NatBindMode
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "This object indicates whether the bind indicated
             by the object natSessionPrivateDstEPBindId
             is an address bind or an address port bind."
    ::= { natSessionEntry 5 }

natSessionDirection OBJECT-TYPE
    SYNTAX     INTEGER {
                   inbound (1),
                   outbound (2)
               }
```

```
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The direction of this session with respect to the
             local network.  'inbound' indicates that this session
             was initiated from the public network into the private
             network.  'outbound' indicates that this session was
             initiated from the private network into the public
             network."
    ::= { natSessionEntry 6 }

natSessionUpTime OBJECT-TYPE
    SYNTAX     TimeTicks
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The up time of this session in one-hundredths of a
             second."
    ::= { natSessionEntry 7 }

natSessionAddrMapIndex OBJECT-TYPE
    SYNTAX     NatAddrMapId
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "This object is a pointer to the natAddrMapTable entry
             (and the parameters of that entry) used in
             creating this session.  This object, in conjunction with
             the ifIndex (which identifies a unique addrMapName), points
             to a unique entry in the natAddrMapTable."
    ::= { natSessionEntry 8 }

natSessionProtocolType OBJECT-TYPE
    SYNTAX     NatProtocolType
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The protocol type of this session."
    ::= { natSessionEntry 9 }

natSessionPrivateAddrType OBJECT-TYPE
    SYNTAX      InetAddressType
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
            "This object specifies the address type used for
             natSessionPrivateSrcAddr and natSessionPrivateDstAddr."
    ::= { natSessionEntry 10 }
```

```
natSessionPrivateSrcAddr OBJECT-TYPE
    SYNTAX     InetAddress
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The source IP address of the session endpoint that
             lies in the private network.

             The value of this object must be zero only when the
             natSessionPrivateSrcEPBindId object has a zero value.
             When the value of this object is zero, the NAT session
             lookup will match any IP address to this field.

             The type of this address is determined by the value of
             the natSessionPrivateAddrType object."
    ::= { natSessionEntry 11 }

natSessionPrivateSrcPort OBJECT-TYPE
    SYNTAX     InetPortNumber
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "When the value of protocol is TCP or UDP, this object
             represents the source port in the first packet of session
             while in private-realm.  On the other hand, when the
             protocol is ICMP, a NAT session is created only for
             query/response type ICMP messages such as ICMP echo,
             Timestamp, and Information request messages, and this
             object represents the private-realm specific identifier
             in the ICMP message, as defined in RFC 792 for ICMPv4
             and in RFC 2463 for ICMPv6.

             The value of this object must be zero when the
             natSessionPrivateSrcEPBindId object has zero value
             and value of natSessionPrivateSrcEPBindMode is
             addressPortBind(2).  In such a case, the NAT session
             lookup will match any port number to this field.

             The value of this object must be zero when the object
             is not a representative field (SrcPort, DstPort, or
             ICMP identifier) of the session tuple in either the
             public realm or the private realm."
    ::= { natSessionEntry 12 }

natSessionPrivateDstAddr OBJECT-TYPE
    SYNTAX     InetAddress
    MAX-ACCESS read-only
    STATUS     current
```

```
    DESCRIPTION
            "The destination IP address of the session endpoint that
             lies in the private network.

             The value of this object must be zero when the
             natSessionPrivateDstEPBindId object has a zero value.
             In such a scenario, the NAT session lookup will match
             any IP address to this field.

             The type of this address is determined by the value of
             the natSessionPrivateAddrType object."
    ::= { natSessionEntry 13 }

natSessionPrivateDstPort OBJECT-TYPE
    SYNTAX     InetPortNumber
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "When the value of protocol is TCP or UDP, this object
             represents the destination port in the first packet
             of session while in private-realm.  On the other hand,
             when the protocol is ICMP, this object is not relevant
             and should be set to zero.

             The value of this object must be zero when the
             natSessionPrivateDstEPBindId object has a zero
             value and natSessionPrivateDstEPBindMode is set to
             addressPortBind(2).  In such a case, the NAT session
             lookup will match any port number to this field.

             The value of this object must be zero when the object
             is not a representative field (SrcPort, DstPort, or
             ICMP identifier) of the session tuple in either the
             public realm or the private realm."
    ::= { natSessionEntry 14 }

natSessionPublicAddrType OBJECT-TYPE
    SYNTAX      InetAddressType
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
            "This object specifies the address type used for
             natSessionPublicSrcAddr and natSessionPublicDstAddr."
    ::= { natSessionEntry 15 }

natSessionPublicSrcAddr OBJECT-TYPE
    SYNTAX     InetAddress
    MAX-ACCESS read-only
```

```
    STATUS     current
    DESCRIPTION
            "The source IP address of the session endpoint that
             lies in the public network.

             The value of this object must be zero when the
             natSessionPrivateSrcEPBindId object has a zero value.
             In such a scenario, the NAT session lookup will match
             any IP address to this field.

             The type of this address is determined by the value of
             the natSessionPublicAddrType object."
    ::= { natSessionEntry 16 }

natSessionPublicSrcPort OBJECT-TYPE
    SYNTAX     InetPortNumber
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "When the value of protocol is TCP or UDP, this object
             represents the source port in the first packet of
             session while in public-realm.  On the other hand, when
             protocol is ICMP, a NAT session is created only for
             query/response type ICMP messages such as ICMP echo,
             Timestamp, and Information request messages, and this
             object represents the public-realm specific identifier
             in the ICMP message, as defined in RFC 792 for ICMPv4
             and in RFC 2463 for ICMPv6.

             The value of this object must be zero when the
             natSessionPrivateSrcEPBindId object has a zero value
             and natSessionPrivateSrcEPBindMode is set to
             addressPortBind(2).  In such a scenario, the NAT
             session lookup will match any port number to this
             field.

             The value of this object must be zero when the object
             is not a representative field (SrcPort, DstPort or
             ICMP identifier) of the session tuple in either the
             public realm or the private realm."
    ::= { natSessionEntry 17 }

natSessionPublicDstAddr OBJECT-TYPE
    SYNTAX     InetAddress
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The destination IP address of the session endpoint that
```

```
             lies in the public network.

             The value of this object must be non-zero when the
             natSessionPrivateDstEPBindId object has a non-zero
             value.  If the value of this object and the
             corresponding natSessionPrivateDstEPBindId object value
             is zero, then the NAT session lookup will match any IP
             address to this field.

             The type of this address is determined by the value of
             the natSessionPublicAddrType object."
    ::= { natSessionEntry 18 }

natSessionPublicDstPort OBJECT-TYPE
    SYNTAX     InetPortNumber
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "When the value of protocol is TCP or UDP, this object
             represents the destination port in the first packet of
             session while in public-realm.  On the other hand, when
             the protocol is ICMP, this object is not relevant for
             translation and should be zero.

             The value of this object must be zero when the
             natSessionPrivateDstEPBindId object has a zero value
             and natSessionPrivateDstEPBindMode is
             addressPortBind(2).  In such a scenario, the NAT
             session lookup will match any port number to this
             field.

             The value of this object must be zero when the object
             is not a representative field (SrcPort, DstPort, or
             ICMP identifier) of the session tuple in either the
             public realm or the private realm."
    ::= { natSessionEntry 19 }

natSessionMaxIdleTime OBJECT-TYPE
    SYNTAX     TimeTicks
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The max time for which this session can be idle
             without detecting a packet."
    ::= { natSessionEntry 20 }

natSessionCurrentIdleTime OBJECT-TYPE
    SYNTAX     TimeTicks
```

```
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The time since a packet belonging to this session was
            last detected."
    ::= { natSessionEntry 21 }

natSessionInTranslates OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The number of inbound packets that were translated for
             this session.

             Discontinuities in the value of this counter can occur at
             reinitialization of the management system and at other
             times, as indicated by the value of
             ifCounterDiscontinuityTime on the relevant interface."
    ::= { natSessionEntry 22 }

natSessionOutTranslates OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The number of outbound packets that were translated for
             this session.

             Discontinuities in the value of this counter can occur at
             reinitialization of the management system and at other
             times, as indicated by the value of
             ifCounterDiscontinuityTime on the relevant interface."
    ::= { natSessionEntry 23 }

--
-- The Protocol table
--

natProtocolTable OBJECT-TYPE
    SYNTAX     SEQUENCE OF NatProtocolEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
            "The (conceptual) table containing per protocol NAT
             statistics."
    ::= { natMIBObjects 10 }
```

```
natProtocolEntry OBJECT-TYPE
    SYNTAX     NatProtocolEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
            "An entry (conceptual row) containing NAT statistics
             pertaining to a particular protocol."
    INDEX   { natProtocol }
    ::= { natProtocolTable 1 }

NatProtocolEntry ::= SEQUENCE {
    natProtocol                 NatProtocolType,
    natProtocolInTranslates     Counter64,
    natProtocolOutTranslates    Counter64,
    natProtocolDiscards         Counter64
}

natProtocol    OBJECT-TYPE
    SYNTAX     NatProtocolType
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
            "This object represents the protocol pertaining to which
             parameters are reported."
    ::= { natProtocolEntry 1 }

natProtocolInTranslates OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The number of inbound packets pertaining to the protocol
             identified by natProtocol that underwent NAT.

             Discontinuities in the value of this counter can occur at
             reinitialization of the management system and at other
             times, as indicated by the value of
             ifCounterDiscontinuityTime on the relevant interface."
    ::= { natProtocolEntry 2 }

natProtocolOutTranslates OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The number of outbound packets pertaining to the protocol
             identified by natProtocol that underwent NAT.
```

```
             Discontinuities in the value of this counter can occur at
             reinitialization of the management system and at other
             times, as indicated by the value of
             ifCounterDiscontinuityTime on the relevant interface."
    ::= { natProtocolEntry 3 }

natProtocolDiscards OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The number of packets pertaining to the protocol
             identified by natProtocol that had to be
             rejected/dropped due to lack of resources.  These
             rejections could be due to session timeout, resource
             unavailability, lack of address space, etc.

             Discontinuities in the value of this counter can occur at
             reinitialization of the management system and at other
             times, as indicated by the value of
             ifCounterDiscontinuityTime on the relevant interface."
     ::= { natProtocolEntry 4 }


--
-- The Shared Address Map Table
--

natSharedAddrMapTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF NatSharedAddrMapEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "This table lists address map parameters for NAT."
    ::= { natMIBObjects 11 }

natSharedAddrMapEntry OBJECT-TYPE
    SYNTAX      NatSharedAddrMapEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "This entry represents an address map to be used for
             NAT and contributes to the dynamic and/or static
             address mapping tables of the NAT device."
    INDEX   { natSharedAddrMapIndex }
    ::= { natSharedAddrMapTable 1 }

NatSharedAddrMapEntry ::= SEQUENCE {
```

```
    natSharedAddrMapIndex               NatSharedAddrMapId,
    natSharedAddrMapName                SnmpAdminString,
    natSharedAddrMapEntryType           NatAssociationType,
    natSharedAddrMapTranslatEntity      NatTranslationEntity,
    natSharedAddrMapLocalAddrType       InetAddressType,
    natSharedAddrMapLocalAddrFrom       InetAddress,
    natSharedAddrMapLocalAddrTo         InetAddress,
    natSharedAddrMapLocalPortFrom       InetPortNumber,
    natSharedAddrMapLocalPortTo         InetPortNumber,
    natSharedAddrMapGlobalAddrType      InetAddressType,
    natSharedAddrMapGlobalAddrFrom      InetAddress,
    natSharedAddrMapGlobalAddrTo        InetAddress,
    natSharedAddrMapGlobalPortFrom      InetPortNumber,
    natSharedAddrMapGlobalPortTo        InetPortNumber,
    natSharedAddrMapProtocol            NatProtocolMap,
    natSharedAddrMapInTranslates        Counter64,
    natSharedAddrMapOutTranslates       Counter64,
    natSharedAddrMapDiscards            Counter64,
    natSharedAddrMapAddrUsed            Gauge32,
    natSharedAddrMapStorageType         StorageType,
    natSharedAddrMapRowStatus           RowStatus
}

natSharedAddrMapIndex  OBJECT-TYPE
    SYNTAX      NatSharedAddrMapId
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "Along with ifIndex, this object uniquely
             identifies an entry in the natAddrMapTable.
             Address map entries are applied in the order
             specified by natAddrMapIndex."
    ::= { natSharedAddrMapEntry 1 }

natSharedAddrMapName OBJECT-TYPE
    SYNTAX      SnmpAdminString (SIZE(1..32))
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "Name identifying all map entries in the table associated
             with the same interface.  All map entries with the same
             ifIndex MUST have the same map name."
    ::= { natSharedAddrMapEntry 2 }

natSharedAddrMapEntryType OBJECT-TYPE
    SYNTAX      NatAssociationType
    MAX-ACCESS  read-create
    STATUS      current
```

```
    DESCRIPTION
            "This parameter can be used to set up static
             or dynamic address maps."
    ::= { natSharedAddrMapEntry 3 }

natSharedAddrMapTranslatEntity OBJECT-TYPE
    SYNTAX      NatTranslationEntity
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "The end-point entity (source or destination) in
             inbound or outbound sessions (i.e., first packets) that
             may be translated by an address map entry.

             Session direction (inbound or outbound) is
             derived from the direction of the first packet
             of a session traversing a NAT interface.
             NAT address (and Transport-ID) maps may be defined
             to effect inbound or outbound sessions.

             Traditionally, address maps for Basic NAT and NAPT are
             configured on a public interface for outbound sessions,
             effecting translation of source end-point.  The value of
             this object must be set to outboundSrcEndPoint for
             those interfaces.

             Alternately, if address maps for Basic NAT and NAPT were
             to be configured on a private interface, the desired
             value for this object for the map entries
             would be inboundSrcEndPoint (i.e., effecting translation
             of source end-point for inbound sessions).

             If TwiceNAT were to be configured on a private interface,
             the desired value for this object for the map entries
             would be a bitmask of inboundSrcEndPoint and
             inboundDstEndPoint."
    ::= { natSharedAddrMapEntry 4 }

natSharedAddrMapLocalAddrType OBJECT-TYPE
    SYNTAX      InetAddressType
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "This object specifies the address type used for
             natAddrMapLocalAddrFrom and natAddrMapLocalAddrTo."
    ::= { natSharedAddrMapEntry 5 }

natSharedAddrMapLocalAddrFrom OBJECT-TYPE
```

```
    SYNTAX      InetAddress
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "This object specifies the first IP address of the range
             of IP addresses mapped by this translation entry.  The
             value of this object must be less than or equal to the
             value of the natAddrMapLocalAddrTo object.

             The type of this address is determined by the value of
             the natAddrMapLocalAddrType object."
    ::= { natSharedAddrMapEntry 6 }

natSharedAddrMapLocalAddrTo OBJECT-TYPE
    SYNTAX      InetAddress
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "This object specifies the last IP address of the range of
             IP addresses mapped by this translation entry.  If only
             a single address is being mapped, the value of this object
             is equal to the value of natAddrMapLocalAddrFrom.  For a
             static NAT, the number of addresses in the range defined
             by natAddrMapLocalAddrFrom and natAddrMapLocalAddrTo must
             be equal to the number of addresses in the range defined by
             natAddrMapGlobalAddrFrom and natAddrMapGlobalAddrTo.
             The value of this object must be greater than or equal to
             the value of the natAddrMapLocalAddrFrom object.

             The type of this address is determined by the value of
             the natAddrMapLocalAddrType object."
    ::= { natSharedAddrMapEntry 7 }

natSharedAddrMapLocalPortFrom OBJECT-TYPE
    SYNTAX      InetPortNumber
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "If this conceptual row describes a Basic NAT address
             mapping, then the value of this object must be zero.  If
             this conceptual row describes NAPT, then the value of
             this object specifies the first port number in the range
             of ports being mapped.

             The value of this object must be less than or equal to the
             value of the natAddrMapLocalPortTo object.  If the
             translation specifies a single port, then the value of this
             object is equal to the value of natAddrMapLocalPortTo."
```

```
    DEFVAL { 0 }
    ::= { natSharedAddrMapEntry 8 }

natSharedAddrMapLocalPortTo OBJECT-TYPE
    SYNTAX      InetPortNumber
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "If this conceptual row describes a Basic NAT address
             mapping, then the value of this object must be zero.  If
             this conceptual row describes NAPT, then the value of
             this object specifies the last port number in the range
             of ports being mapped.

             The value of this object must be greater than or equal to
             the value of the natAddrMapLocalPortFrom object.  If the
             translation specifies a single port, then the value of this
             object is equal to the value of natAddrMapLocalPortFrom."
    DEFVAL { 0 }
    ::= { natSharedAddrMapEntry 9 }

natSharedAddrMapGlobalAddrType OBJECT-TYPE
    SYNTAX      InetAddressType
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "This object specifies the address type used for
             natAddrMapGlobalAddrFrom and natAddrMapGlobalAddrTo."
    ::= { natSharedAddrMapEntry 10 }

natSharedAddrMapGlobalAddrFrom OBJECT-TYPE
    SYNTAX      InetAddress
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "This object specifies the first IP address of the range of
             IP addresses being mapped to.  The value of this object
             must be less than or equal to the value of the
             natAddrMapGlobalAddrTo object.

             The type of this address is determined by the value of
             the natAddrMapGlobalAddrType object."
    ::= { natSharedAddrMapEntry 11 }

natSharedAddrMapGlobalAddrTo OBJECT-TYPE
    SYNTAX      InetAddress
    MAX-ACCESS  read-create
    STATUS      current
```

```
    DESCRIPTION
            "This object specifies the last IP address of the range of
             IP addresses being mapped to.  If only a single address is
             being mapped to, the value of this object is equal to the
             value of natAddrMapGlobalAddrFrom.  For a static NAT, the
             number of addresses in the range defined by
             natAddrMapGlobalAddrFrom and natAddrMapGlobalAddrTo must be
             equal to the number of addresses in the range defined by
             natAddrMapLocalAddrFrom and natAddrMapLocalAddrTo.
             The value of this object must be greater than or equal to
             the value of the natAddrMapGlobalAddrFrom object.

             The type of this address is determined by the value of
             the natAddrMapGlobalAddrType object."
    ::= { natSharedAddrMapEntry 12 }

natSharedAddrMapGlobalPortFrom OBJECT-TYPE
    SYNTAX      InetPortNumber
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "If this conceptual row describes a Basic NAT address
             mapping, then the value of this object must be zero.  If
             this conceptual row describes NAPT, then the value of
             this object specifies the first port number in the range
             of ports being mapped to.


             The value of this object must be less than or equal to the
             value of the natAddrMapGlobalPortTo object.  If the
             translation specifies a single port, then the value of this
             object is equal to the value natAddrMapGlobalPortTo."
    DEFVAL { 0 }
    ::= { natSharedAddrMapEntry 13 }

natSharedAddrMapGlobalPortTo OBJECT-TYPE
    SYNTAX      InetPortNumber
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "If this conceptual row describes a Basic NAT address
             mapping, then the value of this object must be zero.  If
             this conceptual row describes NAPT, then the value of this
             object specifies the last port number in the range of
             ports being mapped to.

             The value of this object must be greater than or equal to
             the value of the natAddrMapGlobalPortFrom object.  If the
```

```
             translation specifies a single port, then the value of this
             object is equal to the value of natAddrMapGlobalPortFrom."
    DEFVAL { 0 }
    ::= { natSharedAddrMapEntry 14 }

natSharedAddrMapProtocol OBJECT-TYPE
    SYNTAX      NatProtocolMap
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "This object specifies a bitmap of protocol identifiers."
    ::= { natSharedAddrMapEntry 15 }

natSharedAddrMapInTranslates OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The number of inbound packets pertaining to this address
             map entry that were translated.

             Discontinuities in the value of this counter can occur at
             reinitialization of the management system and at other
             times, as indicated by the value of
             ifCounterDiscontinuityTime on the relevant interface."
    ::= { natSharedAddrMapEntry 16 }

natSharedAddrMapOutTranslates OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The number of outbound packets pertaining to this
             address map entry that were translated.

             Discontinuities in the value of this counter can occur at
             reinitialization of the management system and at other
             times, as indicated by the value of
             ifCounterDiscontinuityTime on the relevant interface."
    ::= { natSharedAddrMapEntry 17 }

natSharedAddrMapDiscards OBJECT-TYPE
    SYNTAX     Counter64
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The number of packets pertaining to this address map
             entry that were dropped due to lack of addresses in the
```

```
            address pool identified by this address map.  The value of
            this object must always be zero in case of static
            address map.

            Discontinuities in the value of this counter can occur at
            reinitialization of the management system and at other
            times, as indicated by the value of
            ifCounterDiscontinuityTime on the relevant interface."
    ::= { natSharedAddrMapEntry 18 }

natSharedAddrMapAddrUsed OBJECT-TYPE
    SYNTAX     Gauge32
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The number of addresses pertaining to this address map
             that are currently being used from the NAT pool.
             The value of this object must always be zero in the case
             of a static address map."
    ::= { natSharedAddrMapEntry 19 }

natSharedAddrMapStorageType OBJECT-TYPE
    SYNTAX      StorageType
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "The storage type for this conceptual row.
             Conceptual rows having the value 'permanent'
             need not allow write-access to any columnar objects
             in the row."
    REFERENCE
            "Textual Conventions for SMIv2, Section 2."
    DEFVAL { nonVolatile }
    ::= { natSharedAddrMapEntry 20 }

natSharedAddrMapRowStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
            "The status of this conceptual row.

             Until instances of all corresponding columns are
             appropriately configured, the value of the
             corresponding instance of the natAddrMapRowStatus
             column is 'notReady'.

             None of the objects in this row may be modified
```

```
             while the value of this object is active(1)."
    REFERENCE
            "Textual Conventions for SMIv2, Section 2."
    ::= { natSharedAddrMapEntry 21 }


--
-- Notifications section
--

natMIBNotifications OBJECT IDENTIFIER ::= { natMIB 0 }

--
-- Notifications
--

natPacketDiscard NOTIFICATION-TYPE
    OBJECTS { ifIndex }
    STATUS  current
    DESCRIPTION
            "This notification is generated when IP packets are
             discarded by the NAT function; e.g., due to lack of
             mapping space when NAT is out of addresses or ports.

             Note that the generation of natPacketDiscard
             notifications is throttled by the agent, as specified
             by the 'natNotifThrottlingInterval' object."
    ::= { natMIBNotifications 1 }

--
-- Conformance information.
--

natMIBConformance OBJECT IDENTIFIER ::= { natMIB 2 }

natMIBGroups      OBJECT IDENTIFIER ::= { natMIBConformance 1 }
natMIBCompliances OBJECT IDENTIFIER ::= { natMIBConformance 2 }

--
-- Units of conformance
--

natConfigGroup OBJECT-GROUP
    OBJECTS { natInterfaceRealm,
              natInterfaceServiceType,
              natInterfaceStorageType,
              natInterfaceRowStatus,
              natAddrMapName,
```

```
              natAddrMapEntryType,
              natAddrMapTranslationEntity,
              natAddrMapLocalAddrType,
              natAddrMapLocalAddrFrom,
              natAddrMapLocalAddrTo,
              natAddrMapLocalPortFrom,
              natAddrMapLocalPortTo,
              natAddrMapGlobalAddrType,
              natAddrMapGlobalAddrFrom,
              natAddrMapGlobalAddrTo,
              natAddrMapGlobalPortFrom,
              natAddrMapGlobalPortTo,
              natAddrMapProtocol,
              natAddrMapStorageType,
              natAddrMapRowStatus,
              natSharedAddrMapName,
              natSharedAddrMapEntryType,
              natSharedAddrMapTranslatEntity,
              natSharedAddrMapLocalAddrType,
              natSharedAddrMapLocalAddrFrom,
              natSharedAddrMapLocalAddrTo,
              natSharedAddrMapLocalPortFrom,
              natSharedAddrMapLocalPortTo,
              natSharedAddrMapGlobalAddrType,
              natSharedAddrMapGlobalAddrFrom,
              natSharedAddrMapGlobalAddrTo,
              natSharedAddrMapGlobalPortFrom,
              natSharedAddrMapGlobalPortTo,
              natSharedAddrMapProtocol,
              natSharedAddrMapStorageType,
              natSharedAddrMapRowStatus,
              natBindDefIdleTimeout,
              natUdpDefIdleTimeout,
              natIcmpDefIdleTimeout,
              natOtherDefIdleTimeout,
              natTcpDefIdleTimeout,
              natTcpDefNegTimeout,
              natNotifThrottlingInterval }
    STATUS  current
    DESCRIPTION
            "A collection of configuration-related information
             required to support management of devices supporting
             NAT."
    ::= { natMIBGroups 1 }

natTranslationGroup OBJECT-GROUP
    OBJECTS { natAddrBindNumberOfEntries,
              natAddrBindGlobalAddrType,
```

```
              natAddrBindGlobalAddr,
              natAddrBindId,
              natAddrBindTranslationEntity,
              natAddrBindType,
              natAddrBindMapIndex,
              natAddrBindSessions,
              natAddrBindMaxIdleTime,
              natAddrBindCurrentIdleTime,
              natAddrBindInTranslates,
              natAddrBindOutTranslates,
              natAddrPortBindNumberOfEntries,
              natAddrPortBindGlobalAddrType,
              natAddrPortBindGlobalAddr,
              natAddrPortBindGlobalPort,
              natAddrPortBindId,
              natAddrPortBindTranslationEntity,
              natAddrPortBindType,
              natAddrPortBindMapIndex,
              natAddrPortBindSessions,
              natAddrPortBindMaxIdleTime,
              natAddrPortBindCurrentIdleTime,
              natAddrPortBindInTranslates,
              natAddrPortBindOutTranslates,
              natSessionPrivateSrcEPBindId,
              natSessionPrivateSrcEPBindMode,
              natSessionPrivateDstEPBindId,
              natSessionPrivateDstEPBindMode,
              natSessionDirection,
              natSessionUpTime,
              natSessionAddrMapIndex,
              natSessionProtocolType,
              natSessionPrivateAddrType,
              natSessionPrivateSrcAddr,
              natSessionPrivateSrcPort,
              natSessionPrivateDstAddr,
              natSessionPrivateDstPort,
              natSessionPublicAddrType,
              natSessionPublicSrcAddr,
              natSessionPublicSrcPort,
              natSessionPublicDstAddr,
              natSessionPublicDstPort,
              natSessionMaxIdleTime,
              natSessionCurrentIdleTime,
              natSessionInTranslates,
              natSessionOutTranslates }
    STATUS  current

    DESCRIPTION
```

```
            "A collection of BIND-related objects required to support
             management of devices supporting NAT."
    ::= { natMIBGroups 2 }

natStatsInterfaceGroup OBJECT-GROUP
    OBJECTS { natInterfaceInTranslates,
              natInterfaceOutTranslates,
              natInterfaceDiscards }
    STATUS  current
    DESCRIPTION
            "A collection of NAT statistics associated with the
             interface on which NAT is configured, to aid
             troubleshooting/monitoring of the NAT operation."
    ::= { natMIBGroups 3 }

natStatsProtocolGroup OBJECT-GROUP
    OBJECTS { natProtocolInTranslates,
              natProtocolOutTranslates,
              natProtocolDiscards }
    STATUS  current
    DESCRIPTION
            "A collection of protocol specific NAT statistics,
             to aid troubleshooting/monitoring of NAT operation."
    ::= { natMIBGroups 4 }

natStatsAddrMapGroup OBJECT-GROUP
    OBJECTS { natAddrMapInTranslates,
              natAddrMapOutTranslates,
              natAddrMapDiscards,
              natAddrMapAddrUsed,
              natSharedAddrMapInTranslates,
              natSharedAddrMapOutTranslates,
              natSharedAddrMapDiscards,
              natSharedAddrMapAddrUsed }
    STATUS  current
    DESCRIPTION
            "A collection of address map specific NAT statistics,
             to aid troubleshooting/monitoring of NAT operation."
    ::= { natMIBGroups 5 }

natMIBNotificationGroup NOTIFICATION-GROUP
    NOTIFICATIONS { natPacketDiscard }
    STATUS        current
    DESCRIPTION
            "A collection of notifications generated by
            devices supporting this MIB."
    ::= { natMIBGroups 6 }
```

```
--
-- Compliance statements
--

natMIBFullCompliance MODULE-COMPLIANCE
    STATUS  current
    DESCRIPTION
            "When this MIB is implemented with support for
             read-create, then such an implementation can claim
             full compliance.  Such devices can then be both
             monitored and configured with this MIB.

             The following index objects cannot be added as OBJECT
             clauses but nevertheless have the compliance
             requirements:
                "
             -- OBJECT  natAddrBindLocalAddrType
             -- SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
             -- DESCRIPTION
             --         "An implementation is required to support
             --          global IPv4 and/or IPv6 addresses, depending
             --          on its support for IPv4 and IPv6."

             -- OBJECT  natAddrBindLocalAddr
             -- SYNTAX  InetAddress (SIZE(4|16))
             -- DESCRIPTION
             --         "An implementation is required to support
             --          global IPv4 and/or IPv6 addresses, depending
             --          on its support for IPv4 and IPv6."

             -- OBJECT  natAddrPortBindLocalAddrType
             -- SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
             -- DESCRIPTION
             --         "An implementation is required to support
             --          global IPv4 and/or IPv6 addresses, depending
             --          on its support for IPv4 and IPv6."

             -- OBJECT  natAddrPortBindLocalAddr
             -- SYNTAX  InetAddress (SIZE(4|16))
             -- DESCRIPTION
             --         "An implementation is required to support
             --          global IPv4 and/or IPv6 addresses, depending
             --          on its support for IPv4 and IPv6."

    MODULE IF-MIB -- The interfaces MIB, RFC2863
      MANDATORY-GROUPS {
        ifCounterDiscontinuityGroup
      }
```

```
    MODULE  -- this module
      MANDATORY-GROUPS { natConfigGroup, natTranslationGroup,
                         natStatsInterfaceGroup }

      GROUP       natStatsProtocolGroup
      DESCRIPTION
              "This group is optional."
      GROUP       natStatsAddrMapGroup
      DESCRIPTION
              "This group is optional."
      GROUP       natMIBNotificationGroup
      DESCRIPTION
              "This group is optional."

      OBJECT  natAddrMapLocalAddrType
      SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
      DESCRIPTION
             "An implementation is required to support global IPv4
              and/or IPv6 addresses, depending on its support
              for IPv4 and IPv6."

      OBJECT  natAddrMapLocalAddrFrom
      SYNTAX  InetAddress (SIZE(4|16))
      DESCRIPTION
             "An implementation is required to support global IPv4
              and/or IPv6 addresses, depending on its support
              for IPv4 and IPv6."

      OBJECT  natAddrMapLocalAddrTo
      SYNTAX  InetAddress (SIZE(4|16))
      DESCRIPTION
             "An implementation is required to support global IPv4
              and/or IPv6 addresses, depending on its support
              for IPv4 and IPv6."

      OBJECT  natAddrMapGlobalAddrType
      SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
      DESCRIPTION
             "An implementation is required to support global IPv4
              and/or IPv6 addresses, depending on its support
              for IPv4 and IPv6."

      OBJECT  natAddrMapGlobalAddrFrom
      SYNTAX  InetAddress (SIZE(4|16))
      DESCRIPTION
             "An implementation is required to support global IPv4
              and/or IPv6 addresses, depending on its support
              for IPv4 and IPv6."
```

```
      OBJECT  natAddrMapGlobalAddrTo
      SYNTAX  InetAddress (SIZE(4|16))
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support
               for IPv4 and IPv6."

      OBJECT  natAddrBindGlobalAddrType
      SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support
               for IPv4 and IPv6."

      OBJECT  natAddrBindGlobalAddr
      SYNTAX  InetAddress (SIZE(4|16))
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support
               for IPv4 and IPv6."

      OBJECT  natAddrPortBindGlobalAddrType
      SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support
               for IPv4 and IPv6."

      OBJECT  natAddrPortBindGlobalAddr
      SYNTAX  InetAddress (SIZE(4|16))
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support
               for IPv4 and IPv6."

      OBJECT  natSessionPrivateAddrType
      SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support
               for IPv4 and IPv6."

      OBJECT  natSessionPrivateSrcAddr
      SYNTAX  InetAddress (SIZE(4|16))
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support
               for IPv4 and IPv6."
```

```
      OBJECT  natSessionPrivateDstAddr
      SYNTAX  InetAddress (SIZE(4|16))
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support
               for IPv4 and IPv6."

      OBJECT  natSessionPublicAddrType
      SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support
               for IPv4 and IPv6."

      OBJECT  natSessionPublicSrcAddr
      SYNTAX  InetAddress (SIZE(4|16))
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support
               for IPv4 and IPv6."

      OBJECT  natSessionPublicDstAddr
      SYNTAX  InetAddress (SIZE(4|16))
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support
               for IPv4 and IPv6."

    ::= { natMIBCompliances 1 }

natMIBReadOnlyCompliance MODULE-COMPLIANCE
    STATUS  current
    DESCRIPTION
            "When this MIB is implemented without support for
             read-create (i.e., in read-only mode), then such an
             implementation can claim read-only compliance.
             Such a device can then be monitored but cannot be
             configured with this MIB.

             The following index objects cannot be added as OBJECT
             clauses but nevertheless have the compliance
             requirements:
             "
             -- OBJECT  natAddrBindLocalAddrType
             -- SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
             -- DESCRIPTION
             --         "An implementation is required to support
             --          global IPv4 and/or IPv6 addresses, depending
```

```
            --          on its support for IPv4 and IPv6."

            -- OBJECT  natAddrBindLocalAddr
            -- SYNTAX  InetAddress (SIZE(4|16))

            -- DESCRIPTION
            --          "An implementation is required to support
            --           global IPv4 and/or IPv6 addresses, depending
            --           on its support for IPv4 and IPv6."

            -- OBJECT  natAddrPortBindLocalAddrType
            -- SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
            -- DESCRIPTION
            --          "An implementation is required to support
            --           global IPv4 and/or IPv6 addresses, depending
            --           on its support for IPv4 and IPv6."
            -- OBJECT  natAddrPortBindLocalAddr
            -- SYNTAX  InetAddress (SIZE(4|16))
            -- DESCRIPTION
            --          "An implementation is required to support
            --           global IPv4 and/or IPv6 addresses, depending
            --           on its support for IPv4 and IPv6."

    MODULE IF-MIB -- The interfaces MIB, RFC2863
      MANDATORY-GROUPS {
        ifCounterDiscontinuityGroup
      }

    MODULE  -- this module
      MANDATORY-GROUPS { natConfigGroup, natTranslationGroup,
                         natStatsInterfaceGroup }

      GROUP       natStatsProtocolGroup
      DESCRIPTION
               "This group is optional."
      GROUP       natStatsAddrMapGroup
      DESCRIPTION
               "This group is optional."
      GROUP       natMIBNotificationGroup
      DESCRIPTION
               "This group is optional."
      OBJECT natInterfaceRowStatus
      SYNTAX RowStatus { active(1) }
      MIN-ACCESS   read-only
      DESCRIPTION
              "Write access is not required, and active is the only
               status that needs to be supported."
```

```
      OBJECT  natAddrMapLocalAddrType
      SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
      MIN-ACCESS   read-only
      DESCRIPTION
              "Write access is not required.  An implementation is
               required to support global IPv4 and/or IPv6 addresses,
               depending on its support for IPv4 and IPv6."

      OBJECT  natAddrMapLocalAddrFrom
      SYNTAX  InetAddress (SIZE(4|16))
      MIN-ACCESS   read-only
      DESCRIPTION
              "Write access is not required.  An implementation is
               required to support global IPv4 and/or IPv6 addresses,
               depending on its support for IPv4 and IPv6."

      OBJECT  natAddrMapLocalAddrTo
      SYNTAX  InetAddress (SIZE(4|16))
      MIN-ACCESS   read-only
      DESCRIPTION
              "Write access is not required.  An implementation is
               required to support global IPv4 and/or IPv6 addresses,
               depending on its support for IPv4 and IPv6."

      OBJECT  natAddrMapGlobalAddrType
      SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
      MIN-ACCESS   read-only
      DESCRIPTION
              "Write access is not required.  An implementation is
               required to support global IPv4 and/or IPv6 addresses,
               depending on its support for IPv4 and IPv6."

      OBJECT  natAddrMapGlobalAddrFrom
      SYNTAX  InetAddress (SIZE(4|16))
      MIN-ACCESS   read-only
      DESCRIPTION
              "Write access is not required.  An implementation is
               required to support global IPv4 and/or IPv6 addresses,
               depending on its support for IPv4 and IPv6."

      OBJECT  natAddrMapGlobalAddrTo
      SYNTAX  InetAddress (SIZE(4|16))
      MIN-ACCESS   read-only
      DESCRIPTION
              "Write access is not required.  An implementation is
               required to support global IPv4 and/or IPv6 addresses,
               depending on its support for IPv4 and IPv6."
```

```
      OBJECT natAddrMapRowStatus
      SYNTAX RowStatus { active(1) }
      MIN-ACCESS   read-only
      DESCRIPTION
              "Write access is not required, and active is the only
               status that needs to be supported."

      OBJECT  natAddrBindGlobalAddrType
      SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support for
               IPv4 and IPv6."

      OBJECT  natAddrBindGlobalAddr
      SYNTAX  InetAddress (SIZE(4|16))
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support for
               IPv4 and IPv6."

      OBJECT  natAddrPortBindGlobalAddrType
      SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support for
               IPv4 and IPv6."

      OBJECT  natAddrPortBindGlobalAddr
      SYNTAX  InetAddress (SIZE(4|16))
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support for
               IPv4 and IPv6."

      OBJECT  natSessionPrivateAddrType
      SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support for
               IPv4 and IPv6."

      OBJECT  natSessionPrivateSrcAddr
      SYNTAX  InetAddress (SIZE(4|16))
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support for
               IPv4 and IPv6."
```

```
      OBJECT  natSessionPrivateDstAddr
      SYNTAX  InetAddress (SIZE(4|16))
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support for
               IPv4 and IPv6."

      OBJECT  natSessionPublicAddrType
      SYNTAX  InetAddressType { ipv4(1), ipv6(2) }
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support for
               IPv4 and IPv6."

      OBJECT  natSessionPublicSrcAddr
      SYNTAX  InetAddress (SIZE(4|16))
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support for
               IPv4 and IPv6."

      OBJECT  natSessionPublicDstAddr
      SYNTAX  InetAddress (SIZE(4|16))
      DESCRIPTION
              "An implementation is required to support global IPv4
               and/or IPv6 addresses, depending on its support for
               IPv4 and IPv6."

    ::= { natMIBCompliances 2 }

END
```

7. Acknowledgements

The authors would like to thank R. Rohit, P. Srisuresh, Rajiv Raghunarayan, Nalinksh Pai, and Cliff Wang, the original authors of [RFC4008], as well as the following individuals who have participated in the drafting, review, and discussion of this memo:

Cathy Zhou, Juergen Schoenwaelder, Marc Blanchet, and Yu Fu.

8. Security Considerations

[To be reviewed, note about large number of mappings/bindings]

It is clear that this MIB can potentially be useful for configuration. Unauthorized access to the write-able objects could cause a denial of service and/or widespread network disturbance.

Hence, the support for SET operations in a non-secure environment without proper protection can have a negative effect on network operations.

At this writing, no security holes have been identified beyond those that SNMP Security is itself intended to address. These relate primarily to controlled access to sensitive information and the ability to configure a device - or which might result from operator error, which is beyond the scope of any security architecture.

There are a number of managed objects in this MIB that may contain information that may be sensitive from a business perspective, in that they may represent NAT bind and session information. The NAT bind and session objects reveal the identity of private hosts that are engaged in a session with external end nodes. A curious outsider could monitor these two objects to assess the number of private hosts being supported by the NAT device. Further, a disgruntled former employee of an enterprise could use the NAT bind and session information to break into specific private hosts by intercepting the existing sessions or originating new sessions into the host. There are no objects that are sensitive in their own right, such as passwords or monetary amounts. It may even be important to control GET access to these objects and possibly to encrypt the values of these objects when they are sent over the network via SNMP. Not all versions of SNMP provide features for such a secure environment.

SNMP versions prior to SNMPv3 did not include adequate security. Even if the network itself is secure (for example by using IPSec), even then, there is no control as to who on the secure network is allowed to access and GET/SET (read/change/create/delete) the objects in this MIB.

It is recommended that the implementers consider the security features as provided by the SNMPv3 framework (see [RFC3410], section 8), including full support for the SNMPv3 cryptographic mechanisms (for authentication and privacy).

Further, deployment of SNMP versions prior to SNMPv3 is NOT RECOMMENDED. Instead, it is RECOMMENDED to deploy SNMPv3 and to enable cryptographic security. It is then a customer/operator responsibility to ensure that the SNMP entity giving access to an instance of this MIB module is properly configured to give access to the objects only to those principals (users) that have legitimate rights to indeed GET or SET (change/create/delete) them.

9. IANA Considerations

TBD

10. References

10.1. Normative References

[RFC2578] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.

[RFC2579] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Textual Conventions for SMIv2", STD 58, RFC 2579, April 1999.

[RFC2580] McCloghrie, K., Perkins, D., and J. Schoenwaelder, "Conformance Statements for SMIv2", STD 58, RFC 2580, April 1999.

[RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January 2001.

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

[RFC4001] Daniele, M., Haberman, B., Routhier, S., and J. Schoenwaelder, "Textual Conventions for Internet Network Addresses", RFC 4001, February 2005.

[RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, September 1981.

[RFC3489] Rosenberg, J., Weinberger, J., Huitema, C., and R. Mahy, "STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)", RFC 3489, March 2003.

[RFC2863] McCloghrie, K. and F. Kastenholz, "The Interfaces Group MIB", RFC 2863, June 2000.

[RFC2463] Conta, A. and S. Deering, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 2463, December 1998.

[RFC3411] Harrington, D., Presuhn, R., and B. Wijnen, "An

Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks", STD 62, RFC 3411, December 2002.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3413] Levi, D., Meyer, P., and B. Stewart, "Simple Network Management Protocol (SNMP) Applications", STD 62, RFC 3413, December 2002.

10.2. Informative References

[RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.

[RFC3410] Case, J., Mundy, R., Partain, D., and B. Stewart, "Introduction and Applicability Statements for Internet-Standard Management Framework", RFC 3410, December 2002.

[RFC4008] Rohit, R., Srisuresh, P., Raghunarayan, R., Pai, N., and C. Wang, "Definitions of Managed Objects for Network Address Translators (NAT)", RFC 4008, March 2005.

[RFC6333] Durand, A., Droms, R., Woodyatt, J., and Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion", RFC 6333, August 2011.

Authors' Addresses

Simon Perreault
Viagenie
2875 boul. Laurier, suite D2-630
Quebec
Canada

Phone: +1-418-656-9254
EMail: simon.perreault@viagenie.ca

Tina Tsou
Huawei Technologies
2330 Central Expressway
Santa Clara
USA

Phone: +1-408-330-4424
EMail: tena@huawei.com

Behave
Internet-Draft
Intended status: Standards Track
Expires: July 18, 2013

S. Sivakumar
R. Penno
Cisco Systems

January 14, 2013

IPFIX Information Elements for logging NAT Events
draft-sivakumar-behave-nat-logging-06

Abstract

NAT devices are required to log events like creation and deletion of translations and information about the resources it is managing. With the wide deployment of Carrier Grade NAT (CGN) devices, the logging of events have become very important for legal purposes. The logs are required in many cases to identify an attacker or a host that was used to launch malicious attacks and/or for various other purposes of accounting. Since there is no standard way of logging this information, different NAT devices behave differently and hence it is difficult to expect a consistent behavior. The lack of a consistent way makes it difficult to write the collector applications that would receive this data and process it to present useful information. This document describes the information that is required to be logged by the NAT devices.

Table of Contents

1. Terminology

The usage of the term "NAT device" in this document refer to any NAT44 and NAT64 devices. The usage of the term "collector" refers to any device that receives the binary data from a NAT device and converts that into meaningful information. This document uses the term "Session" as it is defined in [RFC2663] and the term BIB as it is defined in [RFC6146]

2. Introduction

This document details the IPFIX Information Elements(IEs) that are required for logging by a NAT device. The document will specify the format of the IE's that are required to be logged by the NAT device and all the optional fields. The fields specified in this document are gleaned from [RFC4787] and [RFC5382].

2.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Scope

This document provides the information model to be used for logging the NAT devices including Carrier Grade NAT (CGN) events. This document focuses exclusively on the specification of IPFIX IE's. This document does not provide guidance on the transport protocol like TCP, UDP or SCTP that is to be used to log NAT events. The log events SHOULD NOT be lost but the choice of the actual transport protocol is beyond the scope of this document.

The existing IANA IPFIX Information Elements registry [IPFIX-IANA] already has assignments for many NAT logging events. For convenience, this document uses those same Information Elements. However, as stated earlier, this document is not defining IPFIX or Netflow 9 as the framework for logging. Rather, the information contained in these elements is within the scope of this document.

This document assumes that the NAT device will use the existing IPFIX framework to send the log events to the collector. This would mean that the NAT device will specify the template that it is going to use for each of the events. The templates can be of varying length and there could be multiple templates that a NAT device could use to log the events.

The implementation details of the collector application is beyond the scope of this document.

The optimization of logging the NAT events are left to the implementation and are beyond the scope of this document.

4. Event based logging

An event in a NAT device can be viewed as a happening as it relates to the management of NAT resources. The creation and deletion of NAT sessions and bindings are examples of events as it results in the resources (addresses and ports) being allocated or freed. The events can happen either through the processing of data packets flowing through the NAT device or through an external entity installing policies on the NAT router or as a result of an asynchronous event like a timer. The list of events are provided in Section 4.1. Each of these events SHOULD be logged, unless they are administratively prohibited. A NAT device MAY log these events to multiple collectors if redundancy is required. The network administrator will specify the collectors to which the log records are to be sent.

A collector may receive NAT events from multiple CGN devices and should be able to distinguish between the devices. Each CGN device should have a unique source ID to identify themselves. The source ID is part of the IPFIX template and data exchange.

Prior to logging any events, the NAT device MUST send the template of the record to the collector to advertise the format of the data record that it is using to send the events. The templates can be exchanged as frequently as required given the reliability of the connection. There SHOULD be a configurable timer for controlling the template refresh. NAT device SHOULD combine as many events as possible in a single packet to effectively utilize the network bandwidth.

4.1. Information Elements

The templates could contain a subset of the Information Elements(IEs) shown in Table 1 depending upon the event being logged. For example a NAT44 session creation template record will contain,

{sourceIPv4Adress, postNATSourceIPv4Address, destinationIpv4Address, postNATDestinationIPv4Address, sourceTransportPort, postNAPTSourceTransportPort, destinationTransportPort, postNAPTDestTransportPort, natOriginatingAddressRealm, natEvent, timeStamp}

An example of the actual event data record is shown below - in a readable form

{192.168.16.1, 201.1.1.100, 207.85.231.104, 207.85.231.104, 14800, 1024, 80, 80, 0, 1, 09:20:10:789}

A single NAT device could be exporting multiple templates and the collector should support receiving multiple templates from the same source.

The following is the table of all the IE's that a CGN device would need to export the events. The formats of the IE's and the IPFIX IDs are listed below.

Field Name	Size (bits)	IANA IPFIX ID	Description
timeStamp	64	323	System Time when the event occured.
vlanID	16	58	VLAN ID in case of overlapping networks
ingressVRFID	32	234	VRF ID in case of overlapping networks
sourceIPv4Address	32	8	Source IPv4 Address
postNATSourceIPv4Address	32	225	Translated Source IPv4 Address
protocolIdentifier	8	4	Transport protocol
sourceTransportPort	16	7	Source Port
postNAPTsourceTransportPort	16	227	Translated Source port
destinationIPv4Address	32	12	Destination IPv4 Address
postNATDestinationIPv4Address	32	226	Translated IPv4 destination address
destinationTransportPort	16	11	Destination port
postNAPTdestinationTransportPort	16	228	Translated Destination port
sourceIPv6Address	27	128	Source IPv6 address
destinationIPv6Address	128	28	Destination IPv6 address

postNATSourceIPv6Address	128	281	Translated source IPv6 addresss	
postNATDestinationIPv6Address	128	282	Translated Destination IPv6 address	
natOriginatingAddressRealm	8	229	Address Realm	
natEvent	8	230	Type of Event	
portRangeStart	16	361	Allocated port block start	
portRangeEnd	16	362	Allocated Port block end	
portRangeStepSize	16	363	Step size of next port	
portRangeNumPorts	16	364	Number of ports	

Table 1: Template format Table

4.2. Definition of NAT Events

The following are the list of NAT events and the proposed event values. The list can be expanded in the future as necessary. The data record will have the corresponding natEvent value to identify the event that is being logged.

Event Name	Values
NAT44 Session create	1
NAT44 Session delete	2
NAT Addresses exhausted	3
NAT64 Session create	4
NAT64 Session delete	5
NAT44 BIB create	6
NAT44 BIB delete	7
NAT64 BIB create	8
NAT64 BIB delete	9
NAT ports exhausted	10
Quota exceeded	11
Address Binding	12
Port block allocation	13
Port block de-allocation	14

Table 2: NAT Event ID table

4.3. Quota exceeded - Sub Event types

The following table shows the sub event types for the Quota exceeded event

Quota Exceeded Event Name	Values
Max Session entries	1
Max BIB entries	2
Max entries per user	3

Table 3: Sub Event ID table

4.4. Templates for NAT Events

The following is the template of events that will have to logged. The events below are identified at the time of this writing but the events are expandable. Depending on the implementation and configuration various IE’s specified can be included or ignored.

4.4.1. NAT44 create and delete session event

This event will be generated when a NAT44 session is created or deleted. The template will be the same, the natEvent will indicate whether it is a create or a delete event. The following is a template of the event.

Field Name	Size (bits)	Mandatory
timeStamp	64	Yes
vlanID/ingressVRFID	32	No
sourceIPv4Address	32	Yes
postNATSourceIPv4Address	32	Yes
protocolIdentifier	8	Yes
sourceTransportPort	16	Yes
postNAPTsourceTransportPort	16	Yes
destinationIPv4Address	32	No
postNATDestinationIPv4Address	32	No
destinationTransportPort	16	No
postNAPTdestinationTransportPort	16	No
natOriginatingAddressRealm	8	No
natEvent	8	Yes

Table 4: NAT44 Session delete/create template

4.4.2. NAT64 create and delete session event

This event will be generated when a NAT64 session is created. The following is a template of the event.

Field Name	Size (bits)	Mandatory
timeStamp	64	Yes
vlanID/ingressVRFID	32	No
sourceIPv6Address	128	Yes
postNATSourceIPv4Address	32	Yes
protocolIdentifier	8	Yes
sourceTransportPort	16	Yes
postNAPTsourceTransportPort	16	Yes
destinationIPv6Address	128	No
postNATDestinationIPv4Address	32	No
destinationTransportPort	16	No
postNAPTdestinationTransportPort	16	No
natOriginatingAddressRealm	8	No
natEvent	8	Yes

Table 5: NAT64 session create/delete event template

4.4.3. NAT44 BIB create and delete event

This event will be generated when a NAT44 Bind entry is created. The following is a template of the event.

Field Name	Size (bits)	Mandatory
timeStamp	64	Yes
vlanID/ingressVRFID	32	No
sourceIPv4Address	32	Yes
postNATSourceIPv4Address	32	Yes
protocolIdentifier	8	No
sourceTransportPort	16	No
postNAPTsourceTransportPort	16	No
natOriginatingAddressRealm	8	No
natEvent	8	Yes

Table 6: NAT44 BIB create/delete event template

4.4.4. NAT64 BIB create and delete event

This event will be generated when a NAT64 Bind entry is created. The following is a template of the event.

Field Name	Size (bits)	Mandatory
timeStamp	64	Yes
vlanID/ingressVRFID	32	No
sourceIPv6Address	128	Yes
postNATSourceIPv4Address	32	Yes
protocolIdentifier	8	No
sourceTransportPort	16	No
postNAPTsourceTransportPort	16	No
natOriginatingAddressRealm	8	No
natEvent	8	Yes

Table 7: NAT64 BIB create/delete event template

4.4.5. Addresses Exhausted event

This event will be generated when a NAT device runs out of global IPv4 addresses in a given pool of addresses. Typically, this event would mean that the NAT device wont be able to create any new translations until some addresses/ports are freed. The following is

a template of the event.

Field Name	Size (bits)	Mandatory
timeStamp	64	Yes
natEvent	8	Yes
natPoolName	String	Yes

Table 8: NAT Address Exhausted event template

4.4.6. Ports Exhausted event

This event will be generated when a NAT device runs out of ports for a global IPv4 address. Port exhaustion shall be reported per protocol (UDP, TCP etc) The following is a template of the event.

Field Name	Size (bits)	Mandatory
timeStamp	64	Yes
natEvent	8	Yes
postNATSourceIPv4Address	32	Yes
protocolIdentifier	8	Yes

Table 9: NAT Ports Exhausted event template

4.4.7. Quota exceeded

This event will be generated when a NAT device cannot allocate resources as a result of an administratively defined policy. The examples of Quota exceeded are to allow only certain number of NAT sessions per device, certain number of NAT sessions per user etc. The following is a template of the event.

Field Name	Size (bits)	Mandatory
timeStamp	64	Yes
natEvent	8	Yes
natLimitEvent	32	Yes
sourceIPv4 address	32	No
sourceIPv6 address	128	No

Table 10: NAT Quota Exceeded event template

4.4.8. Address Binding

This event will be generated when a NAT device binds a local address with a global address. This binding event happens when the first packet of the first flow from a host in the private realm.

Field Name	Size (bits)	Mandatory
timeStamp	64	Yes
natEvent	8	Yes
sourceIPv4 address	32	No
sourceIPv6 address	128	No
Translated Source IPv4 Address	32	8

Table 11: NAT Address Binding template

4.4.9. Port block allocation and de-allocation

This event will be generated when a NAT device allocates/de-allocates ports in a bulk fashion, as opposed to allocating a port on a per flow basis. NAT devices would do this in order to reduce logs and potentially to limit the number of connections a subscriber is allowed to use. In the following Port Block allocation template, the portRangeStart must be specified. Along with portRangeStart, atleast one of portRangeEnd, portRangeStepSize or portRangeNumPorts MUST be specified. If portRangeEnd is specified, it MUST NOT be lesser than portRangeStart. The value of portRangeStepSize MUST be between 1 and 32K.

Field Name	Size (bits)	Mandatory
timeStamp	64	Yes
portRangeStart	16	Yes
portRangeEnd	16	No
portRangeStepSize	16	No
portRangeNumPorts	16	No

Table 12: NAT Port Block Allocation event template

5. Encoding

5.1. IPFIX

This document uses IPFIX as the encoding mechanism to describe the logging of NAT events. However, the information that should be logged SHOULD be the same irrespective of what kind of encoding scheme is used. IPFIX is chosen because is it an IETF standard that meets all the needs for a reliable logging mechanism. IPFIX provides the flexibility to the logging device to define the data sets that it is logging. The information elements specified for logging MUST be the same irrespective of the encoding mechanism used.

6. Acknowledgements

Thanks to Dan Wing, Selvi Shanmugam, Mohamed Boucadir, Jacni Qin Ramji Vaithianathan, Simon Perreault, Jean-Francois Tremblay and Julia Renouard for their review and comments.

7. IANA Considerations

8. Security Considerations

None.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

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

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

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

[RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6

Clients to IPv4 Servers", RFC 6146, April 2011.

9.2. Informative References

[NAT-EVENT-LOG-IANA]
IANA, "NAT event log entities", 2012, <http://www.iana.org/assignments/nat-event-log/nat-event-log.xml>.

[RFC5101] Claise, B., "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information", RFC 5101, January 2008.

[RFC5102] Quittek, J., Bryant, S., Claise, B., Aitken, P., and J. Meyer, "Information Model for IP Flow Information Export", RFC 5102, January 2008.

[RFC5470] Sadasivan, G., Brownlee, N., Claise, B., and J. Quittek, "Architecture for IP Flow Information Export", RFC 5470, March 2009.

Authors' Addresses

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

Renaldo Penno
Cisco Systems
170 W Tasman Drive
San Jose, California 95035
USA

Phone:
Email: repenno@cisco.com

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: March 20, 2014

T. Tsou, Ed.
Huawei Technologies (USA)
S. Perreault
Viagenie
J. Huang
Huawei Technologies

September 16, 2013

An FTP Application Layer Gateway (ALG) for IPv4-to-IPv6 Translation
draft-tsou-behave-ftp46-01

Abstract

An FTP ALG for NAT64 was defined in RFC 6384. Its scope was limited to an IPv6 client connecting to an IPv4 server. This memo supports the case of an IPv4 client connecting to an IPv6 server.

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 http://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 March 20, 2014.

Copyright Notice

Table of Contents

1. Overview

During the transition from IPv4 to IPv6, some operators need to deploy NAT in their network. Some subscribers have the need to run IPv4 based FTP servers at home, and some of the FTP [RFC0959] control messages carry IP address and port number in the payload, which will cause a NAT traversal problem.

[RFC6384] defines FTP ALG for NAT64, but only for the case where the FTP client is on the inside of the NAT64. The case where an FTP server is on the inside of the NAT64 is not covered.

When the FTP server is behind NAT, it can publish its service address via a HTTP redirect server and a DDNS system which needs to support both IP address and port rather than IP address only, or other possible methods. The FTP server can listen on any possible ports, not just port 21; FTP server can get it external IP address and port via some technology like UPnP, and then publish the acquired IP address and port as its URI, ftp://203.0.113.1:1200, port 1200 is allocated by NAT.

1.1. Requirements Language

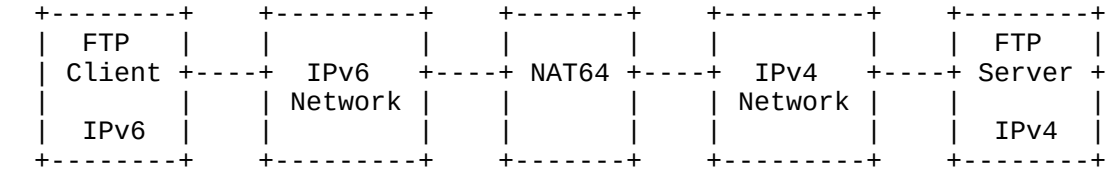
The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Terminology

3. Scenarios

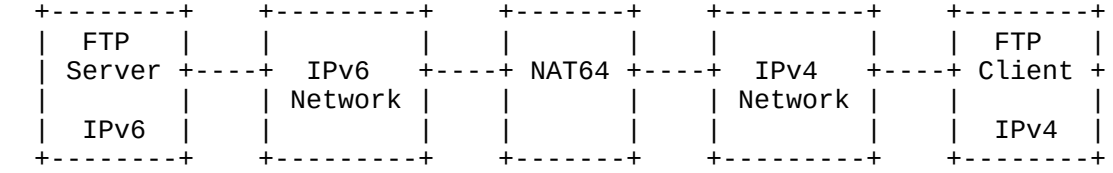
There can be several scenarios if NAT is involved in the network.

a) In this scenario, the FTP client is behind NAT, FTP ALG needs to handle the EPRT / PORT command in FTP active mode, translating the IP address and port. This scenario has been covered by [RFC6384], but only for NAT64. This scenario for other kinds of NAT has not been covered.



FTP Client Behind NAT

b) If the FTP server is behind a NAT, FTP passive mode will be the only working mode, the EPSV / PASV command and the response will be processed by FTP ALG. This memo covers this scenario.



FTP Server Behind NAT

4. PASV to EPSV

If FTP client issues PASV command to FTP server, FTP ALG translates PASV command into EPSV command [RFC2428], setting the "net-prt" field to 2 (IPv6). The response of EPSV command is translated into PASV response. FTP ALG allocates an IPv4 address and port for the EPSV response message, and builds a NAT mapping entry if the NAT is stateful. The source address of the EPSV response message and the "tcp-port" in the payload are used for the NAT mapping. The allocated IPv4 address and port are put into the PASV response message.

For instance, in the IPv4 side of NAT64, FTP server's address is 203.0.113.1. FTP client issues a PASV command to FTP server, and it is translated into EPSV command by FTP AGL, as shown below:

PASV command:

PASV

EPSV command:

EPSV 2

When FTP server returns a success response of EPSV containing tcp-port 3000, FTP AGL allocates an IPv4 address 203.0.113.1 and tcp-port 2000 corresponding to the tcp-port 3000 in the EPSV response message, and puts the allocated IP address and port into PASV response message, as shown below:

EPSV success response:

229 Entering Passive Mode (|||3000|)

PASV success response:

227 Entering Passive Mode (203,0,113,1,7,208)

5. EPSV (IPv4) to EPSV (IPv6)

If FTP client issues EPSV command to FTP server, FTP ALG modifies the "net-prt", change the value from 1 (IPv4) to 2 (IPv6). The response of IPv6 EPSV command is also translated. FTP ALG allocates an IPv4 address and port for the EPSV response message.

[RFC2428] requires that "the network address used to establish the data connection will be the same network address used for the control connection", so NAT MUST to make sure that IPv4 address for control connection and IPv4 address for data connection for a FTP server must be the same, which means all the mappings for an IPv6 address MUST have the same external IPv4 address.

For instance, in the IPv4 side of NAT64, FTP server's address is 203.0.113.1. The FTP client issues an IPv4 EPSV command to FTP server, and it is translated into IPv6 EPSV command by FTP AGL, as shown below:

EPSV (IPv4) command:

EPSV 1

EPSV (IPv6) command:

EPSV 2

When FTP server returns a success response of EPSV containing port 3000, FTP AGL will allocate an IPv4 address 203.0.113.1 and port 2000 corresponding to the port 3000 in the EPSV response message, and put the allocated port into PASV response message, as shown below:

EPSV (IPv6) success response:

229 Entering Passive Mode (|||3000|)

EPSV (IPv4) success response:

229 Entering Passive Mode (|||2000|)

6. Command to disable FTP ALG

Command ALGS defined in [RFC6384] is extended, three more possible arguments are added:

ALGS STATUS46 to return the EPSV and EPRT translation status.

ALGS ENABLE46 to enable translation.

ALGS DISABLE46 to disable translation.

7. IANA Considerations

This memo includes no request to IANA.

8. Security Considerations

RFC6384's security considerations applies to this document.

9. Acknowledgements

10. Normative References

[RFC0959] Postel, J. and J. Reynolds, "File Transfer Protocol", STD 9, RFC 959, October 1985.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2428] Allman, M., Ostermann, S., and C. Metz, "FTP Extensions for IPv6 and NATs", RFC 2428, September 1998.

[RFC6384] van Beijnum, I., "An FTP Application Layer Gateway (ALG) for IPv6-to-IPv4 Translation", RFC 6384, October 2011.

Authors' Addresses

Tina Tsou (editor)
Huawei Technologies (USA)
2330 Central Expressway
Santa Clara CA 95050
USA

Phone: +1 408 330 4424
Email: tina.tsou.zouting@huawei.com

Simon Perreault
Viagenie
246 Aberdeen
Quebec, QC G1R 2E1
Canada

Phone: +1 418 656 9254
Email: simon.perreault@viagenie.ca
URI: http://viagenie.ca

Jing Huang
Huawei Technologies
Huawei Area F, Bantian, Longgang District
Shenzhen 518129
China

Email: James.huang@huawei.com

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: March 10, 2013

Z. Chen
China Telecom
C. Zhou
Huawei Technologies
T. Tsou
Huawei Technologies (USA)
T. Taylor
Huawei Technologies

September 6, 2012

Syslog Format for NAT Logging
draft-zhou-behave-syslog-nat-logging-01

Abstract

Under some circumstances operators will need to maintain a dynamic record of external address and port assignments made by a Carrier Grade NAT (CGN), and will find it feasible and convenient to create such records using SYSLOG (RFC 5424). The present document standardizes a SYSLOG format to meet that recording requirement. It specifies a number of fields that could be a part of the log report, leaving it up to operators to select the fields needed for their specific circumstances.

[*** Subject to discussion*** The log format presented here may also be used by PCP server implementations to log the mappings they implement.]

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 http://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 March 10, 2013.

Copyright Notice

Table of Contents

1. Introduction

Operators already need to record the addresses assigned to subscribers at any point in time, for operational and regulatory reasons. When operators introduce Carrier Grade NATs (CGNs) into their network, both addresses and ports on the external side of the CGN are shared amongst subscribers. To trace back from an external address and port observed at a given point in time to a specific subscriber requires additional information: a record of which subscriber was assigned that address and port by the NAT.

Address-port assignment strategies present a tradeoff between the efficiency with which available external addresses are used, the cost of maintaining a trace back capability, and the need to make port assignments unpredictable to counter the threat of session hijacking. At one extreme, the operator could make a one-time assignment of an external address and a set of ports to each subscriber. Traceback would then be a matter of retrieving configuration information from the NAT. Even in this situation, it is possible that a request for legal interception is placed against a specific subscriber, such that each session involving that subscriber is recorded.

At the opposite extreme, a carrier could assign external addresses and ports to subscribers on demand, in totally random fashion. Such a strategy is not really practical, both because of the volume of records that would be required to support a traceback capability, and because the apparent gain in efficiency with which address-port combinations would be utilized would be attenuated by the need to leave address-port assignments idle for some minimum amount of time after last observed use to make sure they weren’t still being used.

Between these extremes, operators may choose to assign specific addresses and specific blocks of ports to subscribers when they log on to the network, releasing the assignments when they drop off. Such a strategy could be desirable in networks with mobile subscribers, in particular. Compared with the fully dynamic strategy, this strategy reduces the number of times that assignments have to be recorded by orders of magnitude.

The point just made is that under some circumstances operators need to record allocations of external address-port combinations in the NAT dynamically, and the volume of information contained in those records is manageable. Various means are available to create such records. This document assumes that for some operators, the most convenient mechanism to do so will be event logging using SYSLOG [RFC5424], where the SYSLOG records are generated either by the NAT itself or by an off-line device.

The next section specifies a SYSLOG record format for logging of NAT address and port assignments and the format of fields that could be used within such a record. It is up to individual operators to choose the fields that match their specific operating procedures.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in "Key words for use in RFCs to Indicate Requirement Levels" [RFC2119].

2. SYSLOG Record Format For NAT Logging

This section describes the SYSLOG record format for NAT logging in terms of the field names used in [RFC5424] and specified in Section 6 of that document. In particular, this section specifies values for the APP-NAME and MSGID fields in the record header, the SD-ID identifying the STRUCTURED-DATA section, and the PARAM-NAMEs and PARAM-VALUE types for the individual possible parameters within that section.

2.1. SYSLOG HEADER Fields

Within the HEADER portion of the SYSLOG record, the priority (PRI) level is subject to local policy, but a default value of 86 is suggested, representing a Facility value of 10 (security/ authorization) and a Severity level of 6 (informational). Depending on where the SYSLOG record is generated, the HOSTNAME field may identify the NAT or an offline logging device. In the latter case, it may be desirable to identify the NAT using the NID field in the STRUCTURED-DATA section (see below). The value of the HOSTNAME field is subject to the preferences given in Section 6.2.4 of [RFC5424].

The values of the APP-NAME and MSGID fields in the record header determine the semantics of the record. The RECOMMENDED APP-NAME value "NAT" indicates that the record relates to an assignment made autonomously by the NAT itself. [*** Subject to discussion*** The RECOMMENDED APP-NAME "PCP" indicates that the assignment to which the record refers was the result of a Port Control Protocol (PCP) [I-D.PCP-Base] command.] The RECOMMENDED MSGID value "ADD" indicates that the assignment took effect at the time indicated by the record timestamp. The RECOMMENDED MSGID value "DEL" indicates that the assignment was deleted at the time indicated by the record timestamp.

2.2. STRUCTURED-DATA Fields

This document specifies a value of "asgn" (short for "assignment") for the SD-ID field identifying the STRUCTURED-DATA section of the record. In addition it specifies the following parameters for use within that section. All of these parameters are OPTIONAL. All values that are IP addresses are written as a text string in dotted-decimal form (IPv4) or as recommended by [RFC5952] (IPv6).

2.2.1. Incoming IP Source Address Parameter

PARAM-NAME: iSA. PARAM-VALUE: the incoming IP source address of the packet(s) to which the assignment described by this record applies.

2.2.2. Outgoing IP Source Address Parameter

PARAM-NAME: oSA. PARAM-VALUE: the outgoing IP source address of the packet(s) to the assignment described by which this record applies.

2.2.3. Incoming Source Port Parameter

PARAM-NAME: iSP. PARAM-VALUE: the incoming IP source port of the packet(s) to the assignment described by which this record applies.

2.2.4. Outgoing Source Port Parameter

PARAM-NAME: oSP. PARAM-VALUE: the outgoing IP source port of the packet(s) to which the assignment described by this record applies. If the record pertains to the assignment of a range of ports, this parameter gives the lowest port number in the range. In the case of a range, either parameter oSPct or parameter oSPmx SHOULD also be present in the log record.

2.2.5. Number of Port Numbers Parameter

PARAM-NAME: oSPct. PARAM-VALUE: used when the record pertains to the assignment of a range of ports (either consecutive or generated by a known algorithm). This parameter gives the number of port numbers in the range.

2.2.6. Highest Outgoing Port Number Parameter

PARAM-NAME: oSPmx. PARAM-VALUE: used when the record pertains to the assignment of a range of ports (either consecutive or generated by a known algorithm). This parameter gives the highest port number in the range.

2.2.7. Protocol Parameter

PARAM-NAME: Pr. PARAM-VALUE: an integer indicating the value of the Protocol header field (IPv4) or Next Header field (IPv6) in the incoming packet(s) to which the assignment described by this record applies.

2.2.8. Subscriber Identifier Parameter

PARAM-NAME: SID. PARAM-VALUE: an arbitrary UTF-8 string identifying the subscriber to which this assignment applies. This is intended to provide flexibility when the incoming source address will not be unique. The value could be a tunnel identifier, layer 2 address, or any other value that is convenient to the operator and associated with incoming packets.

2.2.9. NAT Identifier Parameter

PARAM-NAME: NID. PARAM-VALUE: an arbitrary UTF-8 string identifying the NAT making the assignment to which this record applies. Needed only if the necessary identification is not provided by the HOSTNAME parameter in the log record header.

3. IANA Considerations

This document requests IANA to make the following assignments to the SYSLOG Structured Data ID Values registry. RFCxxxx refers to the present document when approved.

Structured Data ID	Structured Data Parameter	Required or Optional	Reference
asgn		OPTIONAL	RFCxxxx
	iSA	OPTIONAL	RFCxxxx
	oSA	OPTIONAL	RFCxxxx
	iSP	OPTIONAL	RFCxxxx
	oSP	OPTIONAL	RFCxxxx
	oSPct	OPTIONAL	RFCxxxx
	oSPmx	OPTIONAL	RFCxxxx
	Pr	OPTIONAL	RFCxxxx
	SID	OPTIONAL	RFCxxxx
	NID	OPTIONAL	RFCxxxx

Table 1

4. Security Considerations

When logs are being recorded for regulatory reasons, preservation of their integrity and authentication of their origin is essential. To achieve this result, it is RECOMMENDED that the operator deploy [RFC5848].

Access to the logs defined here while the reported assignments are in force could improve an attacker's chance of hijacking a session through port-guessing. Even after an assignment has expired, the information in the logs SHOULD be treated as confidential, since, if revealed, it could help an attacker trace sessions back to a particular subscriber or subscriber location. It is therefore RECOMMENDED that these logs be transported securely, using [RFC5425], for example, and that they be stored securely at the collector.

5. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5424] Gerhards, R., "The Syslog Protocol", RFC 5424, March 2009.

[RFC5425] Miao, F., Ma, Y., and J. Salowey, "Transport Layer Security (TLS) Transport Mapping for Syslog", RFC 5425, March 2009.

[RFC5848] Kelsey, J., Callas, J., and A. Clemm, "Signed Syslog Messages", RFC 5848, May 2010.

[RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, August 2010.

Authors' Addresses

Zhonghua Chen
China Telecom
P.R. China

Phone:
Email: 18918588897@189.cn

Cathy Zhou
Huawei Technologies
Bantian, Longgang District
Shenzhen 518129
P.R. China

Phone:
Email: cathy.zhou@huawei.com

Tina Tsou
Huawei Technologies (USA)
2330 Central Expressway
Santa Clara, CA 95050
USA

Phone: +1 408 330 4424
Email: tina.tsou.zouting@huawei.com

T. Taylor
Huawei Technologies
Ottawa,
Canada

Phone:
Email: tom.taylor.stds@gmail.com