NAT Working Group
INTERNET-DRAFT
Category: Informational
Expire in six months

P. Srisuresh
Lucent Technologies

November, 1998

IP Network Address Translator Application Programming Interface

<draft-ietf-nat-api-00.txt>

Status of this Memo

Abstract

NAT provides routing transparency for hosts in disparate routing realms to communicate with each other. However, external agents such as Application Level Gateways (ALGs), Host-NAT-clients and Management applications need to interact with NAT and influence its operations. The document identifies the resources and other elements controlled by a NAT device, with specific focus on areas subject to influence from external agents. An Application Programming Interface (API) framework by which external agents could interact with NAT is presented. The intent of this document is to leverage the API specification as a base to identify requirements for the development of one or more protocols by which external agents could interact with NAT.

1. Introduction

NAT provides routing transparency for hosts in disparate routing realms to communicate with each other. [Ref 1] details the various flavors of NAT that abound. Many of the internet applications use IP address as host identifier rather than just as a way to locate a host. For this reason, routing transparency by NAT alone is not sufficient to provide end-to-end transparency for applications operating across realms. Application specific ALGs are required in conjunction with NAT to provide end-to-end transparency for some applications.

In addition to ALGs, there are other kinds of external agents that may like to influence NAT operation. Section 2 below is devoted to describing the resources and other elements controlled by NAT. Section 3 below outlines a selected list of external agents that may likely interface with NAT. Together, the requirements by a selected set of external agents and the nature of NAT resources are used as the basis to derive an API framework, described in section 4.

The intent of the document is two-fold. First, the document suggests an Application programming Interface (API) by which external agents could programmatically interface with NAT. This does not assume or require external agents to reside on the same physical device as NAT, even though assuming they reside on the same physical device does help in understanding. In reality, it is likely to be a combination of both. Some agents are co-located with NAT on the same device and others reside on external devices. The API is merely a suggestion and may vary from vendor to vendor.

Second, the API provides a framework to identify requirements for the development of one or more protocols by which external agents (specified in section 3 below) could communicate with NAT. Such a protocol would need to authenticate clients, locate NAT devices and exchange data elements. The API specified in the document assumes a trusted environment and does not address the first two issues, namely authentication and Service location. The document also does not cover any communication protocol that may be used by external agents to interface with NAT using the API described here. These issues will need to be addressed independently outside the purview of this document.

2. Elements of NAT operation

In order to identify an API for use by external agents, it is

important to understand the resources and other elments managed by NAT. This would help identify the extent to which an external agent may influence NAT operation. This section describes objects within NAT, that could be externalized via Management Information Base (MIB).

2.1. NAT Descriptor

All flavors of NAT are designed to provide routing transparency to hosts in disparate routing realms. A physical device may have multiple NAT instances or there may be multiple NAT devices associated with a specific realm. The following list of attributes identify a specific instance of NAT.

a. NAT IDentifier:

   A NAT Identifier uniquely identifies a NAT instantiation. The External interface address may be one way to specify NAT Identifier.

b. Private and External realm types:

   Every NAT device will have a minimum of two routing interfaces, one connecting to a private realm and one connecting to external realm. An IPv4 NAT device will have both its realm types set to IPv4.

c. NAT type

   NAT type could be one of Basic-NAT, NAPT, Bi-directional-NAT, Twice-NAT, Host-NAT server, Host-NAPT-server or a combination of the above. NAT type is an indication of the direction in which NAT sessions are allowed and the extent of translation within the IP and transport headers. [Ref 1] has a discussion on the nature of various NAT flavors and the extent of their translations.

d. Address(and transport ID) maps

   Address map on a NAT device could consist of one or more of static and dynamic Address maps. Likewise, Transport ID mapping could consists of one or more of static and dynamic Transport ID maps. Transport ID mapping is more specific than address mapping in that a specific TCP/UDP port (or port range) pertaining to an address in external realm is mapped to a specific TCP/UDP port (or port range) in private realm or vice versa. Address (and Transport ID) maps may be defined for both inbound and outbound directions. Outbound address map refers

to mapping a selected set of addresses from private realm to a selected set of addresses in external realm; whereas inbound address map refers to mapping a set of addresses from the external realm to private realm.

e. Miscellaneous parameters

NAT may optionally provide TCP, UDP and other types of session Idle-times used to terminate sessions. It may also provide the current range (and, the maximum range) of session IDs and Bind IDs (to be covered in the follow on sub-sections); and the actual count of session IDs and BIND IDs. Specifically, this information will be of relevance to another NAT (backup NAT) that intends to emulate this NAT, in case of failure. Lastly, NAT may choose to supply any other vendor specific parameters such as log options, session direction failure actions and so forth.

f. Host-NAT (and Host-NAPT) specific parameters

If the NAT device were to provide Host-NAT-Server capability; optionally, the NAT device could specify the Host-NAT tunneling type it supports.

2.2. Address (and Transport-ID) BINDing Descriptor

These bindings can be static or dynamic. Hereafter, the term BIND will be referred in place of BINDing, for ease of use. When external agents do not intervene, dynamic address(and transport-ID) binding is determined by NAT based on the first packet of a session, as described in [Ref 1]. Address binding is between an address in private realm and an address from external realm. Transport-ID BIND is extension of the same concept to the tuple of Address and transport ID (such as TCP/UDP port no.). The following list of attributes identify a BIND within a NAT.

a. Bind ID

A number (say, in the range of 1 through 0xFFFFFFFF) assigned to BIND to uniquely identify this BIND from a different BIND on the same NAT.

b. Direction of Bind

A bind can be uni-directional or bi-directional, same as the orientation of address map based on which this BIND is formed. As before, the direction is with reference to private realm.

c. Bind type

Indicates whether the BIND is Address-BIND (between a pair of addresses) or Transport-ID-Bind (between a pair of Address, transport ID tuples). Note, a transport-ID bind intrinsically assumes an address bind between the addresses specified in the tuples. This also indicates if the Bind is static or dynamic.

d. Private and External addresses (and Transport IDs)

The pair described here essentially identify the BINDing items between private and external realms.

e. Maximum leased time

The validity of a BIND may be limited by the maximum length of leased time it is allowed. Unless the leased time is renewed, the BIND will no longer be valid past this time. As a special case, a value of 0 may be assumed to indicate no lease time limit. Typically, this attribute is of relevance in conjunction with Host-NAT operation.

f. Available leased time

This parameter is of relevance only when Maximum Leased time is set to a non-zero value. At any given instance of time, this parameter indicates the real-time left for the BIND to remain valid. Typically, this attribute is of relevance in conjunction with Host-NAT operation.

g. Maximum Idle time

This parameter indicates maximum amount of time a dynamic BIND is allowed to remain valid, with no NAT session hanging off this BIND. Typically, a dynamic Bind is established when NAT notices the first session that needs such a binding. Subsequent to this, multiple NAT sessions can be maintained using the same binding. When the last of these sessions is terminated, the bind is also terminated. In other words, Maximum Idle time is 0, by default, for native NAT. External agents could control this parameter differently. Static Binds and lease time limited BINDs are not effected by this parameter.

h. Current Idle time

This parameter is of relevance only when Maximum Idle time is

set to a non-zero value. At any given instance of time, this parameter indicates the real-time the BIND has been idle with no sessions attached to it.

i. Controlling Agent IDentification

This indicates the last external Agent who has tried to control (i.e., set) parameters for this BIND. A value of 0 indicates that native NAT is the responsible agent.

2.3. Session State descriptor

NAT maintains soft state for the sessions it tracks. These states are created dynamically during NAT operation and are responsible for translation of packets pertaining to the session. The translation element of a state is based on address (or Transport ID) bind (two binds in case of twice-nat). The following list of attributes identify a session (or session State) within NAT.

a. Session IDentifier

A number (say, in the range of 1 through 0xFFFFFFFF) assigned to session to uniquely identify this from other sessions on the same NAT.

b. Direction of Session.

Direction of first packet of the session. As specified earlier, direction is with reference to private realm.

c. Bind IDentifier

Identifies the Bind based on which this session is created. The Direction of BIND must be same as that of the session, if the BIND is uni-directional. Typically, if a Bind supporting the session translation does not already exist, a Bind is created prior to creating new session state. However, this Identifier may be set to 0, when BIND creation is unnecessary for the session. For example, there can be no more than one ICMP Query session using am ICMP Query based transport-ID-bind. In such a case, it suffices to do away with BIND and keep all requisite information within the session state itself.

d. Second Bind IDentifier

This is of relevance only to Twice-NAT. For all other flavors of NAT, this parameter may be set to zero. If the session is outbound, this parameter refers to binding of the target

destination address from private realm to external realm.

e. Original Session parameters

These parameters identify the session level parameters as they appear in the first packet of session. These parameters include src and dest IP addresses, IP protocol and transport IDentifier info (such as TCP/UDP port numbers or ICMP Query Identifier).

f. Translated Session parameters

These parameters identify the session level parameters as the first packet of session is translated. These parameters are derived from the BIND ID(s) off which this session hangs.

g. Session tag

NAT managed sessions are assigned a session tag, so that sessions bearing the same tag are handled the same way. The tag value is of significance only to the processing agent. Native NAT maintains four types of session tags for TCP, UDP, ICMP QUERY and all other sessions. So, tag numbers selected by the agents will need to be different from the native tags, if the processing were to be done differently.

h. Session Termination heuristic

Session-Idle-time is typically used as a heuristic means by NAT to determine if the session has ended. There may other heuristic approaches. A value of zero is an indication that NAT would not use any heuristic to session termination, unless it is a TCP session and the session has noticeable ended with FIN or RST options. The agent may take the responsibility for terminating the session.

i. Maximum Idle time

This parameter indicates maximum amount of time this session is allowed to remain valid, even as there is no activity. Idle time is typically used as a heuristic means to determine session termination. There may be other heuristic approaches. As a special case, a value of 0 implies that NAT should run the same timer as used for native sessions.

j. Current Idle Time

This parameter is of relevance only when session termination heuristic is set to session-idle-time. Typically, NAT would examine the idle time on the sessions it manages periodically and updates this variable. When the idle time exceeds the maximum allowed idle time, the session is terminated.

k. Packet modifier functions

Typically, NAT modifies IP header and optionally, the transport header. External agents could choose to assume responsibility for payload modification alone, or the entire packet modification. In the case an external agent assumes responsibility for the entire packet modification, NAT will simply redirect the original packet as is to external agent modifier.

l. Bundle ID

Applications that deal with a bundle of sessions may cause multiple sessions to be managed by NAT. Even though these sessions constitute a single session from application stand point, NAT is not congnizant of the relation. In such cases, it is not uncommon for external agents to store a unique application ID (say, the session ID of the first NAT session the application originated) in all sessions it spawns in its incarnation.

m. Controlling Agent IDentification

This indicates the last external Agent who has tried to control parameters for this session. A value of 0 indicates that native NAT is the responsible agent.

3. External agents interfacing with NAT

Many network applications assume the IP address of their host to be host Identifier and embed the Identifier information in application specific payload. When packets from such an application traverse NAT, the IP address of private host remains uncorrected in the payload, as the packet is delivered to hosts in external realm. An Application Level Gateway (ALG) is required to re-interpret such a payload as the payload traverses realms.

In addition, there are applications such as H.323 that use out-of-band signaling to dynamically create newer sessions. While a signaling session itself may be directed to a well-known port, sessions created by it need not be that way. Once again, an ALG may

be required to process payload in the signaling sessions and notify NAT to recognize the newly created sessions.

There may be other instances where an ALG may be required to provide application level transparency. Clearly, there is a need for a variety of ALGs to interface with NAT. The ALGs may reside on the same NAT device or an external device. Independent of this, the NAT interface requirement will remain the same.

In a multi-homed NAT configuration, there is a need for a backup NAT to communicate with the primary and keep in sync, so that when the primary goes away, the backup NAT could instantly assume support for the sessions that primary NAT was responsible for. This is yet another case where an external agent (i.e., backup NAT) has a need to interface with NAT.

A NAT device is uniquely qualified to serve as host-NAT-Server (or host-NAPT-Server) for host-NAT-clients (or host-NAPT-clients). [Ref 1] has a description of Host-NAT terminology. Host-NAT (and Host-NAPT) clients need to interface with the server node to obtain an external address (or a tuple of address and TCP/UDP port) while communicating with hosts in external realms. In addition, if NAT were to act as tunnel end-point, host-NAT clients will need to interface with NAT to setup tunnel state for the lifetime of Host-NAT-client address assignment. So, once again, there is a need for an API for use by an external agent(i.e., host-NAT-client) to communicate with NAT, acting as host-NAT-server.

Lastly, a mangement utility would be useful to interface with NAT for configuration and monitor purposes and to enforce NAT policies. For example, reconfigure a NAT device to switch over from NAPT to Basic-NAT configuration or vice versa. Or, add, terminate and monitor ALGs and other external agents on a NAT box. Such a program would also be useful to notify NAT about the status and setup information concerning ALGs, backup NATs and Host-NAT clients.

Clearly, agents such as Host-NAT-clients and Backup-NATs are likely to reside on a different physical device than the NAT device. Some of the ALG agents may also reside on an external device. The API presented in the follow-on section will provide a base to identify requirements for the development of one or more protocols by which each of these external agents could communicate with NAT. It may be a single protocol applicable to all external agents (or) multiple protocols, specific to each agent type.

The following diagram identifies a selected list of external agents that might interact with NAT using its API.

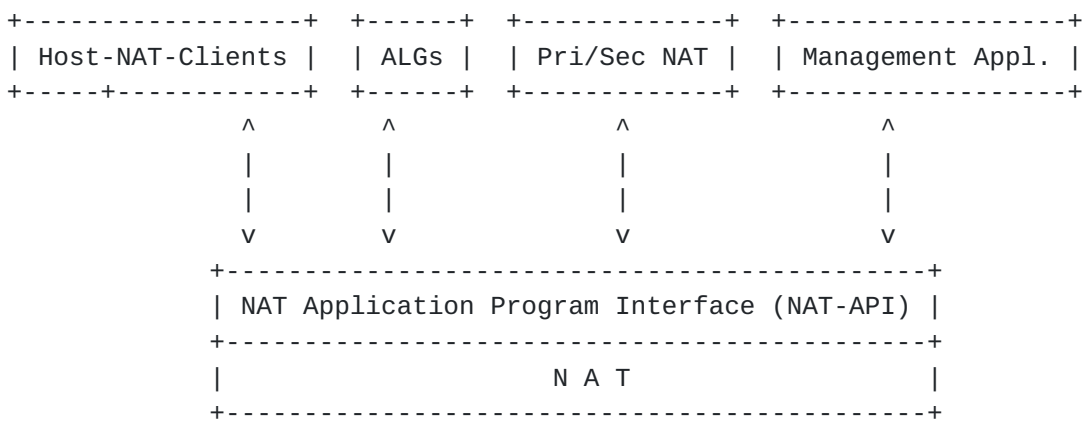

figure 1. External agents interfacing with NAT using NAT-API.

The following list of attributes uniquely identify an external agent with reference to a NAT.

a. Agent IDentifier

A number (say, in the range of 1 through 0xFFFFFFFF) assigned to the agent by the NAT device to distinguish from other agents. Typically, this handle may be assigned when the agent registers with NAT.

b. Agent type

Based on the categories of external agents described thus far, it is clear that the API requirements differ considerably amongst them. A native NAT API may or may not be able to support the requirements of all these agents. It is beneficial for NAT to know the agent type to be one of ALG or Host-NAT-Client or Backup-NAT or Management Application or something else, so it can accept or deny registration.

c. Agent call-back requirements

The agents will typically require NAT to invoke a call-back function within the agent when NAT notices the occurrence of an external event. But, the call-back requirements across the agents vary. For example, an ALG might require NAT to call back when a data packet is received on a session with a certain session-tag. But, other agents do not have such a requirement. There may , however, be some common requirements for call-back upon events such as termination of a session, termination of a Bind and termination of NAT itself. In addition, management applications and Backup-NAT

may have a requirement to have NAT periodically invoke a call-back function.

d. Agent call-back functions

Depending upon call-back requirements, the agent will be required to register one or more call-back function entry points with NAT. Below are three different call-back function prototypes.

```
Event notification - void agent_callback_event(nat_id,
                           agent_id, event_type, event_status)

Periodic notification - void agent_callback_periodic(nat_id,
                           agent_id, info_type, info_length,
                                                information)

Packet notification - void agent_callback_packet(nat_id,
                           agent_id, session_id,
                           pkt_direction, packet)
```

e. Periodic Notification interval

This parameter would be required only when the agent calls for periodic notification. This may be specified in units of seconds.

f. Host-NAT-Server tunnel type requirement

A Host-NAT-client may have a requirement for NAT, acting as Host-NAT-server to support a certain type of tunneling. In such a case, the agent will specify the tunneling requirement through this parameter.

g. Agent access information

In the case the agent is resident on a different physical device than NAT, this parameter is used by the agent to specify a means by which NAT can access the agent. This will include a combination of Agent's IP address, IP protocol (e.g., TCP or UDP), well-known port etc. As a special case, a value of 0 to agent_ip_address would indicate that the agent is on the same device as NAT.

4. NAT Application Programming Interface (NAT API)

The following API is specified in pseudo C language and is by no

means exhaustive in coverage. The API may vary from vendor to vendor. The intent is to provide a framework that could be expanded upon as required in the future. This section is divided into two sub-sections. The first sub-section lists function calls available to external agents. These calls are synchronous and require NAT to return back a value. The second sub-section lists functions that are expected to be provided by external agents in order for NAT to call-back upon some events.

4.1. NAT API functions

4.1.1. int nat_enquire_IDentity(nat_type, &natid_info)

Purpose:

This function is used by external agents to obtain NAT-ID and its characteristics, as described in section 2.1

Input parameters:

nat_type - This parameter is specified to verify if NAT device supports a certain flavor of NAT.

Output Parameters:

natid_info - NAT will fill up the natid_info data structure with its characteristics, as described in section 2.1. Also returned in this block would be an Identifier (nat_id) to uniquely identify this NAT.

Multiple pieces of this information may be returned, if NAT supports multiple instances of the same NAT type.

Return Value:

No-Error(0) - A return value of 0 implies success and that natid_info may be examined for NAT description.

NAT-TYPE-NOT-SUPPORTED - Notify the client that the requested NAT device does not support the specified NAT type.

4.1.2. int nat_enquire_address_bind (nat_id, pvt_address, ext_address, &bind_info)

Purpose:

This function is used by external agents to obtain
Address BIND information.

Input parameters:

nat_id - The identifier that uniquely identifies the NAT instance.

pvt_address, ext_address - The caller might specify both or just
one of either private address or external address and
set the other to zero.

Output Parameters:

bind_info - NAT will fill up the bind_info data structure
with info as described in section 2.2, if NAT were
to find a match for the addresses specified.

Return Value:

No-Error(0) - A return value of 0 implies success
in finding a match.

NO-MATCHING_BIND - Notify the client that there isn't a BIND
matching the specified addresses.

INVALID-NAT-ID - The specified NAT-ID is not operational
or is incorrect.

4.1.3. **int nat_enquire_transport_bind(nat_id, pvt_address, pvt_port, transport_protocol, ext_address, ext_port, &bind_info)**

Purpose:

This function is used by external agents to obtain
Transport ID BIND information.

Input parameters:

nat_id - The identifier that uniquely identifies the NAT instance.

pvt_address, pvt_port,
ext_address, ext_port - The caller might specify both or just
one of either (private address and the port no.) or
external address and the port number.

transport_protocol - This must be one of TCP, UDP or ICMP Query

Output Parameters:

bind_info - NAT will fill up the bind_info data structure with info as described in section 2.2, if NAT were to find a match for the addresses specified.

Return Value:

No-Error(0) - A return value of 0 implies success in finding a match.

NO-MATCHING_BIND - Notify the client that there isn't a BIND matching the specified addresses.

INVALID-NAT-ID - The specified NAT-ID is not operational or is incorrect.

4.1.4. int nat_enquire_sess_range(nat_id, agent_id, sessid_min, sessid_max, &sess_count, &sess_info)

Purpose:

This function is used by external agents to request NAT to send all valid session information for sessions with an ID in the range of sessid_min through sessid_max.

Input parameters:

nat_id - The identifier that uniquely identifies the NAT instance.

agent_id - The agent Identifier that uniquely identifies the agent to NAT.

sessid_min,
sessid_max - The range of session IDs that the agent is interested in knowing about.

Output Parameters:

sess_count - Number of sessions being returned through sess_info pointer.

sess_info - Return one or more sessions maintained by NAT, with an ID in the given range.

Return Value:

No-Error(0) - A return value of 0 implies successful session termination.

INVALID-NAT-ID - The specified NAT-ID is not operational or is incorrect.

INVALID-AGENT-ID - The specified Agent-ID is not currently registered with NAT.

4.1.5. int nat_register_agent (nat_id, &agent_info)

Purpose:

This function is used by external agents to register with NAT.

Input parameters:

nat_id - The identifier that uniquely identifies the NAT instance.

agent_info - The agent is required to provide all the requisite information (with the exception of agent_id) as described in section 3.0. This ID may be used by the caller to control and influence NAT operation.

Output Parameters:

agent_info - NAT will return the agent_id in agent_info structure when registration is successful.

Return Value:

No-Error(0) - A return value of 0 implies successful registration.

AGENT-TYPE-NOT-SUPPORTED - Notify the caller that NAT does not support API requirements of the agent.

TUNNEL-TYPE-NOT-SUPPORTED - Notify the caller that NAT does not support Host-NAT tunnel type requested.

INVALID-NAT-ID - The specified NAT-ID is not operational or is incorrect.

4.1.6. int nat_set_bind (nat_id, agent_id, &bind_info)

Purpose:

This function is used by external agents to create a new Address Bind or set certain parameters of an existing Bind.

Input parameters:

nat_id - The identifier that uniquely identifies the NAT instance.

agent_id - The agent Identifier that uniquely identifies the agent to NAT.

bind_info - The caller supplies the specifics of a new BIND or sets a selected number of parameters of an existing BIND to influence NAT operation. The BIND can be an address BIND or transport BIND. A new BIND request is made by setting the BIND ID within bind_info structure to 0. A non-Zero Bind-ID would be interpreted by NAT to mean that the agent is attempting to set some BIND parameters.

Output Parameters:

bind_info - If the caller requested for a BIND creation and NAT was successful in creating a new BIND, NAT will fill the structure with the assigned BIND ID and any other NAT assigned parameter values. If the caller requested to set some BIND parameters and NAT succeeded in doing so, the bind_info would be filled with the values that NAT holds.

Return Value:

No-Error(0) - A return value of 0 implies successful BIND creation or parameter setting.

BIND-MAKE-FAILED - When NAT was unable to create BIND or was unable to set the requested parameter(s).

INVALID-BIND-INFO - When NAT finds that one or all of the parameters specified is not valid.

INVALID-NAT-ID - The specified NAT-ID is not operational or is incorrect.

INVALID-AGENT-ID - The specified Agent-ID is not currently

registered with NAT.

4.1.7. int nat_set_sess(nat_id, agent_id, &sess_info)

Purpose:

This function is used by external agents to create a new session state or set certain parameters of an existing session.

Input parameters:

nat_id - The identifier that uniquely identifies the NAT instance.

agent_id - The agent Identifier that uniquely identifies the agent to NAT.

sess_info - The caller supplies the specifics of a new session parameters or sets a selected number of parameters of an existing session to influence NAT operation. A new session request is made by setting the session-ID within sess_info structure to 0. A non-Zero session-ID would be interpreted by NAT to mean that the agent is attempting to set some session specific parameters.

Output Parameters:

sess_info - If the caller requested for a session creation and NAT was successful in creating a new session, NAT will fill the structure with the assigned session-ID and any other NAT assigned parameter values. If the caller requested to set some session parameters and NAT succeeded in doing so, the sess_info would be filled with the values that NAT holds.

Return Value:

No-Error(0) - A return value of 0 implies successful session creation or parameter setting.

SESS-MAKE-FAILED - When NAT was unable to create session or was unable to set the requested parameter(s).

INVALID-SESS-INFO - When NAT finds that one or all of the parameters specified is not valid.

INVALID-NAT-ID - The specified NAT-ID is not operational or is incorrect.

INVALID-AGENT-ID - The specified Agent-ID is not currently registered with NAT.

4.1.8. int nat_free_bind(nat_id, agent_id, bind_id)

Purpose:

This function is used by external agents to terminate the specified BIND and any sessions that are based on this BIND.

Input parameters:

nat_id - The identifier that uniquely identifies the NAT instance.

agent_id - The agent Identifier that uniquely identifies the agent to NAT.

bind_id - The ID of the BIND that needs to be terminated.

Output Parameters:

none.

Return Value:

No-Error(0) - A return value of 0 implies successful BIND termination.

INVALID-BIND-ID - The specified BIND ID does not exist.

INVALID-NAT-ID - The specified NAT-ID is not operational or is incorrect.

INVALID-AGENT-ID - The specified Agent-ID is not currently registered with NAT.

4.1.9. int nat_free_sess(nat_id, agent_id, sess_id)

Purpose:

This function is used by external agents to terminate the specified session.

Input parameters:

nat_id - The identifier that uniquely identifies the NAT instance.

agent_id - The agent Identifier that uniquely identifies the agent to NAT.

sess_id - The ID of the session that needs to be terminated.

Output Parameters:

none.

Return Value:

No-Error(0) - A return value of 0 implies successful session termination.

INVALID-SESS-ID - The specified session ID does not exist.

INVALID-NAT-ID - The specified NAT-ID is not operational or is incorrect.

INVALID-AGENT-ID - The specified Agent-ID is not currently registered with NAT.

4.1.10. int nat_free_sess_bundle(nat_id, agent_id, bundle_id)

Purpose:

This function is used by external agents to terminate a bundle of sessions identified by the same bundle ID.

Input parameters:

nat_id - The identifier that uniquely identifies the NAT instance.

agent_id - The agent Identifier that uniquely identifies the agent to NAT.

bundle_id - The ID of the session bundle (group of sessions) that needs to be terminated.

Output Parameters:

none.

Return Value:

No-Error(0) - A return value of 0 implies successful session termination.

INVALID-BUNDLE-ID - The specified bundle ID does not exist.

INVALID-NAT-ID - The specified NAT-ID is not operational or is incorrect.

INVALID-AGENT-ID - The specified Agent-ID is not currently registered with NAT.

4.2. Call-back functions within an external agent

4.2.1. void agent_callback_event(nat_id, agent_id, event_type, &event_info)

Purpose:

This function is used by NAT to notify an agent of an event status.

Input parameters:

nat_id - The identifier that uniquely identifies the NAT instance.

agent_id - The agent Identifier that uniquely identifies the agent to NAT.

event_type - The event can be one of BIND creation, BIND termination, session Creation, and session termination.

event_info - This will return the BIND or session description structure that contains the specific instance identifier and other pertinent information.

4.2.2. void agent_callback_periodic(nat_id, agent_id, info_type, info_length, &periodic_info)

Purpose:

This function is used by NAT to notify an agent of a certain piece of information periodically.

Input parameters:

nat_id - The identifier that uniquely identifies the NAT instance.

agent_id - The agent Identifier that uniquely identifies the agent to NAT.

info_type - NAT may have been requested to periodically notify the agent many types of information. Possible values for this parameter would be statistics update, Incremental BIND update Incremental session update, Incremental BIND termination, Incremental session termination etc..

info_length- Number of bytes included in periodic info block.

periodic_info - This point to the actual periodic information being sent to the agent.

4.2.3. void agent_callback_packet(nat_id, agent_id, sess_id, pkt_direction, packet)

Purpose:

This function is used by NAT to notify an agent of a data packet for processing. The agent is expected to process the packet and forward to the actual destination in the first-in-first-out (FIFO) order. The processing performed by the agent may be limited to just the payload or the entire packet, as set by the agent at session setup time.

Input parameters:

nat_id - The identifier that uniquely identifies the NAT instance.

agent_id - The agent Identifier that uniquely identifies the agent to NAT.

sess_id - The Identifier if NAT session to which the packet belongs.

pkt_direction - This can be inbound or outbound.

packet - IP packet that needs to be processed by the agent. If NAT was required to perform header translation, this packet is post-NAT-translated version of the packet. In the case the agent selected to perform the entire translation, the original packet is sent as is to the agent, without any NAT transformation.

5. Acknowledgement

The author would like to express sincere appreciation and thanks to Yakov Rekhter for his valuable advice and contribution in the presentation of this document.

6. Security considerations.

The security considerations described in [Ref 1] for all variations of NATs are applicable here.

REFERENCES

[1] P. Srisuresh, M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", <draft-ietf-nat-terminology-01.txt> - Work in progress.

[2] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. de Groot, and, E. Lear, "Address Allocation for Private Internets", RFC 1918

[3] J. Reynolds and J. Postel, "Assigned Numbers", RFC 1700

[4] R. Braden, "Requirements for Internet Hosts -- Communication Layers", RFC 1122

[5] R. Braden, "Requirements for Internet Hosts -- Application and Support", RFC 1123

[6] F. Baker, "Requirements for IP Version 4 Routers", RFC 1812

[7] J. Postel, J. Reynolds, "FILE TRANSFER PROTOCOL (FTP)", RFC 959

[8] "TRANSMISSION CONTROL PROTOCOL (TCP) SPECIFICATION", RFC 793

[9] J. Postel, "INTERNET CONTROL MESSAGE (ICMP) SPECIFICATION",

RFC 792

[10] J. Postel, "User Datagram Protocol (UDP)", RFC 768

[11] J. Mogul, J. Postel, "Internet Standard Subnetting Procedure", RFC 950

[12] Brian carpenter, Jon Crowcroft, Yakov Rekhter, "IPv4 Address Behaviour Today", RFC 2101

Author's Address:

Pyda Srisuresh
Lucent technologies
4464 Willow Road
Pleasanton, CA 94588-8519
U.S.A.

Voice: (925) 737-2153
Fax: (925) 737-2110
EMail: suresh@ra.lucent.com