SIPPING WG
Internet-Draft
Expires: August 20, 2005

C. Jennings, Ed.
Cisco Systems
A. Hawrylyshen
Jasomi Networks
February 19, 2005

SIP Conventions for UAs with Outbound Only Connections
draft-jennings-sipping-outbound-01

Status of this Memo

This document is an Internet-Draft and is subject to all provisions of section 3 of RFC 3667. By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she become aware will be disclosed, in accordance with RFC 3668.

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 August 20, 2005.

Copyright Notice

Copyright (C) The Internet Society (2005).

Abstract

Often with SIP a request can only be routed over an existing connection or flow, such as when there is a firewall or network address translation (NAT) device in the network path. TLS is also affected when the user agent (UA) does not have a certificate suitable for mutual TLS authentication. This draft addresses how user agents and proxies need to behave to work in these environments.

This work shows how existing SIP mechanisms can be used to allow the UA to register multiple times over different connections or flows and the proxies can use the instance-id in the contact header to identify that the multiple flows go to the same UA. It can then choose which flow to use to route requests to this UA.

This work is being discussed on the sipping@ietf.org mailing list.

Table of Contents

1. Introduction

There are many environments for SIP deployments in which the user-agent (UA) can form a connection to the Registrar or Proxy but in which the connections in the reverse sense are not possible. This can happen for several reasons. It is important to understand that most IP phones and and soft-phones get their network configurations via a host-configuration protocol such as DHCP; they typically do not have a useful name in DNS; and they definitely do not have a long-term, stable DNS name that is appropriate for binding to a certificate. It is impractical for them to have a certificate that can be used as a client-side TLS certificate. However, they do support TLS and form TLS connections to a proxy or registrar which the UA authenticates using TLS, and the server authenticates the UA using a digest challenge.

Sometimes a firewall device between the UA and proxy or registrar will only allow connections in the "outbound" direction. Similarly there may be a NAT that is only capable of allowing connections in the "outbound" direction. It is worth noting that most UAs in the world are deployed behind a firewall or NAT.

This document describes several concepts that are used to solve this problem using a key idea from the connection reuse draft [10]: A proxy that wishes to route a request to a particular AOR, say alice@example.com, may use any connection to Alice's UA which has been previously authenticated at an appropriate level to allow it to change the registration bindings for Alice.

Secondly, for high reliability systems, the UA needs to keep a connection to the proxy or registrar that it can use at any time. This is achieved by having the UA keep multiple connections, referred to as "flows", to the proxy or registrar and using a keep alive mechanism on each flow so that the UA can detect when it has failed and establish a new one.

The overall approach can be summarized simply: UAs use a keep alive mechanism to keep their flow to the proxy or registrar alive. For TCP, TLS, and other connection oriented protocols this is a burst containing a CRLF payload, and for UDP it is a STUN request over the flow. A UA can create more than one flow using multiple registrations for the same contact and AOR. The instance in the contact is used to identify the UA that a connection is associated with. A new contact parameter called flow-id is used to allow the proxy and registrar to tell the difference between a UA re-registering and registering an additional connection. The proxies keep track of the "flows" or connection mappings for successful registrations.

When a proxy goes to route a message to a UA for which it has a mapping, it can use any one of the flows on which a successful registration has been completed for that contact. A failure on a particular flow can be tried again on an alternate flow.

2. Requirements

Must be able to detect that a UA supports these mechanisms.

Support UAs behind NATs.

Support UAs behind firewalls.

Support TLS to a UA without stable DNS name or IP.

Detect failure of connection and be able to correct for this.

Support many UAs simultaneously rebooting.

Support a NAT rebooting or resetting.

Support proxy farms with multiple hosts for scaling and reliability purposes.

Minimize initial startup load on a proxy.

3. Conventions & 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 RFC 2119 [1].

3.1 Definitions

'Edge Proxy': An Edge Proxy is any proxy that is located topologically between the registering user agent and the registrar.
'Flow': A Flow is a network protocol level connection between two endpoints that is represented by the network address of both ends and the protocol. For TCP and UDP this would include the IP addresses and ports of both ends and the protocol (TCP or UDP). With TCP, a flow would often correspond with a single file descriptor in the OS.
'Outbound Connection': An Outbound Connection is a connection between two network elements that can only be established by one party. Typically this is due to network policy from a firewall or NAT device or to issues with TLS where one end does not have a certificate that can be used as a server certificate so cannot act as a TLS server.

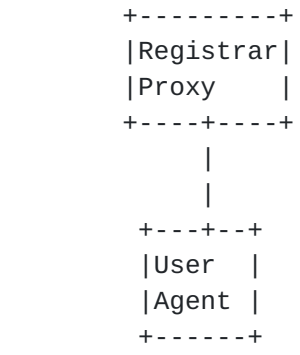
'Third party registration ': A third party registration is defined in section 10.2 of RFC 3261. It is a REGISTER request in which the value of the To header field is not the same as that of the From header field.

4. Overview

Several scenarios in which this technique is useful are discussed below, including the simple collocated registrar and proxy, a user agent desiring multiple connections to a resource (for redundancy for example), and an system that uses Edge Proxies. This section explains the details of the approach while section (Section 5) has the exact details of how various elements handle messages.

4.1 Single Registrar and UA

The network's topology in this example is that there is single server acting as a registrar and proxy, with which the user agent registers.



User Agents forming only a single connection continue to register in the normal way but include the instance identifier as described in the GRUU [8] and can also add a flow-id parameter to the Contact header field value. The flow-id parameters are used to allow the registrar to detect and avoid using invalid contacts when a UA reboots, as described later in this section.

For clarity, here is an example. Alice's UA creates a new TCP flow to the registrar and sends the following register.

```
REGISTER sip:example.com SIP/2.0
Via: SIP/2.0/UDP 192.0.2.1;branch=z9hG4bK-bad0ce
Max-Forwards: 70
From: Bob <sip:bob@example.com>;tag=d879h76
To: Bob <sip:bob@example.com>
Call-ID: 8921348ju72je840.204
CSeq: 1 REGISTER
Contact: <sip:line1@192.168.0.2>; flow-id=1;
  ;+sip.instance="<urn:uuid:00000000-0000-0000-0000-000A95A0E128>"
Content-Length: 0
```

The registrar would, as usual, challenge this registration to authenticate the sender. When the registrar adds an entry for this contact under the AOR for Bob, the registrar also needs to form a sublist under this contact that keeps track of the flow that received this registration and the flow-id value.

Later when Alice sends a request to Bob at the registrar, it acts as a proxy and follows [2] to select a contact to forward the request to. The proxy then looks and finds the flows that have registrations to this contact. It forwards the request on that flow instead of trying to form a new flow to that contact. This allows the proxy to forward a request to a particular contact down the same flow that did the registration for this AOR. If the proxy had multiple flows that all went to this contact, it could choose any one of them that had registered for this AOR and had the same instance value as the selected contact. In general, if two flows have the same flow-id value, the proxy would favor the most recently registered flow. This is so that if a UA reboots, the proxy will prefer to use the most recent flow that goes to this UA instead of trying one of the old flows which will presumably fail.

The keep alive mechanism needs to detect both failure of a connection and changes to the NAT public mapping. When a residential NAT is rebooted, the UA needs to understand that its bindings are no longer valid and it needs to reregister. Simply sending keep alive packets will not detect this failure when using UDP. With connection oriented transports such as TCP or TLS, the keep alive will detect failure after a NAT reboot. Connection oriented transport failures are detected as the UA periodically sends a CRLF over the connection; if the connection has failed, a connection level error will be reported to the UA. A CRLF can be considered the beginning of the next message that will be sent and therefore this approach is backwards compatible with existing standards. The TCP KEEP_ALIVE mechanism is not used because many operating systems do not allow this to be set on a per connection basis.

The keep alive mechanism for UDP is quite different. The UA needs to

detect when the connection is working but also when the flow definition has changed. A flow definition could change because a NAT device in the network path reboots and the resulting pubic address mapping for the UA changes. STUN [4] requests are sent over the connection that is being used for the UDP SIP traffic. The proxy or registrar acts as a STUN server on the SIP signaling port.

The STUN mechanism is very robust and allows the detection of a changed IP address. It may also be possible to do this with OPTIONS messages and rport; although this approach has the advantage of being backwards compatible, it also significantly increases the load on the proxy or registrar server.

If the UA detects that the connection has failed or that the flow definition has changed, it will re-register using a back-off mechanism described in Section 5.1 in order to provide congestion relief when a large number of agents simultaneously reboot.

The registrar saves the instance id (as defined in GRUU [8]) and flow-id (as defined in Section 6) along with the rest of the contact Header. If the instance id and flow-id are the same as a previous registration, the proxy or registrar can decide to fork requests to these contacts' registrations in serial and to choose to use the most recently created registration first. This allows a UA that has rebooted to replace its previous registration for each flow with minimal impact on overall system load.

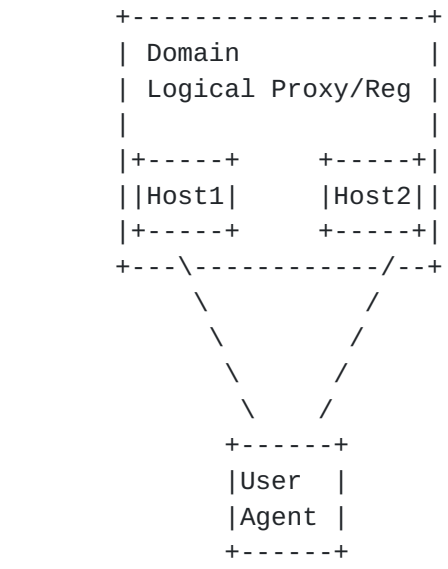
If the TCP flow to the registrar is closed, any map entries referring to that flow must be removed. Similarly, if the registration expires, any map entries created by it need to be removed.

A note about the UUID: a device like a soft-phone, when first installed, should generate a UUID [6] and then save this in persistent storage for all future use. For a device such as a hard phone, which will only ever have a single SIP UA present, the UUID can be generated at any time because it is guaranteed that no other UUID is being generated at the same time on that physical device. This means the value of the time component of the UUID can be arbitrarily selected to be any time less than the time when the device was manufactured. A time of 0 (as shown in the example) is perfectly legal as long as the device knows no other UUIDs were generated at this time.

4.2 Multiple Connections from a User Agent

In this example system, the logical proxy/registrar for the domain is running on two hosts that share appropriate state and can both provide registrar and proxy functionality for the domain. The UA

will form connections to two of the physical hosts for the domain.



A UA that forms two or more flows has similar behavior to a UA that forms a single connection but has some additional requirements. The UA MAY be configured with a primary and backup outbound proxy or it MAY select two flows to form using the DNS selection mechanism described in this section. The registration on each flow needs to contain the instance identifier from the GRUU mechanisms and also needs to add a different flow-id parameter to the Contact header so that the Registrar can differentiate the flows as being distinct connections from the same instance. For example, the flow-id value might be set to 1 for the primary connection and 2 for the backup connection.

A UA that needs to establish multiple flows needs a way to use DNS to select candidate addresses for the formation of flows. The recommended way to do this is to look at the DNS records resulting from the algorithm described in RFC 3263 [3] and select distinct addresses from the target set.

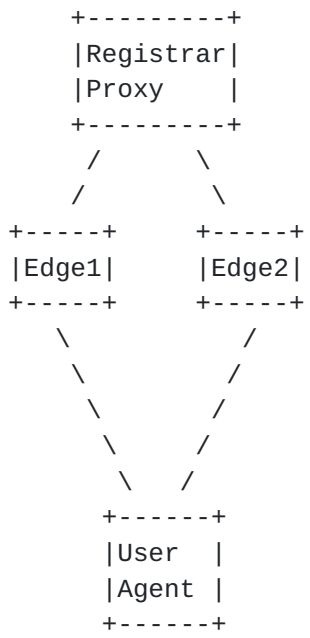
Hosts that are multi-homed can avoid complications by ensuring that interfaces that are in separate routing domains have distinct DNS names for each routing domain. Having different SRV records point to the same host record should also be avoided when deploying proxies. Multiple interfaces in a single network should either be absent from DNS or preferably share an address. These guidelines will help prevent a UA from establishing flows that connect to the same resource and thereby unintentionally eliminating the desired redundancy.

When a proxy goes to route a call to a particular contact, it can use

the flow for any registration to that contact that has the same instance value of the selected contact. If it detects that a flow has failed, it needs to remove that mapping and use the others.

4.3 Edge Proxies

Some SIP deployments use edge proxies such that the UA sends the REGISTER to an edge proxy that then forwards the REGISTER to the Registrar. The edge proxy can include a path header as defined in RFC 3327 [9] so that when the registrar later retargets a request to this UA, the request is routed through the edge proxy.



These systems can use effectively the same mechanism as described in the previous sections but need to use the Path header. When the edge proxy receives a registration, it needs to create an identifier value that is unique to this AOR, contact, flow, and instance-id and put this identifier in the path header. This is done by putting the value in the user portion of a loose route in the path header. If the registration succeeds, the edge proxy needs to map future requests that are routed to the identifier value that was put in the path header to the associated flow. The edge proxy needs to ensure that a 200 response to a register request represents a successful registration and not some spoofed traffic to the edge proxy. One way this can be done is by ensuring that it only pays attention to responses received over a TLS connection from a proxy that is authoritative for the domain of the registration.

As an alternative to actually storing the state for the mapping in the edge proxy, the proxy can form an encrypted version of the flow

identifier and put it in the path header so that the edge proxy will get it back from the registrar at the time it needs it.

5. Mechanisms

5.1 User Agent

User Agents MUST support the the instance identifier as described in the GRUU [8] mechanism. If the UA detects that the binding on a NAT has changed, it MUST treat this as a connection failure and re-register. When registration fails due to a network problem or the Registrar does not respond, the UA maintains a range value for computing when it should next attempt to register. This range value SHOULD have an initial value of 1 minute and SHOULD double after each consecutive failed registration attempt, up to a maximum of 30 minutes. When a registration fails due to network problems, the UA MUST randomly select a time to re-register that is between 50 and 100 percent of the current range value.

User Agents that form two or more connections behave similarly to User Agents that form single connections but also have some additional requirements. All User Agents SHOULD support forming multiple connections. The UA MAY be configured with a primary and backup outbound proxy. It MUST support selecting at least two connections using the mechanism described in Location of SIP Servers [3]. When DNS is used, the UA finds IP addresses used for registration the normal way, but if it discovers more than one possible IP address, it SHOULD connect to two distinct addresses, among the possible IP addresses. If the UA finds multiple records that correspond to different A or AAAA records, it SHOULD select address corresponding to different A or AAAA records.

Each connection MUST contain the instance identifier from the GRUU mechanisms but MUST also add a distinct flow-id parameter to the contact header field value so that the Registrar can differentiate the two connections as being from the same instance but different connections. The flow-id MAY be set to 1 for the primary connection and 2 for the backup connection. Each time the UA is rebooted, it SHOULD use the same flow-id values it previously used.

On connection oriented transports such as TCP or TLS, if no other traffic has been sent for 600 seconds, then the UA MUST send a CRLF to detect whether the connection has failed. On UDP connections, the UA MUST send a STUN [4] request every 30 seconds over the same flow as the SIP signaling. If the UA detects that the flow has changed, it MUST reregister.

The text in this section does not apply to third party registration.

A UA doing third party registration would proceed as described in RFC 3261.

User Agents that form flows with stream oriented protocols such as TCP, TLS, or SCTP SHOULD periodically send a CRLF over the connection to detect liveness of the flow. It is RECOMMENDED that a CRLF be sent if the flow has not had any data sent or received in the previous 600 seconds. The UA SHOULD not send a CRLF if data has been sent or received in the previous 30 seconds.

User Agents that form flows with UDP SHOULD perform STUN requests over the flow every 30 seconds. If the mapped address in the STUN response changes, the UA must treat this as a transport error on the flow. This will cause the UA to form a new registration on a new flow.

5.2 Registrar

The registrar MUST check if the registration is a 3rd party registration. If so it would process it normally and none of the other text in this section would apply.

When a registrar receives a registration that does not contain a path header, it processes the registration as normal; and if the registration is successful, the registrar MUST store the flow-id, instance value and reference to the flow to a list that is maintained for this particular AOR and contact. The reference to flow is the information it needs to send a message back to the UA over this same flow. Typically it would be something like the file descriptor for TCP or the full source and destination addresses and ports for UDP.

This document does not require a registrar to support the path header, but if registrar does there is some special processing based on the path header. Specifically, when a registrar that supports path receives a registration that contains a path header, the registrar still stores the instance value and flow id but does not save the reference information to the flow.

When the registrar, acting as a proxy, proxies a request to a particular contact, it selects a contact to use in the normal way. Next the registrar selects a flow to reach this contact. It forms the list of possible flows by looking at the contacts registered for this AOR and selecting the ones that have the same instance value. The registrar MAY decide, when two flows have the same flow-id, to choose the one that registered most recently. Once a flow is selected, the registrar needs to forward to that flow. If a path header was used for the registration of that flow, the registrar populates the request in the normal way and forwards it. If there

was no path header, then the registrar looks at the flow reference information for that flow and forwards the request over the same flow that was used to receive the registration.

If the registrar receives a transport level error using this flow, it must remove the flow and any associated registration information.

5.3 Edge Proxy

When an edge proxy receives a registration request that does not contain a path header, it MUST form a registration identifier that is unique to this flow and then include that identifier in the path header it adds to the registration. This is done by inserting the unique identifier in the user portion of the path header URI for this edge proxy.

If the edge proxy receives a request that is routed to a registration identifier that it has created, then it MUST forward the request on the flow that created the registration. This can be implemented either by storing the mapping from the unique identifier to the flow when the identifier is created or, if the edge proxy does not wish to save this state information, it can take the flow reference information and encrypt it with a secret known only to the edge proxy and put it in the user portion of the path header. Later when the edge proxy receives a request, it can decrypt this information and use it to route the request over the correct flow.

The edge proxy MUST ensure that data sent from the edge proxy to the registrar or other edge proxies is integrity protected against attackers. This could be achieved by using TLS between the edge proxy and the registrar.

5.4 Receivers of REGISTER Requests

A device that receives register requests directly from a UA needs to behave as specified in this section. Such devices would generally include a Registrar and and an Edge Proxy, as they both receive register requests directly from a UA.

If the server receives UDP SIP requests on a given interface and port, it MUST also provide a limited version of the STUN server on the same interface and port. Specifically it MUST support all of STUN with the exception that it does not need to support STUN requests with the changed port or changed address flag set. This allows the STUN server to run with only one port and IP address.

It is easy to demux STUN and SIP packets because the first byte of a STUN packet is 0 or 1 while the first byte of a SIP packet is in the

range of 'A' to 'Z'.

6. Grammar

This specification defines a new Contact header field parameter, flow-id. The grammar for pvalue and EQUAL is obtained from RFC 3261 [2].

```
contact-params = c-p-q / c-p-expires / contact-extension
                 c-p-connection
c-p-connection = "flow-id" EQUAL pvalue
                 ; defined in RFC3261
```

7. IANA

This specification defines a new Contact header field parameter called flow-id, as per the registry created by [5]. The required information is as follows:

Header field in which the parameter can appear: Contact

Name of the Parameter: flow-id

RFC Reference: RFC AAAA [NOTE TO IANA: Please replace AAAA with the RFC number of this specification.]

8. Security Considerations

One of the key security concerns in this work is making sure that an attacker cannot hijack the sessions of a valid user and cause all calls destined to that user to be sent to the attacker.

The simple case is when there are no edge proxies. In this case, the only time an entry can be added to the routing for a given AOR is when the registration succeeds. SIP protects against attackers being able to successfully register, and this scheme relies on that security. Some implementers have considered the idea of just saving the instance without relating it to the AOR with which it registered. This idea will not work because an attacker's UA can impersonate a valid user's instance value and hijack that user's calls.

The more complex case involves one or more edge proxies. The only time an edge proxy will route over a particular flow is when it has received a route header that has the instance information it has

created. An incoming request would have gotten this information from the registrar. The registrar will only save this information for a given AOR if the registration for the AOR has been successful; and the registration will only be successful if the UA can correctly authenticate. Even if an attacker has spoofed some bad information in the path header sent to the registrar, the attacker will not be able to get the registrar to accept this information for an AOR that does not belong to the attacker. The registrar will not hand out this bad information to others, and others would not be misled into contacting the attacker.

9. Changes from 00 Version

Changed the behavior of the proxy so that it does not automatically remove registrations with the same instance and flow-id but instead just uses the most recently created registration first.

Changed the connection-id to flow-id.

Fixed expiry of edge proxies and rewrote mechanism section to be clearer.

10. Acknowledgments

Rohan Mahy had the insight key to this draft, that registration can be used to authorize connection reuses. Dave Oran came up with the idea of using the most recent registration first in the proxy. The TCP design team consisting of Chris Boulton, Scott Lawrence, Rajnish Jain, Vijay K. Gurbani, and Ganesh Jayadevan provided input. Additionally, many of the concepts here originated at a connection reuse meeting at IETF 60 that included Jon Peterson, Jonathan Rosenberg, Paul Kyzivat and Rohan Mahy.

11. References

11.1 Normative References

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

[2] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[3] Rosenberg, J. and H. Schulzrinne, "Session Initiation Protocol (SIP): Locating SIP Servers", RFC 3263, June 2002.

[4] 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.

[5] Camarillo, G., "The Internet Assigned Number Authority (IANA) Header Field Parameter Registry for the Session Initiation Protocol (SIP)", draft-ietf-sip-parameter-registry-02 (work in progress), June 2004.

[6] Mealling, M., "A UUID URN Namespace", draft-mealling-uuid-urn-03 (work in progress), March 2004.

[7] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

[8] Rosenberg, J., "Obtaining and Using Globally Routable User Agent (UA) URIs (GRUU) in the Session Initiation Protocol (SIP)", draft-ietf-sip-gruu-02 (work in progress), July 2004.

[9] Willis, D. and B. Hoeneisen, "Session Initiation Protocol (SIP) Extension Header Field for Registering Non-Adjacent Contacts", RFC 3327, December 2002.

11.2 Informative References

[10] Mahy, R., "Connection Reuse in the Session Initiation Protocol (SIP)", draft-ietf-sip-connect-reuse-02 (work in progress), July 2004.

[11] Mahy, R., "Requirements for Connection Reuse in the Session Initiation Protocol (SIP)", draft-ietf-sipping-connect-reuse-reqs-00 (work in progress), October 2002.

Authors' Addresses

Cullen Jennings (editor)
Cisco Systems
170 West Tasman Drive
Mailstop SJC-21/2
San Jose, CA 95134
USA

Phone: +1 408 902-3341
EMail: fluffy@cisco.com

Alan Hawrylyshen
Jasomi Networks
310, 602 - 11 Ave SW
Calgary, Alberta T2R 1J8
Canada

Phone: +1 866 617 8647
EMail: alan@jasomi.com

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Copyright Statement

Copyright (C) The Internet Society (2005). This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.