Abstract

The Route header field in the Session Initiation Protocol (SIP) is used to cause a request to visit a set of hops on its way towards the final destination. Several specifications have defined rules for how a user agent obtains and then uses a set of Route header fields in the transmission of a request. These include the SIP specification itself, the Service-Route header field specification, the SIP server option in the Dynamic Host Configuration Protocol (DHCP), and others. Unfortunately, these specifications are not consistent and the
resulting behavior at clients and servers is not clear or complete. This document resolves this problem by defining a consistent set of logic.

Table of Contents

1. Introduction .............................................. 3
2. Existing Sources ......................................... 3
   2.1 Default Outbound Proxies ............................... 3
   2.2 Service Route ........................................... 4
   2.3 Record-Routes .......................................... 4
   2.4 305 Use Proxy .......................................... 4
3. Problems with Current Specifications ......................... 5
4. Overview of Operation ....................................... 7
5. Detailed Processing Rules .................................... 9
   5.1 Registrar Behavior ....................................... 9
   5.2 Proxy Behavior ......................................... 10
   5.3 UAC Behavior ........................................... 11
   5.4 Client Behavior ........................................ 13
   5.5 Server Behavior ........................................ 14
6. Grammar ..................................................... 14
7. Security Considerations ..................................... 15
8. IANA Considerations ........................................ 15
9. Examples .................................................... 15
10. Acknowledgements .......................................... 15
11. References ................................................ 15
   11.1 Normative References .................................. 15
   11.2 Informative References ................................. 16
   Author’s Address .......................................... 16
   Intellectual Property and Copyright Statements .......... 17
1. Introduction

The Route header field in the Session Initiation Protocol (SIP) protocol is used to cause a request to visit a set of hops on its way towards the final destination. RFC 3261 [2] discusses how a client constructs the Route header field for requests. However, this logic is restricted to mid-dialog requests, where the route set was learned as a result of record-routing.

However, additional sources of routes can exist for a UA. These include default outbound proxies, a service route learned from the Service-Route header field [3], and a redirection coming from a 305 response. In total, there are four sources of potential route headers. The way in which these various sources are reconciled is unclear. Furthermore, the various specifications are unclear about which requests these Route headers are applicable to. Do they apply to REGISTER? Do they apply to mid-dialog requests? Finally, the existing specifications are incomplete and inconsistent.

Section 2 reviews the existing sources of route sources. Section 3 discusses problems with the existing specifications. Section 4 overviews the proposed changes in behavior. Section 5 provides a detailed description of element behavior, and Section 6 defines the grammar for the new headers specified here.

2. Existing Sources

This section examines the current set of route header field sources.

2.1 Default Outbound Proxies

RFC 3261 discusses default outbound proxies. In Section 8.1.1.1, it makes reference to its interaction with Route header fields:

In some special circumstances, the presence of a pre-existing route set can affect the Request-URI of the message. A pre-existing route set is an ordered set of URIs that identify a chain of servers, to which a UAC will send outgoing requests that are outside of a dialog. Commonly, they are configured on the UA by a user or service provider manually, or through some other non-SIP mechanism. When a provider wishes to configure a UA with an outbound proxy, it is RECOMMENDED that this be done by providing it with a pre-existing route set with a single URI, that of the outbound proxy.

When a pre-existing route set is present, the procedures for populating the Request-URI and Route header field detailed in Section 12.2.1.1 MUST be followed (even though there is no
dialog), using the desired Request-URI as the remote target URI.

The default outbound proxy can be learned either through DHCP [6], through configuration (such as the SIP configuration framework [8]), or through other means. In the IP Multimedia Subsystem (IMS), the default outbound proxy is the P-CSCF and is learned through GPRS specific techniques.

RFC 3261 does not explicitly say the set of messages to which this route set applies. However, the text above implies that it is for all requests outside of a dialog.

2.2 Service Route

RFC 3608 specifies the Service-Route header field. This header field is provided to the UA in a 2xx response to a REGISTER request. The client uses this to populate its Route header fields for outgoing requests. However, RFC 3608 explicitly says that the decision a UA makes about how it combines the service route with other outbound routes is a matter of local policy. Furthermore, RFC 3608 does not clearly define to which requests the service route applies, and in particular, whether or not it applies to a REGISTER request or a mid-dialog request.

Furthermore, RFC 3608 specifies that a service-route is associated with an Address-of-Record (AOR), and is shared by all contacts associated with the same AOR. It also specifies that the Service-Route URI can only be ones known to the registrar apriori, as opposed to learned through the registration itself.

2.3 Record-Routes

RFC 3261 provides a detailed description of the record-routing mechanism, and how the user agents in a dialog construct route sets from the Record-Route header field values. RFC 3261 is also clear that the resulting route set applies to mid-dialog requests. It implies (though does not explicitly say) that the resulting route set overrides any default outbound proxies (which represent a pre-loaded route set).

2.4 305 Use Proxy

RFC 3261 defines the 305 "Use Proxy" response code, but says extremely little about exactly how it is used. It has this to say:

The requested resource MUST be accessed through the proxy given by the Contact field. The Contact field gives the URI of the proxy. The recipient is expected to repeat this single request via the
proxy. 305 (Use Proxy) responses MUST only be generated by UASs.

It is unclear how the Contact in the redirect is used. Does it populate the request URI of the resulting request? Or, does it get used to populate the Route header field? The restriction to UASs is also not explained.

Historically, the reason for the restriction to UAs was to avoid routing loops. Consider an outbound proxy that generates a 305, instead of proxying the request. The concern was that the client would then recurse on the response, populate the Contact into a new request URI, and send the request to its default outbound proxy, which redirects once more. To avoid this, the RFC says that only a UAS can redirect with a 305, not a proxy.

However, this design decision on 305 handling was made prior to the conception of loose routing, although both ended up in RFC 3261. The design of the 305 mechanism, unfortunately, was not revisited after loose routing was specified. As such, the draft is not clear about whether or not the contact gets utilized as a Route header field value or whether it replaces the Request URI. The assumption, unstated, is that it populates the Request-URI, since redirection works that way in general.

3. Problems with Current Specifications

Numerous problems have arisen as a consequence of the combination of these specifications. These problems fit into two categories. The first are interoperability problems, and the second are missing capabilities.

An interoperability problem that has arisen is keeping an outbound proxy on the path for outbound requests. Consider a proxy in a hotel which a client discovers via DHCP and uses as its outbound proxy. This proxy wishes to be used for incoming and outgoing requests, both in and out of a dialog. If the home proxy provides a service route, the hotel proxy will not be able to determine what it needs to do in order to stay on the path. If the client implementation is such that it appends the service route to its default outbound proxy, then the hotel proxy need not do anything to stay on the path. If, however, the client abandons its default proxy in favor of the service route, the hotel proxy would fall off the path of subsequent requests unless it inserted a Service-Route into the response of a REGISTER request. Interestingly, the latter is illegal behavior according to RFC 3608, but is the only mechanism available for ensuring that a proxy stay on the request path. Since RFC 3608 does not provide any normative behavior for combining service routes and outbound proxies, the hotel proxy cannot know what to do, thus causing the interoperability
This points to the first major functional gap with the existing specifications. There is no standards-based way for keeping an outbound proxy on the path for outbound requests, when it is not the default outbound proxy. Consider a proxy in a hotel, PH-1 which a client discovers via DHCP and uses as its outbound proxy. When the client sends a REGISTER to this proxy, it forwards it to a second proxy in the hotel, PH-2, which then forwards it to the home proxy of the user, PA. PH-2 needs to be on the outbound path for all requests leaving the hotel. PA includes itself in a Service-Route header field in the response. The client receives this Service-Route. For an initial INVITE request, the client constructs a route set which includes its outbound proxy PH-1 followed by the URI from the Service-Route, PA. This request will traverse PH-1, which now follows the next Route header, sending it to PA. This is not the desired behavior. The problem is that the Service-Route URI has provided a route that overrides the default outbound route behavior at PH-1.

Similarly, there is no way in the current specifications to change the outbound proxy, outside of an update in the client configuration. Such changes are extremely useful for many operational reasons. One example is movement of subscribers between geographically distributed sites in cases where a site must be gracefully taken out of service, and the subscribers using it need to be moved gracefully, over a period of an hour or two, from one site to the other. Since, at best, the impact of Service-Route on the outbound proxy is ambiguous, there is generally no way to affect it excepting configuration change. Using configuration updates as the only way to alter the outbound proxy is problematic. In practice, systems providing automated updates to client configuration (when they exist, as they often do not) are decoupled from the operational systems that manage subscriber capacity and software upgrades of sites, making the change hard to affect through configuration. Furthermore, configuration updates are typically passed to clients once they are made. Here, however, the objective is to gracefully move subscribers. Using the randomized nature of REGISTER timings helps provide that; such a function is difficult to accomplish through configuration updates. Finally, many deployments use mechanisms other than [8] for updating client configurations. As a consequence, there is no common way across deployments to provide this very basic operational feature.

Finally, it is important to note that there is an architectural inconsistency between record-routing and service route. With record-route, each proxy on the path of the request inserts a Record-Route header field, and this dictates the path of subsequent messages within a dialog both to and from the UA. With REGISTER, each proxy
on the path of the request inserts a Path [4] header field to receive requests directed towards the client. However, the Service-Route is not the inverse of the Path, and is instead created through proprietary means by the registrar.

4. Overview of Operation

Firstly, new behavior for generation and processing of the 305 Use Proxy is specified. Any element in the network, proxy or UAS, can generate a 305, not just a UAS as specified in RFC3261. The 305 is directed towards a specific upstream element, whether it is a proxy or UAC, through the inclusion of the Redirect-Target header field in the response. This header field contains a counter that is decremented as the response is forwarded upstream. When it hits zero, that element recurses on it.

This only works if the server that sends the 305 can be sure that all of the upstream elements between it and the target of the redirect support this mechanism. To make this determination, the Target-Range header field is used. This header field contains a pair of integers, start-range and end-range. These integers correspond to the values of the Max-Forwards header field across a set of compliant elements. When the first element in a compliant chain (for example, a UAC supporting this mechanism) emits a request, it sets both start-range and end-range to the value of Max-Forwards in the request it emits. A compliant element decrements both Max-Forwards and end-range before forwarding the request if its policy says that downstream elements are permitted to redirect elements upstream from it. If this is not permitted, the element sets both start-range and end-range to Max-Forwards in the outbound request, effectively starting the chain afresh. However, a non-compliant element will only decrement the value of Max-Forwards. As such, a compliant server can determine whether the previous hop was compliant by comparing the value of Max-Forwards in the received request with the value of end-range. If they are identical, it means the previous hop supports the mechanism. Therefore, this proxy is is extended an existing chain of compliant elements, and it decrements both end-range and Max-Forwards before sending the request. However, if the values of Max-Forwards and end-range in the received request were not identical, it means that the previous upstream element (and possibly ones upstream from that) were not compliant. Therefore, this proxy is the first in a chain of compliant elements. It therefore resets start-range and end-range to the value of Max-Forwards in the request it emits.

Now, when a server receives a request, if Max-Forwards equals end-range, the server knows some number of upstream elements support this mechanism. Indeed, the exact number of such elements will be start-range minus end-range plus one. Thus, the server can insert the
Redirect-Target header field into the response with a value between 0 (directing the immediate upstream element to recurse) and start-range minus end-range.

This procedure is shown pictorially in Figure 1. The figure shows a UAC and four proxies, P-1 through P-4. The UAC and proxies P-2 through P-4 support the mechanism, but P-1 does not. The UAC emits an INVITE request. It populates Max-Forwards (shown as MF in the figure) with the initial value of 70. It also adds a Target-Range header field to the request, with a start-range value (SR in the figure) of 70, and an end-range value (ER in the figure) of 70. This request is received by P-1. Since P-1 does not understand Target-Range, it only decrements Max-Forwards. The request is now received by P-2. P-2 sees that the value of Max-Forwards in the received request (69) does not match the value of end-range (70). Thus, it knows that its immediately upstream neighbor didn’t support the mechanism. As such, when it emits its request, it sets the value of both start-range and end-range to the value of Max-Forwards, 68. This request arrives at P-3. P-3 sees that the value of Max-Forwards matches end-range. So, it decrements both in the request it emits. This request arrives at P-4. Again, the value of Max-Forwards equals end-range. It subtracts end-range from start-range (68-67) and adds 1, which results in 2. This means that the 2 upstream elements support the redirect targeting mechanism, and P-4 generates a 305 response with a Redirect-Target value of 1. This is received by P-3, which decrements Redirect-Target to zero. This is received by P-2, which sees that the value is zero. This means that it is the intended target of the redirect. It therefore recurses on the redirect and emits a new request.
[OPEN ISSUE: This backwards compatibility mechanism could actually be used for all redirects; providing a mechanism to know and control where and when recursion is done. Indeed, the mechanism could provide a general framework for allowing downstream servers to determine whether a sequence of upstream servers supports some extension. If one uses a sequence of ranges instead of a single range, full proxy support information can be delivered to the UAS. Is there a need for such a thing?]

In addition to specifying new rules for generation and processing of the 305, this specification updates the behaviors in RFC 3608. In particular, a registrar, upon receipt of a REGISTER, uses the Path header field values to construct the Service-Route in the response. The values from the Path are copied into the Service-Route, and the registrar can then add some additional ones if they are within the domain of the provider. By allowing the registrar to add values, the mechanism defined here is made a superset of the behaviors defined in RFC 3608. There, the registrar can only add URI in its own domain. Here, the registrar can add such URI, but also reflects Path headers from the request, which may have come from other domains. In addition, RFC 3608 defined the service route to be associated with an AOR, rather than a registered contact. This specification modifies that behavior. Instead, the service route will be associated with each registered contact. Note that the registrar never needs to store the service-route; it is computed as a function of the Path header in the REGISTER request.

5. Detailed Processing Rules

5.1 Registrar Behavior

This specification updates the procedures from RFC 3608.

The registrar MUST construct the Service-Route in the registration response by taking each URI from the Path header field in the REGISTER request, and inverting the order. After inversion, the registrar MAY add additional URIs at the end of the list (that is, the right hand side of the list, corresponding to proxy elements that will be the farthest away from the UA).

Furthermore, the registrar MAY replace or remove any URIs that are within a domain under the control of the registrar. When replacing a URI, one or more new ones can take its place. If the registrar is in example.com, this would include any URIs whose domain part is example.com. It would also include any URIs whose domain is a subdomain of example.com, as long as that subdomain is under the control of example.com. It could also include URIs whose domain part is unrelated to example.com, as long as those are within the control...
of example.com. It is difficult to provide a concise definition of "under the control", but generally it means that the administrative policies for the subservient domain are completely defined by the controlling domain.

This behavior ensures that proxies outside of the domain of the registrar have a way to appear on the service route, but provides a way, within a domain, to provide service routes that are not coupled to the Path.

[[OPEN ISSUE: Its unclear whether this mechanism is backwards compatible with current IMS procedures. The P-CSCF will insert Path, but not expect for its URI to be in the outbound Route set. The procedures here, for a UA compliant to this specification, will result in an outgoing INVITE being delivered to the P-CSCF as a consequence of it being the default outbound proxy, but the request will arrive with the topmost route equal to the outbound proxy URI, and the next route will be the Path URI. The route after that will be the S-CSCF (or I-CSCF in the home domain if added to the service route). Not clear that this will work. If it doesn’t, it is easily remedied by including a flag in the Path header which indicates whether it needs to be reflected into Service-Route.]]

5.2 Proxy Behavior

When a proxy receives a request that contains the Target-Range header field, it examines the value of end-range. If end-range is not equal to the value of Max-Forwards in the received request, the proxy MUST set both end-range and start-range equal to the value of the Max-Forwards header field in the request it emits. If they are equal, the proxy MAY extend the chain of compliant elements, or MAY reset it, starting with itself. The decision depends on whether the proxy wishes downstream elements to be able to generate redirects towards upstream elements, and is a matter of local policy. If the proxy decides to reset it, it sets both end-range and start-range equal to the value of the Max-Forwards header field in the request it emits. If it decides to extend it, it sets end-range equal to the value of the Max-Forwards header field in the request it emits and retains the value of start-range.

When a proxy receives a 305 response, it MUST check the value of the Redirect-Target header field. If this value is non-zero, the proxy MUST decrement it by one before forwarding the 305 upstream, and MUST NOT recurse on the 305. If the value is zero, the proxy follows the procedures in Section 5.4.

This specification updates the proxy processing rules in RFC 3608. In particular, if a proxy inserts a Path header field in a REGISTER
request, it means that a compliant registrar will echo the Path header field back into the REGISTER response as a Service-Route header field value. The proxy MAY remove its value from the Service-Route in the response, or MAY modify it. If the REGISTER response does not contain a Service-Route value that includes the Path URI inserted by the proxy, it means that the registrar is not compliant to this specification. [[OPEN ISSUE: and then it does what??]

[[OPEN ISSUE: not sure if the following belongs here or not; its not elaborated on anywhere else and is just a placeholder]]

If the proxy receives a request destined for the AOR of a subscriber, and the proxy is the responsible proxy for that domain, it looks up the AOR in the location database, and retrieves the Path URI and the registered Contact. However, instead of rewriting the request-URI to be equal to the registered contact, if that contact contains the URI loose-route parameter, the proxy retains the request URI, and instead uses that registered contact URI as the last Route header field value. In this way, the UAS will receive new requests with the AOR retained in the request URI, and a topmost Route header field present, with a URI containing the registered contact.

5.3 UAC Behavior

A UAC which supports the 305 recursion mechanism, including the Response-Target and Target-Range header fields, MUST include the Target-Range header field in all requests it emits, excepting CANCEL and ACK. This header field MUST have the start range and end range values equal to the value of Max-Forwards in the request emitted by the UAC.

Determination of the route set for a request is done in two steps. The first is the determination of a baseline route set. This set is the route set determined strictly by protocol mechanisms, and has not yet been subject to UA policies which might require alteration of the route set. Those policies are then applied, and the result is the final route set for the request.

For a request sent by a UAC that is not the result of recursion on a 305, the following logic MUST be used to compute the baseline route set:

- If the request is a mid-dialog request, the route set is computed per the procedures in Section 12.2.1.1 of RFC 3261. The baseline route set will not contain any routes learned from configuration, DHCP, Service-Route or any other mechanism.
o If the request is not a mid-dialog request, the client checks to see if it has learned a service route as a result of registration. The UAC may have learned numerous service routes, one for each unique AOR/Contact that it registered. In the case of registrations using the mechanisms of [5], the Contact includes the flow ID and instance ID, so that the client may have a distinct service route for each unique AOR/Flow ID/Instance ID combination. As such, when sending a request, the client selects the service route corresponding to the contact which is sending the request. [[OPEN ISSUE: Need to say more about this selection.]]. Once a service route has been selected, the URIs from this service route become the baseline route set. Here too, the baseline route set will not contain any routes learned from configuration, DHCP, or other service routes.

o If the request is not a mid-dialog request, and there are no service routes associated with the contact generating the request, the UAC uses the route set learned through configuration. [[OPEN ISSUE: Do we need to specify how to reconcile route sources learned across disparate configuration sources? For example DHCP and a config file? These can come from different providers.]]

If the request is being generated as a consequence of a 305, the baseline route set is computed as described in Section 5.4.

With the baseline route set computed, the UAC applies policy to determine whether this route set needs to be modified. The primary factor involved is whether or not the client needs to send this request through its outbound proxy or not. The following logic is RECOMMENDED. If the topmost URI in the baseline route set equals the configured default outbound proxy for the UAC, then the baseline route set is used unmodified, and used as the final route set. If, however, it does not, the UAC checks a white list of URI that it maintains. If the topmost URI appears on that white list, the baseline route set is used as the final route set. If it is not present, the default outbound proxy URI is appended to the top of the route set.

The white list represents destinations that the UA has confidence are ones permitted to be used. Here, this implies that the provider of the default outbound proxy allows that URI to be used in its place. This white list can be provided by configuration, but this is cumbersome and NOT RECOMMENDED. Instead, the following algorithm is RECOMMENDED. Initially, the white list is empty when a UA starts up. If a UA receives a 305 to a REGISTER request it generates, the URI in the Contact header field of the redirect are added to the white list. This will cause the REGISTER to be sent with one of these URI as the topmost URI in the route set. If that registration succeeds, the URI
are retained in the white list for the duration of the registration. The client maintains a separate white list for each registered contact.

This algorithm works because of two factors. Firstly, registrations are always targeted towards the domain of the subscriber, and are never delivered to user agents. As such, the mechanism relies on trust between the provider of the outbound proxy and provider of the AOR for the subscriber to follow the 305 mechanism described here correctly. However, if that trust doesn’t exist, basic call processing is not possible in any case. The second factor is the 305 mechanism itself. If the outbound proxy doesn’t support this specification, or the provider of the outbound proxy doesn’t wish the provider of the AOR to bypass the outbound proxy, the 305 mechanism allows this to be known to the provider of the AOR. Thus, the provider of the AOR can only bypass the outbound proxy if permitted by the provider of the outbound proxy. Typically, this would only be allowed when they are the same provider. The 305 mechanism allows the outbound proxy to bypass itself, since the outbound proxy can generate a 305 as long as the client supports the mechanism in this specification.

5.4 Client Behavior

The following logic defined here applies to all clients, both UAC and proxies, and applies to the processing of a 305 response.

When a client receives a 305 response, it MUST check for the presence of the Response-Target header field. If this header field is absent, the client MAY recurse on the request. However, in this case, the recursion MUST be accomplished by replacing the request URI of the request it generates with the value of the Contact header field in the 305 response. This provides backwards compatibility with the existing usage of 305, since all redirection defined in RFC 3261 updates the Request-URI. If the Response-Target header field is present, but has a non-zero value, the client MUST NOT recurse on the redirect. If the Response-Target header field is present and has a value of zero, the client MUST recurse on the redirect.

To compute the request that is sent as a result of the recursion, the client MUST take the route set used for the request that generated the 305 response. If that request had a Route header field, the first value MUST be replaced with the value of the Contact header field in the 305 with the highest q-value. If there are multiple such Contacts with the same q-value, one is chosen at random. The result is used as the route set for the new request. If the client is a UAC, it follows the procedures defined in Section 5.3. If the client is a proxy, it follows the procedures defined in Section 5.2.
[[ISSUE: There are three choices about how to process the contact in the 305. The URI there can replace the route set at the client completely, they can be appended to the route set, or they can replace the topmost route. This draft employs the latter technique. Further consideration is needed to determine whether or not this is the right thing. Since the contact can contain multiple URI, we could actually have it contain the entire route set that should replace the one from the request.]]

If a 305 response had multiple Contact header field values, and the recursed request generated a 503 response, and the client had exhausted all alternative servers learned from DNS [7] for the previous Contact header field value, the client SHOULD choose the Contact from the 305 with the next highest q-value, and construct another recursed request using the procedures defined above. In the event the 305 had multiple Contact header field values with equivalent q-values, the next highest one might have a q-value equal to the one that was just tried.

5.5 Server Behavior

Any server, either a UAS or a proxy, MAY generate a 305 in response to a request. However, it MUST NOT do so unless the request contains a Target-Range header field, and the value of end-range in that header field equals the value of Max-Forwards in the received request. If the server generates a 305, it MUST direct that redirect to a specific upstream element. To do so, it includes a Redirect-Target header field in the response. That header field identifies a specific element that is the target of the redirect. A value of 0 indicates that the element immediately upstream is the target, 1 indicates that the target is the second upstream element, and so on. The value of Redirect-Target MUST be between 0 and start range minus end range.

6. Grammar

This specification defines two new header fields - the Target-Range and Redirect-Target header fields. Their syntax is as follows:

Target-Range = "Target-Range" HCOLON start-range SWS "-" SWS end-range
Redirect-Target = "Redirect-Target" HCOLON target-val
start-range = 1*DIGIT
end-range = 1*DIGIT
target-val = 1*DIGIT
7. Security Considerations

The route set used by a user agent for generating initial requests into the network is very sensitive information. If this information is manipulated by an attacker, it can cause calls to be directed towards intermediaries, which can then observe call patterns, intercept communications, and so on. As a consequence, the mechanisms in this specification to take care that this route set can only be updated on very specific conditions. Furthermore, the 305 mechanism defined here gives service providers policy hooks that allow them to control whether such redirection can be employed by external service providers.

[[ISSUE: needs more verbiage here]]

8. IANA Considerations

9. Examples

10. Acknowledgements

The author would like to thank Paul Kyzivat and Anders Kristensen for their comments.

11. References

11.1 Normative References


11.2 Informative References


Author’s Address

Jonathan Rosenberg
Cisco Systems
600 Lanidex Plaza
Parsippany, NJ 07054
US

Phone: +1 973 952-5000
Email: jdrosen@cisco.com
URI: http://www.jdrosen.net
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 (2006). 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.