Network Working Group
Request for Comments: 3327
Category: Standards Track

D. Willis
dynamicsoft Inc.
B. Hoeneisen
Switch

December 2002

Session Initiation Protocol (SIP) Extension Header Field for Registering Non-Adjacent Contacts

Status of this Memo

Copyright Notice

Abstract

The REGISTER function is used in a Session Initiation Protocol (SIP) system primarily to associate a temporary contact address with an address-of-record. This contact is generally in the form of a Uniform Resource Identifier (URI), such as Contact: <sip:alice@pc33.atlanta.com> and is generally dynamic and associated with the IP address or hostname of the SIP User Agent (UA). The problem is that network topology may have one or more SIP proxies between the UA and the registrar, such that any request traveling from the user's home network to the registered UA must traverse these proxies. The REGISTER method does not give us a mechanism to discover and record this sequence of proxies in the registrar for future use. This document defines an extension header field, "Path" which provides such a mechanism.

Table of Contents

1. Background

3GPP established a requirement for discovering intermediate proxies during SIP registration and published this requirement in [5].

Scenario:

UA1----P1-----P2-----P3------REGISTRAR

UA1 wishes to register with REGISTRAR. However, due to network topology, UA1 must use P1 as an "outbound proxy", and all requests between UA1 and REGISTRAR must also traverse P1, P2, and P3 before reaching REGISTRAR. Likewise, all requests between REGISTRAR and UA1 must also traverse P3, P2, and P1 before reaching UA1.

UA1 has a standing relationship with REGISTRAR. How UA1 establishes this relationship is outside the scope of this document. UA1 discovers P1 as a result of configuration, DHCP assignment or other similar operation, also outside the scope of this document. REGISTRAR has a similar "default outbound proxy" relationship with P3.

Eventually, REGISTRAR or a "home proxy" (a proxy serving as the terminal point for routing an address-of-record) closely related to it will receive a request destined for UA1. It needs to know which proxies must be transited by that request in order to get back to UA1. In some cases, this information may be deducible from SIP routing configuration tables or from DNS entries. In other cases, such as that raised by 3GPP, the information is not readily available outside of the SIP REGISTER transaction.

The Path extension header field allows accumulating and transmitting the list of proxies between UA1 and REGISTRAR. Intermediate nodes such as P1 may statefully retain Path information if needed by operational policy. This mechanism is in many ways similar to the operation of Record-Route in dialog-initiating requests. The routing established by the Path header field mechanism applies only to requests transiting or originating in the home domain.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, RFC 2119 [3].

3. Applicability Statement

The Path mechanism is applicable whenever there are intermediate proxies between a SIP UA and a SIP Registrar used by that UA where the following conditions are true:

1. One or more of the intermediate proxies are visited by registration requests from the UA to the Registrar.
2. The same intermediate proxies or a set of proxies known to the intermediate proxies must be traversed, in reverse order, by requests sent through a home proxy to the UA. In the simplest form, the route between the home proxy and the UA is the exact inverse of the route between the UA and the route between the UA and the registrar.
3. The network topology is such that the intermediate proxies which must be visited are NOT implied by SIP routing tables, DNS, or similar mechanisms.

4. Path Header Field Definition and Syntax

The Path header field is a SIP extension header field with syntax very similar to the Record-Route header field. It is used in conjunction with SIP REGISTER requests and with 200 class messages in response to REGISTER (REGISTER responses).

A Path header field MAY be inserted into a REGISTER by any SIP node traversed by that request. Like the Route header field, sequential Path header fields are evaluated in the sequence in which they are present in the request, and Path header fields MAY be combined into compound Path header in a single Path header field. The registrar reflects the accumulated Path back into the REGISTER response, and intermediate nodes propagate this back toward the originating UA. The originating UA is therefore informed of the inclusion of nodes on its registered Path, and MAY use that information in other capacities outside the scope of this document.

The difference between Path and Record-Route is that Path applies to REGISTER and 200 class responses to REGISTER. Record-Route doesn't, and can't be defined in REGISTER for reasons of backward compatibility. Furthermore, the vector established by Record-Route applies only to requests within the dialog that established that Record-Route, whereas the vector established by Path applies to future dialogs.

The syntax for Path is defined as follows:

```
Path = "Path" HCOLON path-value *( COMMA path-value )

path-value = name-addr *( SEMI rr-param )
```

Note that the Path header field values conform to the syntax of a Route element as defined in [1]. As suggested therein, such values MUST include the loose-routing indicator parameter ";lr" for full compliance with [1].

The allowable usage of header fields is described in Tables 2 and 3 of SIP [1]. The following additions to this table are needed for Path.

Support for the Path header field MAY be indicated by a UA by including the option-tag "path" in a Supported header field.

Addition of Path to SIP Table 3:

Header field	where	proxy	ACK	BYE	CAN	INV	OPT	REG
Path	R	ar	-	-	-	-	-	o
Path	2xx	-	-	-	-	-	-	o

5. Usage of Path Header Field

5.1 Procedures at the UA

The UA executes its register operation as usual. The response MAY contain a Path header field. The general operation of the UA is to ignore the Path header field in the response. It MAY choose to display the contents of the Path header field to the user or take other action outside the scope of this document. The Path information in the REGISTER response lets the UA know what intermediate proxies were added during registration. Examination of this information may be important from a security perspective, as such inspection might allow the UA to detect intermediate proxies that have inappropriately added themselves.

The UA SHOULD include the option tag "path" as a header field value in all Supported header fields, and SHOULD include a Supported header field in all requests.

The UA MAY include a Path header field in a request. This is not broadly applicable and caution must be taken to insure proper function, as the Path header field inserted by the UA may have additional Path header field values appended by intermediate proxies. Such proxies are not aware that the Path header field value was inserted by a UA, and will treat it as if it had been inserted by a previously traversed proxy, which could result in unexpected routing behavior wherein the UA is asked to act as a proxy.

5.2 Procedures at Intermediate Proxies

When a proxy processing a REGISTER request wishes to be on the path for future requests toward the UA originating that REGISTER request, the proxy inserts a URI for that proxy as the topmost value in the Path header field (or inserts a new topmost Path header) before proxying that request. It is also possible for a proxy with specific knowledge of network topology to add a Path header field value referencing another node, thereby allowing construction of a Path which is discongruent with the route taken by the REGISTER request. Such a construction is implementation specific and outside the scope of this document.

Intermediate proxies SHOULD NOT add a Path header field to a request unless the UA has indicated support for this extension with a Supported header field value. If the UA has indicated support and the proxy requires the registrar to support the Path extension, then the proxy SHOULD insert a Requires header field value for this extension. If the UA has not indicated support for the extension and the proxy requires support for it in the registrar, the proxy SHOULD

reject the request with a 421 response indicating a requirement for the extension.

Proxies processing a REGISTER response SHOULD NOT alter any Path header field values that may be present in the response. The registrar MAY protect the Path header field in the response by including it in a protected S/MIME body, and alterations of the Path by an intermediate proxy can therefore be detected by the UA as man-in-the-middle attacks. Proxies SHOULD only consider altering the value of a Path header field in the REGISTER response if they have the credentials to correctly alter the S/MIME body to account for the change.

5.3 Procedures at the Registrar

If a Path header field exists in a successful REGISTER request, the registrar constructs an ordered list of route elements (a path vector) from the nodes listed in the Path header field values, preserving the order as indicated in the Path header field values. The registrar then stores this path vector in association with that contact and the address-of-record indicated in the REGISTER request (the "binding" as defined in [1]). The registrar copies the Path header field values into a Path header field in the successful (200 class) REGISTER response. In the event that the home proxy and registrar are not co-located, the registrar MAY apply a locally-determined transformation to the stored path vector.

If a registrar receives a REGISTER request containing a Path header field and there is no indication of support for the extension in the UA (via a Supported header field), the registrar must rely on local policy in determining how to treat this request. The recommended policy is for the registrar to reject the request with a 420 "Bad Extension" response indicating the Path extension. This approach allows the UA to detect that an intermediate proxy has inappropriately added a Path header field. However, the Path mechanism should technically work in the absence of UA support (at some compromise to security), so some registrars MAY choose to support the extension in the absence of a Supported header field value in the request.

5.4 Procedures at the Home Proxy

In the common SIP model, there is a home proxy associated with the registrar for a user. Each incoming request targeted to the public address-of-record for the user is routed to this proxy, which consults the registrar’s database in order to determine the contact to which the request should be retargeted. The home proxy, in its basic mode of operation, rewrites the request-URI from the incoming

request with the value of the registered contact and retransmits the request.

With the addition of Path, the home proxy also copies the stored path vector associated with the specific contact in the registrar database into the Route header field of the outgoing request as a preloaded route. This causes the outgoing request to transit the proxies that were included in the Path header field of the REGISTER request.

In normal processing, the home proxy is the "terminal point" for the user's address-of-record (AOR). Consequentially, the Route header field on the incoming request will have been exhausted in reaching the home proxy. If it isn't, then things get interesting. In the most common case, the home proxy generates the outgoing Route header field by inserting the stored path vector ahead of the Route header field values contained in the incoming request. This procedure may be altered by a local policy at the home proxy.

Loose routes may interact with routing policy in interesting ways. The specifics of how the stored path vector integrates with any locally required default route and local policy are implementation dependent. For example, some devices will use locally-configured explicit loose routing to reach a next-hop proxy, and others will use a default outbound-proxy routing rule. However, for the result to function, the combination must provide valid routing in the local environment. In general, the stored path vector is appended to any locally configured route needed to egress the service cluster. The service proxy (or registrar, as noted earlier) MAY also transform the stored path vector as needed to provide correct functionality. Systems designers must match the Path recording policy of their nodes with the routing policy in order to get a workable system.

5.5 Examples of Usage

Note that some header fields (e.g. Content-Length) and session descriptions are omitted to provide a shorter and hopefully more readable presentation. The node marked REGISTRAR is a registrar and a proxy and serves as a home proxy. Thus, in the DNS the domain EXAMPLEHOME.COM points to the same host as REGISTRAR.EXAMPLEHOME.COM.

5.5.1 Example of Mechanism in REGISTER Transaction

As an example, we use the scenario from the Background section:

UA1----P1-----P2----P3-----REGISTRAR

In this example, UA1 sends a REGISTER request to REGISTRAR. This request transits its default outbound proxy P1, an intermediate proxy P2, and the firewall proxy for the home domain, P3, before reaching REGISTRAR. Due to network topology and operational policy, P1 and and P3 need to be transited by requests from REGISTRAR or other nodes in the home network targeted to UA1. P2 does not. P1 and P3 have been configured to include themselves in Path header fields on REGISTER requests that they process. UA1 has a current IP address of "192.0.2.4".

Message sequence for REGISTER with Path:

F1 Register UA1 -> P1

```
REGISTER sip:REGISTRAR.EXAMPLEHOME.COM SIP/2.0
Via: SIP/2.0/UDP 192.0.2.4:5060;branch=z9hG4bKnashds7
To: UA1 <sip:UA1@EXAMPLEHOME.COM>
From: UA1 <sip:UA1@EXAMPLEHOME.COM>;tag=456248
Call-ID: 843817637684230@998sdasdh09
CSeq: 1826 REGISTER
Contact: <sip:UA1@192.0.2.4>
Supported: path
 . . .
```

F2 Register P1 -> P2

```
REGISTER sip:REGISTRAR.EXAMPLEHOME.COM SIP/2.0
Via: SIP/2.0/UDP 112.68.155.4:5060;branch=z9hG4bK34ghi7ab04
Via: SIP/2.0/UDP 192.0.2.4:5060;branch=z9hG4bKnashds7
To: UA1 <sip:UA1@EXAMPLEHOME.COM>
From: UA1 <sip:UA1@EXAMPLEHOME.COM>;tag=456248
Call-ID: 843817637684230@998sdasdh09
CSeq: 1826 REGISTER
Contact: <sip:UA1@192.0.2.4>
Supported: path
Path: <sip:P1.EXAMPLEVISITED.COM;lr>
 . . .
```

Note: P1 has added itself to the Path.

F3 Register P2 -> P3

```
REGISTER sip:REGISTRAR.EXAMPLEHOME.COM SIP/2.0
Via: SIP/2.0/UDP 178.73.76.230:5060;branch=z9hG4bKiokioukju908
Via: SIP/2.0/UDP 112.68.155.4:5060;branch=z9hG4bK34ghi7ab04
Via: SIP/2.0/UDP 192.0.2.4:5060;branch=z9hG4bKnashds7
To: UA1 <sip:UA1@EXAMPLEHOME.COM>
From: UA1 <sip:UA1@EXAMPLEHOME.COM>;tag=456248
Call-ID: 843817637684230@998sdasdh09
CSeq: 1826 REGISTER
Contact: <sip:UA1@192.0.2.4>
Supported: path
Path: <sip:P1.EXAMPLEVISITED.COM;lr>
 . . .
```

Note: P2 did NOT add itself to the Path.

F4 Register P3 -> REGISTRAR

```
REGISTER sip:REGISTRAR.EXAMPLEHOME.COM SIP/2.0
Via: SIP/2.0/UDP 19.31.97.3:5060;branch=z9hG4bKp3wer654363
Via: SIP/2.0/UDP 178.73.76.230:5060;branch=z9hG4bKiokioukju908
Via: SIP/2.0/UDP 112.68.155.4:5060;branch=z9hG4bK34ghi7ab04
Via: SIP/2.0/UDP 192.0.2.4:5060;branch=z9hG4bKnashds7
To: UA1 <sip:UA1@EXAMPLEHOME.COM>
From: UA1 <sip:UA1@EXAMPLEHOME.COM>;tag=456248
Call-ID: 843817637684230@998sdasdh09
CSeq: 1826 REGISTER
Contact: <sip:UA1@192.0.2.4>
Supported: path
Path: <sip:P3.EXAMPLEHOME.COM;lr>,<sip:P1.EXAMPLEVISITED.COM;lr>
 . . .
```

Note: P3 added itself to the Path.

F5 REGISTRAR executes Register

```
REGISTRAR Stores:
For UA1@EXAMPLEHOME.COM
Contact: <sip:UA1@192.0.2.4>
Supported: path
Path: <sip:P3.EXAMPLEHOME.COM;lr>,<sip:P1.EXAMPLEVISITED.COM;lr>
```

F6 Register Response REGISTRAR -> P3

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 19.31.97.3:5060;branch=z9hG4bKp3wer654363
Via: SIP/2.0/UDP 178.73.76.230:5060;branch=z9hG4bKiokioukju908
Via: SIP/2.0/UDP 112.68.155.4:5060;branch=z9hG4bK34ghi7ab04
Via: SIP/2.0/UDP 192.0.2.4:5060;branch=z9hG4bKnashds7
To: UA1 <sip:UA1@EXAMPLEHOME.COM>;tag=251077
From: UA1 <sip:UA1@EXAMPLEHOME.COM>;tag=456248
Call-ID: 843817637684230@998sdasdh09
CSeq: 1826 REGISTER
Contact: <sip:UA1@192.0.2.4>
Supported: path
Path: <sip:P3.EXAMPLEHOME.COM;lr>,<sip:P1.EXAMPLEVISITED.COM;lr>
 . . .
```

Note: The Path header field in the response is identical to the one received in the REGISTER request.

F7 Register Response P3 -> P2

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 178.73.76.230:5060;branch=z9hG4bKiokioukju908
Via: SIP/2.0/UDP 112.68.155.4:5060;branch=z9hG4bK34ghi7ab04
Via: SIP/2.0/UDP 192.0.2.4:5060;branch=z9hG4bKnashds7
To: UA1 <sip:UA1@EXAMPLEHOME.COM>;tag=251077
From: UA1 <sip:UA1@EXAMPLEHOME.COM>;tag=456248
Call-ID: 843817637684230@998sdasdh09
CSeq: 1826 REGISTER
Contact: <sip:UA1@192.0.2.4>
Supported: path
Path: <sip:P3.EXAMPLEHOME.COM;lr>,<sip:P1.EXAMPLEVISITED.COM;lr>
 . . .
```

F8 Register Response P2 -> P1

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 112.68.155.4:5060;branch=z9hG4bK34ghi7ab04
Via: SIP/2.0/UDP 192.0.2.4:5060;branch=z9hG4bKnashds7
To: UA1 <sip:UA1@EXAMPLEHOME.COM>;tag=251077
From: UA1 <sip:UA1@EXAMPLEHOME.COM>;tag=456248
Call-ID: 843817637684230@998sdasdh09
CSeq: 1826 REGISTER
Contact: <sip:UA1@192.0.2.4>
Supported: path
Path: <sip:P3.EXAMPLEHOME.COM;lr>,<sip:P1.EXAMPLEVISITED.COM;lr>
 . . .
```

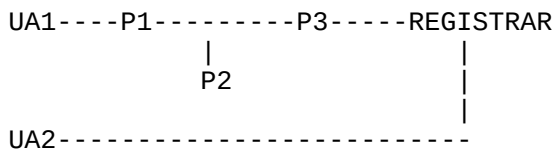
F9 Register Response P1 -> UA1

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.0.2.4:5060;branch=z9hG4bKnashds7
To: UA1 <sip:UA1@EXAMPLEHOME.COM>;tag=251077
From: UA1 <sip:UA1@EXAMPLEHOME.COM>;tag=456248
Call-ID: 843817637684230@998sdasdh09
CSeq: 1826 REGISTER
Contact: <sip:UA1@192.0.2.4>
Supported: path
Path: <sip:P3.EXAMPLEHOME.COM;lr>,<sip:P1.EXAMPLEVISITED.COM;lr>
 . . .
```

5.5.2 Example of Mechanism in INVITE Transaction

This example shows the message sequence for an INVITE transaction originating from UA2 eventually arriving at UA1. REGISTRAR inserts a preloaded Route toward UA1 and retargets the request by replacing the request URI with the registered Contact. It then sends the retargeted INVITE along the Path towards UA1. Note that this example introduces foreign user agent UA2 (address "71.91.180.10") and foreign domain FOREIGN.ELSEWHERE.ORG. We have extended the diagram from the previous example by adding UA2, and by showing P2 out-of-line indicating that it did not include itself in the path during registration.

Scenario



Message sequence for INVITE using Path:

F1 Invite UA2 -> REGISTRAR

```
INVITE UA1@EXAMPLEHOME.COM SIP/2.0
Via: SIP/2.0/UDP 71.91.180.10:5060;branch=z9hG4bKe2i95c5st3R
To: UA1 <sip:UA1@EXAMPLEHOME.COM>
From: UA2 <sip:UA2@FOREIGN.ELSEWHERE.ORG>;tag=224497
Call-ID: 48273181116@71.91.180.10
CSeq: 29 INVITE
Contact: <sip:UA2@71.91.180.10>
 . . .
```

F2 REGISTRAR processing

REGISTRAR looks up name "UA1@EXAMPLEHOME.COM" and returns:
- Contact = <sip:UA1@192.0.2.4>
- Path vector = <sip:P3.EXAMPLEHOME.COM;lr>,
<sip:P1.EXAMPLEVISITED.COM;lr>

Note: The Contact replaces the request-URI. The path vector is pushed onto the Route stack (preloaded Route) of the outgoing INVITE request. The topmost Route is used for making the routing decision (in conjunction with local policy).

F3 Invite REGISTRAR -> P3

```
INVITE UA1@192.0.2.4 SIP/2.0
Via: SIP/2.0/UDP 143.70.6.83:5060;branch=z9hG4bKlj25C107a7b176
Via: SIP/2.0/UDP 71.91.180.10:5060;branch=z9hG4bKe2i95c5st3R
To: UA1 <sip:UA1@EXAMPLEHOME.COM>
From: UA2 <sip:UA2@FOREIGN.ELSEWHERE.ORG>;tag=224497
Call-ID: 48273181116@71.91.180.10
CSeq: 29 INVITE
Contact: <sip:UA2@71.91.180.10>
Route: <sip:P3.EXAMPLEHOME.COM;lr>,<sip:P1.EXAMPLEVISITED.COM;lr>
 . . .
```

Note: In this example REGISTRAR does not want to stay on the Route and therefore does not insert a Record-Route.

F4 Invite P3 -> P1

```
INVITE UA1@192.0.2.4 SIP/2.0
Via: SIP/2.0/UDP 19.31.97.3:5060;branch=z9hG4bKjasg7li7nc9e
Via: SIP/2.0/UDP 143.70.6.83:5060;branch=z9hG4bKlj25C107a7b176
Via: SIP/2.0/UDP 71.91.180.10:5060;branch=z9hG4bKe2i95c5st3R
To: UA1 <sip:UA1@EXAMPLEHOME.COM>
From: UA2 <sip:UA2@FOREIGN.ELSEWHERE.ORG>;tag=224497
Call-ID: 48273181116@71.91.180.10
CSeq: 29 INVITE
Contact: <sip:UA2@71.91.180.10>
Record-Route: <sip:P3.EXAMPLEHOME.COM;lr>
Route: <sip:P1.EXAMPLEVISITED.COM;lr>
 . . .
```

Note: P3 has added a Record-Route entry, indicating that it wants to be traversed by future messages in this dialog.

F5 Invite P1 -> UA1

```
INVITE UA1@192.0.2.4 SIP/2.0
Via: SIP/2.0/UDP 112.68.155.4:5060;branch=z9hG4bKk5l1833o43p
Via: SIP/2.0/UDP 19.31.97.3:5060;branch=z9hG4bKjasg7li7nc9e
Via: SIP/2.0/UDP 143.70.6.83:5060;branch=z9hG4bKlj25C107a7b176
Via: SIP/2.0/UDP 71.91.180.10:5060;branch=z9hG4bKe2i95c5st3R
To: UA1 <sip:UA1@EXAMPLEHOME.COM>
From: UA2 <sip:UA2@FOREIGN.ELSEWHERE.ORG>;tag=224497
Call-ID: 48273181116@71.91.180.10
CSeq: 29 INVITE
Contact: <sip:UA2@71.91.180.10>
Record-Route: <sip:P1.EXAMPLEVISITED.COM;lr>
Record-Route: <sip:P3.EXAMPLEHOME.COM;lr>
 . . .
```

Note: P1 has added a Record-Route entry, indicating that it wants to be traversed by future messages in this dialog.

6. Security Considerations

There are few security considerations for this document beyond those in SIP [1]. From a security perspective, the Path extension and its usage are identical to the Record-Route header field of basic SIP. Note that the transparency of the user expectations are preserved by returning the final Path to the originating UA -- that is, the UA is informed which additional proxies have been inserted into the path for the registration associated with that response.

The Path header field accumulates information in a hop-by-hop manner during REGISTER processing. The return information is essentially end-to-end, that is, it is not altered by intermediate proxies. This leads to two slightly different security approaches.

6.1 Considerations in REGISTER Request Processing

Information accumulated in REGISTER processing causes additional proxies to be included in future requests between the registrar’s location and the UA. An attack that allowed an intruding proxy to add itself to this chain would allow the attacker to intercept future calls intended for the UA.

An attacker could conceivably alter the Path either by altering data "on the wire" or by other manipulations (such as impersonation) that would cause it to be included in the SIP routing chain (a "node insertion" attack). Altering data "on the wire" may be addressed adequately by the use of transport-layer integrity protection mechanisms such as TLS or IPSEC. Proxy insertion can be addressed by

mutual authentication at the proxy layer, which can also be provided by TLS or IPSEC. The "sips:" URI class defined in [1] provides a mechanism by which a UA may request that intermediate proxies provide integrity protection and mutual authentication.

Systems using the Path mechanism SHOULD use appropriate mechanisms (TLS, IPSEC, etc.) to provide message integrity and mutual authentication. UAs SHOULD use "sips:" to request transitive protection.

The registering UA SHOULD use S/MIME mechanisms to provide a protected copy of the original request to the registrar. In this case, the UA SHOULD include a Supported header field with a value indicating support for the Path extension in the protected copy. Registrars receiving such as request MUST honor the Path extension only if support is indicated in the protected header field. Further, they SHOULD compare the unprotected Supported header field with the protected Supported header field and take appropriate action in the event that an intermediate has altered the message to indicate support for Path when it was not indicated by the requesting UA.

6.2 Considerations in REGISTER Response Processing

The data returned to the UA by the Path header field in the response to the REGISTER request is there to provide openness to the UA. The registrar is telling the UA, "These are the intermediate proxies that will be included on future requests to you processed through me". By inspection of this header field, the UA may be able to detect node insertion attacks that involve inserting a proxy into the SIP routing chain. S/MIME techniques may be used to prevent alteration of this header field by intermediate proxies during response processing.

As specified, there is no requirement for arbitrary proxies between the UA and the registrar to modify the Path header field in the REGISTER response. Consequently, we may use an end-to-end protection technique. The S/MIME technique defined in [1] provides an effective mechanism. Using this technique, the registrar makes a copy of the complete response, signs it, and attaches it as a body to the response. The UA may then verify this response, assuring an unmodified Path header field is received.

In addition to the hop-by-hop integrity protection and mutual authentication measures suggested for REGISTER request processing in the preceding section, systems using Path header fields SHOULD implement end-to-end protection using S/MIME. More specifically, registrars returning a Path header field SHOULD attach a signed S/MIME of the response, and UAs receiving a REGISTER response containing a Path header field SHOULD validate the message using the

S/MIME technique. Furthermore, UAs receiving a Path header field in a REGISTER response SHOULD render it to the user, or (where feasible) check it programmatically.

7. IANA Considerations

This document defines the SIP extension header field "Path", which the IANA has added to the registry of SIP header fields defined in SIP [1].

This document also defines the SIP option tag "path" which IANA has added to the registry of SIP option tags defined in SIP [1].

The following is the registration for the Path header field:

RFC Number: RFC3327

Header Field Name: Path

Compact Form: none

The following is the registration for the path option tag:

RFC Number: RFC3327

Option Tag: path

8. Acknowledgements

Min Huang and Stinson Mathai, who put together the original proposal in 3GPP for this mechanism, and worked out most of the 3GPP procedures in 24.229.

Keith Drage, Bill Marshall, and Miguel Angel Garcia-Martin who argued with everybody a lot about the idea as well as helped refine the requirements.

Juha Heinanen, who argued steadfastly against standardizing the function of discovering the home proxy with this technique in this document.

Normative References

[1] 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.

[2] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, October 1996.

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

[4] Postel, J. and J. Reynolds, "Instructions to RFC Authors", RFC 2223, October 1997.

Non-Normative References

[5] Garcia-Martin, MA., "3GPP Requirements On SIP", Work in Progress.

[6] Mankin, A., "SIP Change Process", Work in Progress.

Authors' Addresses

Dean Willis
dynamicsoft Inc.
5100 Tennyson Parkway
Suite 1200
Plano, TX 75028
US

Phone: +1 972 473 5455
EMail: dean.willis@softarmor.com
URI: http://www.dynamicsoft.com/

Bernie Hoeneisen
Switch
Limmatquai 138
CH-8001 Zuerich
Switzerland

Phone: +41 1 268 1515
EMail: hoeneisen@switch.ch, b.hoeneisen@ieee.org
URI: http://www.switch.ch/

Full Copyright Statement

Acknowledgement

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