Network Working Group
Internet-Draft
Expires: July 2, 2002

R. Sparks
dynamicsoft
Jan 2002

Exploring the Loose Route Proposal
draft-sparks-sip-looseroute-00

Status of this Memo

Copyright Notice

Abstract

This memo documents a proposal to allow SIP elements to use local policy when processing requests containing Route header fields. It identifies two restrictions on that local policy and provides an algorithm for satisfying one of those restrictions. It also presents several example flows utilizing policies enabled by this proposal.

Table of Contents

1. Introduction

Several proposals have been presented in draft form and on the SIPWG mailing list to address the problem of routing the initial request in a dialog through a particular set of proxies. Several problems with the strict routing mechanisms already available in SIP were identified, including the endpoint's lack of complete knowledge of what that set of proxies should be and the potential loss of information about the desired target due to the strict replacement of the Request-URI.

This memo reflects an effort by participants to unify those proposals, addressing the issues with a single mechanism. It addresses how an endpoint and intermediaries can cause a request to visit certain proxies once they know they need to do so. It does NOT address how the endpoint or intermediaries would discover that they needed to do this. In particular, if an endpoint is to use a preloaded Route header field, it must be told through some mechanism (either protocol or configuration) what to use. Those mechanisms are not discussed in this document.

This proposal is backwards-compatible with existing strict routing implementations and corrects some problems with the use of default outbound proxies. The heart of the proposal is in relaxing the restrictions on what elements can do with messages that contain Route headers. Only Route (and route-set) processing is affected. The Record-Route mechanism is not modified in any way.

2. The Loose Routing Proposal

The proposal is simply stated:

> When an element processes a message containing a Route header field, it MAY observe local policy in determining rewriting the Request-URI, where to send the request and whether to remove the topmost Route header field value. It MUST NOT modify or remove subsequent Route header field values.

There are only a few restrictions on the local policy to make sure we still have the Route behavior we've come to expect.

The first restriction ensures a Route header value is consumed once the resource it identifies has been reached.

> If the resource in the topmost Route header field value belongs to this element, the local policy MUST include removing the topmost header field value.

"Belongs" here means it is the element that the Route targeted. In operations to date, that URI would have been moved from the Route header field to the Request-URI by the previous strict-routing hop. If, however the previous hop is a loose-routing element that didn't remove the topmost Route header field value, it must get removed here so that the request is routed towards the next thing in the Route set.

When this restriction is invoked, the removed Route value may or may not be placed in the Request-URI. If it is, the request will spiral through this element.

The second restriction complements the first, ensuring that a request-URI is modified once the resource it identifies is actually reached.

> If the Request-URI identifies a resource owned by this element, the local policy SHOULD include modifying the Request-URI.

This modification would most likely take the form of moving the URI from the topmost Route header field value of the arriving request into the Request-URI (as is done in strict routing).

The third restriction discourages completely ignoring the Route header field.

> The local policy SHOULD be strict-routing. The local policy MUST direct the request to the resource indicated in the topmost Route header field value (using the standard methods for determining where to send a request) or to a proxy it trusts ensure this property.

This proposal is backwards compatible with deployed strict-routing systems as long as the loose-routing elements are careful to construct their Request-URIs so that they don't induce loops in unsuspecting current implementations. This care is captured in the last restriction on local policy:

> The Request-URI created by a loose-routing element MUST differ from the URI in the topmost Route header.

This restriction should be kept even if the proposal to deprecate loop detection at proxies is accepted to avoid triggering loop detections in currently deployed systems. Likewise, Whether or not we accept this loose-routing proposal, this restriction needs to be kept for UACs configured to use default outbound proxies (DOP) to avoid inducing a loop at the DOP. One simple way to satisfy this restriction is through ensuring the maddr parameters in the Request-

URI and topmost Route header field value are different (added or removed if necessary). An algorithm for ensuring this restriction is met is presented later in this document.

3. A step beyond Loose Routing

A loose-routing element has a choice on removing the topmost Route header field value. Further, a new behavior is now possible. A loose-routing element MAY push new values onto the Route stack.

This allows network elements to place more than one service node on the path of a request (similar to service-route, but without the necessary participation of the originating UA). Examples of its use are included below.

4. Algorithm for Ensuring Request-URI and Topmost Route Differ

A loose-routing element (such as an endpoint configured to use an outbound proxy) may wish to send a request where the user and hostport portions of the Request-URI and topmost Route header field value are the same. It can use the following algorithm to ensure the remainder of the Request-URI and topmost Route header field value are sufficiently different to avoid inducing a loop at the next hop. For each of these points, D is the address of the next hop (which may or may not be equivalent to A).

- If the topmost element in the received Route header field is <sip:a@A>, the outgoing request will contain

```
METHOD sip:a@A;maddr=D
Route: <sip:a@A>
```

- If the topmost element in the received Route header field is <sip:a@A;maddr=D>, the outgoing request will contain

```
METHOD sip:a@A
Route: <sip:a@A;maddr=D>
```

- If the topmost element in the received Route header field is <sip:a@A;maddr=B> and D!=B, the outgoing request will contain

```
METHOD sip:a@A;maddr=D
Route: <sip:a@A;maddr=B>
```

5. Examples

These examples show that the mechanisms proposed here can be used to satisfy various scenarios. They are NOT intended to be normative in any sense. In particular, they are NOT definitions of how any scenario must be realized. They are simply examples of how they can be realized.

5.1 Current Strict Routing Behavior

Strict routing as defined in rfc2543bis (up to version 5) is just a particular local routing policy. The policy in this case is to remove the topmost Route header field value, place it in the Request-URI, and use the contents of the new Request-URI to determine where to send the message. Existing strict-route implementations are thus compatible with this proposal.

5.2 Using Preloaded Route-sets

If a UA has been given a Route header field to include in every initial request to a dialog, it SHOULD append the intended destination as the last value in the Route header field. As the examples will show, this ensures that the intended destination is not lost when traversing a strict routing element.

5.3 Using a Default Outbound Proxy

Elements configured to send all requests to a default outbound proxy as discussed in rfc2543bis-05 are already implementing another particular loose-routing policy.

In this case, the policy is to form requests normally, but not to remove the topmost element from the route set before constructing the Route header field. Once the message has been constructed, it is sent directly to the default outbound proxy instead of locating the server based on the contents of the Request-URI.

There is an error in the normative definition of this behavior in bis-05. If it is followed correctly, the topmost Route and Request-URI will be identical and the DOP will detect a loop. The algorithm presented in section Section 4 can be used to ensure the topmost Route header field and Request-URI are sufficiently different to avoid that problem. In this case, of course, the input to the algorithm is the topmost element of the route set instead of the topmost Route header field value of a received message.

Cases using default outbound proxies both with and without preloaded routes are included in the examples below.

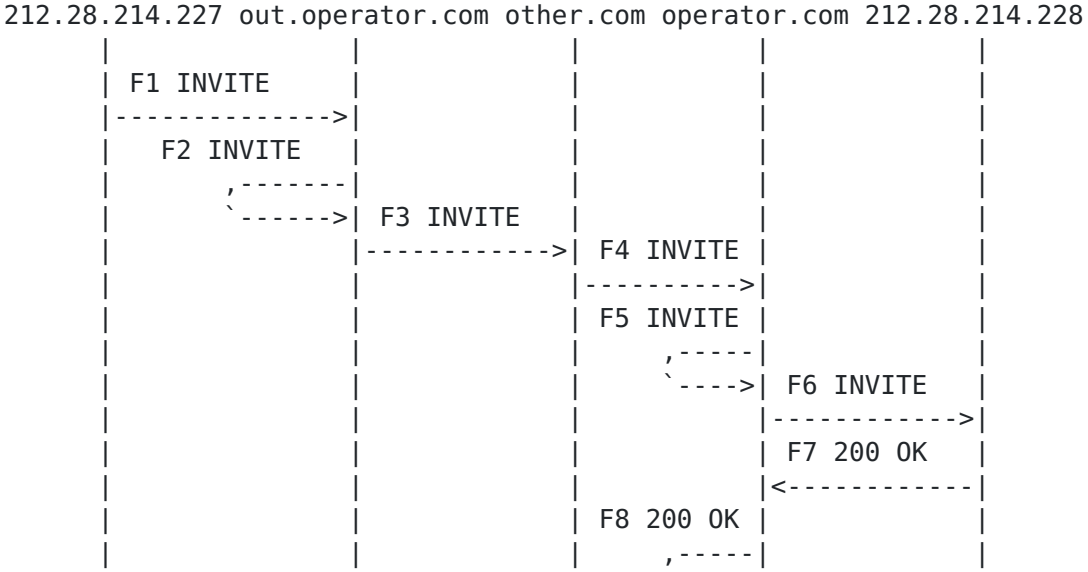
5.4 Being a Default Outbound Proxy

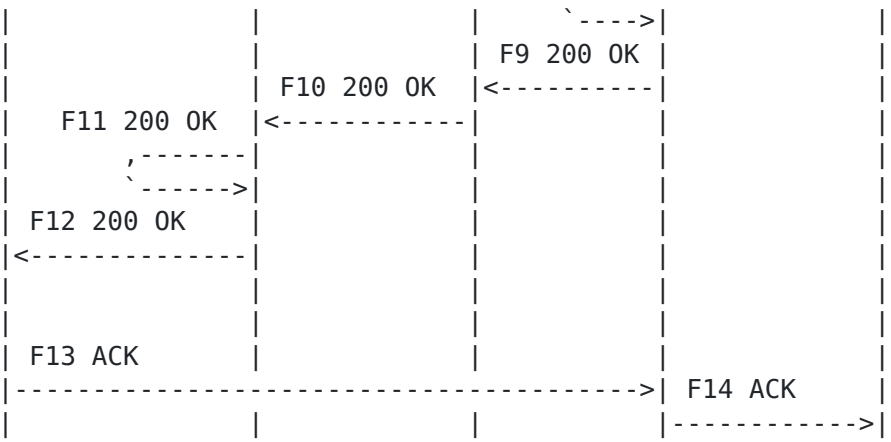
Under this proposal, a default outbound proxy is not a special element. It does not have to know it is a default outbound proxy, that burden lies on the UACs instructed to use it. A default outbound proxy may utilize any routing policy (including a strict routing policy). Examples of a default outbound proxy in use are included below.

5.5 Service Route

A service element on the path of an initial request in a dialog with a preloaded Route could choose to retarget the request (fully rewriting the request URI) or simply to rewrite the maddr parameter and leave the Route header field alone - moving the request towards the resource identified in the topmost Route header field value. It can also choose whether it wishes to Record-Route or not.

The following shows how the scenario described in draft-bjorkner-sip-serviceroute-00 can be satisfied under this proposal. In that draft, the originating UA has knowledge that the request needs to visit two service proxies. Here, that UA places the two proxies along with the desired destination into a preloaded Route header field. Adding the desired destination to the end of the preloaded route protects against encountering proxies along the path that implement a strict routing policy. Note that having out.operator.com appear in the preloaded route set is only one way of implementing this behavior. The service could also have instructed the UA to use out.operator.com as a default outbound proxy (shown in a later example).






```
F1  INVITE sip:hegu@operator.com SIP/2.0
    Via: . . .
    From: sip:jbj@operator.com;tag=42
    To: sip:hegu@operator.com
    Call-ID: . . .
    Cseq: . . .
    Contact: sip:jbj@212.28.214.227
    Content-Length: . . .
    Content-Type: application/sdp
    Route: <sip:out.operator.com>,
           <sip:other.com>,
           <sip:hegu@operator.com>
```

The service proxies process the Route headers using a local policy that preserves the Request-URI.

```
F2 (spirals through out.operator.com)
    INVITE sip:hegu@operator.com SIP/2.0
    Via: . . .
    From: sip:jbj@operator.com;tag=42
    To: sip:hegu@operator.com
    Call-ID: . . .
    Cseq: . . .
    Contact: sip:jbj@212.28.214.227
    Content-Length: . . .
    Content-Type: application/sdp
    Route:<sip:other.com>,<sip:hegu@operator.com>

F3  (sent by loose route policy to other.com)
    INVITE sip:hegu@operator.com;maddr=212.28.214.2 SIP/2.0
    Via: . . .
    From: sip:jbj@operator.com;tag=42
```

```
       To: sip:hegu@operator.com
       Call-ID: . . .
       Cseq: . . .
       Contact: sip:jbj@212.28.214.227
       Content-Length: . . .
       Content-Type: application/sdp
       Route: <sip:hegu@operator.com>

   F4  (sent by loose route policy to operator.com)
       INVITE sip:hegu@operator.com;maddr=212.28.214.2 SIP/2.0
       Via: . . .
       From: sip:jbj@operator.com;tag=42
       To: sip:hegu@operator.com
       Call-ID: . . .
       Cseq: . . .
       Contact: sip:jbj@212.28.214.227
       Content-Length: . . .
       Content-Type: application/sdp

   F5  (spirals through operator.com)
       INVITE sip:hegu@operator.com SIP/2.0
       Via: . . .
       From: sip:jbj@operator.com;tag=42
       To: sip:hegu@operator.com
       Call-ID: . . .
       Cseq: . . .
       Contact: sip:jbj@212.28.214.227
       Content-Length: . . .
       Content-Type: application/sdp

   F6  INVITE sip:hegu@212.28.214.228 SIP/2.0
       Via: . . .
       From: sip:jbj@operator.com;tag=42
       To: sip:hegu@operator.com
       Call-ID: . . .
       Cseq: . . .
       Contact: sip:jbj@212.28.214.227
       Content-Length: . . .
       Content-Type: application/sdp
       Record-Route: <sip:hegu@operator.com>

   F7  SIP/2.0 200 OK
   /   Via: . . .
   F12 From: sip:jbj@operator.com;tag=42
       To: sip:hegu@operator.com;tag=11
       Call-ID: . . .
       Cseq: . . .
       Contact: sip:jbj@212.28.214.227
```

```
    Content-Length: . . .
    Content-Type: application/sdp
    Record-Route: <sip:hegu@operator.com>

F13 ACK sip:hegu@operator.com SIP/2.0
    Via: . . .
    From: sip:jbj@operator.com;tag=42
    To: sip:hegu@operator.com;tag=11
    Call-ID: . . .
    Cseq: . . .
    Contact: sip:jbj@212.28.214.227
    Content-Length: . . .
    Content-Type: application/sdp
    Route: <sip:hegu@212.28.214.228>

F14 ACK sip:hegu@212.28.214.228 SIP/2.0
    Via: . . .
    From: sip:jbj@operator.com;tag=42
    To: sip:hegu@operator.com;tag=11
    Call-ID: . . .
    Cseq: . . .
    Contact: sip:jbj@212.28.214.227
    Content-Length: . . .
    Content-Type: application/sdp
```

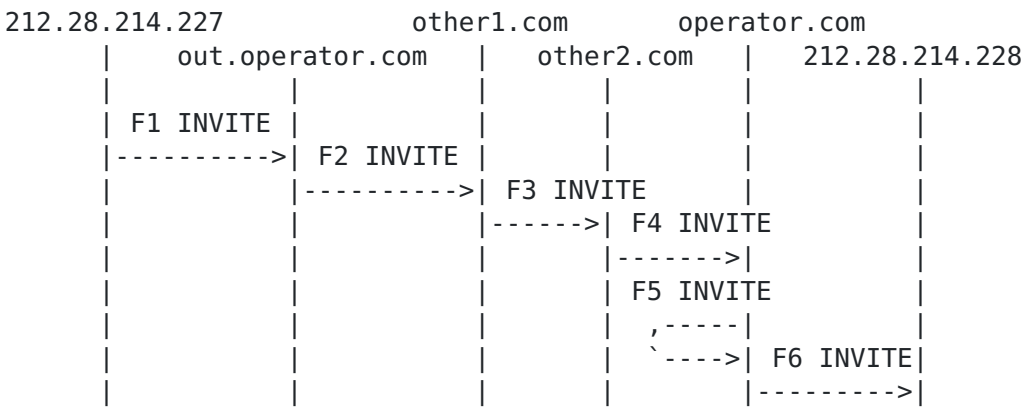
5.6 Traversing a Home proxy while traveling

When attached to a foreign network (while visiting another company for example), a UA may want all dialog initiating requests to visit its "Home" proxy for some processing. This is just a special case of the Service Route example above - the UA need only include a preloaded Route containing its home proxy.

5.7 Service nodes provided by the service

In some cases, a service provider may not be able to rely on an endpoint providing trustworthy information for the initial routing of a request through their service nodes. They may prefer, instead, to have a proxy in their domain of control provide that information.

Consider operator.com again. Suppose they have given the UA out.operator.com as a default outbound proxy, and had two other services they wished the request to visit. Suppose further that one of those services wishes to remain in the dialog. The flow above can be modified as follows:



F1 (sent by default outbound proxy policy to out.operator.com)

```
INVITE sip:hegu@operator.com SIP/2.0
Via: . . .
From: sip:jbj@operator.com;tag=42
To: sip:hegu@operator.com
Call-ID: . . .
Cseq: . . .
Contact: sip:jbj@212.28.214.227
Content-Length: . . .
Content-Type: application/sdp
```

Note the lack of a preloaded Route. The proxy at out.operator.com can now add the service destinations.

F2 (sent by loose-route policy to other1.com)

```
INVITE sip:hegu@operator.com SIP/2.0
Via: . . .
From: sip:jbj@operator.com;tag=42
To: sip:hegu@operator.com
Call-ID: . . .
Cseq: . . .
Contact: sip:jbj@212.28.214.227
Content-Length: . . .
Content-Type: application/sdp
Route:<sip:other2.com>,<sip:hegu@operator.com>
```

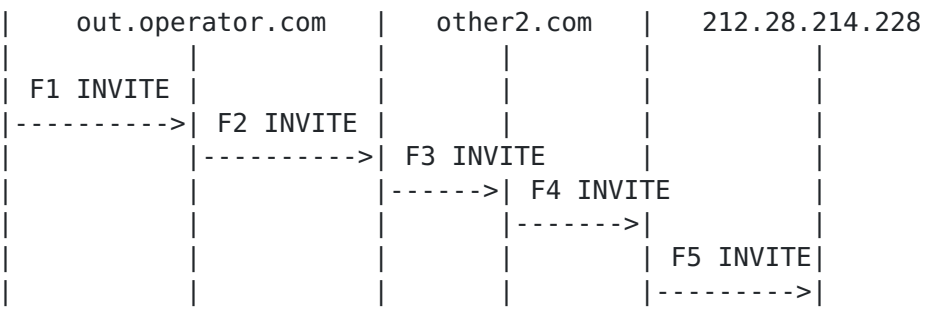
F3 (sent by loose route policy to other2.com)

```
INVITE sip:hegu@operator.com;maddr=212.28.214.9 SIP/2.0
Via: . . .
From: sip:jbj@operator.com;tag=42
To: sip:hegu@operator.com
Call-ID: . . .
Cseq: . . .
```

```
      Contact: sip:jbj@212.28.214.227
      Content-Length: . . .
      Content-Type: application/sdp
      Route: <sip:hegu@operator.com>

  F4  (sent by loose route policy to operator.com)
      INVITE sip:hegu@operator.com;maddr=212.28.214.2 SIP/2.0
      Via: . . .
      From: sip:jbj@operator.com;tag=42
      To: sip:hegu@operator.com
      Call-ID: . . .
      Cseq: . . .
      Contact: sip:jbj@212.28.214.227
      Content-Length: . . .
      Record-Route: <sip:hegu@operator.com;maddr=212.28.214.9>
      Content-Type: application/sdp
      Route: <sip:hegu@operator.com>

  F5  (spirals through operator.com)
      INVITE sip:hegu@operator.com SIP/2.0
      Via: . . .
      From: sip:jbj@operator.com;tag=42
      To: sip:hegu@operator.com
      Call-ID: . . .
      Cseq: . . .
      Contact: sip:jbj@212.28.214.227
      Content-Length: . . .
      Record-Route: <sip:hegu@operator.com;maddr=212.28.214.9>
      Content-Type: application/sdp

  F6  INVITE sip:hegu@212.28.214.228 SIP/2.0
      Via: . . .
      From: sip:jbj@operator.com;tag=42
      To: sip:hegu@operator.com
      Call-ID: . . .
      Cseq: . . .
      Contact: sip:jbj@212.28.214.227
      Content-Length: . . .
      Content-Type: application/sdp
      Record-Route: <sip:hegu@operator.com;maddr=212.28.214.9>
      Record-Route: <sip:hegu@operator.com>
```

Now suppose that other1.com and other2.com utilize current proxy implementations with strict route policies. The flow would look like this:

```
212.28.214.227           other1.com       operator.com
```



F1 (sent by default outbound proxy policy to out.operator.com)

```
INVITE sip:hegu@operator.com SIP/2.0
Via: . . .
From: sip:jbj@operator.com;tag=42
To: sip:hegu@operator.com
Call-ID: . . .
Cseq: . . .
Contact: sip:jbj@212.28.214.227
Content-Length: . . .
Content-Type: application/sdp
```

F2 (sent by loose-route policy to other1.com)

```
INVITE sip:hegu@operator.com SIP/2.0
Via: . . .
From: sip:jbj@operator.com;tag=42
To: sip:hegu@operator.com
Call-ID: . . .
Cseq: . . .
Contact: sip:jbj@212.28.214.227
Content-Length: . . .
Content-Type: application/sdp
Route:<sip:other2.com>,<sip:hegu@operator.com>
```

F3 (sent by strict route policy to other2.com)

```
INVITE sip:other2.com SIP/2.0
Via: . . .
From: sip:jbj@operator.com;tag=42
To: sip:hegu@operator.com
Call-ID: . . .
Cseq: . . .
Contact: sip:jbj@212.28.214.227
Content-Length: . . .
Content-Type: application/sdp
Route: <sip:hegu@operator.com>
```

Note that the service at other2.com now has to go digging into the message to see what resource the service is being invoked against

(looking at the last Route header value).

```
F4  (sent by strict route policy to operator.com)
    INVITE sip:hegu@operator.com SIP/2.0
    Via: . . .
    From: sip:jbj@operator.com;tag=42
    To: sip:hegu@operator.com
    Call-ID: . . .
    Cseq: . . .
    Contact: sip:jbj@212.28.214.227
    Content-Length: . . .
    Record-Route: <sip:other2.com>
    Content-Type: application/sdp
```

Note that from other2 will now remain on the path, but the information about what resource the service was being invoked against and, quite likely, information about which service to invoke is gone (just like strict routing to date). If other2 really needed that information for future requests, it would have to construct a different Record-Route value.

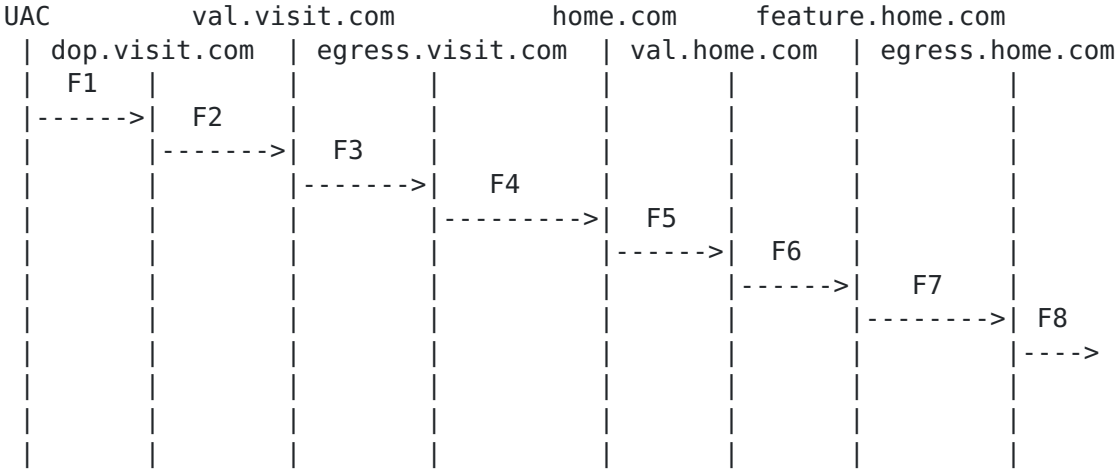
```
F5  INVITE sip:hegu@212.28.214.228 SIP/2.0
    Via: . . .
    From: sip:jbj@operator.com;tag=42
    To: sip:hegu@operator.com
    Call-ID: . . .
    Cseq: . . .
    Contact: sip:jbj@212.28.214.227
    Content-Length: . . .
    Content-Type: application/sdp
    Record-Route: <sip:other2.com>
    Record-Route: <sip:hegu@operator.com>
```

5.8 Multiple Service Providers

The case for pushing route values becomes even more compelling when you look at a call that is traversing multiple service providers. Consider a UA visiting a network that needs to place a call through its home services. The visiting network has services it must provide, and has given the UA a default outbound proxy to utilize. A new request from the UA visits the outbound proxy, which routes the request through some validation service and then to an egress proxy to the UAs home service. The home service runs the request through

its own validation service and some feature service associated with the caller before senting the request to a third service provider for delivery.

Here's a possible flow through the visited and home services. The activity in the third service is not shown since it is similar to what is shown for the first two. The common headers have also been omitted for the sake of brevity.



F1 (sent to dop.visit.com by default outbound proxy policy)

```
INVITE sip:someone@thirdparty.com SIP/2.0
Contact: <sip:uacaddress>
Route: <sip:homeuser@home.com>,
       <sip:someone@thirdparty.com>
```

F2 (sent to val.visit.com and directed to egress.visit.com by loose-route policy)

```
INVITE sip:someone@thirdparty.com SIP/2.0
Route: <sip:egress.visit.com>,
       <sip:homeuser@home.com>,
       <sip:someone@thirdparty.com>
Contact: <sip:uacaddress>
```

F3 (sent to egress.visit.com through a route-stripping but Request-URI preserving loose-route policy)

```
INVITE sip:someone@thirdparty.com SIP/2.0
Contact: <sip:uacaddress>
Route: <sip:homeuser@home.com>,
       <sip:someone@thirdparty.com>
Record-Route: <sip:egress.visit.com>
```

F4 (sent to home.com through a route-stripping and request-URI modifying policy)

```
       INVITE sip:homeuser@home.com SIP/2.0
       Contact: <sip:uacaddress>
       Route: <sip:someone@thirdparty.com>
       Record-Route: <sip:egress.visit.com>

   F5  (sent to val.home.com and directed through feature
        and egress.home.com)
       INVITE sip:homeuser@home.com SIP/2.0
       Contact: <sip:uacaddress>
       Route: <sip:feature.home.com>,
              <sip:egress.home.com>,
              <sip:someone@thirdparty.com>
       Record-Route: <sip:egress.visit.com>

   F6  (sent to feature.home.com using a stripping policy)
       INVITE sip:homeuser@home.com SIP/2.0
       Contact: <sip:uacaddress>
       Route: <sip:egress.home.com>,
              <sip:someone@thirdparty.com>
       Record-Route: <sip:egress.visit.com>

   F7  (sent to egress.home.com using a stripping policy)
       INVITE sip:homeuser@home.com SIP/2.0
       Contact: <sip:uacaddress>
       Route: <sip:someone@thirdparty.com>
       Record-Route: <sip:egress.visit.com>,
                     <sip:egress.home.com>

   F8  (sent to thirdparty.com using a strict routing policy)
       INVITE sip:someone@thirdparty.com SIP/2.0
       Contact: <sip:uacaddress>
       Record-Route: <sip:egress.visit.com>,
                     <sip:egress.home.com>
```

6. Acknowledgements

This proposal reflects the work of many on the SIPWG mailing list. The following contributed special efforts to ensure its correctness during IETF52:

Henrik Gustafsson
Sean Olsen
Jo Hornsby
Ben Campbell
Jonathan Rosenberg

References

Author's Address

Robert J. Sparks
dynamicsoft
5100 Tennyson Parkway
Suite 1200
Plano, TX 75024

EMail: rsparks@dynamicsoft.com

Full Copyright Statement

Acknowledgement

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