SIP WG
Internet-Draft
Updates: 3261 (if approved)
Expires: December 18, 2006

S. Lawrence
Pingtel Corp.
A. Hawrylyshen
Ditech Networks Inc.
R. Sparks
Estacado Systems
June 16, 2006

Diagnostic Responses for Session Initiation Protocol Hop Limit Errors
draft-ietf-sip-hop-limit-diagnostics-03

Status of this Memo

Copyright Notice

Abstract

The Session Initiation Protocol (SIP) imposes a limit on the number of hops a request can transit on the way to its destination. When this limit is reached, a 483 (Too Many Hops) error response is returned. The present form of the 483 response does not provide enough information for the UAC or proxy on the path to diagnose

failures whose symptom is that the hop limit is reached. This document specifies additional diagnostic information to be returned in a 483 response.

Table of Contents

1. Conventions and Definitions

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 [RFC2119].

2. Diagnosing Hop Limit Exceeded Failures

The SIP protocol imposes a limit on the number of hops a request can transit on the way to its destination. The number of hops remaining for the request is carried in the Max-Forwards header, and is decremented each time the request is forwarded. When a SIP User Agent receives a request whose Max-Forwards is zero (0), it returns a 483 error response to indicate that the limit was reached.

The 483 response alone does not provide enough information for the originating UAC to determine where the problem lies. The problem is rarely that the target of the request was actually further away than the Max-Forwards limit allowed. The problem is usually incorrect routing; often a routing loop.

2.1. Limitations of the 483 Error Response

Section 20.22 of RFC 3261 [RFC3261] says:

> The Max-Forwards header field must be used with any SIP method to limit the number of proxies or gateways that can forward the request to the next downstream server. This can also be useful when the client is attempting to trace a request chain that appears to be failing or looping in mid-chain.

In practice, there is too little information returned in a 483 response for it to be of much use as a diagnostic tool. When a request has traversed a series of proxies, the response follows the Vias back to the requester - in the case of a typical 483 response it can be difficult to determine even what server the response came from. Even when the rejecting server does identify itself, it can be difficult to figure out why the request got there.

The following is an actual example request; the IP addresses and domain names have been changed, but it is otherwise complete (it was intentionally sent without SDP for brevity):

```
INVITE sip:9999@example.com SIP/2.0
Via: SIP/2.0/TCP 10.1.1.20:59449
     ;branch=z9hG4bK-56ec69968c31f498c9a5573a00c8fc04
To: sip:9999@example.com
From: Sip Send <sip:sipsend@10.1.1.20>;tag=08e2f515
Call-ID: 159213b1aa5a67bc6eca6c4c2bad9f94@10.1.1.20
Cseq: 1 INVITE
Max-Forwards: 1
User-Agent: sipsend/0.02
Date: Wed, 12 Oct 2005 20:09:29 GMT
Content-Length: 0
```

This request was sent with the Max-Forwards header field value set to only 1 to force the error response: it should traverse only the first outbound proxy, and then be rejected by the next system that it encounters.

The response received in this case was:

```
SIP/2.0 483 Too Many Hops
Via: SIP/2.0/TCP 10.1.1.20:59449
     ;branch=z9hG4bK-56ec69968c31f498c9a5573a00c8fc04
To: sip:9999@example.com;tag=-1574266585
From: Sip Send <sip:sipsend@10.1.1.20>;tag=08e2f515
Call-ID: 159213b1aa5a67bc6eca6c4c2bad9f94@10.1.1.20
Cseq: 1 INVITE
Content-Length: 0
```

There is no indication in the response of what server returned the error. Even with the error only one hop beyond the first proxy, there is no way to determine if that first proxy has routed the request incorrectly.

2.2. Improved Diagnostic Information in Responses

In some ways, the SIP Max-Forwards mechanism is analogous to the Time To Live (TTL) field in an IP datagram. The TTL field was originally [RFC0791] intended to be the maximum number of seconds that a datagram should remain in the network. In practice, IP TTL has evolved into a hop count, since each system forwarding a datagram was (is) required to decrement the TTL by (at least) one. As an aid to diagnosing problems, the Internet Control Message Protocol [RFC0792] defines a "Time Exceeded Message" to be sent by any system that discards an IP datagram because it was received with a TTL value of zero (0). The Time Exceeded Message is sent to the source address of the discarded datagram, and includes a field that carries the "Internet Header + 64 bits of Original Data Datagram". This allows the originator to see the datagram as it appeared where it was discarded. The 'traceroute' tool determines the route followed between a given pair of IP addresses by sending a series of IP packets from the source to the destination with gradually increasing TTL values. As each packet reaches its limit, an ICMP Time Exceeded Message is returned by the router that is discarding it; some checks on the route can be made by examining the original packet as it arrived at each hop.

As an aid to diagnosing problems that result in 483 responses, it would be useful to know how the failed request arrived at the rejecting system; both what path it followed to get there, and what

the request looked like when it ran out of hops. One way to accomplish this is to return the SIP header of the rejected message to the UAC that originated it. Doing so is already allowed by existing rules:

RFC 3261 [RFC3420] (section 7.4) says: "All responses MAY include a body.".

RFC 3420 [RFC3420] defines the Content-Type "message/sipfrag" to "allow SIP entities to make assertions about a subset of a SIP message".

3. Proxy Behavior

This document adds the following new rule for all SIP Proxy implementations:

> Any 483 response SHOULD be constructed with both:
>
> > A message body of type message/sipfrag containing as much as possible of the SIP header from the rejected request.
> >
> > A Warning header with a warn-code of 399 that identifies the system returning the error.
>
> Exceptions to this are allowed so that a system is allowed to omit parts of the message either to limit the size of the response or to conform to a local security policy. See Section 3.1.

3.1. Pruning Responses

A server may be unable or unwilling to return the full request message in every 483 response. The returned message may exceed the maximum message size it can handle, or may include security-sensitive information.

It is RECOMMENDED that when the complete message cannot be returned, that at least all of the Route and Via headers be included in the message/sipfrag body. In the example (Section 6), this would at least enable the end user to determine which proxies were in the routing loop and how the request arrived there, but not the specific address transformations that caused the loop.

If including all Via and Route headers is still too large, the implementation SHOULD remove the oldest Vias (those nearest the message originator) until the size is acceptable; the only exception to this rule is when . This way, the originator can detect that some Vias were removed (because the one that it put on is missing).

4. UAS Behavior

Since a UAS is not required to validate the Max-Forwards header field value when processing a received message, it is not required to return the received message header as described above for proxies, but it MAY do so.

5. UAC Behavior

This specification does not mandate any new behavior for a UAC. The returned message is available to the UAC to either pass on to the user or to use for any automated diagnostic process.

Note that a UAC MUST be prepared to receive message bodies in a response that it does not understand and did not request; this is already required by [RFC3261] section 20.1 Accept:

> The Accept header field follows the syntax defined in [H14.1]. The semantics are also identical, with the exception that if no Accept header field is present, the server SHOULD assume a default value of application/sdp.

[H14.1] refers to RFC 2616 [RFC2616], which specifically limits the semantics of the Accept header fields in section 10.4.7 '406 Not Acceptable' as follows:

> Note: HTTP/1.1 servers are allowed to return responses which are not acceptable according to the accept headers sent in the request. In some cases, this may even be preferable to sending a 406 response. User agents are encouraged to inspect the headers of an incoming response to determine if it is acceptable.

6. Example

This example shows how this proposed change is used to diagnose an example routing problem.

Here is a request sent to a proxy that implements the suggested content in a 483 response.

```
INVITE sip:9999@example.com SIP/2.0
Via: SIP/2.0/TCP 10.1.1.20:40221
;branch=z9hG4bK-931ea14405e9da8c95cf4ed60a71f59f
To: sip:9999@example.com
From: Sip Send <sip:sipsend@10.1.1.20>;tag=612f37e7
Call-ID: 7a26fdad2cb40d48e81e10d6fce39825@10.1.1.20
Cseq: 1 INVITE
Max-Forwards: 9
User-Agent: sipsend/0.02
Date: Fri, 14 Oct 2005 15:35:53 GMT
Content-Length: 0
```

The target user '9999' is one that has been deliberately configured to go into a forwarding loop alternating between two addresses (neither of them the original target); a situation that is currently difficult to diagnose. A relatively low Max-Forwards header field value of 9 was chosen to improve readability.

The response received was:

```
SIP/2.0 483 Too many hops
Warning: 399 192.0.2.162:5080 Too Many Hops
From: Sip Send <sip:sipsend@10.1.1.20>;tag=612f37e7
To: sip:9999@example.com
Call-Id: 7a26fdad2cb40d48e81e10d6fce39825@10.1.1.20
Cseq: 1 INVITE
Via: SIP/2.0/TCP 10.1.1.20:40221
;branch=z9hG4bK-931ea14405e9da8c95cf4ed60a71f59f
Content-Type: message/sipfrag
Content-Length: 1014
Date: Fri, 14 Oct 2005 15:27:47 GMT

INVITE sip:InfiniteLoop@example.com SIP/2.0
Record-Route: <sip:192.0.2.162:5080
;lr;a;t=612f37e7;s=96e09e8e8c93a8c60bf460029847f4b1>
Via: SIP/2.0/TCP 192.0.2.162
;branch=z9hG4bK-42e47bba67559bd9a3da1934a70bbc37
Via: SIP/2.0/TCP 192.0.2.162:5080
;branch=z9hG4bK-50d909f1209f7a820de85c7831846330
Via: SIP/2.0/TCP 192.0.2.162
;branch=z9hG4bK-994f162bc179fb75093166fabbfd13c7
Via: SIP/2.0/TCP 192.0.2.162:5080
;branch=z9hG4bK-708842ad6ea22f8fa6e39c503d3d803e
Via: SIP/2.0/TCP 192.0.2.162
;branch=z9hG4bK-50b581a06ca023ebcddbc82c5221149c
Via: SIP/2.0/TCP 192.0.2.14:1084
;branch=z9hG4bK994220327571023745d7996c13560a11.0
Via: SIP/2.0/TCP 192.0.2.14:40221
;branch=z9hG4bK-931ea14405e9da8c95cf4ed60a71f59f
To: sip:9999@example.com
From: Sip Send <sip:sipsend@10.1.1.20>;tag=612f37e7
Call-Id: 7a26fdad2cb40d48e81e10d6fce39825@10.1.1.20
Cseq: 1 INVITE
Max-Forwards: 0
User-Agent: sipsend/0.02
Date: Fri, 14 Oct 2005 15:35:53 GMT
Content-Length: 0
```

The Warning header in this response identifies the server returning the error (192.0.2.162:5080). The Via headers of the returned message/sipfrag body show the path the failed message took. The returned request line also shows that the target URI has been changed to the user 'InfiniteLoop'.

Resending the request with a hop limit one less than before (8), shows that at that hop the request URI is to user 'LoopForever':

```
INVITE sip:LoopForever@example.com SIP/2.0
Record-Route: <sip:192.0.2.162:5080
;lr;a;t=4f18a30b;s=a9711e1704ccd5273955589c5fe94745>
Via: SIP/2.0/TCP 192.0.2.162
;branch=z9hG4bK-9b7f69455266f7cccd4ae8a285c0417c
Via: SIP/2.0/TCP 192.0.2.162:5080
;branch=z9hG4bK-b9d3e2aff65e68497849a2609bf8c373
Via: SIP/2.0/TCP 192.0.2.162
;branch=z9hG4bK-a178f4c5a3b8bbf35f979bc6c6d33022
Via: SIP/2.0/TCP 192.0.2.162:5080
;branch=z9hG4bK-3c098b8d58e4b6ce98fca3495263e795
Via: SIP/2.0/TCP 192.0.2.162
;branch=z9hG4bK-e7ceb06ed917d59024c905b3ee60e4cc
Via: SIP/2.0/TCP 192.0.2.14:1085
;branch=z9hG4bK8d685f52450e87da45d7996c13560a11.0
Via: SIP/2.0/TCP 192.0.2.14:56114
;branch=z9hG4bK-4ae5a563b0cbc76aef7be17115836dea
To: sip:9999@example.com
From: Sip Send <sip:sipsend@10.1.1.20>;tag=4f18a30b
Call-Id: 39106d45526cb5e78bf8dac378e05817@10.1.1.20
Cseq: 1 INVITE
Max-Forwards: 0
User-Agent: sipsend/0.02
Date: Fri, 14 Oct 2005 15:42:21 GMT
Content-Length: 0
Route: <sip:192.0.2.162:5070;transport=tcp;lr>
```

Reducing the limit one at a time (or starting from 1 and working forward), a UAC can determine that the InfiniteLoop/LoopForever forwarding loop exists (in reality, of course, the user names would rarely be such good hints), and where in the forwarding sequence the original '9999' was changed to enter the loop.

Without the returned request headers, the 483 response does not help the request originator (or any proxy administrator on the path) diagnose why the error has occurred. With it, in this case a diagnostic application running as a User Agent is able to at least identify that there is a routing problem and which proxy is misrouting the request.

7. IANA Considerations

None.

8. Security Considerations

The proposed mechanism provides a means by which topology and some routing information about a set of SIP systems can be discovered. The mechanism is very similar to that provided for IP routing by the traceroute tool.

Some systems may not want to expose as much information as is available in the full set of SIP request headers by returning them in the error response body. In this case, the system returning the error should prune the response as recommended in Section 3.1.

It may be possible for an attacker to forge very large messages with a deliberately low Max-Forwards header field values so that a Server will send error responses. If those responses are fragemented and sent on a transport that does not have congestion control, this could cause a small number more response packets than the attacker sent requests. The server is allowed to prune the response as recommended in Section 3.1 to reduce the response size, which reduces the opportunity for the attacker to generate many fragments. Other than this, the returned response message is roughly twice the size of the original request, and gets smaller as the Via and Route headers are removed in transit, so there is little amplification.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC3261] 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.

[RFC3420] Sparks, R., "Internet Media Type message/sipfrag", RFC 3420, November 2002.

9.2. Informative References

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, September 1981.

Authors' Addresses

Scott Lawrence
Pingtel Corp.
400 West Cummings Park
Suite 2200
Woburn, MA 01801
USA

Phone: +1 781 938 5306
Email: slawrence@pingtel.com

Alan Hawrylyshen
Ditech Networks Inc.
1167 Kensington Rd NW
Suite 200
Calgary, Alberta T2N 1X7
Canada

Phone: +1 403 806 3366
Email: ahawrylyshen@ditechnetworks.com

Robert Sparks
Estacado Systems
17210 Campbell Road
Suite 250
Dallas, Texas 75254-4203
USA

Email: RjS@nostrum.com

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

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