Correct transaction handling for 200 responses to Session Initiation Protocol INVITE requests
draft-sparks-sip-invfix-01

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on August 10, 2008.

Copyright Notice

Copyright (C) The IETF Trust (2008).

Abstract

This document normatively updates RFC 3261, the Session Initiation Protocol (SIP), to address an error in the specified handling of success (200 class) responses to INVITE requests. Elements following RFC 3261 exactly will misidentify retransmissions of the request as a new, unassociated, request. The correction involves modifying the INVITE transaction state machines. The correction also changes the
way responses that cannot be matched to an existing transaction are handled to address a security risk.

Table of Contents

1. Conventions and Definitions ........................................... 3
2. Introduction .......................................................... 3
3. Reason for Change .................................................... 3
4. Summary of Change .................................................... 4
5. Consequences if Not Approved ....................................... 4
6. The Change ............................................................ 4
7. Change Details ........................................................ 4
   7.1. UAS Impacts ..................................................... 5
   7.2. UAC Impacts ..................................................... 7
   7.3. Proxy Considerations .......................................... 8
8. Exact changes to RFC3261 .............................................. 9
   8.1. Page 85 .......................................................... 9
   8.2. Page 107 ........................................................ 9
   8.3. Page 114 ........................................................ 9
   8.4. Pages 126 through 128 .......................................... 10
   8.5. Pages 134 to 135 ............................................... 13
   8.6. Page 136 ........................................................ 13
   8.7. Page 137 ........................................................ 15
   8.8. Page 144 ........................................................ 15
   8.9. Page 146 ........................................................ 15
   8.10. Page 265 ......................................................... 16
9. IANA Considerations ................................................... 16
10. Security Considerations .............................................. 16
11. Acknowledgments ..................................................... 16
12. References ............................................................ 17
   12.1. Normative References ......................................... 17
   12.2. Informative References ....................................... 17
Author’s Address ........................................................ 17
Intellectual Property and Copyright Statements ....................... 18
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. Introduction

This document describes an essential correction to the Session Initiation Protocol (SIP), defined in [RFC3261], using the process defined in [I-D.drage-sip-essential-correction]. The change addresses an error in the handling of 200 class responses to INVITE requests that leads to retransmissions of the INVITE being treated as new requests and forbids forwarding stray INVITE responses.

3. Reason for Change

One use of the INVITE method in SIP is to establish new sessions. These "initial" INVITEs may fork at intermediaries, and more than one receiving endpoint may choose to accept the request. SIP is designed such that the requester receives all of these success responses.

Two sets of requirements in [RFC3261] work together to allow multiple 200s to be processed correctly by the requester. First, all elements are required to immediately destroy any INVITE client transaction state upon forwarding a matching 200 OK response. This requirement applies to both proxies and user agents (proxies forward the response upstream, the transaction layer at user agents forward the response to its "UA core"). Second, all proxies are required to statelessly forward any 200 OK responses that do not match an existing transaction, also called stray responses, upstream. The transaction layer at user agents is required to forward these responses to its UA core. Logic in the UA core deals with acknowledging each of these responses.

This technique for specifying the behavior was chosen over adjusting INVITE client transaction state machines as a simpler way to specify the correct behavior.

Over time, implementation experience demonstrated the existing text is in error. Once any element with a server transaction (say, a proxy in the path of the INVITE) deletes that transaction state, any retransmission of the INVITE will be treated as a new request, potentially forwarded to different locations than the original. Many implementations in the field have made proprietary adjustments to their transaction logic to avoid this error.
The requirement to statelessly forward stray responses has also been identified as a security risk. Through it, elements compliant to [RFC3261] are compelled to do work (forward packets) that is not protected by the admission policies applied to requests. This can be leveraged to, for instance, use a SIP proxy as an anonymizing forwarder of packets in a distributed DOS attack. General internet endpoints can also collude to tunnel non-SIP content through such proxies by wrapping them in an SIP response envelope.

4. Summary of Change

This correction document updates [RFC3261], changing the transitions in the INVITE client state machine such that the INVITE client transaction remains in place to receive multiple 200 OK responses. It adds a state to the INVITE server state machine to absorb retransmissions of the INVITE after a 200 OK response has been sent. It also forbids forwarding stray responses to INVITE requests (not just 200 OK responses), which RFC3261 required.

5. Consequences if Not Approved

Implementations strictly conformant to [RFC3261] will process retransmitted initial INVITE requests as new requests. Proxies may forward them to different locations than the original. Proxies may also be used as anonymizing forwarders of bulk traffic.

6. The Change

An element sending or receiving a 200 OK to an INVITE transaction MUST NOT destroy any matching INVITE transaction state. This state is necessary to ensure correct processing of retransmissions of the request and the retransmission of the 200 OK and ACK that follow.

When receiving any response SIP response, a transaction-stateful proxy MUST compare the transaction identifier in that response against its existing transaction state machines. The proxy MUST NOT forward the response if there is no matching transaction state machine.

7. Change Details

These changes impact requirements in several sections of RFC3261. The exact effect on that text is detailed in Section 8. This section describes the details of the change, particularly the impact on the
INVITE state machines, more succinctly to facilitate review and simplify implementation.

7.1.  UAS Impacts

To allow a UAS to recognize retransmissions of an INVITE as retransmissions instead of new requests, a new state, "Accepted" is added to the INVITE server transaction state machine. A new timer, Timer L, is also added to ultimately allow the state machine to terminate. A UAS in the "Proceeding" state will transition to the "Accepted" state when it issues a 2xx response, and will remain in that state just long enough to absorb any retransmissions of the INVITE.

If the UAC’s TU issues a 2xx response for this transaction while the state machine is in the "Proceeding" state, it MUST transition to the "Accepted" state and set Timer L to 64*T1.

While in the "Accepted" state, any retransmissions of the INVITE received will match this transaction state machine and will be absorbed by the machine without changing its state. These retransmissions are not passed onto the TU. RFC3261 requires the TU to periodically retransmit the 2xx response until it receives an ACK. The server transaction MUST NOT generate 2xx retransmissions on its own. Any retransmission of the 2xx response passed from the TU to the transaction while in the "Accepted" state MUST be passed to the transport layer for transmission.

When Timer L fires and the state machine is in the "Accepted" state, the machine MUST transition to the "Terminated" state. Once the transaction is in the "Terminated" state, it MUST be destroyed immediately. Timer L reflects the amount of time the TU will wait to receive an ACK for the 2xx it is emitting before considering the transaction failed.

Figure 1 graphically shows the part of the INVITE server state machine that has changed. The entire new INVITE server state machine is shown in Figure 4.
Figure 1: Changes to the INVITE server transaction state machine
7.2. UAC Impacts

In order to correctly distinguish retransmissions of 2xx responses from stray 2xx responses, the INVITE client state machine is modified to not transition immediately to "Terminated" on receipt of a 2xx response. Instead, the machine will transition to "Completed". Specifically, if a 2xx response is received while the client INVITE state machine is in the "Calling" or "Completed" states, it MUST transition to the "Completed" state and pass the 2xx response to the TU. A 2xx response received while in the "Completed" state MUST be passed to the TU and the machine remains in the "Completed" state. The client transaction MUST NOT generate an ACK to any 2xx response on its own. The TU responsible for the transaction will generate the ACK.

Any response received which does not match an existing client transaction state machine is simply dropped. (Implementations are, of course, free to log or do other implementation specific things with such responses, but the implementer should be sure to consider the impact of large numbers of malicious stray responses).

Figure 2 graphically shows the part of the INVITE client state machine that has changed. The entire new INVITE client state machine is shown in Figure 3.
Figure 2: Changes to the INVITE client transaction state machine

7.3. Proxy Considerations

A direct consequence of the change to the UAS state machine is that a transaction-stateful proxy will not forward any stray INVITE responses. When receiving any response SIP response, a transaction-stateful proxy MUST compare the transaction identifier in that response against its existing transaction state machines. The proxy
MUST NOT forward the response if there is no matching transaction state machine.

8. Exact changes to RFC3261

This section describes exactly the same changes as above, but shows exactly which text in RFC3261 is affected.

8.1. Page 85

Section 13.3.1.4 paragraph 4 is replaced entirely by

Once the response has been constructed, it is passed to the INVITE server transaction. In order to ensure reliable end-to-end transport of the response, it is necessary to periodically pass the response directly to the transport until the ACK arrives. The 2xx response is passed to the transport with an interval that starts at T1 seconds and doubles for each retransmission until it reaches T2 seconds (T1 and T2 are defined in Section 17). Response retransmissions cease when an ACK request for the response is received. This is independent of whatever transport protocols are used to send the response.

8.2. Page 107

Section 16.7 paragraphs 1 and 2 are replaced entirely by

When a response is received by an element, it first tries to locate a client transaction (Section 17.1.3) matching the response. If none is found, the element MUST NOT forward the response. If a transaction is found, the response is handed to the client transaction.

8.3. Page 114

Section 16.7, part 9, first paragraph. Replace this sentence

If the server transaction is no longer available to handle the transmission, the element MUST forward the response statelessly by sending it to the server transport.

with

If the server transaction is no longer available to handle the transmission, the response is simply discarded.
When in either the "Calling" or "Proceeding" states, reception of a response with status code from 300-699 MUST cause the client transaction to transition to "Completed". The client transaction MUST pass the received response up to the TU, and the client transaction MUST generate an ACK request, even if the transport is reliable (guidelines for constructing the ACK from the response are given in Section 17.1.1.3) and then pass the ACK to the transport layer for transmission. The ACK MUST be sent to the same address, port, and transport to which the original request was sent.

When in either the "Calling" or "Proceeding" states, reception of a 2xx response MUST cause the client transaction to transition to "Completed", and the response MUST be passed up to the TU. The client transaction MUST NOT generate an ACK to the 2xx response — its handling is delegated to the TU. A UAC core will send an ACK to the 2xx response using a new transaction. A proxy core will always forward the 2xx response upstream.

The client transaction SHOULD start timer D when it enters the "Completed" state for any reason, with a value of at least 32 seconds for unreliable transports, and a value of zero seconds for reliable transports. Timer D reflects the amount of time that the server transaction can remain in the "Completed" state when unreliable transports are used. This is equal to Timer H in the INVITE server transaction, whose default is 64*T1, and is also equal to the time a UAS core will wait for an ACK once it sends a 2xx response. However, the client transaction does not know the value of T1 in use by the server transaction or any downstream UAS cores, so an absolute minimum of 32s is used instead of basing Timer D on T1.

Any retransmissions of a response with status code 300-699 that are received while in the "Completed" state MUST cause the ACK to be re-passed to the transport layer for retransmission, but the newly received response MUST NOT be passed up to the TU.

Any retransmissions of a 2xx response that are received while in the "Completed" state MUST be passed up to the TU. The client transaction MUST NOT generate an ACK to the 2xx response. The client transaction takes no further action.
A retransmission of the response is defined as any response which would match the same client transaction based on the rules of Section 17.1.3.

If timer D fires while the client transaction is in the "Completed" state, the client transaction MUST move to the terminated state.

The client transaction MUST be destroyed the instant it enters the "Terminated" state.

Replace Figure 5 with
Figure 3: INVITE client transaction
8.5. Pages 134 to 135

Section 17.2.1 paragraph 4 is replaced with

If, while in the "Proceeding" state, the TU passes a 2xx response to the server transaction, the server transaction MUST pass this response to the transport layer for transmission. It is not retransmitted by the server transaction; retransmissions of 2xx responses are handled by the TU. The server transaction MUST then transition to the "Accepted" state.

8.6. Page 136

Replace Figure 7 with
INVITE
V send 100 if TU won’t in 200ms

send response

+--------+ 101-199 from TU
|        | send response
+--------+ Transport Err.
              | Inform TU

+--------++-+
|        |--|
+--------+ Invokes TU

300-699 from TU | 2xx from TU
send response | send response

INVITE V
Timer G fires
send response

+--------+ send response
|        | Completed
+--------+ INVITE

ACK

V

V

Accept

Timer H fires or Transport Error
Inform TU

Confirmed

^ 2xx from TU
send response

Timer I fires

Timer L fires

V


Terminated

Figure 4: INVITE server transaction
Section 17.2.1 - Replace the last paragraph (starting "Once the transaction") with

The purpose of the "Accepted" state is to absorb retransmissions of an accepted INVITE request. Any such retransmissions are absorbed entirely within the server transaction. They are not passed up to the TU since any downstream UAS cores that accepted the request have taken responsibility for reliability and will already retransmit their 2xx responses if necessary.

While in the "Accepted" state, if the TU passes a 2xx response, the server transaction MUST pass the response to the transport layer for transmission.

When the INVITE server transaction enters the "Accepted" state, Timer L MUST be set to fire in 64*T1 for all transports. This value matches both Timer B in the next upstream client state machine (the amount of time the previous hop will wait for a response when no provisionals have been sent) and the amount of time this (or any downstream) UAS core might be retransmitting the 2xx while waiting for an ACK. If Timer L fires while the INVITE server transaction is in the "Accepted" state, the transaction MUST transition to the "Terminated" state.

Once the transaction is in the "Terminated" state, it MUST be destroyed immediately.

Section 18.1.2 - Replace the second paragraph with

The client transport uses the matching procedures of Section 17.1.3 to attempt to match the response to an existing transaction. If there is a match, the response MUST be passed to that transaction. Otherwise, the response MUST be silently discarded.

Section 18.2.1 - Replace the last paragraph with

Next, the server transport attempts to match the request to a server transaction. It does so using the matching rules described in Section 17.2.3. If a matching server transaction is found, the request is passed to that transaction for processing. If no match is found, the request is passed to the core, which may decide to
construct a new server transaction for that request.

8.10. Page 265

Add to Table 4:

| Timer L | 64*Tl | Section 17.2.1 | Wait time for accepted INVITE request retransmits |

9. IANA Considerations

None.

10. Security Considerations

This document makes two changes to the Session Initiation Protocol to address the error discussed in Section 3. It changes the behavior of both the client and server INVITE transaction state machines, and it changes the way "stray" responses (those that don’t match any existing transaction) are handled.

The changes to the state machines cause elements to hold onto each INVITE transaction state longer (32 seconds) than what was specified in RFC 3261. This will have a direct impact on the amount of work an attacker leveraging state exhaustion will have to exert against the system. However, this additional state is necessary to achieve correct operation.

RFC 3261 required SIP proxies to forward any stray 200 class responses to an INVITE request upstream statelessly. As a result, conformant proxies can be forced to forward packets (that look sufficiently like SIP responses) to destinations of the sender’s choosing. Section 3 discusses some of the malicious behavior this enables. This document reverses the stateless forwarding requirement, making it a violation of the specification to forward stray responses.

11. Acknowledgments

Pekka Pessi reported the improper handling of INVITE retransmissions.

12. References
12.1. Normative References


12.2. Informative References


Author’s Address

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

Email: RjS@nostrum.com