Response Code for Indication of Terminated Dialog
draft-ietf-sip-199-08.txt

Status of this Memo

   This Internet-Draft is submitted to IETF in full conformance with the
   provisions of BCP 78 and 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 October 11, 2009.

Copyright Notice

Abstract

   This specification defines a new SIP response code, 199 Early Dialog
   Terminated, which a SIP forking proxy and a UAS can use to indicate
   upstream towards the UAC that an early dialog has been terminated,
   before a final response is sent towards the UAC.

Table of Contents

1. Introduction

As defined in SIP (Session Initiation Protocol) specification
[RFC3261], an early SIP dialog is created when a non-100 provisional
response is sent to the dialog initiation request (e.g. INVITE).
The dialog is considered to be in early state until a final response
is sent.

When a proxy receives an initial request (outside an existing dialog,
and without a pre-defined route set), it can forward it towards
multiple remote destinations. When the proxy does that, it performs
forking.

When a forking proxy receives non-100 provisional responses, it
forwards the responses upstream towards the sender of the associated
request. When a forking proxy receives a 2xx final response, it
forwards the response upstream towards the sender of the associated
request. At that point the proxy normally sends a CANCEL request
downstream towards all remote destinations where it previously sent
the request associated with the 2xx final response, and from which it
has yet not received a final response, in order to terminate
associated outstanding early dialogs. It is possible to receive
multiple 2xx final responses. When SIP entities upstream receive the
first 2xx final response, and they do not to intend to accept
subsequent 2xx final responses, they will automatically terminate
other associated outstanding early dialogs. If additional 2xx final
responses are received, those SIP entities will normally send a BYE
request using the dialog identifier retrieved from the subsequent 2xx
final response.

NOTE: A UAC can use the Request-Disposition header [RFC3841] to
request that proxies do not send CANCEL requests downstream once they
have received the first final 2xx response.

When a forking proxy receives a non-2xx final response, it does not
always immediately forward the response upstream towards the sender
of the associated request. Instead, the forking proxy "stores" it
and waits for further final responses from remote destinations where
the forked request was forwarded. At some point the proxy uses a
specified mechanism to determine the "best" final response code, and
forwards that final response upstream towards the sender of the
associated request. When SIP entities upstream receive the non-2xx
final response they will release resources associated with the
session, and the UAC will terminate the session setup.

Since the forking proxy does not always immediately forward non-2xx
final responses, SIP entities upstream (including the UAC that
initiated the request) do not know that a specific early dialog has

been terminated, and the SIP entities keep possible resources associated with the early dialog until they receive a final response from the forking proxy.

This specification defines a new SIP response code, 199 Early Dialog Terminated, which a forking proxy and a UAS can use to indicate upstream that an early dialog has been terminated. The 199 response can also be sent by a UAS, prior to sending a non-2xx final response. SIP entities that receive the 199 provisional response MAY release resources associated with the specific early dialog. The SIP entities MAY also use the 199 provisional response to make policy related decisions related to early dialogs.

The 199 response code is an optimization, which allows the UAC to be informed about terminated early dialogs. However, since the support of the 199 response is optional, a UAC cannot assume that it will always receive a 199 provisional response for all terminated early dialogs.

2. **Conventions**

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 [RFC2119].

3. **Requirements**

REQ 1: It must be possible to indicate to the UAC that an early dialog has been terminated before a final response is sent.

4. **User Agent Client behavior**

When a UAC sends an initial request, and if it wants to receive 199 responses, it MUST insert the 199 option-tag in the Supported header, which indicates that the client supports the 199 Early Dialog Terminated response code. The UAC SHOULD NOT insert the 199 option-tag in the Require header, unless the particular session usage requires the UAS to support the response code. Also, the UAC SHOULD NOT insert the 199 option-tag in the Proxy-Require header, unless the particular session usage requires every proxy on the path to support the response code. Using Require or Proxy-Require with the 199 option-tag will in many cases result in unnecessary session establishment failures.

When a UAC receives a 199 response it MAY release resources and
procedures associated with the early dialog on which the 199 response
is received. Examples of resources and procedures are e.g.
procedures for the establishment of media plane resources (bandwidth,
radio, codecs etc), media security procedures or procedures related
to NAT traversal. In addition, the UAC may use the 199 response for
policy decisions related to early dialogs, e.g. when choosing to
process media associated with a particular early dialog.

If multiple usages [RFC5057] are used within an early dialog, and it
is not clear which dialog usage the 199 response terminates, SIP
entities that keep dialog state shall not release resources
associated with the early dialog when they receive the 199 response.

If a client receives an unreliable 199 response on a dialog which has
not previously been created (this can happen if a 199 response
reaches the client before a 18x response) the client SHALL discard
the 199 responses. If a client receives a reliable 199 response on a
dialog which has not previously been created the UAC SHOULD
acknowledge the 199 response, as described in [RFC3262].

4.1. Examples of resource types

Examples which benefit from resource-release are:

1. Codec release - when resources for a specific codec has been
reserved only for the stream that is terminated. In that case the
resources associated with that codec can be released.

2. Pre-conditions - when the dialog is terminated, procedures and
resources associated to the pre-conditions for that dialog can be
released.

3. In-band security negotiation - when the dialog is terminated,
procedures and resources associated with the in-band security
negotiation for that dialog can be released.

4. ICE [I-D.ietf-mmusic-ice] mechanism - when the dialog is
terminated, procedures and resources associated with the ICE related
in-band procedures for that dialog can be released.

5. Limited access resources - in case of forking and multiple stream
it may not be possible to allow early media on all dialogs, so media
sessions associated with some dialogs may e.g. be set to "inactive".
When a dialog is terminated, media sessions associated with other
dialogs can be allowed.

6. Secure media selection - when SRTP is used to encrypt the media.

In some cases SIP entities are only able to render media associated
with a single early dialog. If a 199 response is received on a
dialog, and media associated with that media has been rendered, the
SIP entity can start rendering media associated with another early
dialog.

If the client is able to associate the 199 response with a specific
media stream, it MAY choose to discard media on that specific media
stream, it MAY release all resources associated with that media
stream and it MAY start to process media streams received on other
early dialogs. When the P-Early-Media header [RFC5009] is used, a UA
MAY trigger different actions depending on whether the header has
been used for the terminated dialog. How the association between the
dialog and the associated media stream is done is outside the scope
of this document.

NOTE: When using SRTP [RFC3711], the secure media stream is bound to
the crypto context setup for the dialog, and can be identified using
the MKI (if used) of SRTP.

If the client only has a single early dialog (other early dialogs MAY
not have been established, or they MAY have been established and
later terminated) and a 199 response is received for that early
dialog, the client terminates the early dialog. Afterwards, the
client SHOULD act as before the first early dialog was established.

4.2. Examples of policy procedures

1. UAC early media selection - when media associated with multiple
early dialogs is received, SIP endpoint normally chooses to process
media associated with a single early dialog (e.g. the recently
established early dialog). If a 199 response is received on such
early dialog, the SIP endpoint can start processing media associated
with another early dialog. For example, an early dialog may be used
for an announcement message, and when the message is finished a 199
response will be sent on that dialog, in order for the SIP endpoint
to stop processing media associated with that early dialog. This
kind of policy is normal especially in PSTN gateways, where the
calling user cannot control which media is processed.

2. SBC early media selection - when an SBC is used to control which
media is processed and forwarded. In many cases, the SBC only
processes media associated with a single early dialog. Typical for
NAT traversal, the SBC often "latches" onto a media stream. If a 199
response is received, the SBC can choose to start processing media
associated with another dialog. If the SBC performs latching, it can
trigger a "re-latch" onto a new media stream when the 199 response is
received.

3. UAC ringing tone generation - when a UAC receives a 180 response,
    it may choose to generate a local ringing tone. If early media is
    received, the UAC may stop the local ringing tone generation and play
    the incoming early media packets. If a 199 response is received on
    the early dialog associated with the early media, and the UAC has
    previously received a 180 response for another early dialog, it can
    start to generate local ringing tone again. Having knowledge that
    the early dialog associated with early media has been terminated, the
    UAC can also start generating local ringing tone if a 180 is received
    on another early dialog after the early dialog has been terminated."

5. User Agent Server behavior

If the received initial request contains an 199 option tag, the UAS
SHOULD NOT send a 199 response on a dialog on which it intends to
send a final response, unless it e.g. has been configured to do so
due to lack of 199 support by forking proxies or other intermediate
SIP entities.

NOTE: If the UAS has created multiple early dialogs (the UAS is
acting similar to a forking proxy), it does not always intend to send
a final response for all of those dialogs.

When a UAS generates a 199 response, the response MUST contain a To
header tag parameter, which identifies the early dialog that has been
terminated. The UAS MUST also insert a Reason header [RFC3326] which
contains a response code which describes the reason why the dialog
was terminated.

If the UAS intends to send 199 responses, and if it supports the
procedures defined in [RFC3840], it MAY during the registration
procedure use the sip.extensions feature tag [RFC3840] to indicate
support of the 199 response code.

A 199 response SHOULD NOT contain an SDP offer/answer message body,
unless required by the rules in [RFC3264].

NOTE: If the INVITE request did not contain an SDP offer, and the 199
response is the first reliably sent response, the 199 response is
required to contain an SDP offer.

When a 199 response is sent by a UAS, since the provisional response
is only used for information purpose, the UAS SHOULD send it
unreliably even if the 100rel option tag [RFC3262] is present in the
Require header of the associated request.

6. Proxy behavior

   When a proxy receives a 199 response on an early dialog, it MUST
   process the response as any other non-100 provisional responses. The
   proxy will forward the response upstream towards the sender of the
   associated request. The proxy MAY release resources it has reserved
   associated with the early dialog on which the response is received.
   If a proxy receives a 199 response out of dialog, it processes it as
   other non-100 provisional responses received out of dialog.

   When a forking proxy receives a non-2xx final response that it
   recognizes as terminating one or more early dialogs, it SHOULD
   generate and send a 199 response upstream for each of the terminated
   early dialogs that satisfy each of the following conditions:

   - the forking proxy does not intend to forward the final response
   immediately (in accordance with rules for a forking proxy)

   - the UAC has indicated support (using the 199 option tag) for the
   199 response code

   - the forking proxy has not already received and forwarded a 199
   response for that early dialog

   - the forking proxy has not already sent a final response for any of
   the early dialogs

   When the forking proxy forks the initial request, it generates a
   unique Via header branch parameter value for each forked leg. The
   proxy can determine whether additional forking has occurred
   downstream of the proxy by storing the top Via branch value from each
   response which creates an early dialog. If the same top Via branch
   value is received for multiple early dialogs, the proxy knows that
   additional forking has occured downstream of the proxy. A non-2xx
   final response received for a specific early dialog also terminates
   all other early dialog for which the same top Via branch value was
   received in the responses which created those early dialogs.

   Based on implementation policy, the forking proxy MAY wait before
   sending the 199 response, e.g. if it expects to receive a 2xx final
   response on another dialog shortly after it received the non-2xx
   final response which triggered the 199 response.

   When a forking proxy generates a 199 response, the response MUST
   contain a To header tag parameter, which identifies the early dialog
   that has been terminated. The proxy MUST also insert a Reason header
   [RFC3326] which contains the response code of the response that
   triggered the 199 response. If the proxy has stored the Contact and

Record-Route header fields received in the responses that created
early dialogs, it SHALL insert those header fields in the 199
responses.

A forking proxy which supports generating of 199 responses MUST keep
track of early dialogs, in order to determine whether to generate a
199 response when the proxy receives a non-2xx final response. In
addition, the proxy MUST keep track on which early dialogs it has
received and forwarded 199 responses, in order to not generate
additional 199 responses for those early dialogs.

If a forking proxy receives a reliably sent 199 response for a
dialog, for which it has previously generated and sent a 199
response, it MUST forward the 199 response. In case of a unreliably
sent 199 response, the proxy MAY forward the 199 response, or it MAY
discard it.

When a forking proxy generates a 199 response, the response MUST NOT
be sent reliably.

[7](#). **Backward compatibility**

Since all SIP entities involved in a session setup do not necessarily
support the specific meaning of the 199 Early Dialog Terminated
provisional response, the sender of the response MUST be prepared to
receive SIP requests and responses associated with the dialog for
which the 199 response was sent (a proxy can receive SIP messages
from either direction). If such request is received by a UA, it MUST
act in the same way as if it had received the request after sending
the final non-2xx response to the INVITE, as specified in [[RFC3261](#)].
A UAC that receives a 199 response for an early dialog MUST NOT send
any further requests on that dialog, except for requests which
acknowledge reliable responses. A proxy MUST forward requests
according to [[RFC3261](#)], even if the proxy has knowledge that the
early dialog has been terminated.

The 199 Early Dialog Terminated response code does not "replace" a
final response. [RFC 3261](#) [[RFC3261](#)] specifies when a final response
is sent.

[8](#). **199 Early Dialog Terminated**

The 199 Early Dialog Terminated response code allows a SIP entity to
indicate upstream that a specific dialog has been terminated, before
a final response is sent by the entity. The To header tag value is
used to identify the dialog.

9. Usage with SDP offer/answer

A 199 Early Dialog Terminated provisional response SHOULD NOT contain
an SDP offer/answer message body, unless required by the rules in
[RFC3264].

NOTE: If the INVITE request did not contain an SDP offer, and the 199
response is the first reliably sent response, the 199 response is
required to contain an SDP offer.

10. Usage with 100rel

When a 199 Early Dialog Terminated provisional response is sent by a
UAS, since the provisional response is only used for information
purpose, the UAS SHOULD send it unreliably even if the 100rel option
tag [RFC3262] is present in the Require header of the associated
request.

When a forking proxy generates a 199 response, the response MUST NOT
be sent reliably.

11. Examples

11.1. Example with a forking proxy which generates 199

The figure shows an example, where a proxy (P1) forks an INVITE
received from UAC. The forked INVITE reaches UAS_2, UAS_3 and UAS_4,
which send 18x provisional responses in order to create early dialogs
between themselves and the UAC. UAS_2 and UAS_3 reject the INVITE by
sending a 4xx error response each. When P1 receives the 4xx
responses it immediately sends 199 responses, associated with the
dialogs where the 4xx responses were received, towards the UAC. The
early dialog leg is shown in parenthesis.

```
    UAC              P1                   UAS_2    UAS_3    UAS_4
     |                |                     |        |        |
     |-- INVITE -->|                        |        |        |
     |                |--- INVITE (2) ->|        |        |
     |                |--- INVITE (3) --------->|        |
     |                |--- INVITE (4) ----------------->|
     |                |<-- 18x (2) -----|        |        |
     |<- 18x (2) --|                        |        |        |
     |                |<-- 18x (3) ------------|        |
     |<- 18x (3) --|                        |        |        |
     |                |<-- 18x (4) --------------------|
     |<- 18x (4) --|                        |        |        |
     |                |<-- 4xx (2) -----|        |        |
     |                |--- ACK (2) ---->|        |        |
     |<- 199 (2) --|                        |        |        |
     |                |<-- 4xx (3) ------------|        |
     |                |--- ACK (3) ----------->|        |
     |<- 199 (3) --|                        |        |        |
     |                |<-- 200 (4) --------------------|
     |<- 200 (4) --|                        |        |        |
     |-- ACK (4) ->|                        |        |        |
     |                |--- ACK (4) -------------------->|
     |                |                     |        |        |
```
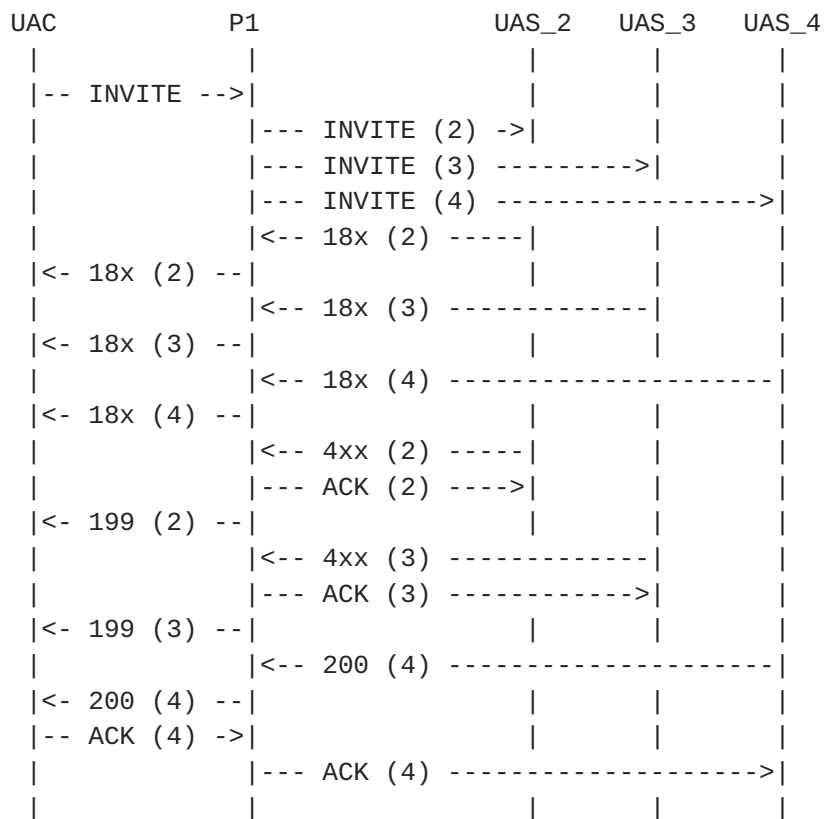
                      Figure 1: Example call flow

11.2. **Example with a forking proxy which receives 200 OK**

   The figure shows an example, where a proxy (P1) forks an INVITE
   received from UAC. The forked INVITE reaches UAS_2, UAS_3 and UAS_4,
   which send 18x provisional responses in order to create early dialogs
   between themselves and the UAC. UAS_4 accepts the session by sending
   a 200 OK final response. When P1 receives the 200 OK responses it
   immediately forwards it towards the UAC. P1 does not send 199
   responses for the early dialogs from UAS_2 and UAS_3, since P1 has
   yet not received any final responses on those early dialogs (even if
   P1 sends CANCEL request to UAS_2 and UAS_3 P1 may still receive 200
   OK final response from UAS_2 or UAS_3, which P1 would have to forward
   towards the UAC. The early dialog leg is shown in parenthesis.

```
    UAC               P1                UAS_2   UAS_3   UAS_4
     |                 |                  |       |       |
     |-- INVITE -->|                      |       |       |
     |                 |--- INVITE (2) ->|        |       |
     |                 |--- INVITE (3) --------->|         |
     |                 |--- INVITE (4) ----------------->|
     |                 |<-- 18x (2) -----|        |       |
     |<- 18x (2) --|                      |       |       |
     |                 |<-- 18x (3) ------------|          |
     |<- 18x (3) --|                      |       |       |
     |                 |<-- 18x (4) --------------------|
     |<- 18x (4) --|                      |       |       |
     |                 |<-- 200 (4) --------------------|
     |<- 200 (4) --|                      |       |       |
     |-- ACK (4) ->|                      |       |       |
     |                 |--- ACK (4) -------------------->|
     |                 |                  |       |       |
```
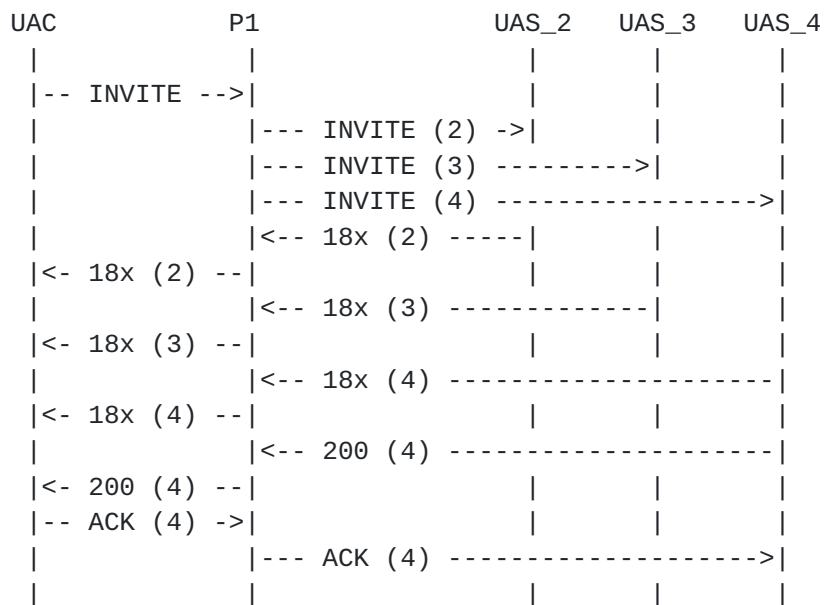
Figure 2: Example call flow

11.3. Example with two forking proxies, of which one generates 199

   The figure shows an example, where a proxy (P1) forks an INVITE
   received from UAC. One of the forked INVITEs reaches UAS_2. The
   other forked INVITE reaches another proxy (P2), which forks the
   INVITE to UAS_3 and UAS_4, which send 18x provisional responses in
   order to create early dialogs between themselves and the UAC. UAS_3
   and UAS_4 reject the INVITE by sending a 4xx error response each. P2
   does not support the 199 response code, and forwards a single 4xx
   response. When P1 receives the 4xx responses from P2, it manages to
   associate the response with the early dialogs from both UAS_3 and
   UAS_4, so it generates and sends two 199 response to indicate that
   the early dialogs from UAS_3 and UAS_4 have been terminated. The
   early dialog leg is shown in parenthesis.

```
   UAC           P1              P2              UAS_2   UAS_3   UAS_4
    |             |               |               |       |       |
    |-- INVITE -->|               |               |       |       |
    |             |-- INVITE (2) ----------------->|       |       |
    |             |-- INVITE ---->|               |       |       |
    |             |               |--- INVITE (3) --------->|       |
    |             |               |--- INVITE (4) ----------------->|
    |             |               |<-- 18x (3) -------------|       |
    |             |<- 18x (3) ----|               |       |       |
    |<- 18x (3) --|               |               |       |       |
    |             |               |<-- 18x (4) --------------------|
    |             |<- 18x (4) ----|               |       |       |
    |<- 18x (4) --|               |               |       |       |
    |             |               |<-- 4xx (3) -------------|       |
    |             |               |--- ACK (3) ------------>|       |
    |             |               |<-- 4xx (4) --------------------|
    |             |               |--- ACK (4) -------------------->|
    |             |<- 4xx (3) ----|               |       |       |
    |             |-- ACK (3) --->|               |       |       |
    |<- 199 (3) --|               |               |       |       |
    |<- 199 (4) --|               |               |       |       |
    |             |<- 200 (2) --------------------|       |       |
    |<- 200 (2) --|               |               |       |       |
    |-- ACK (2) ->|               |               |       |       |
    |             |-- ACK (2) -------------------->|       |       |
    |             |               |               |       |       |
```
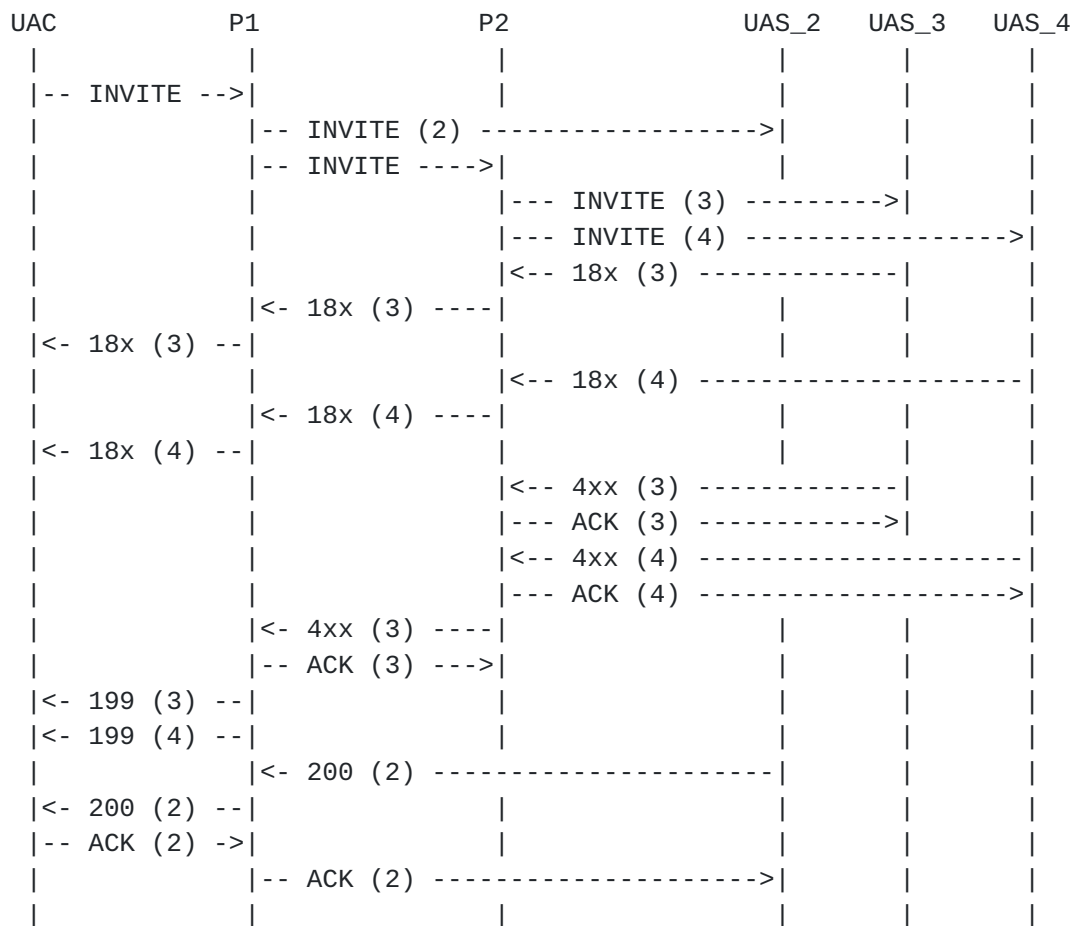
                    Figure 3: Example call flow

12. Security Considerations

   General security issues related to SIP responses are described in
   [RFC3261]. Due to the nature of the 199 response, it may be
   attractive to use it for launching attacks in order to terminate
   specific early dialogs. SIP security mechanisms (e.g. hop-to-hop
   TLS) can be used to minimize, or eliminate, the risk for such
   attacks.

13. IANA Considerations

   This section registers a new SIP response code and a new option tag,
   according to the procedures of RFC 3261.

13.1. IANA Registration of the 199 response code

This section registers a new SIP response code, 199. The required
information for this registration, as specified in RFC 3261, is:

  RFC Number: RFC XXXX [[NOTE TO IANA: Please replace XXXX with the RFC
number of this specification]]

  Response Code Number: 199

  Default Reason Phrase: Early Dialog Terminated

13.2. IANA Registration of the 199 Option Tag

This section registers a new SIP option tag, 199. The required
information for this registration, as specified in RFC 3261, is:

 Name: 199

 Description: This option tag is for indicating support of the 199
      Early Dialog Terminated provisional response code. When present
      in a Supported header, it indicates that the UA supports the
      response code. When present in a Require header in a request,
      it indicates that the UAS MUST support the sending of the
      response code.

14. Acknowledgements

Thanks to Paul Kyzivat, Dale Worley, Gilad Shaham, Francois Audet,
Attila Sipos, Robert Sparks, Brett Tate, Ian Elz, Hadriel Kaplan,
Timothy Dwight, Dean Willis, Serhad Doken, John Elwell, Gonzalo
Camarillo, Adam Roach, Bob Penfield,Tom Taylor, Ya Ching Tan, Keith
Drage and Hans Erik van Elburg for their feedback and suggestions.

15. References

15.1. Normative References

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

[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.

[RFC3262] Rosenberg, J. and H. Schulzrinne, "Reliability of

                    Provisional Responses in Session Initiation Protocol
                    (SIP)", RFC 3262, June 2002.

   [RFC3264]   Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model
               with Session Description Protocol (SDP)", RFC 3264,
               June 2002.

   [RFC3326]   Schulzrinne, H., Oran, D., and G. Camarillo, "The Reason
               Header Field for the Session Initiation Protocol (SIP)",
               RFC 3326, December 2002.

   [RFC3711]   Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K.
               Norrman, "The Secure Real-time Transport Protocol (SRTP)",
               RFC 3711, March 2004.

   [RFC3840]   Rosenberg, J., Schulzrinne, H., and P. Kyzivat,
               "Indicating User Agent Capabilities in the Session
               Initiation Protocol (SIP)", RFC 3840, August 2004.

   [RFC3841]   Rosenberg, J., Schulzrinne, H., and P. Kyzivat, "Caller
               Preferences for the Session Initiation Protocol (SIP)",
               RFC 3841, August 2004.

   [RFC5009]   Ejza, R., "Private Header (P-Header) Extension to the
               Session Initiation Protocol (SIP) for Authorization of
               Early Media", RFC 5009, September 2007.

   [I-D.ietf-mmusic-ice]
               Rosenberg, J., "Interactive Connectivity Establishment
               (ICE): A Protocol for Network Address Translator (NAT)
               Traversal for Offer/Answer Protocols",
               draft-ietf-mmusic-ice-19 (work in progress), October 2007.

15.2. Informational References

   [RFC5057]   Sparks, R., "Multiple Dialog Usages in the Session
               Initiation Protocol", RFC 5057, November 2007.

Author's Address

    Christer Holmberg
    Ericsson
    Hirsalantie 11
    Jorvas 02420
    Finland

    Email: christer.holmberg@ericsson.com