       Re-INVITE and Target-refresh Request Handling in the Session Initiation
                              Protocol (SIP)
                  draft-camarillo-sipcore-reinvite-01.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 April 29, 2010.

Copyright Notice

Abstract

   In this document, we clarify the handling of re-INVITEs in SIP. We
   clarify in which situations a UAS (User Agent Server) should generate
   a success response and in which situations a UAS should generate an
   error response to a re-INVITE. Additionally, we clarify issues
   related to target-refresh requests.

Table of Contents

1. Introduction

As discussed in Section 14 of RFC 3261 [RFC3261], an INVITE request
sent within an existing dialog is known as a re-INVITE. A re-INVITE
is used to modify session parameters, dialog parameters, or both.
That is, a single re-INVITE can change both the parameters of its
associated session (e.g., changing the IP address where a media
stream is received) and the parameters of its associated dialog
(e.g., changing the remote target of the dialog). A re-INVITE can
change the remote target of a dialog because it is a target refresh
request, as defined in Section 6 of RFC 3261 [RFC3261].

A re-INVITE transaction has an offer/answer [RFC3264] exchange
associated to it. The UAC (User Agent Client) generating a given re-
INVITE can act as the offerer or as the answerer. A UAC willing to
act the offerer includes an offer in the re-INVITE. The UAS then
provides an answer in a response to the re-INVITE. A UAC willing to
act as answerer does not include an offer in the re-INVITE. The UAS
then provides an offer in a response to the re-INVITE becoming, thus,
the offerer.

Certain transactions within a re-INVITE (e.g., UPDATE [RFC3311]
transactions) can also have offer/answer exchanges associated to
them. A UA (User Agent) can act as the offerer or the answerer in
any of these transactions regardless of whether the UA was the
offerer or the answerer in the umbrella re-INVITE transaction.

There has been some confusion among implentors regarding how a UAS
(User Agent Server) should handle re-INVITEs. In particular,
implementors requested clarifications on which type of response a UAS
should generate in different situations. In this document, we
clarify these issues.

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 RFC 2119 [RFC2119].

UA: User Agent.

UAC: User Agent Client.

UAS: User Agent Server.

3. **Background on Re-INVITE Handling by UASs**

   A UAS receiving a re-INVITE will need to, eventually, generate a
   response to it. Some re-INVITEs can be responded to immediately
   because their handling does not require user interaction (e.g.,
   changing the IP address where a media stream is received). The
   handling of other re-INVITEs requires user interaction (e.g., adding
   a video stream to an audio-only session). Therefore, these re-
   INVITEs cannot be responded to immediately.

   An error response to a re-INVITE has the following semantics. As
   specified in Section 12.2.2 of RFC 3261 [RFC3261], if a re-INVITE is
   rejected, no state changes are performed. These state changes
   include state changes associated to the re-INVITE transaction and all
   other transactions within the re-INVITE (target refreshes, which are
   discussed in Section 10, are an exception to this rule because in
   certain cases they are performed even if the re-INVITE is rejected).
   That is, the session and dialog states are the same as before the re-
   INVITE was received. The example in Figure 1 illustrates this point.


```
              UAC                                          UAS

                |                                           |
                |-------------(1) INVITE SDP1--------------->|
                |                                           |
                |<------------(2) 200 OK SDP2---------------|
                |                                           |
                |-----------------(3) ACK------------------>|
                |                                           |
                |                                           |
                |-------------(4) INVITE SDP3-------------->|
                |                                           |
                |<---------------(5) 4xx-------------------|
                |                                           |
                |-----------------(6) ACK------------------>|
                |                                           |
```
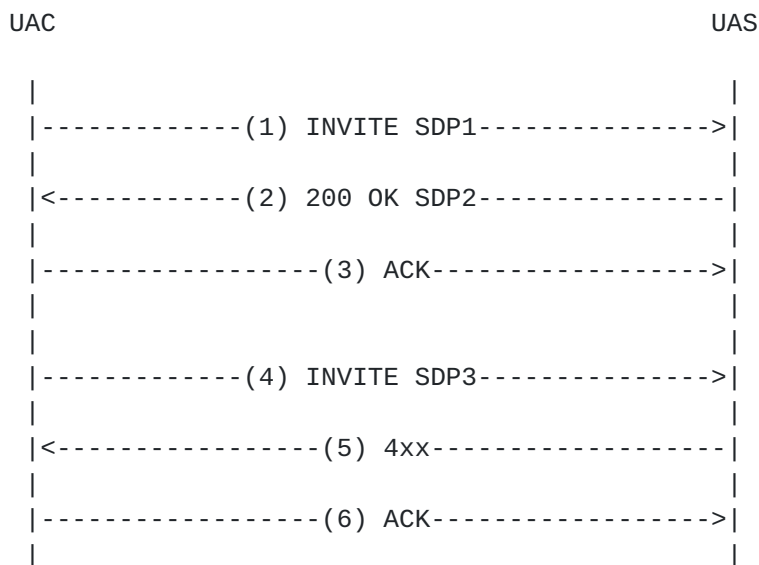

             Figure 1: Rejection of a re-INVITE

   The UAs perform an offer/answer exchange to establish an audio-only
   session:

```
SDP1:
    m=audio 30000 RTP/AVP 0

SDP2:
    m=audio 31000 RTP/AVP 0
```

At a later point, the UAC sends a re-INVITE (4) in order to add a
video stream to the session.

```
SDP3:
    m=audio 30000 RTP/AVP 0
    m=video 30002 RTP/AVP 31
```

The UAS is automatically configured to reject video streams.
Consequently, the UAS returns an error response (5). At that point,
the session parameters in use are still those resulting from the
initial offer/answer exchange, which are described by SDP1 and SDP2.
That is, the session and dialog states are the same as before the re-
INVITE was received.

In the previous example, the UAS rejected all the changes requested
in the re-INVITE by returning an error response. However, there are
situations where a UAS wants to accept some but not all the changes
requested in a re-INVITE. In these cases, the UAS generates a 200
(OK) response with an SDP indicating which changes were accepted and
which were not. The example in Figure 2 illustrates this point.

```
                  UAC                                    UAS

                   |                                      |
                   |-------------(1) INVITE SDP1--------------->|
                   |                                      |
                   |<------------(2) 200 OK SDP2----------------|
                   |                                      |
                   |-----------------(3) ACK------------------>|
                   |                                      |
                   |                                      |
                   |-------------(4) INVITE SDP3--------------->|
                   |                                      |
                   |<------------(5) 200 OK SDP4----------------|
                   |                                      |
                   |-----------------(6) ACK------------------>|
                   |                                      |
```
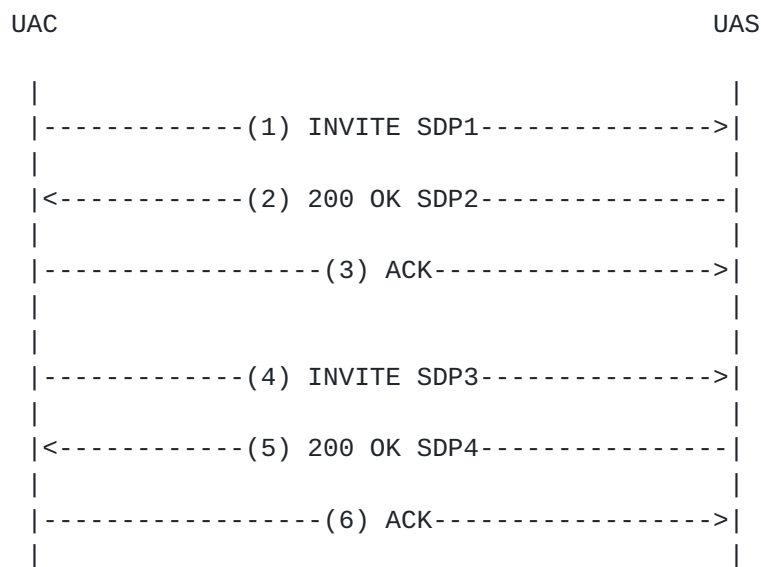
                Figure 2: Automatic rejection of a video stream

   The UAs perform an offer/answer exchange to establish an audio only
   session:

        SDP1:
            m=audio 30000 RTP/AVP 0
            c=IN IP4 192.0.2.1

        SDP2:
            m=audio 31000 RTP/AVP 0
            c=IN IP4 192.0.2.5

   At a later point, the UAC moves to an access that provides a higher-
   bandwidth. Therefore, the UAC sends a re-INVITE (4) in order to
   change the IP address where it receives the audio stream to its new
   IP address, and add a video stream to the session.

        SDP3:
            m=audio 30000 RTP/AVP 0
            c=IN IP4 192.0.2.2
            m=video 30002 RTP/AVP 31
            c=IN IP4 192.0.2.2

   The UAS is automatically configured to reject video streams.
   However, the UAS needs to accept the change of the audio stream's
   remote IP address. Consequently, the UAS returns a 200 (OK) response
   and sets the port of the video stream to zero in its SDP.

        SDP4:
            m=audio 31000 RTP/AVP 0
            c=IN IP4 192.0.2.5
            m=video 0 RTP/AVP 31
            c=IN IP4 192.0.2.2

   In the previous example, the UAS was configured to automatically
   reject the addition of video streams. The example in Figure 3
   assumes that the UAS requires its user's input in order to accept or
   reject the addition of a video stream and uses reliable provisional
   responses [RFC3262] (PRACK transactions are not shown for clarity).

```
                 UAC                                      UAS

                  |                                        |
                  |-------------(1) INVITE SDP1--------------->|
                  |                                        |
                  |<-----------(2) 200 OK SDP2---------------|
                  |                                        |
                  |----------------(3) ACK----------------->|
                  |                                        |
                  |                                        |
                  |------------(4) INVITE SDP3--------------->|
                  |                                        |
                  |<----(5) 183 Session Progress SDP4----------|
                  |                                        |
                  |                                        |
                  |<-----------(6) UPDATE SDP5---------------|
                  |                                        |
                  |------------(7) 200 OK SDP6-------------->|
                  |                                        |
                  |<--------------(8) 200 OK-----------------|
                  |                                        |
                  |----------------(9) ACK----------------->|
                  |                                        |
```
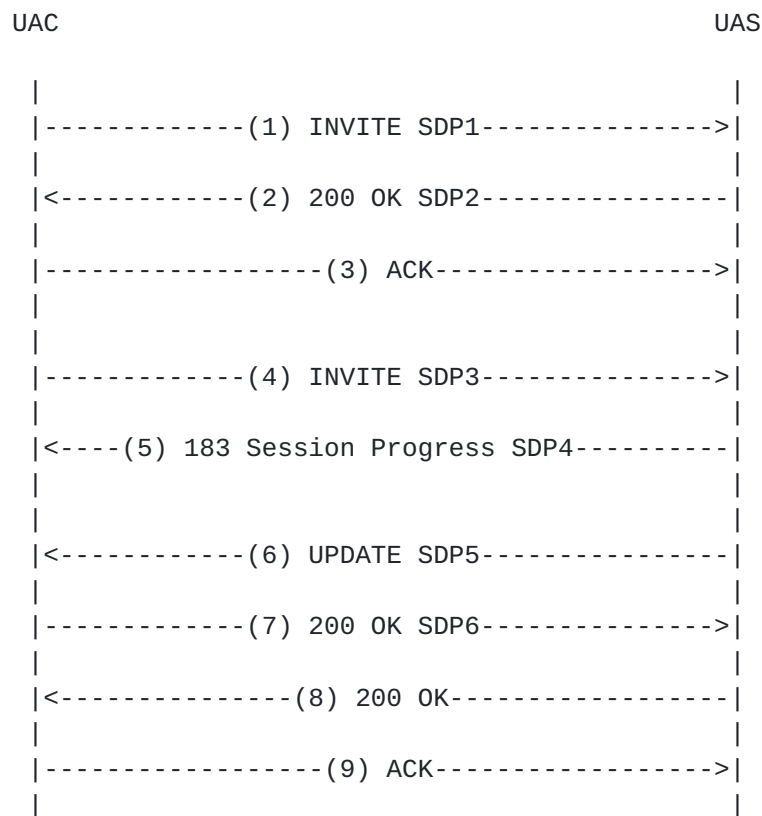
         Figure 3: Rejection of a video stream by the user

   Everything up to (4) is identical to the previous example. In (5),
   the UAS accepts the change of the audio stream's remote IP address
   but does not accept the video stream yet (it provides a null IP
   address instead of setting the stream to 'inactive' because inactive
   streams still need to exchange RTCP traffic).

```
        SDP4:
           m=audio 31000 RTP/AVP 0
           c=IN IP4 192.0.2.5
           m=video 31002 RTP/AVP 31
           c=IN IP4 0.0.0.0
```

   At a later point, the UAS's user rejects the addition of the video
   stream. Consequently, the UAS sends an UPDATE request setting the
   port of the video stream to zero in its SDP.

```
     SDP5:
         m=audio 31000 RTP/AVP 0
         c=IN IP4 192.0.2.5
         m=video 0 RTP/AVP 31
         c=IN IP4 0.0.0.0
```

The UAS now returns a 200 (OK) response to the re-INVITE.

In all the previous examples, the UAC was the offerer in the re-INVITE transaction. Examples with UACs acting as the answerers would be similar.

4. **Problems with Error Responses and Already-executed Changes**

Section 3 contains examples on how a UAS rejects all the changes requested in a re-INVITE without executing any of them by returning an error response (Figure 1), and how a UAS executes some of the changes requested in a re-INVITE and rejects some of them by returning a 2xx response (Figure 2 and Figure 3). A UAS can accept and reject different sets of changes simultaneously (Figure 2) or at different times (Figure 3).

The scenario that created confusion among implementors consists of a UAS that receives a re-INVITE, executes some of the changes requested in it, and then wants to reject all those already-executed changes and revert to the pre-re-INVITE state. Such a UAS may consider returning an error response to the re-INVITE (the message flow would be similar to the one in Figure 1), or using an UPDATE request to revert to the pre-re-INVITE state and then returning a 2xx response to the re-INVITE (the message flow would be similar to the one in Figure 3). This section explains the problems associated with returning an error response in these circumstances. In order to avoid these problems, the UAS should use the latter option (UPDATE request plus a 2xx response). Section 5 and Section 6 contain the normative statements needed to avoid these problems.

The reason for not using an error response to undo already executed changes is that an error response to a re-INVITE for which changes have already been executed is effectively requesting a change in the session or the dialog state. However, the UAC has no means to reject those changes if it is unable to execute them. That is, if the UAC is unable to revert to the pre-re-INVITE state, it will not be able to communicate this fact to the UAS.

Using an error response to undo already executed changes presents an additional problem. Section 4 of [RFC3264] specifies rules to avoid glare situations (i.e., to avoid offer/answer collisions in race

conditions). Even when both UAs generate an offer at the same time,
there are rules to determine which one should be processed first.
However, there are no rules to avoid a collision between an offer in
an UPDATE request and an error response for a re-INVITE. Since both
the UPDATE request and the error response would request changes, it
would not be clear which changes would need to be executed first.
This is yet another reason why UASs should not use error responses to
undo already-executed changes.

5. **UAS Behavior**

UASs should only return an error response to a re-INVITE if no
changes to the session or to the dialog state have been executed
since the re-INVITE was received. Such an error response indicates
that no changes have been executed as a result of the re-INVITE or
any other transaction within it.

If any of the changes requested in a re-INVITE or in any transaction
within it have already been executed (with the exception of target
refreshes), the UAS MUST always return a 2xx response.

A change to the session state is considered to have been executed
when the new media parameters are being used. Therefore, a change to
a stream subject to preconditions [RFC4032] is considered to have
been executed when the new media parameters start being used; not
when the preconditions for the stream are met. Connection
establishment messages (e.g., TCP SYN) and connectivity checks (e.g.,
when using ICE [I-D.ietf-mmusic-ice]) are not considered media
either. A UA considers the new parameters to be in use when it sends
media using them, or when media that uses the new parameters is
received, which should be interpreted as follows. From Section 8.3.1
of RFC 3264 [RFC3264]:

    "Received, in this case, means that the media is passed to a media
    sink. This means that if there is a playout buffer, the agent
    would continue to listen on the old port until the media on the
    new port reached the top of the playout buffer. At that time, it
    MAY cease listening for media on the old port."

TODO: RFC3264 assumes media streams that carry media continuously.
So, it considers that an UA should continue listening to the old port
(i.e., using the old parameters) until it sends media or receives
media on the new port. However, if two UASs perform an offer/answer
exchange on a stream that only carries media every now and then, the
UAs will need to be ready to receive media on both the old and the
new port for a long time. Shall we define some type of timeout for
this?

A UAS MUST NOT generate an error response to a re-INVITE if it has
generated a prior offer for which it has not yet received an answer
or a rejection.

6. UAC Behavior

A UAC that receives an error response to a re-INVITE that undoes
already-executed changes within the re-INVITE may be facing a legacy
UAS that does not support this specification (i.e., a UAS that does
not follow the guidelines in Section 5). There are certain race
condition situations that get both user agents out of
synchronization. In order to cope with these race condition
situations, a UAC that receives an error response to a re-INVITE for
which changes have been already executed SHOULD generate a new re-
INVITE or UPDATE request in order to make sure that both UAs have a
common view of the state of the session. The purpose of this new
offer/answer exchange is to synchronize both UAs, not to request
changes that the UAS may choose to reject. Therefore, the session
parameters in the offer/answer exchange SHOULD be as close as those
in the pre-re-INVITE state as possible.

7. Example of UAS Behavior

This section contains an example of a UAS that supports this
specification using an UPDATE request and a 2xx response to a re-
INVITE in order to revert to the pre-re-INVITE state. The example,
which is shown in Figure 4, assumes that the UAS requires its user's
input in order to accept or reject the addition of a video stream and
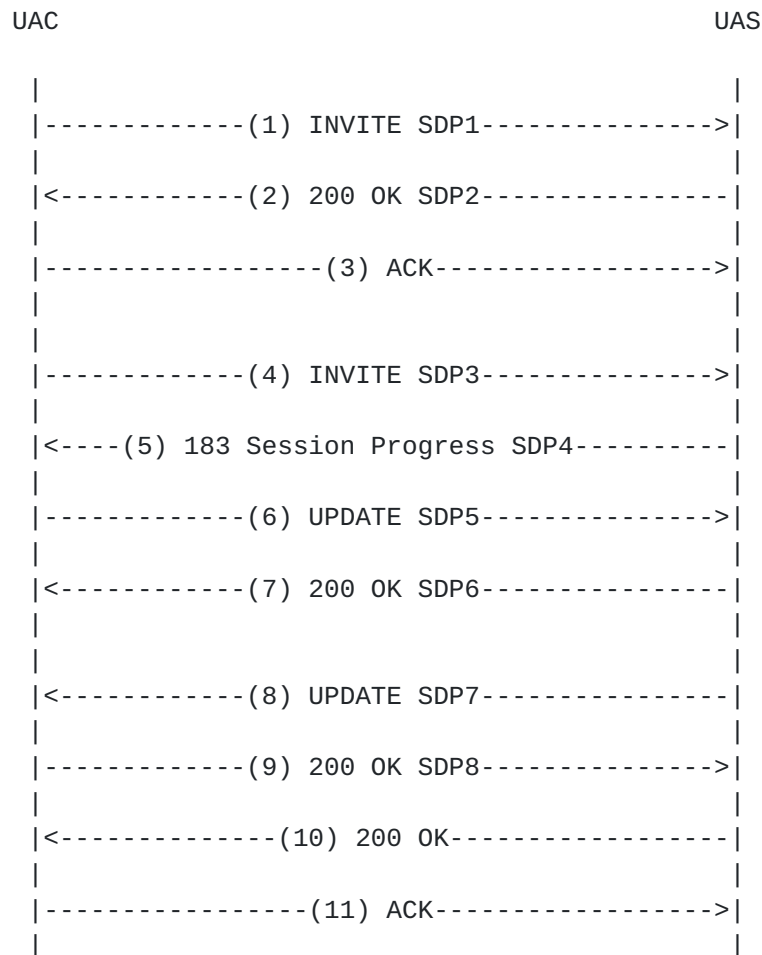uses reliable provisional responses [RFC3262] (PRACK transactions are
not shown for clarity).

```
                     UAC                                  UAS

                      |                                    |
                      |-------------(1) INVITE SDP1--------------->|
                      |                                    |
                      |<-----------(2) 200 OK SDP2---------------|
                      |                                    |
                      |----------------(3) ACK------------------>|
                      |                                    |
                      |                                    |
                      |------------(4) INVITE SDP3-------------->|
                      |                                    |
                      |<----(5) 183 Session Progress SDP4----------|
                      |                                    |
                      |-------------(6) UPDATE SDP5--------------->|
                      |                                    |
                      |<-----------(7) 200 OK SDP6---------------|
                      |                                    |
                      |                                    |
                      |<-----------(8) UPDATE SDP7---------------|
                      |                                    |
                      |------------(9) 200 OK SDP8-------------->|
                      |                                    |
                      |<-------------(10) 200 OK-----------------|
                      |                                    |
                      |----------------(11) ACK------------------>|
                      |                                    |
```

Figure 4: Rejection of a video stream by the user

The UAs perform an offer/answer exchange to establish an audio only
session:

```
        SDP1:
           m=audio 30000 RTP/AVP 0
           c=IN IP4 192.0.2.1

        SDP2:
           m=audio 31000 RTP/AVP 0
           c=IN IP4 192.0.2.5
```

At a later point, the UAC sends a re-INVITE (4) in order to add a new
codec to the audio stream and to add a video stream to the session.

```
     SDP3:
         m=audio 30000 RTP/AVP 0 3
         c=IN IP4 192.0.2.1
         m=video 30002 RTP/AVP 31
         c=IN IP4 192.0.2.1
```

In (5), the UAS accepts the addition of the audio codec but does not accept the video stream yet (it provides a null IP address instead of setting the stream to 'inactive' because inactive streams still need to exchange RTCP traffic).

```
     SDP4:
         m=audio 31000 RTP/AVP 0 3
         c=IN IP4 192.0.2.5
         m=video 31002 RTP/AVP 31
         c=IN IP4 0.0.0.0
```

At a later point, the UAC sends an UPDATE request (6) to remove the original audio codec from the audio stream (the UAC could have also used the PRACK to (5) to request this change).

```
     SDP5:
         m=audio 30000 RTP/AVP 3
         c=IN IP4 192.0.2.1
         m=video 30002 RTP/AVP 31
         c=IN IP4 192.0.2.1

     SDP6:
         m=audio 31000 RTP/AVP 3
         c=IN IP4 192.0.2.5
         m=video 31002 RTP/AVP 31
         c=IN IP4 0.0.0.0
```

Yet at a later point, the UAS's user rejects the addition of the video stream. Additionally, the UAS decides to revert to the original audio codec. Consequently, the UAS sends an UPDATE request (8) setting the port of the video stream to zero and offering the original audio codec in its SDP.

```
     SDP7:
         m=audio 31000 RTP/AVP 0
         c=IN IP4 192.0.2.5
         m=video 0 RTP/AVP 31
         c=IN IP4 0.0.0.0
```

   The UAC accepts the change in the audio codec in its 200 (OK)
   response (9) to the UPDATE request.

         SDP8:
             m=audio 30000 RTP/AVP 0
             c=IN IP4 192.0.2.1
             m=video 0 RTP/AVP 31
             c=IN IP4 192.0.2.1

   The UAS now returns a 200 (OK) response (10) to the re-INVITE. Note
   that the media state after this 200 (OK) response is the same as the
   pre-re-INVITE media state.

8. **Example of UAC Behavior**

   Figure 5 shows an example of a race condition situation in which the
   UAs end up with different views of the state of the session. The UAs
   in Figure 5 are involved in a session that, just before the message
   flows in the figures starts, includes a sendrecv audio stream and an
   inactive video stream. UA1 sends a re-INVITE (1) requesting to make
   the video stream sendrecv.

         SDP1:
             m=audio 20000 RTP/AVP 0
             a=sendrecv
             m=video 20002 RTP/AVP 31
             a=sendrecv

   UA2 is configured to automatically accept incoming video streams but
   to ask for user input before generating an outgoing video stream.
   Therefore, UAS2 makes the video stream sendonly by returning a 183
   (Session Progress) response (2).

         SDP2:
             m=audio 30000 RTP/AVP 0
             a=sendrecv
             m=video 30002 RTP/AVP 31
             a=sendonly

   When asked for input, UA2's user chooses not to have either incoming
   or outgoing video. In order to make the video stream inactive, UA2
   returns a 4xx error response (5) to the re-INVITE. The ACK request
   (6) for this error response is generated by the proxy between both
   user agents. Note that this error response undoes already-executed

changes. So, UA2 is a legacy UA that does not support this
specification.

The proxy relays the 4xx response (7) towards UA1. However, the 4xx
response (7) takes time to arrive to UA1 (e.g., the response may have
been sent over UDP and the first few retransmissions were lost). In
the meantime, UA2's user decides to put the audio stream on hold.
UA2 sends an UPDATE request (8) making the audio stream recvonly.
The video stream, which is inactive, is not modified and, thus,
continues being inactive.

```
    SDP3:
        m=audio 30000 RTP/AVP 0
        a=recvonly
        m=video 30002 RTP/AVP 31
        a=inactive
```

The proxy relays the UPDATE request (9) to UA1. The UPDATE request
(9) arrives at UA1 before the 4xx response (7) that had been
previously sent. UA2 accepts the changes in the UPDATE request and
returns a 200 (OK) response (10) to it .

SDP4: m=audio 20000 RTP/AVP 0 a=sendonly m=video 30002 RTP/AVP 31
a=inactive

At a later point, the 4xx response (7) finally arrives at UA1. This
response makes the session return to its pre-re-INVITE state.
Therefore, for UA1, the audio stream is sendrecv and the video stream
is inactive. However, for UA2, the audio stream is recvonly (the
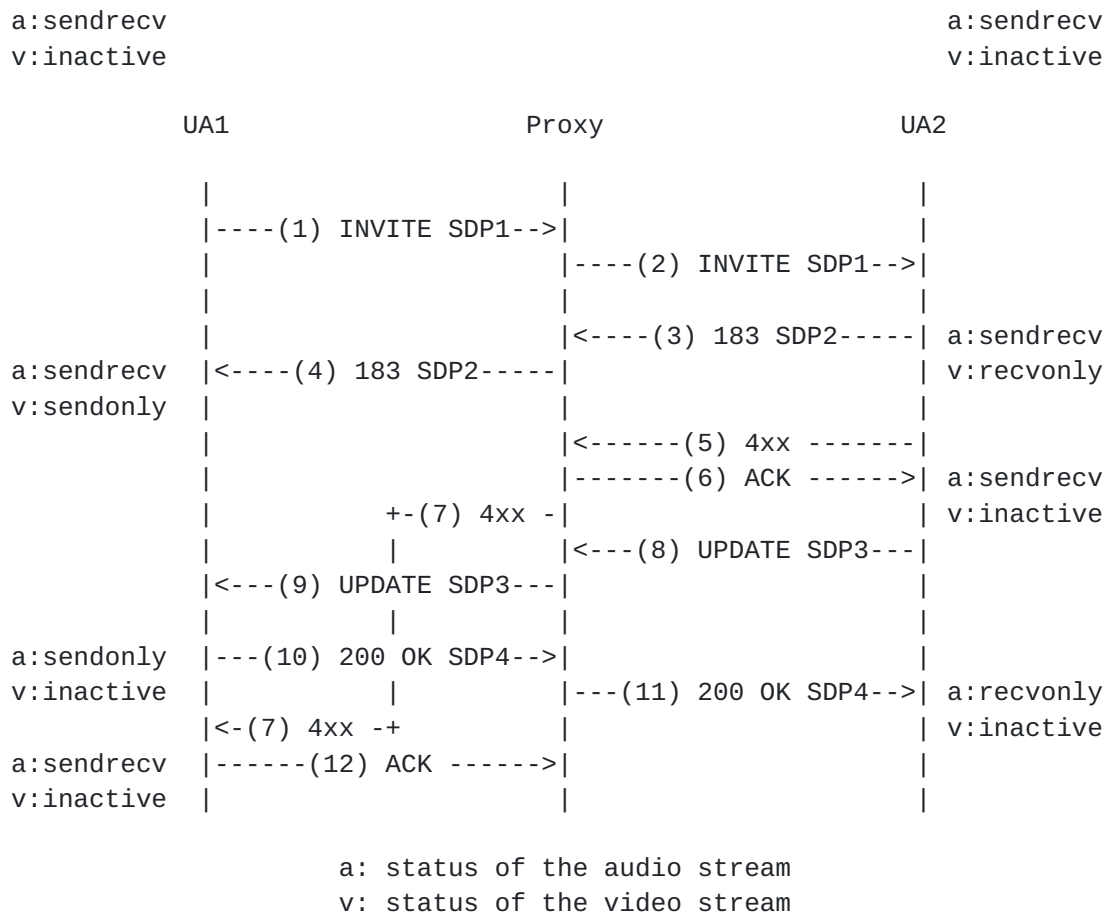video stream is also inactive).

```
  a:sendrecv                                             a:sendrecv
  v:inactive                                             v:inactive

              UA1                 Proxy                 UA2

               |                    |                    |
               |----(1) INVITE SDP1-->|                  |
               |                    |----(2) INVITE SDP1-->|
               |                    |                    |
               |                    |<----(3) 183 SDP2-----| a:sendrecv
  a:sendrecv   |<----(4) 183 SDP2-----|                  | v:recvonly
  v:sendonly   |                    |                    |
               |                    |<------(5) 4xx -------|
               |                    |-------(6) ACK ------>| a:sendrecv
               |          +-(7) 4xx -|                    | v:inactive
               |          |         |<---(8) UPDATE SDP3---|
               |<---(9) UPDATE SDP3---|                  |
               |          |         |                    |
  a:sendonly   |---(10) 200 OK SDP4-->|                  |
  v:inactive   |          |         |---(11) 200 OK SDP4-->| a:recvonly
               |<-(7) 4xx -+         |                    | v:inactive
  a:sendrecv   |------(12) ACK ------>|                  |
  v:inactive   |                    |                    |
```

```
                    a: status of the audio stream
                    v: status of the video stream
```

                      Figure 5: Message flow with race condition

   After the message flow in Figure 5, following the recommendations in
   this section, when UA1 received an error response (7) that undid
   already-executed changes, UA1 would generate an UPDATE request with
   an SDP reflecting the pre-re-INVITE state (i.e., sendrecv audio and
   inactive video). UA2 could then return a 200 (OK) response to the
   UPDATE request making the audio stream recvonly, which is the state
   UA2's user had requested. Such an UPDATE transaction would get the
   UAs back into synchronization.

9. Clarifications on Cancelling Re-INVITEs

   Section 9.2 of RFC 3261 [RFC3261] specifies the behavior of a UAS
   responding to a CANCEL request. Such a UAS responds to the INVITE
   request with a 487 (Request Terminated) at the 'should' level. Per
   the rules specified in Section 5, if the INVITE request was a re-
   INVITE and some of its requested changes had already been executed,
   the UAS would return a 2xx response instead.

10. **Background on Target-Refresh Requests**

   A target-refresh request is defined as follows in Section 6 of RFC
   3261 [RFC3261]:

      "A target-refresh request sent within a dialog is defined as a
      request that can modify the remote target of the dialog."

   Additionally, 2xx responses to target-refresh requests can also
   update the remote target of the dialog.. As discussed in Section
   12.2 of RFC 3261 [RFC3261], re-INVITEs are target-refresh requests.

   RFC 3261 [RFC3261] specifies the behavior of UASs receiving target-
   refresh requests and of UACs receiving a 2xx response for a target-
   refresh request.

   Section 12.2.2 of RFC 3261 [RFC3261] says:

      "When a UAS receives a target-refresh request, it MUST replace the
      dialog's remote target URI with the URI from the Contact header
      field in that request, if present."

   Section 12.2.1.2 of RFC 3261 [RFC3261] says:

      "When a UAC receives a 2xx response to a target-refresh request,
      it MUST replace the dialog's remote target URI with the URI from
      the Contact header field in that response, if present."

   The fact that re-INVITEs can be long-lived transactions and can have
   other transactions within them makes it necessary to revise these
   rules. Section 11 specifies new rules for the handing of target-
   refresh requests. Note that the new rules apply to any target-
   refresh request, not only to re-INVITEs.

11. **Clarification on the Atomicity of Target-Refresh Requests**

   The remote target of a dialog is a special type of state information
   because of its essential role in the exchange of SIP messages between
   UAs in a dialog. A UA involved in a dialog receives the remote
   target of the dialog from the remote UA. The UA uses the remote
   target to send SIP requests to the remote UA.

   The remote target is a piece of state information that is not meant
   to be negotiated. When a UAC changes its address, the UAC simply
   communicates its new address to the UAS in order to remain reachable
   by the UAS. UAs need to follow the behavior specified in Section 12
   and Section 12 instead of that specified in RFC 3261 [RFC3261] and

discussed in [Section 10](#). The new behavior regarding target-refresh
requests implies that a target-refresh request can, in some cases,
update the remote target even if the request is responded with a
final error response. This means that target-refresh requests are
not atomic.

[12](#). UAC Behavior

Behavior of a UAC after having sent a target-refresh request updating
the remote target:

If the UAC receives an error response to the target-refresh request,
the UAS has not updated its remote target.

   This allows UASs to authenticate target-refresh requests.

If the UAC receives a reliable provisional response or a 2xx response
to the target-refresh request, or the UAC receives a request on the
new target, the UAS has updated its remote target. The UAC can
consider the target refresh operation completed.

   Even if the target request was a re-INVITE and the final response
   to the re-INVITE was an error response, the UAS would not revert
   to the pre-re-INVITE remote target.

If the UAC receives a reliable provisional response or a 2xx response
to the target-refresh request, the UAC MUST replace the dialog's
remote target URI with the URI from the Contact header field in that
response, if present.

When interacting with a UACs that does not support reliable
provisional responses or UPDATE requests, a UAC SHOULD NOT use the
same target refresh request to refresh the target and to make session
changes unless the session changes can be trivially accepted by the
UAS (e.g., a change IP address change). Piggybacking a target
refresh with more complicated session changes in this situation would
make it unnecessarily complicated for the UAS to accept the target
refresh while rejecting the session changes.

[13](#). UAS Behavior

Behavior of a UAS after having received a target-refresh request
updating the remote target:

If the UAS receives a target-refresh request that has been properly
authenticated, the UAS SHOULD generate a reliable provisional

response or a 2xx response to the target-refresh request. If
generating such responses is not possible (e.g., the UAS does not
support reliable provisional responses and needs user input before
generating a final response), the UAS SHOULD send a request to the
UAC using the new remote target (if the UAS does not need to send a
request for other reasons, the UAS can send an UPDATE request). On
sending a reliable provisional response or a 2xx response to the
target-refresh request, or a request to the new remote target, the
UAS MUST replace the dialog's remote target URI with the URI from the
Contact header field in the target-refresh request.

   Reliable provisional responses in SIP are specified in RFC 3262
   [RFC3262]. In this document, reliable provisional responses are
   those that use the mechanism defined in RFC 3262 [RFC3262] on any
   other SIP-based mechanism that may be specified in the future.
   Other specifications may define ways to send provisional responses
   reliably using non-SIP mechanisms (e.g., using media-level
   messages to acknowledge the reception of the SIP response). For
   the purposes of this document, provisional responses using those
   non-SIP mechanisms are considered unreliable responses.

If before sending a reliable provisional response or a 2xx response
to the target-refresh request, or a request to the new target, the
UAS generates an error response to the target-refresh request, the
UAS MUST NOT update its dialog's remote target.

14. Race Conditions and Target Refreshes

TODO: this is a corner case but we should describe it anyway. A UA
that changes its own contact twice in a row may create a race
condition if, for example, the first time it refreshes it using a 2xx
response (to an UPDATE or a re-INVITE) and the second with an UPDATE.
If the offer/answer glare-avoidance rules do not apply (and they
don't if there is no offer/answer exchange), the remote UA could
receive first the UPDATE and then the 2xx response for the previous
request.

15. Background on re-INVITE Transaction Routing

Re-INVITEs are routed using the dialog's route set, which contains
all the proxy servers that need to be traversed by requests send
within the dialog. Responses to the re-INVITE are routed using the
Via entries in the re-INVITE.

ACK requests for 2xx responses and for non-2xx final responses are
generated in different ways. As specified in Sections 14.1 and

13.2.1 of RFC 3261 [RFC3261], ACK requests for 2xx responses are
generated by the UAC core and are routed using the dialog's route
set. As specified in Section 17.1.1.2 of RFC 3261 [RFC3261], ACK
requests for non-2xx final responses are generated by the INVITE
client transaction (i.e., they are generated in a hop-by-hop fashion
by the proxy servers in the path) and are sent to the same transport
address as the re-INVITE.

16. Problems with UAs Losing their Contact

Refreshing the dialog's remote target during a re-INVITE transaction
(see Section 11) presents some issues because of the fact that Re-
INVITE transactions can be long lived. As described in Section 15,
the way responses to the re-INVITE and ACKs for non-2xx final
responses are routed is fixed once the re-INVITE is sent. The
routing of this messages does not depend on the dialog's route set
and, thus, target refreshes within an ongoing re-INVITE do not affect
their routing. A UA that changes its location (i.e., performs a
target refresh) but is still reachable at its old location will be
able to receive those messages (which will be sent to the old
location). However, a UA that cannot be reachable at its old
location any longer will not be able to receive them.

17. UAS Losing its Contact: UAC Behavior

When a UAS that moves to a new contact and loses its old contact
generates a non-2xx final response to the re-INVITE, it will not be
able to receive the ACK request. The entity receiving the response
and, thus, generating the ACK request will either get a transport
error or a timeout error, which, as described in Section 8.1.3.1 of
RFC 3261 [RFC3261], will be treated as a 503 (Service Unavailable)
response and as a 408 (Request Timeout) response, respectively. If
the sender of the ACK request is a proxy server, it will typically
ignore this error. If the sender of the ACK request is the UAC,
according to Section 12.2.1.2 of RFC 3261 [RFC3261], it is supposed
to (at the "should" level) terminate the dialog by sending a BYE
request. However, because of the special properties of ACK requests
for non-2xx final responses, most existing UACs do not terminate the
dialog when ACK request fails, which is fortunate.

A UAC that accepts a target refresh within a re-INVITE MUST ignore
transport and timeout errors when generating an ACK request for a
non-2xx final response if the UAC is communicating directly with the
UAS (i.e., there are no proxy servers in the dialog's route set).

18. UAC Losing its Contact: UAS Behavior

   When a UAC moves to a new contact and loses its old contact, it will
   not be able to receive responses to the re-INVITE. Consequently, it
   will never generate an ACK request.

   As described in Section 16.9 of RFC 3261 [RFC3261], a proxy server
   that gets an error when forwarding a response does not take any
   measurements. Consequently, proxy servers relaying responses will
   effectively ignore the error.

   If there are no proxy servers in the dialog's route set, the UAS will
   get an error when sending a non-2xx final response. The UAS core
   will be notified of the transaction failure, as described in Section
   17.2.1 of RFC 3261 [RFC3261]. Most existing UASs do not terminate
   the dialog on encountering this failure, which is fortunate.

   A UAS that accepts a target refresh within a re-INVITE MUST ignore
   transport and timeout errors when generating a non-2xx final response
   to the re-INVITE if the UAS is communicating directly with the UAC
   (i.e., there are no proxy servers in the dialog's route set).

   Regardless of the presence or absence of proxy servers in the
   dialog's route set, a UAS generating a 2xx response to the re-INVITE
   will never receive an ACK request for it. According to Section 14.2
   of RFC 3261 [RFC3261], such a UAS is supposed to (at the "should"
   level) terminate the dialog by sending a BYE request.

   A UAS that accepts a target refresh within a re-INVITE and never
   receives an ACK request after having sent a 2xx response to the re-
   INVITE SHOULD NOT terminate the dialog. If the UA has received a new
   re-INVITE with a higher CSeq sequence number than the original one,
   the UA SHOULD just ignore the error. If the UA has not received such
   a re-INVITE, UA SHOULD generate a new re-INVITE in order to make sure
   that both UAs have a common view of the state of the session.

      Note that the UA generates a re-INVITE and not an UPDATE request
      because UPDATE requests can be sent within a re-INVITE. By
      accepting the incoming re-INVITE, the remote UA indicates that the
      old re-INVITE transaction has already been terminated.

   A 500 (Server Internal Error) response to the new re-INVITE would
   mean that the remote UA was still processing the original re-INVITE.
   This may be because the remote UA is a legacy UA that does not
   support this specification. In this situation, the UA SHOULD follow
   the original recommendation in RFC 3261 [RFC3261] and terminate the
   dialog.

19. UAC Losing its Contact: UAC Behavior

   When a UAC moves to a new contact and loses its old contact, it will
   not be able to receive responses to the re-INVITE. Consequently, it
   will never generate an ACK request.

   Such a UAC SHOULD generate a CANCEL request to cancel the re-INVITE
   and cause the INVITE client transaction corresponding to the re-
   INVITE to enter the "Terminated" state. The UAC SHOULD also send a
   new re-INVITE in order to make sure that both UAs have a common view
   of the state of the session.

      Per Section 14.2 of RFC 3261 [RFC3261], the UAS will accept new
      incoming re-INVITEs as soon as it has generated a final response
      to the previous INVITE request, which had a lower CSeq sequence
      number.

20. Security Considerations

   This document does not introduce any new security issue. It just
   clarifies how certain transactions should be handled in SIP.
   Security issues related to re-INVITEs and UPDATE requests are
   discussed in RFC 3261 [RFC3261] and RFC 3311 [RFC3311].

21. IANA Considerations

   There are no IANA actions associated with this document.

22. Acknowledgements

   Paul Kyzivat provided useful ideas on the topics discussed in this
   document.

23. 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.

   [RFC3311] Rosenberg, J., "The Session Initiation Protocol (SIP)
              UPDATE Method", RFC 3311, October 2002.

   [RFC4032] Camarillo, G. and P. Kyzivat, "Update to the Session
              Initiation Protocol (SIP) Preconditions Framework",
              RFC 4032, March 2005.

   [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.

Authors' Addresses

   Gonzalo Camarillo (editor)
   Ericsson
   Hirsalantie 11
   Jorvas 02420
   Finland

   Email: Gonzalo.Camarillo@ericsson.com

   Christer Holmberg
   Ericsson
   Hirsalantie 11
   Jorvas 02420
   Finland

   Email: Christer.Holmberg@ericsson.com

   Yang Gao
   ZTE
   China

   Email: gao.yang2@zte.com.cn