SIPCORE
Internet-Draft
Updates: 3261 (if approved)
Intended status: Standards Track
Expires: September 5, 2010

G. Camarillo, Ed.
C. Holmberg
Ericsson
Y. Gao
ZTE

March 4, 2010

# Re-INVITE and Target-refresh Request Handling in the Session Initiation Protocol (SIP)

draft-ietf-sipcore-reinvite-03.txt

## 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.

## 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 September 5, 2010.

## Copyright Notice

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 as 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 implementors regarding how a UAS (User Agent Server) should handle re-INVITEs. In particular, implementors requested clarification on which type of response a UAS should generate in different situations. In this document, we clarify these issues.

Additionally, there has also been some confusion among implementors regarding target refresh requests, which include but are not limited to re-INVITEs. In this document, we also clarify the process by which remote targets are refreshed.

## 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.

Note that the terms UAC and UAS are used with respect to an INVITE or re-INVITE transaction and do not necessarily reflect the role of the UA concerned with respect to any other transaction, such as an UPDATE transaction occurring within the INVITE transaction.

## 3. Re-INVITE Handling

The following sections discuss re-INVITE handling.

### 3.1. 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 4.1, 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.

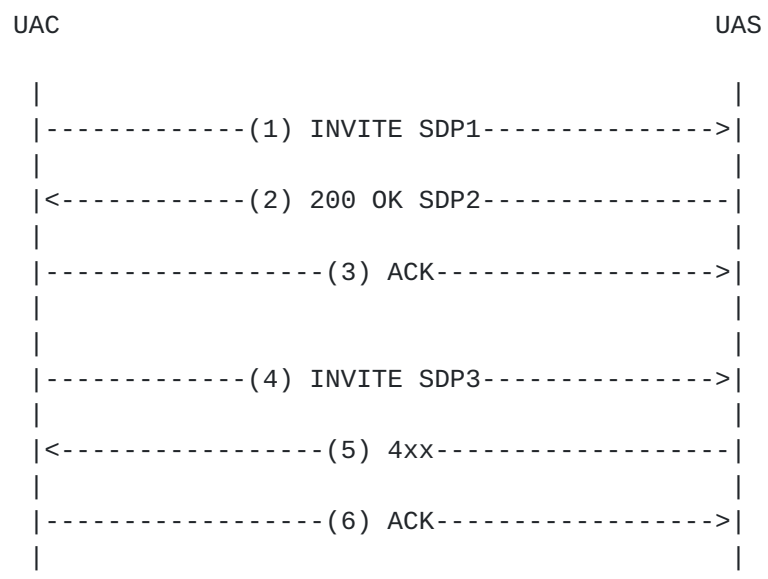


Figure 1: Rejection of a re-INVITE

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.

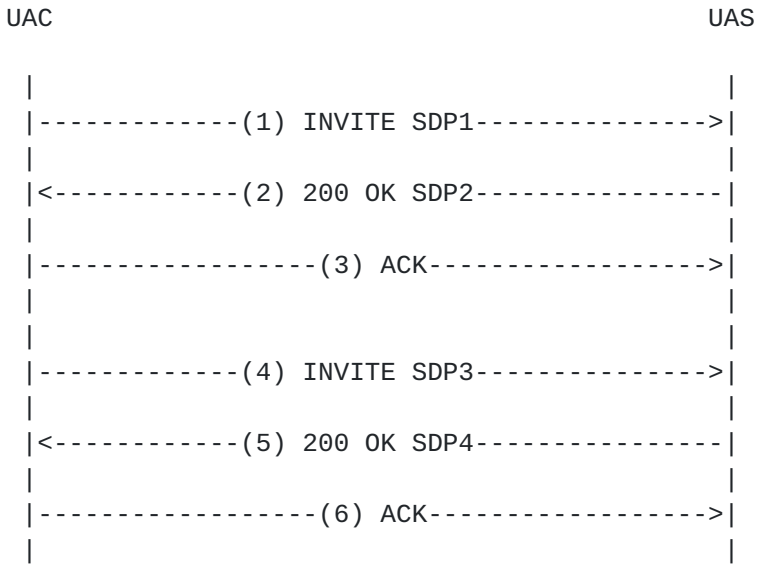


Figure 2: Automatic rejection of a video stream

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
```

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).

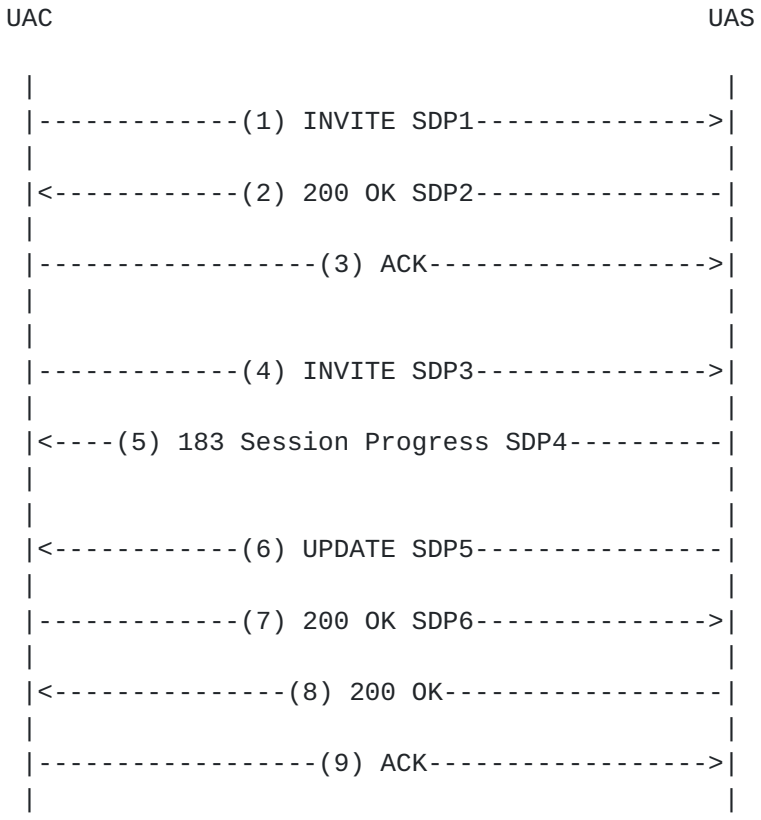

Figure 3: Rejection of a video stream by the user

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 (6) setting the port of the video stream to zero in its offer.

```
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 UAC returns a 200 (OK) response (7) to the UPDATE with the following answer:

```
SDP6:
   m=audio 30000 RTP/AVP 0
   c=IN IP4 192.0.2.2
   m=video 0 RTP/AVP 31
```

The UAS now returns a 200 (OK) response (8) to the re-INVITE.

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

## 3.2. Problems with Error Responses and Already-executed Changes

Section 3.1 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 3.3 and Section 3.4 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.

## 3.3. 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 SHOULD return a 2xx response.

A change to the session state is considered to have been executed if an offer/answer without preconditions [RFC4032] for the stream has completed successfully or the UAs have exchanged media using the new parameters. Connection establishment messages (e.g., TCP SYN), connectivity checks (e.g., when using ICE [I-D.ietf-mmusic-ice]), and any other messages used in the process of meeting the preconditions for a stream are not considered media.

> Normally, a UA receiving media can easily detect when the new parameters for the media stream are used (e.g,. media is received on a new port). However, in some scenarios the UA will have to

process incoming media packets in order to detect whether they use the old or the new parameters.

The successful completion of an offer/answer exchange without preconditions indicates that the new parameters for the media stream are already considered to be in use. The successful completion of an offer/answer exchange with preconditions means something different. The fact that all mandatory preconditions for the stream are met indicates that the new parameters for the media stream are ready to be used. However, they will not actually be used until the UAS decides so. During a session establishment, the UAS can wait before using the media parameters until the callee starts being alerted or until the callee accepts the session. During a session modification, the UAS can wait until its user accepts the changes to the session. When dealing with streams where the UAS sends media more or less continuously, the UAC notices that the new parameters are in use because the UAC receives media that uses the new parameters. However, this mechanism does not work with other types of streams. Therefore, it is RECOMMENDED that when a UAS decides to start using the new parameters for a stream for which all mandatory preconditions have been met, the UAS either sends media using the new parameters or sends a new offer where the precondition-related attributes for the stream have been removed. As indicated above, the successful completion of an offer/answer exchange without preconditions indicates that the new parameters for the media stream are already considered to be in use.

The point a change to the dialog state is considered to have been executed depends on the particular dialog parameter being modified. The specifications of different dialog parameters describe when the new value of the parameter needs to be taken into use.

## 3.4. 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 3.3). There are also 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 dialog and the session (the UAC uses the criteria in Section 3.3 in order to decide whether or not changes have been executed for the stream). 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 dialog parameters

and the session parameters in the offer/answer exchange SHOULD be as close as those in the pre-re-INVITE state as possible.

## 3.5. Glare Situations

Section 4 of RFC 3264 [RFC3264] specifies rules to avoid and detect glare situations (i.e., to avoid offer/answer collisions in race conditions). Section 14.1 of RFC 3261 [RFC3261] specifies general rules to handle glare situations in SIP. Section 5.1 of RFC 3311 [RFC3311] specifies when UPDATE requests can be sent. The specified rules include, among other things, procedures to cope with situations where both UAs generate an offer at the same time. However, there are no rules to avoid a collision between an offer in an UPDATE request and an error response to a re-INVITE. Since both the UPDATE request and the error response could be requesting changes, it would not be clear which changes would need to be executed first. The following rules avoid types of glare conditions that were not covered by previous specifications.

When checking for glare situations, UAs MUST treat the exchange of a non-2xx final response to a re-INVITE and its corresponding ACK request as an offer/answer exchange. Consequently, the rules regarding glare situations applicable to offer/answer exchanges are also applicable to those exchanges. These rules imply that if the UAS of a re-INVITE transaction receives and UPDATE request with an offer after having sent a non-2xx final response to the re-INVITE but before having received the corresponding ACK request, the UA SHOULD return a 491 (Request Pending) response to the UPDATE request. If the UAC of a re-INVITE transaction sends an UPDATE request with an offer, receives a non-2xx response to the re-INVITE, and then a 2xx response to the UPDATE request, the UA 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, as described in Section 3.4.

An UPDATE request without an offer can change dialog parameters. So can a non-2xx final response to a re-INVITE request or a 2xx response to an INVITE request (re-INVITE or initial INVITE). However, there are no rules to avoid a collision between an offerless UPDATE request and a final response to an INVITE request. The rules in Section 4.5, which are given in the context of target refreshes, cover these types of collisions as well. Therefore, there is no need to specify further rules here.

## 3.6. Example of UAS Behavior

This section contains an example of a UAS that implements 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).

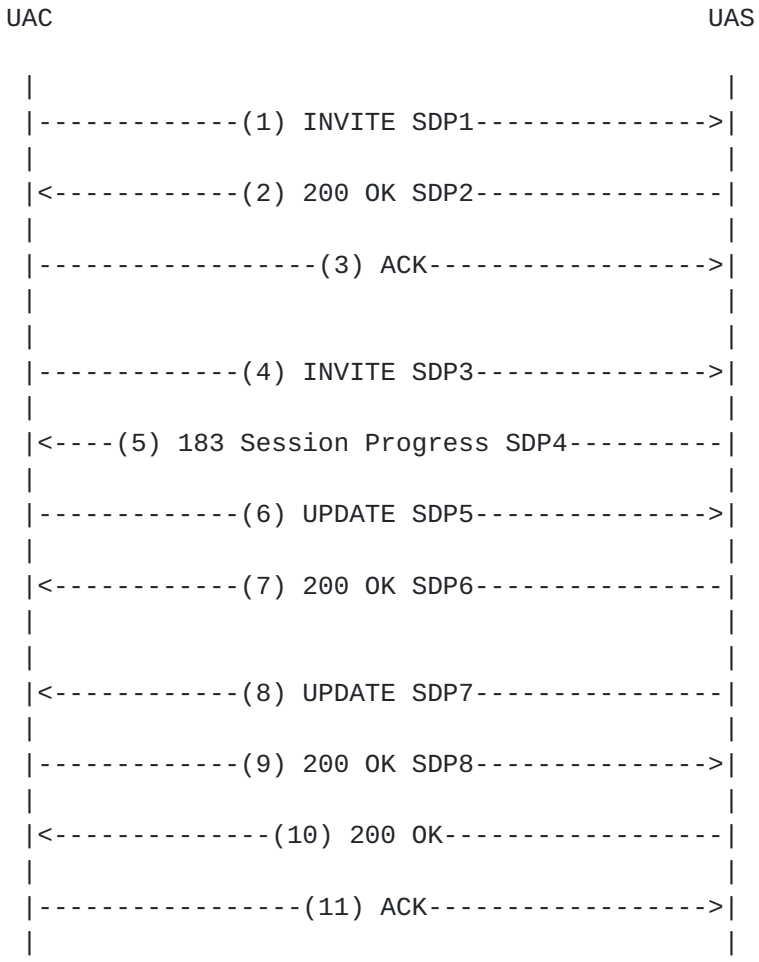


Figure 4: Rejection of a video stream by the user

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.

## 3.7. 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 recvonly by returning a 183 (Session Progress) response (2).

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

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. UA1 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).

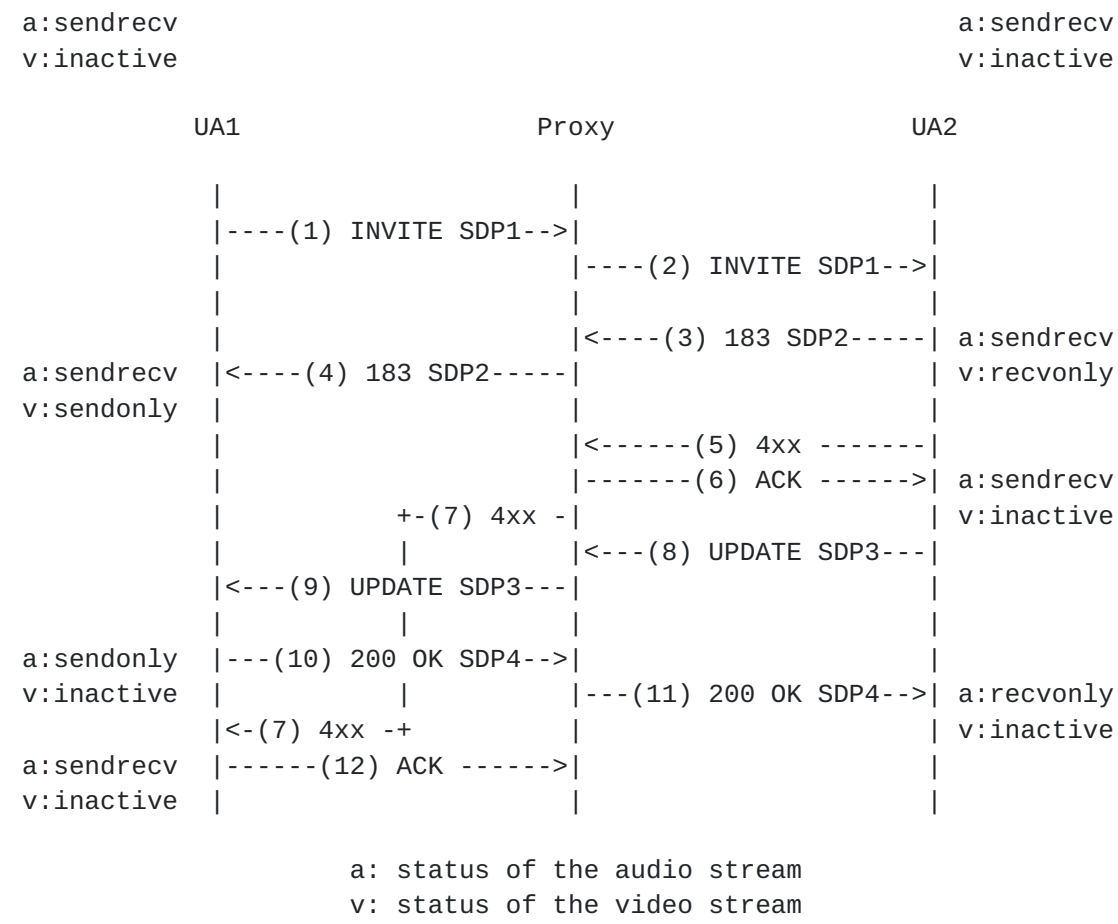


Figure 5: Message flow with race condition

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.

## 3.8. 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 3.3, 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.

## 4. Target-refresh Handling

The following sections discuss target-refresh request handling.

### 4.1. 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 4.2 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.

### 4.2. 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 4.3 and Section 4.4 of this specification instead of that specified in RFC 3261 [RFC3261], which was discussed in Section 4.1. 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.

## 4.3. 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., an 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.

## 4.4. 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] or 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 instead of 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.

### 4.5. Race Conditions and Target Refreshes

SIP provides request ordering by using the Cseq header field. That is, a UAS that receives two requests at roughly the same time can know which one is newer. However, SIP does not provide ordering between responses and requests. For example, if a UA receives a 200 (OK) response to an UPDATE request and an UPDATE request at roughly the same time, the UA cannot know which one was sent last. Since both messages can refresh the remote target, the UA needs to know which message was sent last in order to know which remote target needs to be used.

Some of the procedures discussed in Section 3.5 could avoid these types of situations. However, they are currently defined only for SIP messages involved in offer/answer exchanges (e.g., the procedures do not apply to an UPDATE request that does not carry an offer). The following rules make those procedures applicable to the race conditions described above so that UASs can cope with them.

When checking for glare situations, UAs MUST treat every UPDATE request as if it contained an offer. Additionally, UAs MUST treat

the exchange of a 2xx response to an INVITE request and its corresponding ACK request as an offer/answer exchange. Consequently, the rules regarding glare situations applicable to offer/answer exchanges are also applicable to those exchanges.

## 5. Re-INVITE Transaction Routing

The following sections discuss re-INVITE transaction routing.

### 5.1. 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.

### 5.2. Problems with UAs Losing their Contact

Refreshing the dialog's remote target during a re-INVITE transaction (see Section 4.2) presents some issues because of the fact that Re-INVITE transactions can be long lived. As described in Section 5.1, 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.

### 5.3. 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).

## 5.4. 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.

### 5.5. 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.

## 6. 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].

## 7. IANA Considerations

There are no IANA actions associated with this document.

## 8. Acknowledgements

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

## 9. 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