Internet Engineering Task Force (IETF)
Request for Comments: 6141
Updates: 3261
Category: Standards Track
ISSN: 2070-1721

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

March 2011

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

Abstract

The procedures for handling SIP re-INVITEs are described in RFC 3261. Implementation and deployment experience has uncovered a number of issues with the original documentation, and this document provides additional procedures that update the original specification to address those issues. In particular, this document defines 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, this document defines further details of procedures related to target-refresh requests.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc6141.

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 with 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 (User Agent Server) 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 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.

> Indented passages such as this one are used in this document to provide additional information and clarifying text. They do not contain normative protocol behavior.

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. Changing the Session State during a Re-INVITE

The following sub-sections discuss how to change the state of the session during a re-INVITE transaction.

3.1. Background on Re-INVITE Handling by UASs

Eventually, a UAS receiving a re-INVITE will need to 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 (this section deals with changes to the session state; target refreshes are discussed in Section 4.2). That is, the session state is the same as before the re-INVITE was received. The example in Figure 1 illustrates this point.

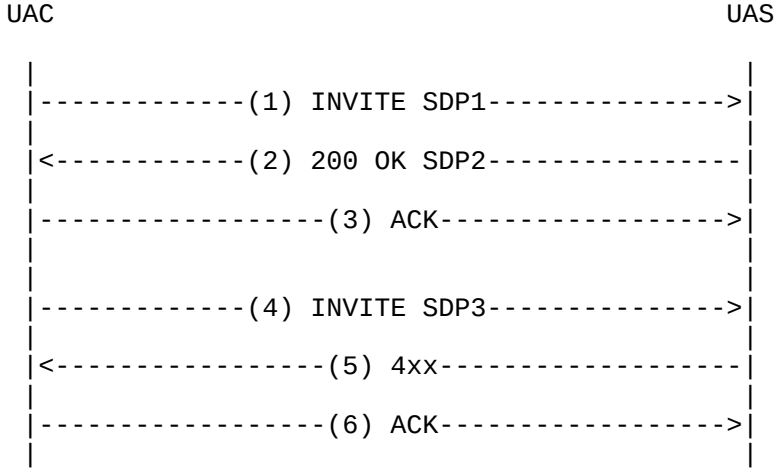

Figure 1: Rejection of a re-INVITE

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 configured to automatically 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 state is 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 a Session Description Protocol (SDP) indicating which changes were accepted and which were not. The example in Figure 2 illustrates this point.

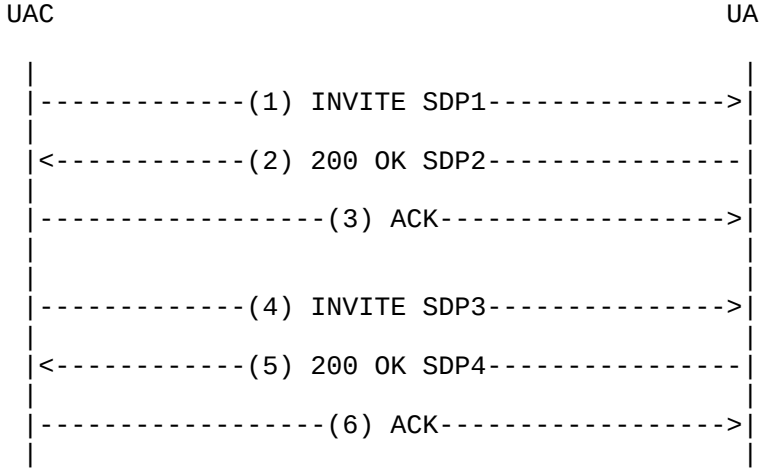

Figure 2: Automatic rejection of a video stream

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).

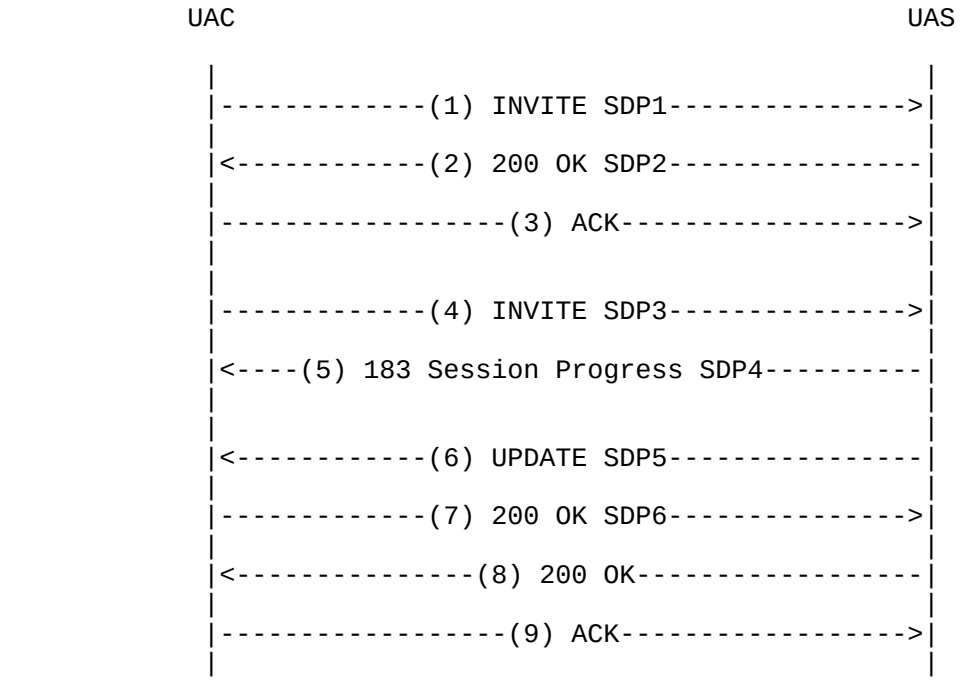

Figure 3: Manual rejection of a video stream by the user

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 RTP Control Protocol (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 (Figures 2 and 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). Sections 3.3 and 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 (e.g., as a result of UPDATE transactions or reliable provisional responses) is effectively requesting a change in the session state. However, the UAC has no means to reject that change 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 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, 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 UA has sent or received media using the new parameters. Connection establishment messages (e.g., TCP SYN), connectivity checks (e.g., when using Interactive Connectivity Establishment (ICE) [RFC5245]), 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 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 to use them. 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.

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 session (the UAC uses the criteria in Section 3.3 in order to decide whether or not changes have been executed for a particular 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, session parameters in the offer/answer exchange SHOULD be as close to those in the pre-re-INVITE state as possible.

3.5. Glare Situations

Section 4 of RFC 3264 [RFC3264] defines glare conditions as a user agent receiving an offer after having sent one but before having received an answer to it. That section specifies rules to avoid glare situations in most cases. When, despite following those rules, a glare condition occurs (as a result of a race condition), it is handled as specified in Sections 14.1 and 14.2 of RFC 3261 [RFC3261]. The UAS returns a 491 (Request Pending) response and the UAC retries the offer after a randomly selected time, which depends on which user agent is the owner of the Call-ID of the dialog. The rules in RFC 3261 [RFC3261] not only cover collisions between re-INVITEs that contain offers, they cover collisions between two re-INVITEs in general, even if they do not contain offers. Sections 5.2 and 5.3 of RFC 3311 [RFC3311] extend those rules to also cover collisions between an UPDATE request carrying an offer and another message (UPDATE, PRACK, or INVITE) also carrying an offer.

The rules in RFC 3261 [RFC3261] do not cover collisions between an UPDATE request and a non-2xx final response to a re-INVITE. Since both the UPDATE request and the reliable response could be requesting changes to the session state, it would not be clear which changes would need to be executed first. However, the procedures discussed in Section 3.4 already cover this type of situation. 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 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).

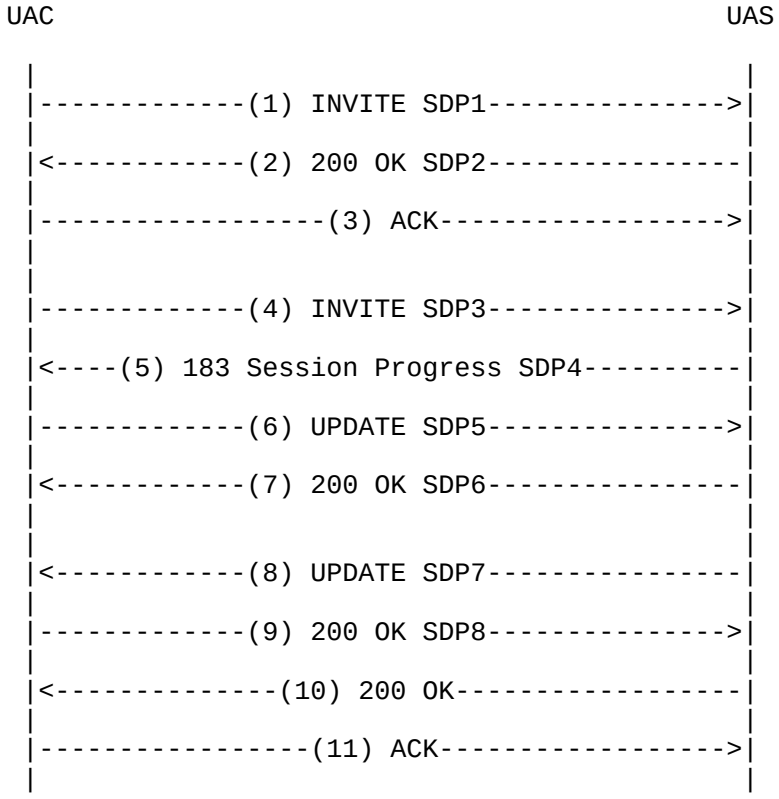

Figure 4: Rejection of a video stream by the user

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.

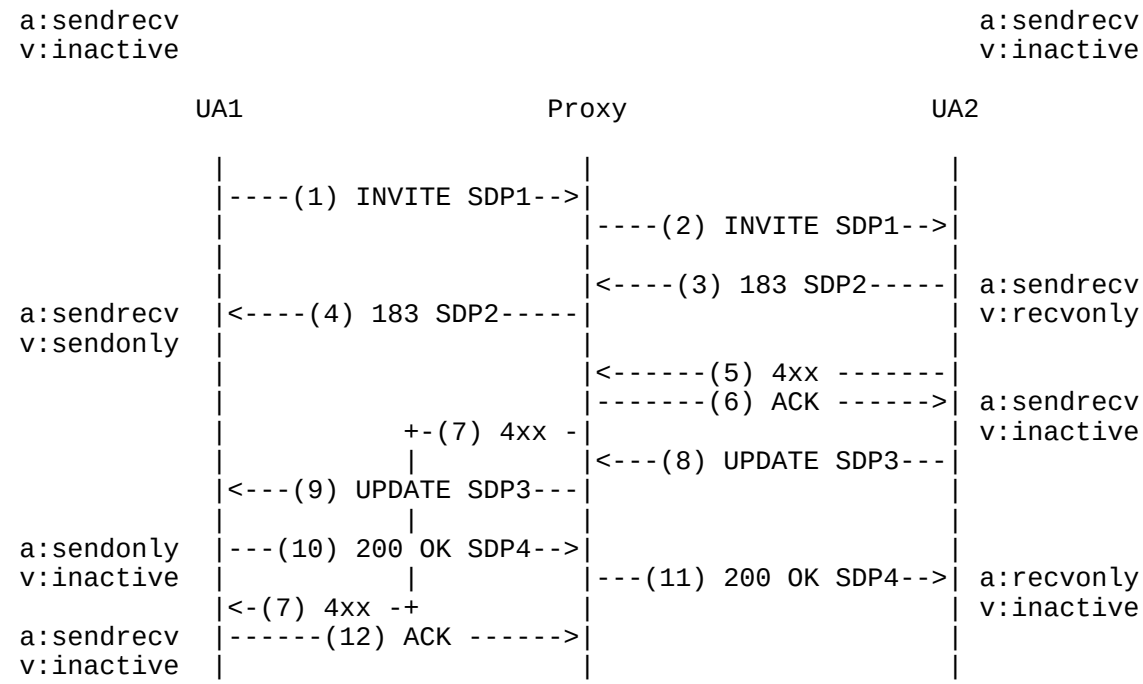

Figure 5: Message flow with race condition

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).

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 Canceling 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. Refreshing a Dialog's Targets

The following sections discuss how to refresh the targets of a dialog.

4.1. Background and Terminology on a Dialog's Targets

As described in Section 12 of RFC 3261 [RFC3261], a UA involved in a dialog keeps a record of the SIP or Session Initiation Protocol Secure (SIPS) URI at which it can communicate with a specific instance of its peer (this is called the "dialog's remote target URI" and is equal to the URI contained in the Contact header of requests and responses it receives from the peer). This document introduces the complementary concept of the "dialog's local target URI", defined as a UA's record of the SIP or SIPS URI at which the peer can communicate with it (equal to the URI contained in the Contact header of requests and responses it sends to the peer). These terms are complementary because the "dialog's remote target URI" according to one UA is the "dialog's local target URI" according to the other UA, and vice versa.

4.2. 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 4.3 specifies new rules for the handling of target-refresh requests. Note that the new rules apply to any target-refresh request, not only to re-INVITEs.

4.3. Clarification on the Atomicity of Target-Refresh Requests

The local and remote targets of a dialog are special types of state information because of their 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 received remote target to send SIP requests to the remote UA.

The dialog's local target is a piece of state information that is not meant to be negotiated. When a UA changes its local target (i.e., the UA changes its IP address), the UA simply communicates its new local target to the remote UA (e.g., the UA communicates its new IP address to the remote UA in order to remain reachable by the remote UA). UAs need to follow the behavior specified in Sections 4.4, 4.5, 4.6, and 4.7 of this specification instead of that specified in RFC 3261 [RFC3261], which was discussed in Section 4.2. 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 to with a final error response. This means that target-refresh requests are not atomic.

4.4. UA Updating the Dialog's Local Target in a Request

In order to update its local target, a UA can send a target-refresh request. If the UA receives an error response to the target-refresh request, the remote UA has not updated its remote target.

> This allows UASs to authenticate target-refresh requests (see Section 26.2 of RFC 3261 [RFC3261]).

If the UA receives a reliable provisional response or a 2xx response to the target-refresh request, or the UA receives an in-dialog request on the new local target, the remote UA has updated its remote target. The UA 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.

A UA 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 remote UA (e.g., an IP address change). Piggybacking a target refresh with more complicated session changes would make it unnecessarily complicated for the remote UA to accept the target refresh while rejecting the session changes. Only in case the target refresh request is a re-INVITE and the UAS supports reliable provisional response or UPDATE requests, the UAC MAY piggyback session changes and a target refresh in the same re-INVITE.

4.5. UA Updating the Dialog's Local Target in a Response

A UA processing an incoming target refresh request can update its local target by returning a reliable provisional response or a 2xx response to the target-refresh request. The response needs to contain the updated local target URI in its Contact header field. On sending the response, the UA can consider the target refresh operation completed.

4.6. A Request Updating the Dialog's Remote Target

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

If the UA receives a target-refresh request that has been properly authenticated (see Section 26.2 of RFC 3261 [RFC3261]), the UA 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 UA does not support reliable provisional responses and needs user input before generating a final response), the UA SHOULD

send an in-dialog request to the remote UA using the new remote target (if the UA 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 UA 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]. 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. Note that non-100 provisional responses are only applicable to INVITE transactions [RFC4320].

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 UA generates an error response to the target-refresh request, the UA MUST NOT update its dialog's remote target.

4.7. A Response Updating the Dialog's Remote Target

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

If a UA receives an unreliable provisional response to a target-refresh request, the UA MUST NOT refresh the dialog's remote target.

4.8. Race Conditions and Target Refreshes

SIP provides request ordering by using the Cseq header field. That is, a UA 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.

This document specifies the following rule to avoid the situation just described. If the protocol allows a UA to use a target-refresh request at the point in time that the UA wishes to refresh its local target, the UA MUST use a target-refresh request instead of a response to refresh its local target. This rule implies that a UA only uses a response (i.e., a reliable provisional response or a 2xx response to a target-refresh request) to refresh its local target if the UA is unable to use a target-refresh request at that point in time (e.g., the UAS of an ongoing re-INVITE without support for UPDATE).

4.9. Early Dialogs

The rules given in this section about which messages can refresh the target of a dialog also apply to early dialogs created by an initial INVITE transaction. Additionally, as specified in Section 13.2.2.4 of RFC 3261 [RFC3261], on receiving a 2xx response to the initial INVITE, the UAC recomputes the whole route set of the dialog, which transitions from the "early" state to the "confirmed" state.

Section 12.1 of RFC 3261 allows unreliable provisional responses to create early dialogs. However, per the rules given in this section, unreliable provisional responses cannot refresh the target of a dialog. Therefore, the UAC of an initial INVITE transaction will not perform any target refresh as a result of the reception of an unreliable provisional response with an updated Contact value on an (already established) early dialog. Note also that a given UAS can establish additional early dialogs, which can have different targets, by returning additional unreliable provisional responses with different To tags.

5. A UA Losing Its Contact

The following sections discuss the case where a UA loses its transport address during an ongoing re-INVITE transaction. Such a UA will refresh the dialog’s local target so that it reflects its new transport address. Note that target refreshes that do not involve changes in the UA’s transport address are outside of the scope of this section. Also, UAs losing their transport address during a non-re-INVITE transaction (e.g., a UA losing its transport address right after having sent an UPDATE request before having received a response to it) are out of scope as well.

The rules given in this section are also applicable to initial INVITE requests that have established early dialogs.

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 sent 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 4.3) 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.

The following sections describe the errors UAs face when they lose their transport address during a re-INVITE. On detecting some of these errors, UAs following the rules specified in RFC 3261 [RFC3261] will terminate the dialog. When the dialog is terminated, the only option for the UAs is to establish a new dialog. The following sections change the requirements RFC 3261 [RFC3261] places on UAs when certain errors occur so that the UAs can recover from those errors. In short, the UAs generate a new re-INVITE transaction to synchronize both UAs. Note that there are existing UA implementations deployed that already implement this behavior.

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. Additionally, UAC SHOULD generate a new re-INVITE in order to make sure that both UAs have a common view of the state of the session.

> It is possible that the errors ignored by the UAC were not related to the target refresh operation. If that was the case, the second re-INVITE would fail and the UAC would terminate the dialog because, per the rules above, UACs only ignore errors when they accept a target refresh within the re-INVITE.

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 measures. 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.

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 final 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.

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].

In particular, in order not to reduce the security level for a given session, re-INVITEs and UPDATE requests SHOULD be secured using a mechanism equivalent to or stronger than the initial INVITE request that created the session. For example, if the initial INVITE request was end-to-end integrity protected or encrypted, subsequent re-INVITEs and UPDATE requests should also be so.

7. Acknowledgements

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

8. References

8.1. 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.

[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.

8.2. Informative References

[RFC4320] Sparks, R., "Actions Addressing Identified Issues with the Session Initiation Protocol's (SIP) Non-INVITE Transaction", RFC 4320, January 2006.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

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