Internet Engineering Task Force (IETF)
Request for Comments: 6337
Category: Informational
ISSN: 2070-1721

S. Okumura
Softfront
T. Sawada
KDDI Corporation
P. Kyzivat

August 2011

Session Initiation Protocol (SIP) Usage of the Offer/Answer Model

Abstract

The Session Initiation Protocol (SIP) utilizes the offer/answer model to establish and update multimedia sessions using the Session Description Protocol (SDP). The description of the offer/answer model in SIP is dispersed across multiple RFCs. This document summarizes all the current usages of the offer/answer model in SIP communication.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

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). Not all documents approved by the IESG are a candidate for any level of Internet Standard; see 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/rfc6337.

Table of Contents

1. Introduction

SIP utilizes the offer/answer model to establish and update sessions. The rules that govern the offer/answer behaviors in SIP are described in several RFCs: [RFC3261], [RFC3262], [RFC3264], [RFC3311], and [RFC6141].

The primary purpose of this document is to describe all forms of SIP usage of the offer/answer model in one document to help the readers to fully understand it. Also, this document tries to incorporate the results of the discussions on the controversial issues to avoid repeating the same discussions later.

This document describes ambiguities in the current specifications and the authors' understanding of the correct interpretation of these specifications. This document is not intended to make any changes to those specifications, but rather is intended to provide a reference for future standards development work on the SIP offer/answer model and to developers looking for advice on how to implement in compliance with the standards.

2. Summary of SIP Usage of the Offer/Answer Model

The offer/answer model itself is independent from the higher layer application protocols that utilize it. SIP is one of the applications using the offer/answer model. [RFC3264] defines the offer/answer model, but does not specify which SIP messages should convey an offer or an answer. This should be defined in the SIP core and extension RFCs.

In theory, any SIP message can include a session description in its body. But a session description in a SIP message is not necessarily an offer or an answer. Only certain session description usages that conform to the rules described in Standards-Track RFCs can be interpreted as an offer or an answer. The rules for how to handle the offer/answer model are defined in several RFCs.

The offer/answer model defines a mechanism for update of sessions. In SIP, a dialog is used to associate an offer/answer exchange with the session that it is to update. In other words, only the offer/answer exchange in the SIP dialog can update the session that is managed by that dialog.

2.1. 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 [RFC2119].

The following abbreviations are used in this document.

UA: User Agent.

UAC: User Agent Client.

UAS: User Agent Server.

SDP: Session Description Protocol [RFC4566].

2.2. Offer/Answer Exchange Pairs in SIP Messages

Currently, the rules on the offer/answer model are defined in [RFC3261], [RFC3262], [RFC3264], [RFC3311], and [RFC6141]. In these RFCs, only the six patterns shown in Table 1 are defined for exchanging an offer and an answer with SIP messages.

Note that an offer/answer exchange initiated by an INVITE request must follow exactly one of the Patterns 1, 2, 3, 4. When an initial INVITE causes multiple dialogs due to forking, an offer/answer exchange is carried out independently in each distinct dialog. When an INVITE request contains no offer, only Pattern 2 or Pattern 4 apply. According to Section 13.2.1 of [RFC3261], 'The first reliable non-failure message' must have an offer if there is no offer in the INVITE request. This means that the User Agent (UA) that receives the INVITE request without an offer must include an offer in the first reliable response with 100rel extension. If no reliable provisional response has been sent, the User Agent Server (UAS) must include an offer when sending 2xx response.

In Pattern 3, the first reliable provisional response may or may not have an answer. When a reliable provisional response contains a session description, and is the first to do so, then that session description is the answer to the offer in the INVITE request. The answer cannot be updated, and a new offer cannot be sent in a subsequent reliable response for the same INVITE transaction.

In Pattern 5, a Provisional Response ACKnowledgement (PRACK) request can contain an offer only if the reliable response that it acknowledges contains an answer to the previous offer/answer exchange.

> NOTE: It is legal to have UPDATE/2xx exchanges without offer/answer exchanges (Pattern 6). However, when re-INVITEs are sent for non-offer/answer purposes, an offer/answer exchange is required. In that case, the prior SDP will typically be repeated.

There may be ONLY ONE offer/answer negotiation in progress for a single dialog at any point in time. Section 4 explains how to ensure this. When an INVITE results in multiple dialogs, each has a separate offer/answer negotiation.

NOTE: This is when using a Content-Disposition of "session". There may be a second offer/answer negotiation in progress using a Content-Disposition of "early-session" [RFC3959]. That is not addressed by this document.

Offer	Answer	RFC	Ini	Est	Early
1. INVITE Req.	2xx INVITE Resp.	RFC 3261	Y	Y	N
2. 2xx INVITE Resp.	ACK Req.	RFC 3261	Y	Y	N
3. INVITE Req.	1xx-rel INVITE Resp.	RFC 3262	Y	Y	N
4. 1xx-rel INVITE Resp.	PRACK Req.	RFC 3262	Y	Y	N
5. PRACK Req.	200 PRACK Resp.	RFC 3262	N	Y	Y
6. UPDATE Req.	2xx UPDATE Resp.	RFC 3311	N	Y	Y

Table 1: Summary of SIP Usage of the Offer/Answer Model

In Table 1, '1xx-rel' corresponds to the reliable provisional response that contains the 100rel option defined in [RFC3262].

The 'Ini' column shows the ability to exchange the offer/answer to initiate the session. 'Y' indicates that the pattern can be used in the initial offer/answer exchange, while 'N' indicates that it cannot. Only the initial INVITE transaction can be used to exchange the offer/answer to establish a multimedia session.

The 'Est' column shows the ability to update the established session.

The 'Early' column indicates which patterns may be used to modify the established session in an early dialog. There are two ways to exchange a subsequent offer/answer in an early dialog.

2.3. Rejection of an Offer

It is not always clear how to reject an offer when it is unacceptable, and some methods do not allow explicit rejection of an offer. For each of the patterns in Table 1, Table 2 shows how to reject an offer.

When a UA receives an INVITE request with an unacceptable offer, it should respond with a 488 response, preferably with Warning header field indicating the reason of the rejection, unless another response code is more appropriate to reject it (Pattern 1 and Pattern 3).

If this is a re-INVITE, extra care must be taken, as detailed in [RFC6141]. Specifically, if the offer contains any changes or additions to media stream properties, and those have already been used to transmit/receive media before the final response is sent, then a 2xx response should be sent, with a syntactically correct session description. This may optionally be followed by an UPDATE request to rearrange the session parameters if both ends support the UPDATE method. Alternatively, the UA may send an error response to the (re-)INVITE request to terminate the dialog or to roll back the offer/answer status before sending re-INVITE request. In this case, the UAS should not continue to retransmit the unacknowledged reliable provisional response; the User Agent Client (UAC) should not continue to retransmit a PRACK request.

When a UA receives an UPDATE request with an offer that it cannot accept, it should respond with a 488 response, preferably with Warning header field indicating the reason of the rejection, unless another response code is more appropriate to reject it (Pattern 6).

When a UA receives a PRACK request with an offer that it cannot accept, it may respond with a 200 response with a syntactically correct session description. Optionally, this may be followed by an UPDATE request to rearrange the session parameters if both ends support the UPDATE method. Alternatively, the UA may terminate the dialog and send an error response to the INVITE request (Pattern 5).

In addition, there is a possibility for UAC to receive a 488 response for an PRACK request. In that case, UAC may send again a PRACK request without an offer or send a CANCEL request to terminate the INVITE transaction.

> NOTE: In [RFC3262], the following restriction is defined with regard to responding to a PRACK request.
>
> "If the PRACK does match an unacknowledged reliable provisional response, it MUST be responded to with a 2xx response."
>
> This restriction is not clear. There are cases where it is unacceptable to send a 2xx response. For example, the UAS may need to send an authentication challenge in a 401 response. This is an open issue and out of scope for this document.

When a UA receives a response with an offer that it cannot accept, the UA does not have a way to reject it explicitly. Therefore, a UA should respond to the offer with the correct session description and rearrange the session parameters by initiating a new offer/answer

exchange, or alternatively terminate the session (Pattern 2 and Pattern 4). When initiating a new offer/answer, a UA should take care not to cause an infinite offer/answer loop.

Section 14.2 of [RFC3261], "UAS Behavior", states:

> The UAS MUST ensure that the session description overlaps with its previous session description in media formats, transports, or other parameters that require support from the peer. This is to avoid the need for the peer to reject the session description.

This is a rule for an offer within 2xx response to a re-INVITE. This rule should be applied to an offer within a reliable provisional response and a PRACK request.

Offer	Rejection
1. INVITE Req. (*)	488 INVITE Response
2. 2xx INVITE Resp.	Answer in ACK Req. followed by new offer OR termination of dialog
3. INVITE Req.	488 INVITE Response (same as Pattern 1)
4. 1xx-rel INVITE Resp.	Answer in PRACK Req. followed by new offer
5. PRACK Req. (**)	200 PRACK Resp. followed by new offer OR termination of dialog
6. UPDATE Req.	488 UPDATE Response

(*) If this was a re-INVITE, a failure response should not be sent if media has already been exchanged using the new offer.

(**) A UA should only use PRACK to send an offer when it has strong reasons to expect the receiver will accept the offer.

Table 2: Rejection of an Offer

2.4. Session Description That Is Not an Offer or an Answer

As previously stated, a session description in a SIP message is not necessarily an offer or an answer. For example, SIP can use a session description to describe capabilities apart from offer/answer exchange. Examples of this are a 200 OK response for OPTIONS and a 488 response for INVITE.

3. Detailed Discussion of the Offer/Answer Model for SIP

3.1. Offer/Answer for the INVITE method with 100rel Extension

The INVITE method provides the basic procedure for offer/answer exchange in SIP. Without the 100rel option, the rules are simple as described in [RFC3261]. If an INVITE request includes a session description, Pattern 1 is applied and if an INVITE request does not include a session description, Pattern 2 is applied.

With 100rel, Patterns 3, 4, and 5 are added and this complicates the rules. An INVITE request may cause multiple responses. Note that even if both UAs support the 100rel extension, not all the provisional responses may be sent reliably.

3.1.1. INVITE Request with SDP

When a UAC includes an SDP body in the INVITE request as an offer, only the first SDP in a reliable non-failure response to the INVITE request is the real answer. No other offer/answer exchanges can occur within the messages (other responses and ACK) of the INVITE transaction.

In [RFC3261] there are some descriptions about an offer/answer exchange, but those cause a little confusion. We interpret those descriptions as follows,

UAC behavior:

1. If the first SDP that the UAC received is included in an unreliable provisional response to the INVITE request, [RFC3261] (Section 13.2.1, second bullet) requires that this be treated as an answer. However, because that same section states that the answer has to be in a reliable non-failure message, this SDP is not the true answer and therefore the offer/answer exchange is not yet completed.

2. After the UAC has received the answer in a reliable provisional response to the INVITE, [RFC3261] requires that any SDP in subsequent responses be ignored.

3. If the second and subsequent SDP (including a real answer) is different from the first SDP, the UAC should consider that the SDP is equal to the first SDP. Therefore, the UAC should not switch to the new SDP.

UAS behavior:

1. [RFC3261] requires all SDP in the responses to the INVITE request to be identical.

2. After the UAS has sent the answer in a reliable provisional response to the INVITE, the UAS should not include any SDPs in subsequent responses to the INVITE.

3. [RFC3261] permits the UAS to send any provisional response without SDP regardless of the transmission of the answer.

A session description in an unreliable response that precedes a reliable response can be considered a "preview" of the answer that will be coming.

NOTE: This "preview" session description rule applies to a single offer/answer exchange. In parallel offer/answer exchanges (caused by forking), a UA may obviously receive a different "preview" of an answer in each dialog. UAs are expected to deal with this.

Although [RFC3261] says a UA should accept media once an INVITE with an offer has been sent, in many cases, an answer (or, at least a preview of it) is required in order for media to be accepted. Two examples of why this might be required are as follows:

- To avoid receiving media from undesired sources, some User Agents assume symmetric RTP will be used, ignore all incoming media packets until an address/port has been received from the other end, and then use that address/port to filter incoming media packets.

- In some networks, an intermediate node must authorize a media stream before it can flow and requires a confirming answer to the offer before doing so.

Therefore, a UAS should send an SDP answer reliably (if possible) before it starts sending media. And, if neither the UAC nor the UAS support 100rel, the UAS should send a preview of the answer before it starts sending media.

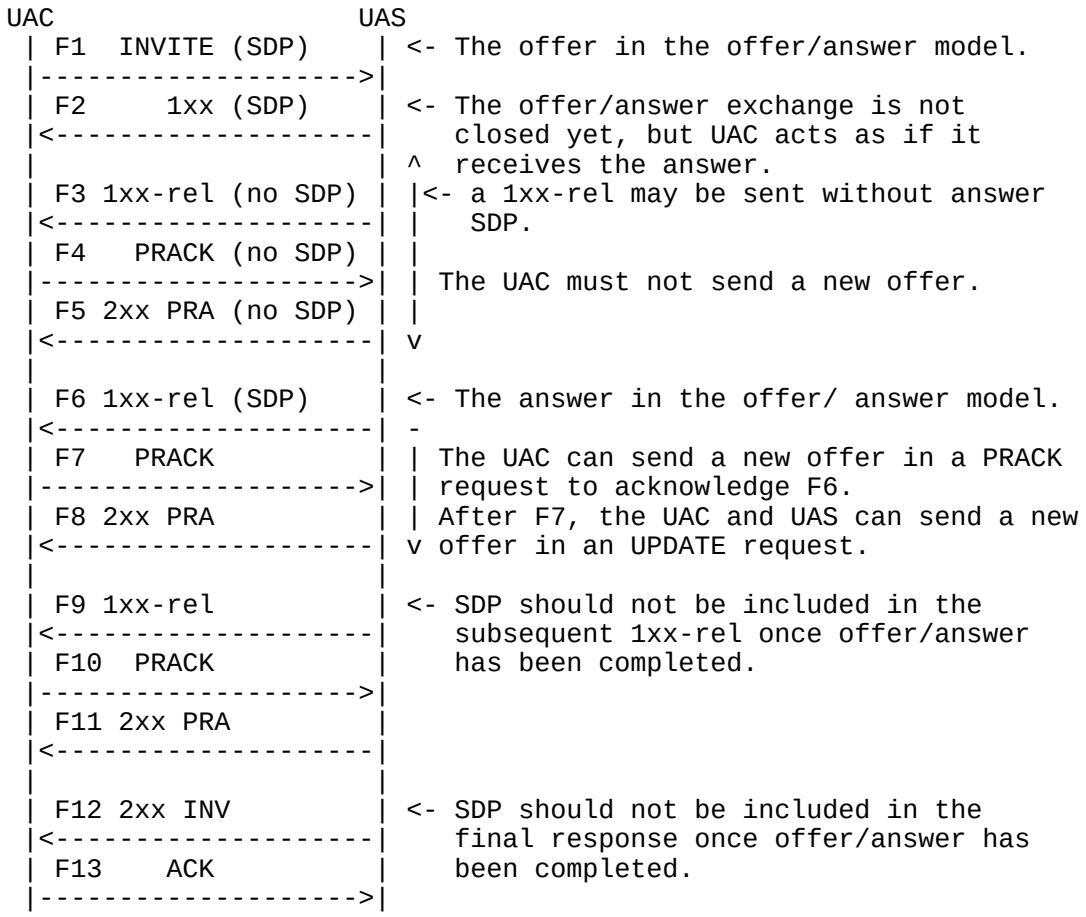

Figure 1: Example of Offer/Answer with 100rel Extension (1)

For example, in Figure 1, only the SDP in F6 is the answer. The SDP in the non-reliable response (F2) is the preview of the answer and must be the same as the answer in F6. Receiving F2, the UAC should act as if it receives the answer. However, offer/answer exchange is not completed yet and the UAC must not send a new offer until it receives the same SDP in a reliable non-failure response, which is the real answer. After sending the SDP in F6, the UAS must prepare to receive a new offer from the UAC in a PRACK request or in an UPDATE request if the UAS supports UPDATE.

The UAS does not include SDP in responses F9 and F12. However, the UAC should prepare to receive SDP bodies in F9 and/or F12, and just ignore them, to handle a peer that does not conform to the recommended implementation.

3.1.2. INVITE Request without SDP

When a UAC does not include an SDP body in the INVITE request, [RFC3261] (Section 13.2.1, first bullet) requires that the UAS include an offer in the first reliable non-failure response. However, a UAC might not expect an SDP in the other responses to the INVITE request because RFC 3261 simply does not anticipate the possibility. Therefore, the UAS ought not include any SDP in the other responses to the INVITE request.

NOTE: In Figure 2, the UAS should not include SDP in the responses F6 and F9. However, the UAC should prepare to receive SDP bodies in F6 and/or F9, and just ignore them to handle a peer that does not conform to the recommended implementation.

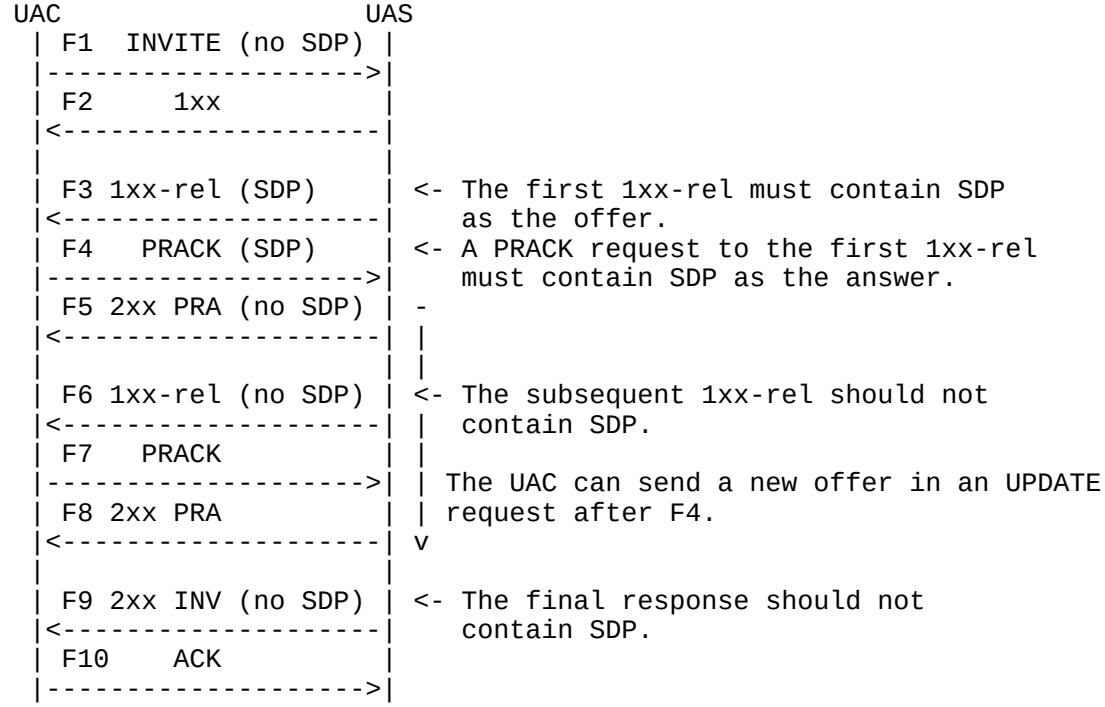

Figure 2: Example of Offer/Answer with 100rel Extension (2)

Note that in the case that the UAC needs to prompt the user to accept or reject the offer, the reliable provisional response with SDP as an offer (Pattern 4) can result in the retransmission until the PRACK request can be sent. The UAC should take care to avoid this situation when it sends the INVITE request without SDP.

3.2. Offer/Answer Exchange in Early Dialog

When both UAs support the 100rel extension, they can update the session in the early dialog once the first offer/answer exchange has been completed.

From a UA sending an INVITE request:

A UA can send an UPDATE request with a new offer if both ends support the UPDATE method. Note that if the UAS needs to prompt the user to accept or reject the offer, the delay can result in retransmission of the UPDATE request.

A UA can send a PRACK request with a new offer only when acknowledging the reliable provisional response carrying the answer to an offer in the INVITE request. Compared to using the UPDATE method, using PRACK can reduce the number of messages exchanged between the UAs. However, to avoid problems or delays caused by PRACK offer rejection, the UA is recommended to send a PRACK request only when it has strong reasons to expect the receiver will accept it. For example, the procedure used in precondition extension [RFC3312] is a case where a PRACK request should be used for updating the session status in an early dialog. Note also that if a UAS needs to prompt the user to accept or reject the offer, the delay can result in retransmission of the PRACK request.

From a UA receiving an INVITE request:

A UA can send an UPDATE request with a new offer if both ends support the UPDATE method. A UAS cannot send a new offer in the reliable provisional response, so the UPDATE method is the only method for a UAS to update an early session.

3.3. Offer/Answer Exchange in an Established Dialog

Both the re-INVITE and UPDATE methods can be used in an established dialog to update the session.

The UPDATE method is simpler and can save at least one message compared with the INVITE method. But both ends must support the UPDATE method for it to be used.

The INVITE method needs at least three messages to complete but no extensions are needed. Additionally, the INVITE method allows the peer to take time to decide whether or not it will accept a session update by sending provisional responses. That is, re-INVITE allows the UAS to interact with the user at the peer, while UPDATE needs to be answered automatically by the UAS. It is noted that re-INVITE

should be answered immediately unless such a user interaction is needed. Otherwise, some Third Party Call Control (3PCC) [RFC3725] flows will break.

3.4. Recovering from a Failed Re-INVITE

Section 14.1 of [RFC3261] requires that the session parameters in effect prior to a re-INVITE remain unchanged if the re-INVITE fails, as if no re-INVITE had been issued. This remains the case even if multiple offer/answer exchanges have occurred between the sending of the re-INVITE and its failure, and even if media has been exchanged using the proposed changes in the session. Because this can be difficult to achieve in practice, a newer specification [RFC6141] recommends the UAS to send a 2xx response to a re-INVITE in cases where rolling back changes would be problematic.

Nevertheless, a UAC may receive a failure response to a re-INVITE after proposed changes that must be rolled back have already been used. In such a case, the UAC should send an UPDATE offering the SDP that has been reinstated. (See [RFC6141] for details.)

4. Exceptional Case Handling

In [RFC3264], the following restrictions are defined with regard to sending a new offer.

> At any time, either agent MAY generate a new offer that updates the session. However, it MUST NOT generate a new offer if it has received an offer which it has not yet answered or rejected. Furthermore, it MUST NOT generate a new offer if it has generated a prior offer for which it has not yet received an answer or a rejection.

Assuming that the above rules are guaranteed, there seem to be two possible 'exceptional' cases to be considered in SIP offer/answer usage: the 'message crossing' case and the 'glare' case. One of the reasons why the usage of SIP methods to exchange offer/answer needs to be carefully restricted in the RFCs is to ensure that the UA can detect and handle appropriately the 'exceptional' cases to avoid incompatible behavior.

4.1. Message Crossing Case Handling

When message packets cross in the transport network, an offer may be received before the answer for the previous offer/answer exchange, as shown in Figure 3. In such a case, UA A must detect that the session description SDP-2 is not the answer to offer1.

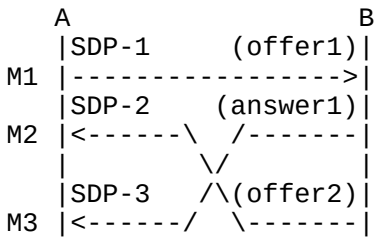

Figure 3: Message Crossing Case

Because of the restrictions on placement of offers and answers (summarized in Table 1), there are a limited number of valid exchanges of messages that may lead to this message crossing case. These are enumerated in Table 3. (This table only shows messages containing offers or answers. There could be other messages, without session descriptions, which are not shown.)

When a response to an UPDATE request crosses a reliable response to an INVITE request, there are variants shown in Figures 4 and 5, which are dependent on an INVITE (Mx) that contains no offer. These are also included in Table 3.

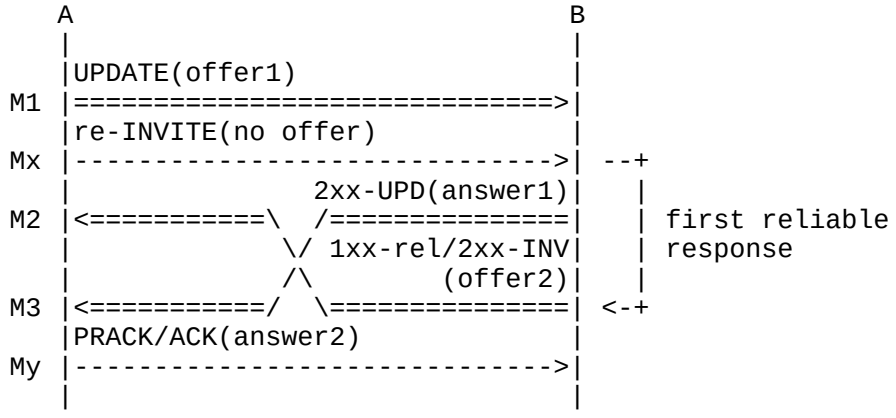

Figure 4: Avoidable Message Crossing Cases

To avoid the message crossing condition shown in Figure 4, UA A should not send this re-INVITE request until an UPDATE transaction has been completed. If UA B encounters this message crossing condition, it should reject this re-INVITE request with a 500 response.

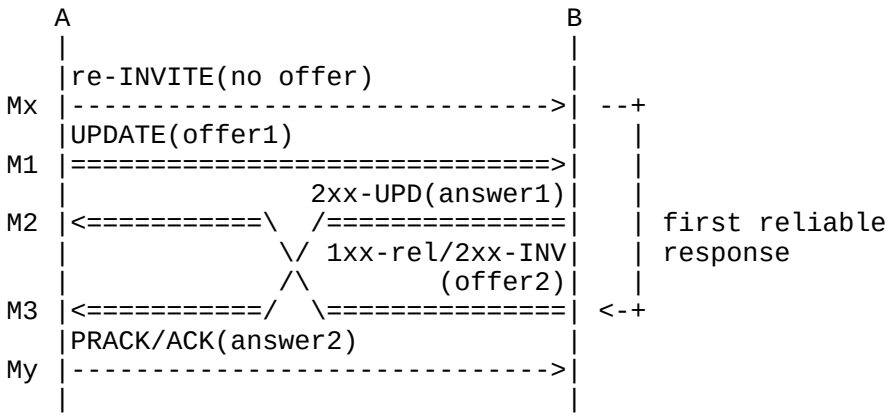

Figure 5: Avoidable Message Crossing Cases

To avoid the message crossing condition shown in Figure 5, UA A should not send this UPDATE request until an ACK or a PRACK transaction associated with an offer/answer has been completed. If UA B encounters this message crossing condition, it should reject this UPDATE request with a 500 response.

The situation when a PRACK request crosses UPDATE request is shown in Figure 6.

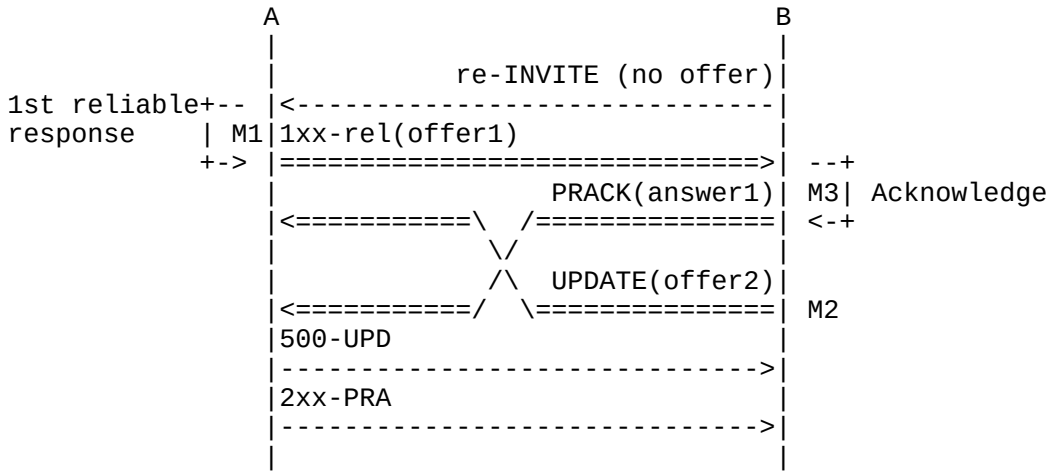

Figure 6: Avoidable Message Crossing Cases

To avoid the message crossing condition shown in Figure 6, UA B should not send this UPDATE request until a PRACK transaction associated with an offer/answer has been completed. If UA A encounters this message crossing condition, it should reject this UPDATE request with a 500 response.

The situation when a reliable provisional response to an INVITE request crosses UPDATE request is shown in Figure 7.

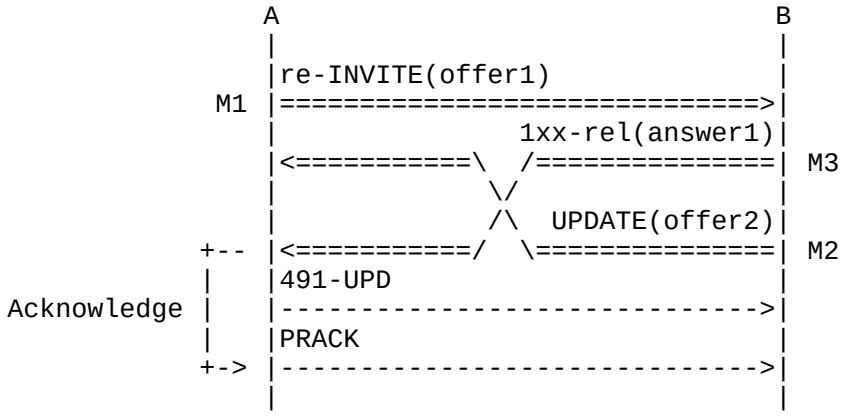

Figure 7: Avoidable Message Crossing Cases

To avoid the message crossing condition shown in Figure 7, UA B should not send this UPDATE request until a PRACK transaction associated with an offer/answer has been completed. If UA A encounters this message crossing condition, it should reject this UPDATE request with a 491 response.

The situation when a 2xx response to an INVITE request crosses UPDATE request is shown in Figure 8.

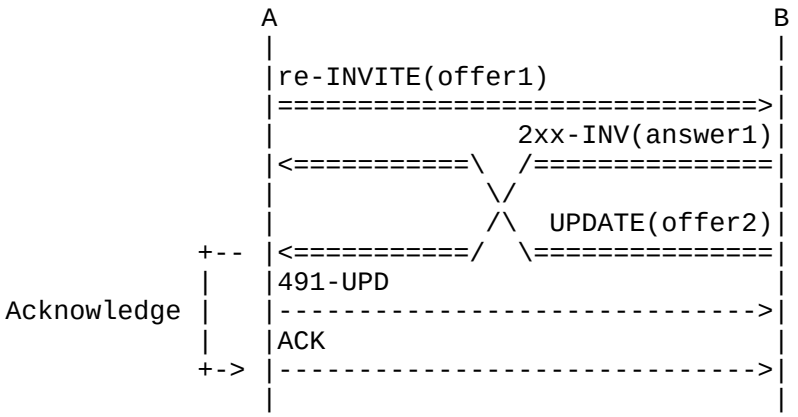

Figure 8: Avoidable Message Crossing Cases

This is a true glare. To avoid the message crossing condition shown in Figure 8, UA B should not send the UPDATE request until it has received an ACK request. But there is no problem even if UA B sends it. If UA A encounters this message crossing condition, it should reject this UPDATE request with a 491 response.

The situation when a response to an UPDATE request crosses a PRACK request is shown in Figure 9.

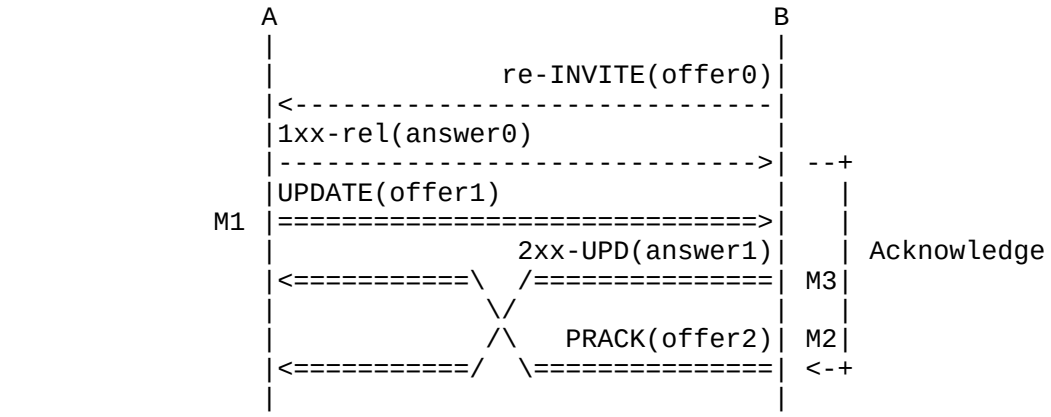

Figure 9: Avoidable Message Crossing Case

To avoid the message crossing condition shown in Figure 9, UA A should not send this UPDATE request until a PRACK transaction associated with an offer/answer has been completed. If UA B encounters this message crossing condition, it should reject this UPDATE request with a 491 response.

Table 3 summarizes this section. Each action is described in Section 4.3.

M1 (offer1)	M3 (answer1)	M2 (offer2)	Action of A	Action of B	Figure
UPDATE	2xx-UPD	UPDATE	UAS-UcU	-	
		INVITE	UAS-UcI		
		1xx-INV	UAC-UI, UAC-IU	UAS-UsI UAS-IsU	4,5
		2xx-INV			
		PRACK (*)	UAC-IU	UAS-IcU	9
PRACK	2xx-PRA	UPDATE	UAS-IcU		
2xx-INV	ACK	UPDATE	UAS-IsU	-	
		INVITE	UAS-IsI		
1xx-rel	PRACK	UPDATE	UAS-IsU	UAC-IU	6
INVITE	1xx-rel	UPDATE (*)	UAS-IcU	-	7
	2xx-INV	UPDATE (*)			8

(*) invalid sequences if INVITE request is an initial one

Table 3: Offer/Answer Crossing Message Sequences

4.2. Glare Case Handling

When both ends in a dialog send a new offer at nearly the same time, as described in Figure 10, a UA may receive a new offer before it receives the answer to the offer it sent. This case is usually called a 'glare' case.

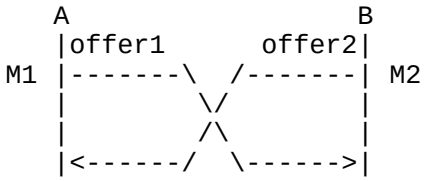

Figure 10: Glare Case

When offer2 is in an UPDATE request or (re-)INVITE request, it must be rejected with a 491 or 500 response.

There is a variant of Figure 7. When offer2 is in a PRACK request (within the current rules, only possible if offer1 is in an UPDATE request), as shown in Figure 11, UA A has a dilemma.

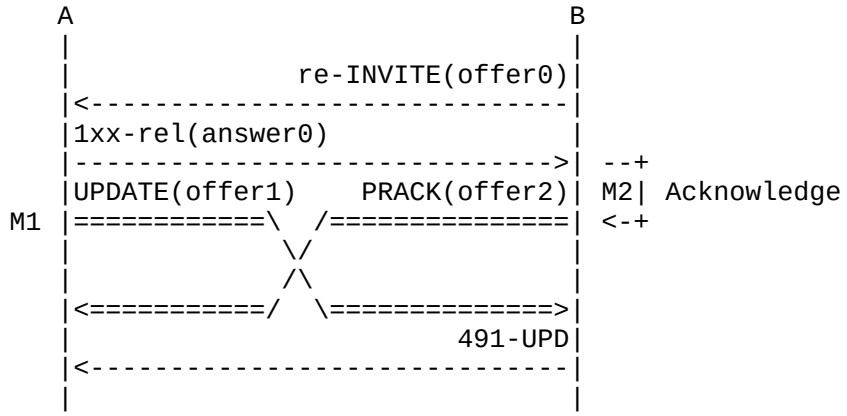

Figure 11: Avoidable Glare Case

All PRACKs are supposed to be accepted with a 200 response, yet there is no way to indicate the problem with a 200 response. At best, it could proceed on the assumption that the UPDATE will be rejected with a 491. To avoid the glare condition shown in Figure 11, UA A should not send this UPDATE request until a PRACK transaction associated with an offer/answer has been completed. If UA B encounters this glare condition, it should reject this UPDATE request with a 491 response.

Glare can also occur when offer2 is in a 1xx or 2xx response. This is a variant of Figure 5, as shown in Figure 12.

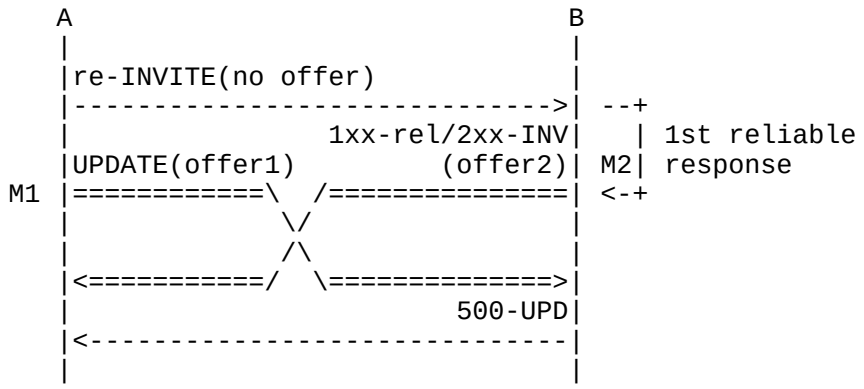

Figure 12: Avoidable Glare Case

To avoid the glare condition shown in Figure 12, UA A should not send this UPDATE request until an ACK or a PRACK transaction associated with an offer/answer has been completed. If UA B encounters this glare condition, it should reject this UPDATE request with a 500 response.

There is a variant of Figure 4, as shown in Figure 13.

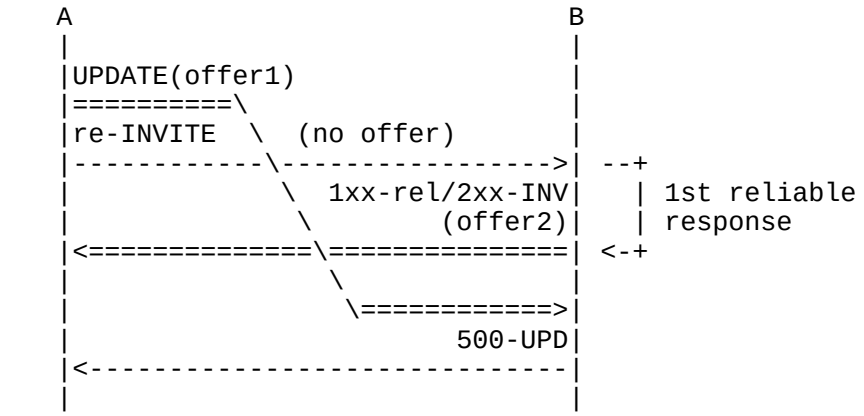

Figure 13: Avoidable Glare Case

To avoid the glare condition shown in Figure 13, UA A should not send this re-INVITE request until an UPDATE transaction has been completed. If UA B encounters this glare condition, it should reject this UPDATE request with a 500 response.

Table 4 summarizes this section. Each action is described in Section 4.3.

offer1 M1	offer2 M2	Action of A	Action of B	Figure
re-INVITE	re-INVITE	UAS-IcI	UAS-IcI	
	UPDATE	UAS-IcU	UAS-UcI	
	UPDATE	UAS-UcU	UAS-UcU	
UPDATE	1xx-rel	UAC-IU, UAC-UI	UAS-IsU	12,13
	2xx-INV			
	PRACK (*)	UAC-IU	UAS-IcU	11

(*) invalid sequences if INVITE request is an initial one

Table 4: Offer/Answer Glare Message Sequences

4.3. Interworking of UPDATE and Re-INVITE

Almost all exceptional cases are caused by an interworking of UPDATE and re-INVITE. The interworking is described in Section 5 of [RFC3311]. And UAC behavior sending an UPDATE is described in Section 5.1 of [RFC3311]. There are two concerns in this section:

1. It seems to describe different rules for each of initial INVITE and re-INVITE. But there is no particular reason why the rules are separated. The lack of restrictions for sending a re-INVITE request cause a lot of problems shown in Section 4.1.

2. It seems to describe that a UA may send an UPDATE request after sending or receiving a PRACK request. But it should be "after PRACK transaction is completed by 2xx response", because it causes the message-crossing case shown in Figure 6.

Since it is assumed that the language in this section itself is non-normative and is justified as a corollary of [RFC3261], we interpret it as follows:

UAC-II: While an INVITE transaction is incomplete or ACK transaction associated with an offer/answer is incomplete, a UA must not send another INVITE request.

UAC-UU: While an UPDATE transaction is incomplete, a UA must not send another UPDATE request.

UAC-UI: While an UPDATE transaction is incomplete, a UA should not send a re-INVITE request.

UAC-IU: While an INVITE transaction is incomplete, and an ACK or a PRACK transaction associated with an offer/answer is incomplete, a UA should not send an UPDATE request.

When a 2xx response to an INVITE includes an offer, the ACK transaction is considered to be associated with an offer/answer.

When a reliable provisional response to an INVITE includes an offer or an answer, the PRACK transaction is considered to be associated with an offer/answer.

UAS behavior receiving an UPDATE is described in Section 5.2 of [RFC3311]. There are two concerns in this section:

1. There is no description about the interworking of an UPDATE request and an INVITE request without an offer.

2. There is no description about the interworking of an UPDATE request and reliable response to an INVITE with an offer.

We interpret this section as follows:

UAS-IcI: While an INVITE client transaction is incomplete or ACK transaction associated with an offer/answer is incomplete, a UA must reject another INVITE request with a 491 response.

UAS-IsI: While an INVITE server transaction is incomplete or ACK transaction associated with an offer/answer is incomplete, a UA must reject another INVITE request with a 500 response.

UAS-UcU: While an UPDATE client transaction is incomplete, a UA must reject another UPDATE request with a 491 response.

UAS-UsU: While an UPDATE server transaction is incomplete, a UA must reject another UPDATE request with a 500 response.

UAS-UcI: While an UPDATE client transaction is incomplete, a UA should reject a re-INVITE request with a 491 response.

UAS-UsI: While an UPDATE server transaction is incomplete, a UA should reject a re-INVITE request with a 500 response.

UAS-IcU: While an INVITE client transaction is incomplete, and an ACK or a PRACK transaction associated with an offer/answer is incomplete, a UA should reject an UPDATE request with a 491 response.

UAS-IsU: While an INVITE server transaction is incomplete, and an ACK or a PRACK transaction associated with an offer/answer is incomplete, a UA should reject an UPDATE request with a 500 response.

These rules are shown in following figures.

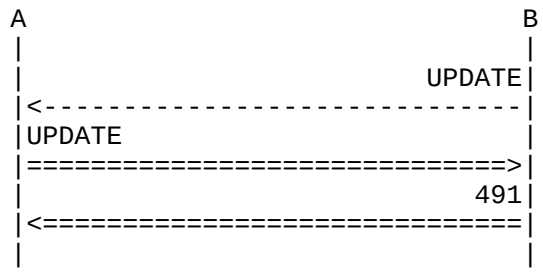

Figure 14: Example of UAC-UU and UAS-UcU

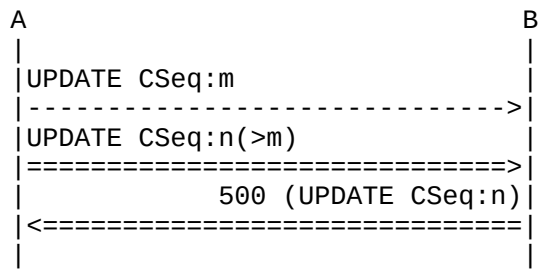

Figure 15: Example of UAC-UU and UAS-UsU

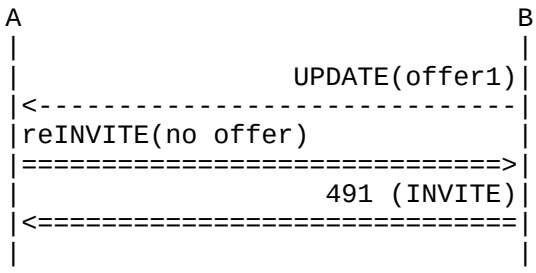

Figure 16: Example of UAC-UI and UAS-UcI

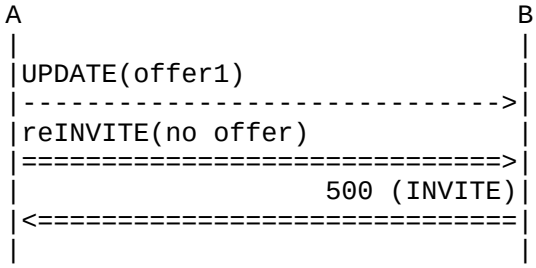

Figure 17: Example of UAC-UU and UAS-UsI

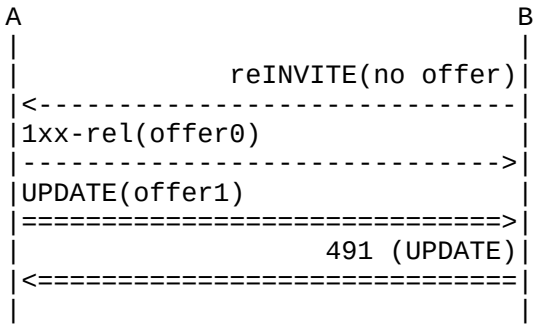

Figure 18: Example of UAC-IU and UAS-IcU

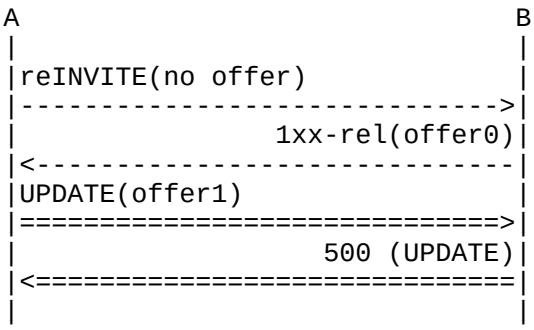

Figure 19: Example of UAC-IU and UAS-IsU

In addition, it is assumed that the UPDATE request in this section includes an offer. The interworking of a re-INVITE and an UPDATE without an offer is out of scope for this document.

5. Content of Offers and Answers

While [RFC3264] and [RFC3312] give some guidance, questions remain about exactly what should be included in an offer or answer. This is especially a problem when the common "hold" feature has been activated, and when there is the potential for a multimedia call.

Details of behavior depend on the capabilities and state of the User Agent. The kinds of recommendations that can be made are limited by the model of device capabilities and state that is presumed to exist.

This section focuses on a few key aspects of offers and answers that have been identified as troublesome, and will consider other aspects to be out of scope. This section considers:

- choice of supported media types and formats to include and exclude
- hold and resume of media

The following are out of scope for this document:

- NAT traversal and Interactive Connectivity Establishment (ICE)
- specific codecs and their parameters
- the negotiation of secure media streams
- grouping of media streams
- preconditions

5.1. General Principle for Constructing Offers and Answers

A UA should send an offer that indicates what it, and its user, are interested in using/doing at that time, without regard for what the other party in the call may have indicated previously. This is the case even when the offer is sent in response to an INVITE or re-INVITE that contains no offer. (However, in the case of re-INVITE, the constraints of [RFC3261] and [RFC3264] must be observed.)

A UA should send an answer that includes as close an approximation to what the UA and its user are interested in doing at that time, while remaining consistent with the offer/answer rules of [RFC3264] and other RFCs.

> NOTE: "at that time" is important. The device may permit the user to configure which supported media are to be used by default.

In some cases, a UA may not have direct knowledge of what it is interested in doing at a particular time. If it is an intermediary, it may be able to delegate the decision. In the worst case, it may apply a default, such as assuming it wants to use all of its capabilities.

5.2. Choice of Media Types and Formats to Include and Exclude

5.2.1. Sending an Initial INVITE with Offer

When a UAC sends an initial INVITE with an offer, it has complete freedom to choose which media type(s) and media format(s) (payload types in the case of RTP) it should include in the offer.

The media types may be all or a subset of the media the UAC is capable of supporting, with the particular subset being determined by the design and configuration (e.g., via [RFC6080]) of the UAC combined with input from the user interface of the UAC.

The media formats may be all or a subset of the media formats the UAC is capable of supporting for the corresponding media type, with the particular subset being determined by the design and configuration of the UAC combined with input from the user interface of the UAC.

Including all supported media formats will maximize the possibility that the other party will have a supported format in common. But including many can result in an unacceptably large SDP body.

5.2.2. Responding with an Offer When the Initial INVITE Has No Offer

When a UAS has received an initial INVITE without an offer, it must include an offer in the first reliable response to the INVITE. It has largely the same options as when sending an initial INVITE with an offer, but there are some differences. The choice may be governed by both static (default) selections of media types as well as dynamic selections made by a user via interaction with the device while it is alerting.

> NOTE: The offer may be sent in a reliable provisional response, before the user of the device has been alerted and had an opportunity to select media options for the call. In this case, the UAS cannot include any call-specific options from the user of the device. If there is a possibility that the user of the device will wish to change what is offered before answering the call, then special care should be taken. If PRACK and UPDATE are supported by caller and callee then an initial offer can be sent reliably, and changed with an UPDATE if the user desires a change. If PRACK and UPDATE are not supported, then the initial offer cannot be changed until the call is fully established. In that case, the offer in a 200 response for the initial INVITE should include only the media types and formats believed to be acceptable to the user.

5.2.3. Answering an Initial INVITE with Offer

When a UAS receives an initial INVITE with an offer, what media lines the answer may contain is constrained by [RFC3264]. The answer must contain the same number of "m=" lines as the offer, and they must contain the same media types. Each media line may be accepted, by including a non-zero port number, or rejected by including a zero port number in the answer. The media lines that are accepted should typically be those with types and formats the UAS would have included if it were the offerer.

The media formats the answer may contain are constrained by [RFC3264]. For each accepted "m=" line in the answer, there must be at least one media format in common with the corresponding "m=" line of the offer. The UAS may also include other media formats it is able to support at this time. Doing so establishes an asymmetric media format situation, where these "other" media formats may only be sent from the offerer to the answerer. This asymmetric media situation is also limited because it cannot be sustained if there is a subsequent offer/answer exchange in the opposite direction. Also, there is limited value in including these other media formats because there is no assurance that the offerer will be able to use them.

If the UAS does not wish to indicate support for any of the media types in a particular media line of the offer it must reject the corresponding media line, by setting the port number to zero.

When the UAS wishes to reject all of the media lines in the offer, it may send a 488 failure response. Alternatively, it may send a reliable non-failure response including all media lines with port numbers set to zero.

5.2.4. Answering When the Initial INVITE Had No Offer

When a UAC has sent an initial INVITE without an offer, and then receives a response with the first offer, it should answer in the same way as a UAS receiving an initial INVITE with an offer.

Because the offer arrives in a response to the INVITE, the UAC cannot reject the message containing the offer. If the UAC wishes to reject the entire offer, it must send a PRACK or ACK request including all the media lines with ports set to zero. Then, if it does not wish to continue the session, it may send a CANCEL or BYE request to terminate the dialog.

5.2.5. Subsequent Offers and Answers

The guidelines above (Sections 5.1 and 5.2.1 through Section 5.2.4) apply, but constraints in [RFC3264] must also be followed. The following are of particular note because they have proven troublesome:

- The number of "m=" lines may not be reduced in a subsequent offer. Previously rejected media streams must remain, or be reused to offer the same or a different stream. (Section 6 of [RFC3264].)

- In the "o=" line, only the version number may change, and if it changes, it must increment by one from the one previously sent as an offer or answer. (Section 8 of [RFC3264].) If it doesn't change, then the entire SDP body must be identical to what was previously sent as an offer or answer. Changing the "o=" line, except version number value, during the session is an error case. The behavior when receiving such a non-compliant offer/answer SDP body is implementation dependent. If a UA needs to negotiate a 'new' SDP session, it should use the INVITE/Replaces method.

- In the case of RTP, the mapping from a particular dynamic payload type number to a particular codec within that media stream ("m=" line) must not change for the duration of the session. (Section 8.3.2 of [RFC3264].)

NOTE: This may be impossible for a back-to-back user agent (B2BUA) to follow in some cases (e.g., 3PCC transfer) if it does not terminate media.

When the new offer is sent in response to an offerless (re-)INVITE, it should be constructed according to the General Principle for Constructing Offers and Answers (Section 5.1): all codecs the UA is currently willing and able to use should be included, not just the ones that were negotiated by previous offer/answer exchanges. The same is true for media types -- so if UA A initially offered audio and video to UA B, and they end up with only audio, and UA B sends an offerless (re-)INVITE to UA A, A's resulting offer should most likely re-attempt video, by reusing the zeroed "m=" line used previously.

NOTE: The behavior above is recommended, but it is not always achievable, for example, in some interworking scenarios. Or, the offerer may simply not have enough resources to offer "everything" at that point. Even if the UAS is not able to offer any other SDP that the one currently being used, it should not reject the re-INVITE. Instead, it should generate an offer with the currently used SDP with "o=" line unchanged.

5.3. Hold and Resume of Media

[RFC3264] specifies (using non-normative language) that "hold" should be indicated in an established session by sending a new offer containing "a=sendonly" attribute for each media stream to be held. An answerer is then to respond with "a=recvonly" attribute to acknowledge that the hold request has been understood.

Note that the use of sendonly/recvonly is not limited to hold. These may be used for other reasons, such as devices that are only capable of sending or receiving. So receiving an offer with "a=sendonly" attribute must not be treated as a certain indication that the offerer has placed the media stream on hold.

This model is based on an assumption that the UA initiating the hold will want to play Music on Hold, which is not always the case. A UA may, if desired, initiate hold by offering "a=inactive" attribute if it does not intend to transmit any media while in hold status.

The rules of [RFC3264] constrain what may be in an answer when the offer contains "sendonly", "recvonly", or "inactive" in an "a=" line. But they do not constrain what must be in a subsequent offer. The "General Principle for Constructing Offers and Answers" (Section 5.1) is important here. The initiation of "hold" is a local action. It should reflect the desired state of the UA. It then affects what the UA includes in offers and answers until the local state is reset.

The receipt of an offer containing "a=sendonly" attribute or "a=inactive" attribute and the sending of a compatible answer should not change the desired state of the recipient. However, a UA that has been "placed on hold" may itself desire to initiate its own hold status, based on local input.

If UA2 has previously been "placed on hold" by UA1, via receipt of "a=sendonly" attribute, then it may initiate its own hold by sending a new offer containing "a=sendonly" attribute to UA1. Upon receipt of that, UA1 will answer with "a=inactive" attribute because that is the only valid answer that reflects its desire not to receive media.

> NOTE: Section 8.4 of [RFC3264] contains a conflicting recommendation that the offer contain "a=inactive" attribute in this case. We interpret that recommendation to be non-normative. The use of "a=sendonly" attribute in this case will never produce a worse outcome, and can produce a better outcome in useful cases.

Once in this state, to resume a two-way exchange of media, each side must reset its local hold status. If UA1 is first to go off hold, it will then send an offer with "a=sendrecv" attribute. The UA2 will respond with its desired state of "a=sendonly" attribute because that is a permitted response. When UA2 desires to also resume, it will send an offer with "a=sendrecv" attribute. In this case, because UA1 has the same desire it will respond with "a=sendrecv" attribute. In the same case, when UA2 receives the offer with "a=sendrecv" attribute, if it has decided it wants to reset its local hold but has not yet signaled the intent, it may send "a=sendrecv" attribute in the answer.

If UA2 has been "placed on hold" by UA1 via receipt of "a=inactive" attribute, and subsequently wants to initiate its own hold, also using "a=inactive" attribute, it need not send a new offer, since the only valid response is "a=inactive" attribute and that is already in effect. However, its local desired state will now be either "inactive" or "a=sendonly" attribute. This affects what it will send in future offers and answers.

If a UA has occasion to send another offer in the session, without any desire to change the hold status (e.g., in response to a re-INVITE without an offer, or when sending a re-INVITE to refresh the session timer), it should follow the "General Principle for Constructing Offers and Answers" (Section 5.1). If it previously initiated a "hold" by sending "a=sendonly" attribute or "a=inactive" attribute, then it should offer that again. If it had not previously initiated "hold", then it should offer "a=sendrecv" attribute, even

if it had previously been forced to answer something else. Without this behavior it is possible to get "stuck on hold" in some cases, especially when a 3pcc is involved.

5.4. Behavior on Receiving SDP with c=0.0.0.0

[RFC3264] requires that an agent be capable of receiving SDP with a connection address of 0.0.0.0, in which case it means that neither RTP nor RTCP should be sent to the peer.

If a UA generates an answer to the offer received with "c=IN IP4 0.0.0.0", the direction attribute of the accepted media stream in the answer must still be based on direction attribute of the offered stream and rules specified in [RFC3264] to form the direction "a=" line in the answer. There is no clear rule about the use of "c=IN IP4 0.0.0.0" in the answer; it may be used or "c=" line with a valid IP address may be used. RTP/RTCP will not be sent toward an address of 0.0.0.0 because it is an invalid address.

6. Security Considerations

This document clarifies ambiguities in the intended behavior of the two SIP User Agents engaged in a dialog. The primary specification of offer/answer behavior that is being clarified resides in [RFC3261] and [RFC3264], with extensions in [RFC3311], [RFC3312], and [RFC6141]. The focus of this document is on cases where ambiguities can result failed or degraded calls when there is no attacker. The clarifications exclude call flows that lead to difficulties, without legitimizing any formerly invalid call flows. Thus, the security considerations of the above mentioned documents continue to apply and need not be extended to handle any additional cases.

The offer/answer process can be disrupted in numerous ways by an attacker. SIP provides mechanisms to protect the offer/answer exchange from tampering by third parties. Of note is "Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)" [RFC4474], as well as Section 26.3.2, "Security Solutions", of [RFC3261].

7. Acknowledgements

The authors would like to thank Christer Holmberg, Rajeev Seth, Nataraju A B, Byron Campen, Jonathan Rosenberg, Gonzalo Camarillo, and Gao Yang for their thorough reviews and comments. Many of their suggestions and ideas have been incorporated 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.

[RFC3312] Camarillo, G., Marshall, W., and J. Rosenberg, "Integration of Resource Management and Session Initiation Protocol (SIP)", RFC 3312, October 2002.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

[RFC6141] Camarillo, G., Holmberg, C., and Y. Gao, "Re-INVITE and Target-Refresh Request Handling in the Session Initiation Protocol (SIP)", RFC 6141, March 2011.

8.2. Informative References

[RFC3725] Rosenberg, J., Peterson, J., Schulzrinne, H., and G. Camarillo, "Best Current Practices for Third Party Call Control (3pcc) in the Session Initiation Protocol (SIP)", BCP 85, RFC 3725, April 2004.

[RFC3959] Camarillo, G., "The Early Session Disposition Type for the Session Initiation Protocol (SIP)", RFC 3959, December 2004.

[RFC4474] Peterson, J. and C. Jennings, "Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)", RFC 4474, August 2006.

[RFC6080] Petrie, D. and S. Channabasappa, "A Framework for Session Initiation Protocol User Agent Profile Delivery", RFC 6080, March 2011.

Authors' Addresses

OKUMURA Shinji
Softfront
28-196, Noth9, West15, Chuo-ku
Sapporo, Hokkaido 060-0009
Japan

EMail: shinji.okumura@softfront.jp

Takuya Sawada
KDDI Corporation
3-10-10, Iidabashi, Chiyoda-ku
Tokyo
Japan

EMail: tu-sawada@kddi.com

Paul H. Kyzivat
Hudson, MA 01749
USA

EMail: pkyzivat@alum.mit.edu