Internet Engineering Task Force (IETF)
Request for Comments: 7088
Category: Informational
ISSN: 2070-1721

D. Worley
Ariadne

February 2014

Session Initiation Protocol Service Example -- Music on Hold

Abstract

"Music on hold" is one of the features of telephone systems that is most desired by buyers of business telephone systems. Music on hold means that when one party to a call has the call "on hold", that party's telephone provides an audio stream (often music) to be heard by the other party. Architectural features of SIP make it difficult to implement music on hold in a way that is fully standards-compliant. The implementation of music on hold described in this document is fully effective, is standards-compliant, and has a number of advantages over the methods previously documented. In particular, it is less likely to produce peculiar user interface effects and more likely to work in systems that perform authentication than the music-on-hold method described in Section 2.3 of RFC 5359.

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/rfc7088.

Copyright Notice

Table of Contents

1. Introduction

Within systems based on SIP [RFC3261], it is desirable to be able to provide features that are similar to those provided by traditional telephony systems. A frequently requested feature is "music on hold": with this feature, when one party to a call has the call "on hold", that party’s telephone provides an audio stream (often music) to be heard by the other party.

Architectural features of SIP make it difficult to implement music on hold in a way that is fully standards-compliant. The purpose of this document is to describe a method that is reasonably simple yet fully effective and standards-compliant. This method has significant advantages over other methods now in use, as described in Section 3.

All current methods of implementing music on hold interoperate with each other, in that the two user agents in a call can use different methods for implementing music on hold with the same functionality as if either of the methods was used by both user agents. Thus, there is no loss of functionality if different music-on-hold methods are used by different user agents within a telephone system or if a single user agent uses different methods within different calls or at different times within one call.

1.1. Requirements Language

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].

2. Technique

The essence of the technique is that when the executing user agent (UA) (the user’s UA) performs a re-INVITE of the remote UA (the other user’s UA) to establish the hold state, it provides no Session Description Protocol (SDP) [RFC4566] offer [RFC3264] [RFC6337], thus compelling the remote UA to provide an SDP offer. The executing UA then extracts the offer SDP from the remote UA’s 2xx response and uses that as the offer SDP in a new INVITE to the external media source. The external media source is thus directed to provide media directly to the remote UA. The media source’s answer SDP is returned to the remote UA in the ACK to the re-INVITE.

2.1. Placing a Call on Hold and Establishing an External Media Stream

1. The executing user instructs the executing UA to put the dialog on hold.

2. The executing UA sends a re-INVITE without SDP to the remote UA, which forces the remote UA to provide an SDP offer in its 2xx response. The Contact header of the re-INVITE includes the '+sip.rendering="no"' field parameter to indicate that it is putting the call on hold ([RFC4235], Section 5.2).

3. The remote UA sends a 2xx to the re-INVITE and includes an SDP offer giving its own listening address/port. If the remote UA understands the sip.rendering feature parameter, the offer may indicate that it will not send media by specifying the media directionalities as "recvonly" (the reverse of "on hold") or "inactive". But the remote UA may offer to send media.

4. The executing UA uses this offer to derive the offer SDP of an initial INVITE that it sends to the configured music-on-hold (MOH) source. The SDP in this request is largely copied from the SDP returned by the remote UA in the previous step, particularly regarding the provided listening address/port and payload type numbers. But the media directionalities are restricted to "recvonly" or "inactive" as appropriate. The executing UA may want or need to change the "o=" line. In addition, some "a=rtpmap" lines may need to be added to control the assignment of RTP payload type numbers (Section 2.8).

5. The MOH source sends a 2xx response to the INVITE, which contains an SDP answer that should include its media source address as its listening address/port. This SDP must necessarily specify "sendonly" or "inactive" as the directionality for all media streams [RFC3264].

   Although this address/port should receive no RTP, the specified port determines the port for receiving the RTP Control Protocol (RTCP) (and conventionally, for sending RTCP [RFC4961]).

   By convention, UAs use their declared RTP listening ports as their RTP source ports as well [RFC4961]. The answer SDP will reach the remote UA, thus informing it of the address/port from which the MOH media will come and presumably preventing the remote UA from ignoring the MOH media if the remote UA filters media packets based on the source address. This functionality requires the SDP answer to contain the sending address in the "c=" line, even though the MOH source does not receive RTP.

6. The executing UA sends this SDP answer as its SDP answer in the ACK for the re-INVITE to the remote UA. The "o=" line in the answer must be modified to be within the sequence of "o=" lines previously generated by the executing UA in the dialog. Any dynamic payload type number assignments that have been created in the answer must be recorded in the state of the original dialog.

7. Due to the sip.rendering feature parameter in the Contact header of the re-INVITE and the media directionality in the SDP answer contained in the ACK, the on-hold state of the dialog is established (at the executing end).

8. After this point, the MOH source generates RTP containing the music-on-hold media and sends it directly to the listening address/port of the remote UA. The executing UA maintains two dialogs (one to the remote UA, one to the MOH source) but does not see or handle the MOH RTP.

2.2. Taking a Call off Hold and Terminating the External Media Stream

1. The executing user instructs the executing UA to take the dialog off hold.

2. The executing UA sends a re-INVITE to the remote UA with SDP that requests to receive media. The Contact header of the re-INVITE does not include the '+sip.rendering="no"' field parameter. (It may contain a sip.rendering field parameter with value "yes" or "unknown", or it may omit the field parameter.) Thus, this re-INVITE removes the on-hold state of the dialog (at the executing end). (Note that the version in "o=" line of the offered SDP must account for the SDP versions that were passed through from the MOH source. Also note that any payload type numbers that were assigned in SDP provided by the MOH source must be respected.)

3. When the remote UA sends a 2xx response to the re-INVITE, the executing UA sends a BYE request in the dialog to the MOH source.

4. After this point, the MOH source does not generate RTP and ordinary RTP flow is reestablished in the original dialog.

2.3. Example Message Flow

This section shows a message flow that is an example of this technique. The scenario is as follows. Alice establishes a call with Bob. Bob then places the call on hold, with music on hold provided from an external source. Bob then takes the call off hold. In this scenario, Bob's user agent is the executing UA, while Alice's

UA is the remote UA. Note that this is just one possible message flow that illustrates this technique; numerous variations on these operations are allowed by the applicable standards.

```
   Alice             Bob       Music Source
```

Alice establishes the call:

```
     |                |              |
     |    INVITE F1   |              |
     |--------------->|              |
     | 180 Ringing F2 |              |
     |<---------------|              |
     |    200 OK F3   |              |
     |<---------------|              |
     |     ACK F4     |              |
     |--------------->|              |
     |       RTP      |              |
     |<==============>|              |
     |                |              |
```

Bob places Alice on hold, compelling Alice’s UA to provide SDP:

```
     |                |              |
     |   INVITE F5    |              |
     |   (no SDP)     |              |
     |<---------------|              |
     |   200 OK F6    |              |
     |   (SDP offer)  |              |
     |--------------->|              |
     |                |              |
```

Bob’s UA initiates music on hold:

```
     |                |              |
     |                |  INVITE F7   |
     |                |  (SDP offer, |
     |                |   rev. hold) |
     |                |------------->|
     |                | 200 OK F8    |
     |                | (SDP answer, |
     |                |  hold)       |
     |                |<-------------|
     |                |    ACK F9    |
     |                |------------->|
     |                |              |
```

Bob’s UA provides an SDP answer containing the address/port of Music Source:

```
     |                |              |
     | ACK F10        |              |
     | (SDP answer,   |              |
     |  hold)         |              |
     |<---------------|              |
     |    no RTP      |              |
     |<..............>|              |
     |     Music-on-hold RTP         |
     |<==============================|
     |                |              |
```

The music on hold is active.

Bob takes Alice off hold:

```
     |                |              |
     |  INVITE F11    |              |
     |  (SDP offer)   |              |
     |<---------------|              |
     |   200 OK F12   |              |
     |   (SDP answer) |              |
     |--------------->|              |
     |     ACK F13    |              |
     |<---------------|              |
     |                |    BYE F14   |
     |                |------------->|
     |                |    200 F15   |
     |                |<-------------|
     |       RTP      |              |
     |<==============>|              |
     |                |              |
```

The normal media session between Alice and Bob is resumed.

```
/* Alice calls Bob. */

F1 INVITE Alice -> Bob

INVITE sips:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/TLS atlanta.example.com:5061
 ;branch=z9hG4bK74bf9
Max-Forwards: 70
From: Alice <sips:alice@atlanta.example.com>;tag=1234567
To: Bob <sips:bob@biloxi.example.com>
Call-ID: 12345600@atlanta.example.com
CSeq: 1 INVITE
Contact: <sips:a8342043f@atlanta.example.com;gr>
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY
Supported: replaces, gruu
Content-Type: application/sdp
Content-Length: [omitted]

v=0
o=alice 2890844526 2890844526 IN IP4 atlanta.example.com
s=
c=IN IP4 atlanta.example.com
t=0 0
m=audio 49170 RTP/AVP 0
a=rtpmap:0 PCMU/8000

F2 180 Ringing Bob -> Alice

SIP/2.0 180 Ringing
Via: SIP/2.0/TLS atlanta.example.com:5061
 ;branch=z9hG4bK74bf9
 ;received=192.0.2.103
From: Alice <sips:alice@atlanta.example.com>;tag=1234567
To: Bob <sips:bob@biloxi.example.com>;tag=23431
Call-ID: 12345600@atlanta.example.com
CSeq: 1 INVITE
Contact: <sips:bob@biloxi.example.com>
Content-Length: 0
```

F3 200 OK Bob -> Alice

```
SIP/2.0 200 OK
Via: SIP/2.0/TLS atlanta.example.com:5061
 ;branch=z9hG4bK74bf9
 ;received=192.0.2.103
From: Alice <sips:alice@atlanta.example.com>;tag=1234567
To: Bob <sips:bob@biloxi.example.com>;tag=23431
Call-ID: 12345600@atlanta.example.com
CSeq: 1 INVITE
Contact: <sips:bob@biloxi.example.com>
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY
Supported: replaces
Content-Type: application/sdp
Content-Length: [omitted]

v=0
o=bob 2890844527 2890844527 IN IP4 biloxi.example.com
s=
c=IN IP4 biloxi.example.com
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

F4 ACK Alice -> Bob

```
ACK sips:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/TLS atlanta.example.com:5061
 ;branch=z9hG4bK74bfd
Max-Forwards: 70
From: Alice <sips:alice@atlanta.example.com>;tag=1234567
To: Bob <sips:bob@biloxi.example.com>;tag=23431
Call-ID: 12345600@atlanta.example.com
CSeq: 1 ACK
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY
Supported: replaces
Content-Length: 0
```

```
   /* Bob places Alice on hold. */

   /* The re-INVITE contains no SDP, thus compelling Alice’s UA
      to provide an offer. */

   F5 INVITE Bob -> Alice

   INVITE sips:a8342043f@atlanta.example.com;gr SIP/2.0
   Via: SIP/2.0/TLS biloxi.example.com:5061
    ;branch=z9hG4bK874bk
   To: Alice <sips:alice@atlanta.example.com>;tag=1234567
   From: Bob <sips:bob@biloxi.example.com>;tag=23431
   Call-ID: 12345600@atlanta.example.com
   CSeq: 712 INVITE
   Contact: <sips:bob@biloxi.example.com>;+sip.rendering="no"
   Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY
   Supported: replaces
   Content-Length: 0

   /* Alice’s UA provides an SDP offer.
      Since it does not know that it is being put on hold,
      the offer is the same as the original offer and describes
      bidirectional media. */

   F6 200 OK Alice -> Bob

   SIP/2.0 200 OK
   Via: SIP/2.0/TLS biloxi.example.com:5061
    ;branch=z9hG4bK874bk
    ;received=192.0.2.105
   To: Alice <sips:alice@atlanta.example.com>;tag=1234567
   From: Bob <sips:bob@biloxi.example.com>;tag=23431
   Call-ID: 12345600@atlanta.example.com
   CSeq: 712 INVITE
   Contact: <sips:a8342043f@atlanta.example.com;gr>
   Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY
   Supported: replaces, gruu
   Content-Type: application/sdp
   Content-Length: [omitted]

   v=0
   o=alice 2890844526 2890844526 IN IP4 atlanta.example.com
   s=
   c=IN IP4 atlanta.example.com
   t=0 0
   m=audio 49170 RTP/AVP 0
   a=rtpmap:0 PCMU/8000
   a=active
   /* Bob’s UA initiates music on hold. */

   /* This INVITE contains Alice’s offer, but with the media
      direction set to "reverse hold", receive-only. */

   F7 INVITE Bob -> Music Source

   INVITE sips:music@source.example.com SIP/2.0
   Via: SIP/2.0/TLS biloxi.example.com:5061
    ;branch=z9hG4bKnashds9
   Max-Forwards: 70
   From: Bob <sips:bob@biloxi.example.com>;tag=02134
   To: Music Source <sips:music@source.example.com>
   Call-ID: 4802029847@biloxi.example.com
   CSeq: 1 INVITE
   Contact: <sips:bob@biloxi.example.com>
   Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY
   Supported: replaces, gruu
   Content-Type: application/sdp
   Content-Length: [omitted]

   v=0
   o=bob 2890844534 2890844534 IN IP4 atlanta.example.com
   s=
   c=IN IP4 atlanta.example.com
   t=0 0
   m=audio 49170 RTP/AVP 0
   a=rtpmap:0 PCMU/8000
   a=recvonly
```

F8 200 OK Music Source -> Bob

```
SIP/2.0 200 OK
Via: SIP/2.0/TLS biloxi.example.com:5061
 ;branch=z9hG4bKnashds9
 ;received=192.0.2.105
From: Bob <sips:bob@biloxi.example.com>;tag=02134
To: Music Source <sips:music@source.example.com>;tag=56323
Call-ID: 4802029847@biloxi.example.com
Contact: <sips:music@source.example.com>;automaton
     ;+sip.byeless;+sip.rendering="no"
CSeq: 1 INVITE
Content-Length: [omitted]

v=0
o=MusicSource 2890844576 2890844576 IN IP4 source.example.com
s=
c=IN IP4 source.example.com
t=0 0
m=audio 49170 RTP/AVP 0
a=rtpmap:0 PCMU/8000
a=sendonly
```

F9 ACK Bob -> Music Source

```
ACK sips:music@source.example.com SIP/2.0
Via: SIP/2.0/TLS source.example.com:5061
 ;branch=z9hG4bK74bT6
From: Bob <sips:bob@biloxi.example.com>;tag=02134
To: Music Source <sips:music@source.example.com>;tag=56323
Max-Forwards: 70
Call-ID: 4802029847@biloxi.example.com
CSeq: 1 ACK
Content-Length: 0
```

```
/* Bob’s UA now sends the ACK that completes the re-INVITE
   to Alice and completes the SDP offer/answer.
   The ACK contains the SDP received from Music Source and thus
   contains the address/port from which Music Source will send media,
   and implies the address/port that Music
   Source will use to send/receive RTCP. */
```

F10 ACK Bob -> Alice

```
ACK sips:a8342043f@atlanta.example.com;gr SIP/2.0
Via: SIP/2.0/TLS biloxi.example.com:5061
 ;branch=z9hG4bKq874b
To: Alice <sips:alice@atlanta.example.com>;tag=1234567
From: Bob <sips:bob@biloxi.example.com>;tag=23431
Call-ID: 12345600@atlanta.example.com
CSeq: 712 ACK
Contact: <sips:bob@biloxi.example.com>;+sip.rendering="no"
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY
Supported: replaces
Content-Length: [omitted]

v=0
o=bob 2890844527 2890844528 IN IP4 biloxi.example.com
s=
c=IN IP4 source.example.com
t=0 0
m=audio 49170 RTP/AVP 0
a=rtpmap:0 PCMU/8000
a=sendonly
```

/* Bob picks up the call by sending a re-INVITE to Alice. */

F11 INVITE Bob -> Alice

```
INVITE sips:a8342043f@atlanta.example.com;gr SIP/2.0
Via: SIP/2.0/TLS biloxi.example.com:5061
 ;branch=z9hG4bK874bk
To: Alice <sips:alice@atlanta.example.com>;tag=1234567
From: Bob <sips:bob@biloxi.example.com>;tag=23431
Call-ID: 12345600@atlanta.example.com
CSeq: 713 INVITE
Contact: <sips:bob@biloxi.example.com>
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY
Supported: replaces
Content-Type: application/sdp
Content-Length: [omitted]

v=0
o=bob 2890844527 2890844529 IN IP4 biloxi.example.com
s=
c=IN IP4 biloxi.example.com
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

F12 200 OK Alice -> Bob

```
SIP/2.0 200 OK
Via: SIP/2.0/TLS biloxi.example.com:5061
 ;branch=z9hG4bK874bk
 ;received=192.0.2.105
To: Alice <sips:alice@atlanta.example.com>;tag=1234567
From: Bob <sips:bob@biloxi.example.com>;tag=23431
Call-ID: 12345600@atlanta.example.com
CSeq: 713 INVITE
Contact: <sips:a8342043f@atlanta.example.com;gr>
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY
Supported: replaces, gruu
Content-Type: application/sdp
Content-Length: [omitted]

v=0
o=alice 2890844526 2890844527 IN IP4 atlanta.example.com
s=
c=IN IP4 atlanta.example.com
t=0 0
m=audio 49170 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

F13 ACK Bob -> Alice

```
ACK sips:a8342043f@atlanta.example.com;gr SIP/2.0
Via: SIP/2.0/TLS biloxi.example.com:5061
 ;branch=z9hG4bKq874b
To: Alice <sips:alice@atlanta.example.com>;tag=1234567
From: Bob <sips:bob@biloxi.example.com>;tag=23431
Call-ID: 12345600@atlanta.example.com
CSeq: 713 ACK
Contact: <sips:bob@biloxi.example.com>
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY
Supported: replaces
Content-Length: 0
```

```
   F14 BYE Bob -> Music Source

   BYE sips:music@source.example.com SIP/2.0
   Via: SIP/2.0/TLS biloxi.example.com:5061
    ;branch=z9hG4bK74rf
   Max-Forwards: 70
   From: Bob <sips:bob@biloxi.example.com>;tag=02134
   To: Music Source <sips:music@source.example.com>;tag=56323
   Call-ID: 4802029847@biloxi.example.com
   CSeq: 2 BYE
   Contact: <sips:bob@biloxi.example.com>
   Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY
   Supported: replaces, gruu
   Content-Length: [omitted]


   F15 200 OK Music Source -> Bob

   SIP/2.0 200 OK
   Via: SIP/2.0/TLS atlanta.example.com:5061
    ;branch=z9hG4bK74rf
    ;received=192.0.2.103
   From: Bob <sips:bob@biloxi.example.com>;tag=02134
   To: Music Source <sips:music@source.example.com>;tag=56323
   Call-ID: 4802029847@biloxi.example.com
   Contact: <sips:music@source.example.com>;automaton
        ;+sip.byeless;+sip.rendering="no"
   CSeq: 2 BYE
   Content-Length: 0

   /* Normal media session between Alice and Bob is resumed. */
```

2.4. Receiving Re-INVITE and UPDATE from the Remote UA

While the call is on hold, the remote UA can send a request to modify the SDP or the feature parameters of its Contact header. This can be done with either an INVITE or UPDATE method, both of which have much the same effect in regard to MOH.

A common reason for a re-INVITE is when the remote UA desires to put the dialog on hold on its end. And because of the need to support this case, an implementation must process INVITEs and UPDATEs during the on-hold state as described below.

The executing UA handles these requests by echoing requests and responses: an incoming request from the remote UA causes the executing UA to send a similar request to the MOH source, and an incoming response from the MOH source causes the executing UA to send a similar response to the remote UA. In all cases, SDP offers or answers that are received are added as bodies to the stimulated request or response to the other UA.

The passed-through SDP will usually need its "o=" line modified. The directionality attributes may need to be restricted by changing "active" to "recvonly" and "sendonly" to "inactive", as the executing UA will not render media from the remote UA. (If all passed-through directionality attributes are "inactive", the optimization described in Section 2.10 may be applied.) In regard to payload type numbers, since the mapping has already been established within the MOH dialog, "a=rtpmap" lines need not be added.

2.5. Receiving INVITE with Replaces

The executing UA must be prepared to receive an INVITE request with a Replaces header that specifies the dialog with the remote UA. If the executing UA wants to create this new dialog in the on-hold state, it creates a new dialog with the MOH source to obtain MOH. The executing UA negotiates the SDP within the dialog created by the INVITE with Replaces by passing the offer through to the new MOH dialog (if the INVITE contains an offer) or by creating the new MOH dialog with an offerless INVITE (if the INVITE does not contain an offer).

Continuing the example of Section 2.3, the executing UA receives an INVITE with Replaces that contains an offer:

```
   Alice             Bob       Music Source          Carol

   (For example, Alice has called Carol and initiates an attended
   transfer by sending a REFER to Carol, causing Carol to send an
   INVITE with Replaces to Bob.)

   Bob receives INVITE with Replaces from Carol:

     |                |              |                 |
     |                |              | INVITE/Replaces |
     |                |              | From: Carol     |
     |                |              | To: Bob         |
     |                |              | (SDP offer)     |
     |                |<-------------------------------|
     |                | INVITE       |                 |
     |                | From: Bob    |                 |
     |                | To: Music Source               |
     |                | (SDP offer,  |                 |
     |                |  rev. hold)  |                 |
     |                |------------->|                 |
     |                | 200 OK       |                 |
     |                | From: Bob    |                 |
     |                | To: Music Source               |
     |                | (SDP answer, |                 |
     |                |  hold)       |                 |
     |                |<-------------|                 |
     |                | ACK          |                 |
     |                | From: Bob    |                 |
     |                | To: Music Source               |
     |                |------------->|                 |
     |                |              | 200 OK          |
     |                |              | From: Carol     |
     |                |              | To: Bob         |
     |                |              | (SDP answer,    |
     |                |              |  hold)          |
     |                |------------------------------->|
     |                |              | ACK             |
     |                |              | From: Carol     |
     |                |              | To: Bob         |
     |                |<-------------------------------|
     |                |              | Music-on-hold RTP
     |                |              |================>|
     |                |              |                 |
```

Bob terminates the previous dialog with Alice:

```
     |                |              |                 |
     | BYE            |              |                 |
     | From: Bob      |              |                 |
     | To: Alice      |              |                 |
     |<---------------|              |                 |
     | 200 OK         |              |                 |
     | From: Bob      |              |                 |
     | To: Alice      |              |                 |
     |--------------->|              |                 |
     |                |              |                 |
```

Bob terminates the MOH dialog for the dialog with Alice:

```
     |                |              |                 |
     |                | BYE          |                 |
     |                | From: Bob    |                 |
     |                | To: Music Source               |
     |                |------------->|                 |
     |                | 200 OK       |                 |
     |                | From: Music Source             |
     |                | To: Bob      |                 |
     |                |<-------------|                 |
     |                |              |                 |
```

The new session continues on hold, between Bob and Carol.

2.6. Receiving REFER from the Remote UA

The executing UA must be prepared to receive a REFER request within the dialog with the remote UA. The SDP within the dialog created by the REFER is negotiated by sending an offerless INVITE (or offerless re-INVITE) to the MOH source to obtain an offer and then using that offer in the INVITE to the refer target.

Similar processing is used for an out-of-dialog REFER whose Target-Dialog header refers to the dialog with the remote UA.

Continuing the example of Section 2.3, the executing UA receives an INVITE with Replaces that contains an offer:

```
   Alice             Bob       Music Source          Carol

   (For example, Alice initiates an unattended transfer of the call to
   Carol by sending a REFER to Bob.)

   Bob receives REFER from Alice:

     |                |              |                 |
     | REFER          |              |                 |
     | From: Bob      |              |                 |
     | To: Alice      |              |                 |
     | Refer-To: Carol|              |                 |
     |--------------->|              |                 |
     |                | re-INVITE    |                 |
     |                | From: Bob    |                 |
     |                | To: Music Source               |
     |                | (no SDP)     |                 |
     |                |------------->|                 |
     |                | 200 OK       |                 |
     |                | From: Bob    |                 |
     |                | To: Music Source               |
     |                | (SDP offer,  |                 |
     |                |  hold)       |                 |
     |                |<-------------|                 |
     |                |              | INVITE          |
     |                |              | From: Bob       |
     |                |              | To: Carol       |
     |                |              | (SDP offer,     |
     |                |              |  hold)          |
     |                |------------------------------->|
     |                |              | 200 OK          |
     |                |              | From: Bob       |
     |                |              | To: Carol       |
     |                |              | (SDP answer,    |
     |                |              |  rev. hold)     |
     |                |------------------------------->|
     |                | ACK          |                 |
     |                | From: Bob    |                 |
     |                | To: Music Source               |
     |                | (SDP answer, |                 |
     |                |  rev. hold)  |                 |
     |                |------------->|                 |
     |                |              | ACK             |
     |                |              | From: Bob       |
     |                |              | To: Carol       |
     |                |------------------------------->|
     |                |              | Music-on-hold RTP
     |                |              |================>|
     |                |              |                 |
```

Bob terminates the previous dialog with Alice:

```
     |                |              |                 |
     | BYE            |              |                 |
     | From: Bob      |              |                 |
     | To: Alice      |              |                 |
     |<---------------|              |                 |
     | 200 OK         |              |                 |
     | From: Bob      |              |                 |
     | To: Alice      |              |                 |
     |--------------->|              |                 |
     |                |              |                 |
```

2.7. Receiving Re-INVITE and UPDATE from the Music-on-Hold Source

It is possible for the MOH source to send a re-INVITE or UPDATE request, and the executing UA can support doing so in similar manner as requests from the remote UA. However, if the MOH source is within the same administrative domain as the executing UA, the executing UA may have knowledge that the MOH source will not (or need not) make such requests and so can respond to any such request with a failure response, avoiding the need to pass the request through. The 403 (Forbidden) response is suitable for this purpose because [RFC3261] specifies that this response indicates "the request SHOULD NOT be repeated".

However, in an environment in which Interactive Connectivity Establishment (ICE) [RFC5245] is supported, the MOH source may need to send requests as part of ICE negotiation with the remote UA. Hence, in environments that support ICE, the executing UA must be able to pass through requests from the MOH source as well as requests from the remote UA.

Again, as SDP is passed through, its "o=" line will need to be modified. In some cases, the directionality attributes will need to be restricted.

2.8. Handling Payload Type Numbers

2.8.1. Analysis

In this technique, the MOH source generates an SDP answer that the executing UA presents to the remote UA as an answer within the original dialog. In basic functionality, this presents no problem, because [RFC3264], Section 6.1 (at the very end) specifies that the payload type numbers used in either direction of RTP are the ones specified in the SDP sent by the recipient of the RTP. Thus, the MOH source will send RTP to the remote UA using the payload type numbers specified in the offer SDP it received (ultimately) from the remote UA.

But strict compliance to [RFC3264], Section 8.3.2 requires that payload type numbers used in SDP may only duplicate the payload type numbers used in any previous SDP sent in the same direction if the payload type numbers represent the same media format (codec) as they did previously. However, the MOH source has no knowledge of the payload type numbers previously used in the original dialog, and it may accidentally specify a different media format for a previously used payload type number in its answer (or in a subsequently generated INVITE or UPDATE). This would cause no problem with media decoding, as it cannot send any format that was not in the remote UA’s offer, but it would violate [RFC3264].

Strictly speaking, it is impossible to avoid this problem because the generator of a first answer in its dialog can choose the payload numbers independently of the payload numbers in the offer, and the MOH server believes that its answer is first in the dialog. Thus, the only absolute solution is to have the executing UA rewrite the SDP that passes through it to reassign payload type numbers, which would also require it to rewrite the payload type numbers in the RTP packets -- a very undesirable solution.

The difficulty solving this problem (and similar problems in other situations) argues that strict adherence should not be required to the rule that payload type numbers not be reused for different codecs.

If an implementation of this technique were to interact with a remote UA that requires strict compliance to [RFC3264], the remote UA might reject the SDP provided by the MOH server. (In Section 2.3, this SDP is in message F10.) As a result, the MOH session will not be established, and the call will remain in its initial state. Implementors that wish to avoid this situation need to implement the solution in Section 2.8.2.

2.8.2. Solution to the Problem

We can construct a technique that will strictly adhere to the payload type rule by exploiting a SHOULD-level requirement in [RFC3264], Section 6.1: "In the case of RTP, if a particular codec was referenced with a specific payload type number in the offer, that same payload type number SHOULD be used for that codec in the answer". Or rather, we exploit the "implied requirement" that if a specific payload number in the offer is used for a particular codec, then the answer should not use that payload number for a different codec. If the MOH source obeys this restriction, the executing UA can modify the offer SDP to "reserve" all payload type numbers that have ever been offered by the executing UA to prevent the MOH source from using them for different media formats.

When the executing UA is composing the INVITE to the MOH source, it compiles a list of all the (dynamically assigned) payload type numbers and associated media formats that have been used by it (or by MOH sources on its behalf) in the original dialog. (The executing UA must maintain a list of all previously used payload type numbers anyway, in order to comply with [RFC3264].)

Any payload type number that is present in the offer but has been used previously by the executing UA in the original dialog for a different media format is rewritten to describe a dummy media format. (One dummy media format name can be used for many payload type numbers as multiple payload type numbers can refer to the same media format.) A payload type number is added to describe the deleted media format, the number being either previously unused or previously used by the executing UA for that media format.

Any further payload type numbers that have been used by the executing UA in the original dialog but that are not mapped to a media format in the current offer are then mapped to a dummy media format.

The result is that the modified offer SDP:

1. offers the same set of media formats (ignoring dummies) as the original offer SDP (though possibly with different payload type numbers),

2. associates every payload type number either with a dummy media format or with the media format that the executing UA has previously used it for, and

3. provides a (real or dummy) media format for every payload type number that the executing UA has previously used.

These properties are sufficient to force an MOH server that obeys the implied requirement to generate an answer that is a correct answer to the original offer and is also compatible with previous SDP from the executing UA.

Note that any re-INVITEs from the remote UA that the executing UA passes through to the MOH server require similar modification, as payload type numbers that the MOH server receives in past offers are not absolutely reserved against its use (as they have not been sent in SDP by the MOH server) nor is there a SHOULD-level proscription against using them in the current answer (as they do not appear in the current offer).

This should provide an adequate solution to the problems with payload type numbers, as it will fail only if (1) the remote UA is particular that other UAs follow the rule about not redefining payload type numbers, and (2) the MOH server does not follow the implied requirement of [RFC3264], Section 6.1.

2.8.3. Example of the Solution

Let us show how this process works by modifying the example of Section 2.3 with this specific assignment of supported codecs:

Alice supports formats X and Y.

Bob supports formats X and Z.

Music Source supports formats Y and Z.

In this case, the SDP exchanges are:

F1 offers X and Y, F3 answers X and Z. (Only X can be used.)

F6 offers X and Y, but F7 offers X, Y, and a place-holder to block use of type 92.

F8/F10 answers Y.

The messages that are changed from Section 2.3 are:

```
F1 INVITE Alice -> Bob

INVITE sips:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/TLS atlanta.example.com:5061
 ;branch=z9hG4bK74bf9
Max-Forwards: 70
From: Alice <sips:alice@atlanta.example.com>;tag=1234567
To: Bob <sips:bob@biloxi.example.com>
Call-ID: 12345600@atlanta.example.com
CSeq: 1 INVITE
Contact: <sips:a8342043f@atlanta.example.com;gr>
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY
Supported: replaces, gruu
Content-Type: application/sdp
Content-Length: [omitted]

v=0
o=alice 2890844526 2890844526 IN IP4 atlanta.example.com
s=
c=IN IP4 atlanta.example.com
t=0 0
m=audio 49170 RTP/AVP 90 91
a=rtpmap:90 X/8000
a=rtpmap:91 Y/8000


F3 200 OK Bob -> Alice

SIP/2.0 200 OK
Via: SIP/2.0/TLS atlanta.example.com:5061
 ;branch=z9hG4bK74bf9
 ;received=192.0.2.103
From: Alice <sips:alice@atlanta.example.com>;tag=1234567
To: Bob <sips:bob@biloxi.example.com>;tag=23431
Call-ID: 12345600@atlanta.example.com
CSeq: 1 INVITE
Contact: <sips:bob@biloxi.example.com>
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY
Supported: replaces
Content-Type: application/sdp
Content-Length: [omitted]
```

```
    v=0
    o=bob 2890844527 2890844527 IN IP4 biloxi.example.com
    s=
    c=IN IP4 biloxi.example.com
    t=0 0
    m=audio 3456 RTP/AVP 90 92
    a=rtpmap:90 X/8000
    a=rtpmap:92 Z/8000
```

F6 200 OK Alice -> Bob

```
    SIP/2.0 200 OK
    Via: SIP/2.0/TLS biloxi.example.com:5061
     ;branch=z9hG4bK874bk
     ;received=192.0.2.105
    To: Alice <sips:alice@atlanta.example.com>;tag=1234567
    From: Bob <sips:bob@biloxi.example.com>;tag=23431
    Call-ID: 12345600@atlanta.example.com
    CSeq: 712 INVITE
    Contact: <sips:a8342043f@atlanta.example.com;gr>
    Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY
    Supported: replaces, gruu
    Content-Type: application/sdp
    Content-Length: [omitted]

    v=0
    o=alice 2890844526 2890844526 IN IP4 atlanta.example.com
    s=
    c=IN IP4 atlanta.example.com
    t=0 0
    m=audio 49170 RTP/AVP 90 91
    a=rtpmap:90 X/8000
    a=rtpmap:91 Y/8000
    a=active
```

F7 INVITE Bob -> Music Source

```
INVITE sips:music@source.example.com SIP/2.0
Via: SIP/2.0/TLS biloxi.example.com:5061
 ;branch=z9hG4bKnashds9
Max-Forwards: 70
From: Bob <sips:bob@biloxi.example.com>;tag=02134
To: Music Source <sips:music@source.example.com>
Call-ID: 4802029847@biloxi.example.com
CSeq: 1 INVITE
Contact: <sips:bob@biloxi.example.com>
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY
Supported: replaces, gruu
Content-Type: application/sdp
Content-Length: [omitted]

v=0
o=bob 2890844534 2890844534 IN IP4 atlanta.example.com
s=
c=IN IP4 atlanta.example.com
t=0 0
m=audio 49170 RTP/AVP 90 91 92
a=rtpmap:90 X/8000
a=rtpmap:91 Y/8000
a=rtpmap:92 x-reserved/8000
a=recvonly
```

F8 200 OK Music Source -> Bob

```
SIP/2.0 200 OK
Via: SIP/2.0/TLS biloxi.example.com:5061
 ;branch=z9hG4bKnashds9
 ;received=192.0.2.105
From: Bob <sips:bob@biloxi.example.com>;tag=02134
To: Music Source <sips:music@source.example.com>;tag=56323
Call-ID: 4802029847@biloxi.example.com
Contact: <sips:music@source.example.com>;automaton
     ;+sip.byeless;+sip.rendering="no"
CSeq: 1 INVITE
Content-Length: [omitted]
```

```
v=0
o=MusicSource 2890844576 2890844576 IN IP4 source.example.com
s=
c=IN IP4 source.example.com
t=0 0
m=audio 49170 RTP/AVP 91
a=rtpmap:91 Y/8000
a=sendonly
```

2.9. Dialog/Session Timers

The executing UA may discover that either the remote UA or the MOH source wishes to use dialog/session liveness timers [RFC4028]. Since the timers verify the liveness of dialogs, not sessions (despite the terminology of [RFC4028]), the executing UA can support the timers on each dialog (to the remote UA and to the MOH source) independently. (If the executing UA becomes obliged to initiate a refresh transaction, it must send an offerless UPDATE or re-INVITE, as if it sends an offer, the remote element has the opportunity to provide an answer that is different from its previous SDP, which could not easily be conveyed to the other remote element.)

2.10. When the Media Stream Directionality is "inactive"

The directionality of the media stream in the SDP offer in an INVITE or re-INVITE to the music source can be "inactive" if the SDP offer from the remote UA was "sendonly" or "inactive". Generally, this happens when the remote UA also has put the call on hold and provided a directionality of "sendonly". In this situation, the executing UA can omit establishing the dialog with the music source (or can terminate the existing dialog with the music source).

If the executing UA uses this optimization, it creates the SDP answer itself, with directionality "inactive" and using its own RTP/RTCP ports, and returns that answer to the remote UA.

The executing UA must be prepared for the remote UA to send a re-INVITE with directionality "active" or "recvonly", in which case the executing UA must initiate a dialog with the music source, as described above.

2.11. Multiple Media Streams

There may be multiple media streams (multiple "m=" lines) in any of the SDPs involved in the dialogs. As the SDPs are manipulated, each media description (each starting with an "m=" line) is manipulated as described above for a single media stream, largely independently of the manipulation of the other media streams. But there are some elaborations that the executing UA may implement to achieve specific effects.

If the executing UA desires to present only certain media types as on-hold media, when passing the offer SDP through, it can reject any particular media streams by setting the port number in the "m=" line to zero [RFC3264]. This ensures that the answer SDP will also have a rejection for that "m=" line.

If the executing UA wishes to provide its own on-hold media for a particular "m=" line, it can do so by providing the answer information for that "m=" line. The executing UA may decide to do this when the offer SDP is received (by modifying the "m=" line to rejected state when sending it to the music source) or upon receiving the answer from the music source and discovering that the "m=" line has been rejected.

The executing UA may not want to pass a rejected "m=" line from the music source to the remote UA (when the remote UA provided a non-rejected "m=" line) and may instead provide an answer with directionality "inactive" (and specifying its own RTP/RTCP ports).

3. Advantages

This technique for providing music on hold has advantages over other methods now in use, including:

1. The original dialog is not transferred to another UA, so the "remote endpoint URI" displayed by the remote endpoint’s user interface and dialog event package [RFC4235] does not change during the call, as contrasted to the method in [RFC5359], Section 2.3. This URI is usually displayed to the user as the name and number of the other party on the call, and it is desirable for it not to change to that of the MOH server.

2. Compared to [RFC5359], this method does not require use of an out-of-dialog REFER, which is not otherwise used much in SIP. Out-of-dialog REFERs may not be routed correctly, since neither the From nor Contact URI of the original dialog may route correctly to the remote UA. Also, out-of-dialog requests to UA URIs may not be handled correctly by authorization mechanisms.

3. The music-on-hold media are sent directly from the music-on-hold source to the remote UA, rather than being relayed through the executing UA. This reduces the computational load on the executing UA and can reduce the load on the network (by eliminating "hairpinning" of the media through the link serving the executing UA).

4. The remote UA sees, in the incoming SDP, the address/port that the MOH source will send MOH media from (assuming that the MOH source follows the convention of sending its media from its advertised media-listening address/port). Thus, the remote UA will render the MOH media even if it is filtering incoming media based on originating address as a media security measure.

5. The technique requires relatively simple manipulation of SDP; in particular, (1) it does not require a SIP element to modify unrelated SDP to be acceptable to be sent within an already established sequence of SDP (a problem with [SIP-SERV-EX], Section 2.3), and (2) it does not require converting an SDP answer into an SDP offer (which was a problem with the initial draft version of this document, as well as with [SIP-SERV-EX]).

4. Caveats

4.1. Offering All Available Media Formats

Unnecessary failures can happen if SDP offerers do not always offer all media formats that they support. Doing so is considered best practice ([RFC6337], Sections 5.1 and 5.3), but some SIP elements offer only formats that have already been in use in the dialog.

An example of how omitting media formats in an offer can lead to failure is as follows. Suppose that the UAs in Section 2.3 each support the following media formats:

Alice supports formats X and Y.

Bob supports formats X and Z.

Music Source supports formats Y and Z.

In this case, the SDP exchanges are:

1. Alice calls Bob:
   Alice offers X and Y (message F1).
   Bob answers X (F3).

2. Bob puts Alice on hold:
   Alice (via Bob) offers X and Y (F6 and F7).
   Music Source (via Bob) answers Y (F8 and F10).

3. Bob takes Alice off hold:
   Bob offers X and Z (F11).
   Alice answers X (F12).

Note that in exchange 2, if Alice assumes that because only format X is currently in use that she should offer only X, the exchange fails. In exchange 3, Bob offers formats X and Z, even though neither is in use at the time (because Bob is not involved in the media streams).

4.2. Handling Re-INVITES in a B2BUA

Many UAs provide MOH in the interval during which it is processing a blind transfer, between receiving the REFER and receiving the final response to the stimulated INVITE. This process involves switching the user’s interface between three media sources: (1) the session of the original dialog, (2) the session with the MOH server, and (3) the session of the new dialog. It also involves a number of race conditions that must be handled correctly. If the UA is a back-to-back user agent (B2BUA) whose "other side" is maintaining a single dialog with another UA, each switching of media sources potentially causes a re-INVITE transaction within the other-side dialog. Since re-INVITEs take time and must be sequenced correctly ([RFC3261], Section 14), such a B2BUA must allow the events on each side to be non-synchronous and must coordinate them correctly. Failing to do so will lead to "glare" errors (491 or 500), leaving the other-side UA not rendering the correct session.

5. Security Considerations

5.1. Network Security

Some mechanism outside the scope of this document must inform the executing UA of the MOH server that it should use. Care must be exercised in selecting the MOH server, because signaling information that is part of the original dialog will be transmitted along the path from the executing UA to the server. If the path between the executing UA and the server is not entirely contained within every network domain that contains the executing UA, the signaling between the UA and the server may be protected by different network security than is applied to the original dialog.

Care must also be exercised because media information that is part of the original dialog will be transmitted along the path between the remote UA and the server. If the path between the remote UA and the server does not pass through the same network domains as the path between the remote UA and the executing UA, the media between the UA and the server may be protected by different network security than is applied to the original dialog.

These requirements may be satisfied by selecting an MOH server that is in the same administrative and network domain as the executing UA and whose path to all external addresses is the same as the UA’s path to those addresses.

5.2. SIP (Signaling) Security

The executing UA and the MOH server will usually be within the same administrative domain, and the SIP signaling path between them will lie entirely within that domain. In this case, the administrator of the domain should configure the UA and server to apply to the dialog between them a level of security that is appropriate for the administrative domain.

If the executing UA and the MOH server are not within the same administrative domain, the SIP signaling between them should be at least as secure as the SIP signaling between the executing UA and the remote UA. Thus, the MOH server should support all of the SIP security facilities that are supported by the executing UA, and the executing UA should use in its dialog with the MOH server all SIP security facilities that are used in its dialog with the remote UA.

5.3. RTP (Media) Security

The RTP for the MOH media will pass directly between the MOH server and the remote UA and thus may pass outside the administrative domain of the executing UA. While it is uncommon for the contents of the MOH media to be sensitive (and the remote UA will not usually be generating RTP when it is on hold), the MOH RTP should be at least as secure as the RTP between the executing UA and the remote UA. In order to make this possible, the MOH server should support all of the RTP security facilities that are supported by the executing UA.

It is possible that the remote UA and the MOH server support an RTP security facility that the executing UA does not support and that it is desirable to use this facility for the MOH RTP. To enable doing so, the executing UA should pass the SDP between the remote UA and the MOH server completely, not omitting elements that it does not understand.

5.4. Media Filtering

Some UAs filter incoming RTP based on the address of origin as a media security measure, refusing to render the contents of RTP packets that originate from an address that is not shown in the remote SDP as an RTP destination address. The remote UA in the original dialog may use this form of media filtering, and if the executing UA does not update the SDP to inform the remote UA of the source address of the MOH media, the remote UA may not render the MOH media. Note that the executing UA has no means for detecting that the remote UA uses media filtering, so the executing UA must assume that any remote UA uses media filtering.

The technique described in this document ensures that any UA that should render MOH media will be informed of the source address of the media via the SDP that it receives. This allows such UAs to filter media without interfering with MOH operation.

6. Acknowledgments

The original version of this proposal was derived from Section 2.3 of [SIP-SERV-EX] and the similar implementation of MOH in the snom UA. Significant improvements to the sequence of operations, allowing improvements to the SDP handling, were suggested by Venkatesh [VENKATESH].

John Elwell [ELWELL] pointed out the need for the executing UA to pass through re-INVITEs/UPDATEs in order to allow ICE negotiation, suggested mentioning the role of RTCP listening ports, suggested the possibility of omitting the dialog to the music source if the directionality would be "inactive", and pointed out that if there are multiple media streams, the executing UA may want to select which streams receive MOH.

Paul Kyzivat [KYZIVAT-1] [KYZIVAT-2] pointed out the difficulties regarding reuse of payload type numbers and considerations that could be used to avoid those difficulties, leading to the writing of Section 2.8.

Paul Kyzivat suggested adding Section 4.1 showing why offerers should always include all supported formats.

M. Ranganathan pointed out the difficulties experienced by a B2BUA (Section 4.2) due to the multiple changes of media source.

Section 4.1 was significantly clarified based on advice from Attila Sipos [SIPOS].

The need to discuss dialog/session timers (Section 2.9) was pointed out by Rifaat Shekh-Yusef [SHEKH-YUSEF].

Robert Sparks clarified the purpose of the "Best Current Practice" status, leading to revising the intended status of this document to "Informational".

In his SecDir review, Stephen Kent pointed out that the Security Considerations should discuss the use of SIP and SDP security features by the MOH server.

Numerous improvements to the text were due to reviewers, including Rifaat Shekh-Yusef and Richard Barnes.

7. References

7.1. 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.

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

[RFC4028] Donovan, S. and J. Rosenberg, "Session Timers in the Session Initiation Protocol (SIP)", RFC 4028, April 2005.

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

7.2. Informative References

[RFC4235] Rosenberg, J., Schulzrinne, H., and R. Mahy, "An INVITE-Initiated Dialog Event Package for the Session Initiation Protocol (SIP)", RFC 4235, November 2005.

[RFC4961] Wing, D., "Symmetric RTP / RTP Control Protocol (RTCP)", BCP 131, RFC 4961, July 2007.

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

[RFC5359] Johnston, A., Sparks, R., Cunningham, C., Donovan, S., and K. Summers, "Session Initiation Protocol Service Examples", BCP 144, RFC 5359, October 2008.

[RFC6337] Okumura, S., Sawada, T., and P. Kyzivat, "Session Initiation Protocol (SIP) Usage of the Offer/Answer Model", RFC 6337, August 2011.

[ELWELL] Elwell, J., "Subject: [Sipping] RE: I-D Action:draft-worley-service-example-00.txt", message to the IETF Sipping mailing list, November 2007, <http://www1.ietf.org/mail-archive/web/sipping/current/msg14678.html>.

[KYZIVAT-1] Kyzivat, P., "Subject: Re: [Sipping] I-D ACTION:draft-ietf-sipping-service-examples-11.txt", message to the IETF Sipping mailing list, October 2006, <http://www1.ietf.org/mail-archive/web/sipping/current/msg12181.html>.

[KYZIVAT-2] Kyzivat, P., "Subject: [Sip-implementors] draft-worley-service-example-02", message to the sip-implementors mailing list, September 2008, <http://lists.cs.columbia.edu/pipermail/sip-implementors/2008-September/020394.html>.

[SHEKH-YUSEF] Shekh-Yusef, R., "Subject: [sipcore] draft-worley-service-example-03", message to the IETF Sipcore mailing list, July 2009, <http://www.ietf.org/mail-archive/web/sipcore/current/msg00580.html>.

[SIPOS] Sipos, A., "Subject: [Sip-implementors] draft-worley-service-example-02", message to the sip-implementors mailing list, March 2009, <http://lists.cs.columbia.edu/pipermail/sip-implementors/2009-March/021970.html>.

[SIP-SERV-EX] Johnston, A., Sparks, R., Cunningham, C., Donovan, S., and K. Summers, "Session Initiation Protocol Service Examples", Work in Progress, October 2006.

[VENKATESH] Venkatesh, "Subject: Re: [Sipping] I-D ACTION:draft-ietf-sipping-service-examples-11.txt", message to the IETF Sipping mailing list, October 2006, <http://www1.ietf.org/mail-archive/web/sipping/current/msg12180.html>.

Author's Address

Dale R. Worley
Ariadne Internet Services, Inc.
738 Main St.
Waltham, MA 02451
US

Phone: +1 781 647 9199
EMail: worley@ariadne.com