

SPLICES Working Group  
Internet-Draft  
Intended status: Informational  
Expires: December 27, 2011

G. Camarillo  
S. Loreto  
Ericsson  
R. Shekh-Yusef  
Avaya  
June 25, 2011

Disaggregated Media in the Session Initiation Protocol (SIP)  
draft-loreto-splices-disaggregated-media-02.txt

Abstract

Disaggregated media refers to the ability for a user to create a multimedia session combining different media streams, coming from different devices under his or her control, so that they are treated by the far end of the session as a single media session. This document lists several use cases that involve disaggregated media in SIP. Additionally, this document analyzes what types of disaggregated media can be implemented using existing protocol mechanisms, and the pros and cons of using each of those mechanisms. Finally, this document describes scenarios that are not covered by current mechanisms and proposes new IETF work to cover them.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 26, 2011.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

(<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
2. Use Cases . . . . .	3
2.1. Using Two Separate devices to Start a Conversation . . . . .	3
2.2. Showing a Pre-recorded Video During a Conversation . . . . .	4
2.3. Sending a File from a PC During a Conversation . . . . .	4
2.4. Including Live Video in a Conversation . . . . .	4
2.5. Including Remote Live Video in a Conversation . . . . .	5
2.6. Answering a call using Two Separate Devices . . . . .	5
2.7. Other possible use cases . . . . .	6
3. Existing Mechanisms to Implement Disaggregated Media . . . . .	6
3.1. Message Bus (Mbus) . . . . .	7
3.1.1. Mbus issues . . . . .	8
3.2. Megaco (H.248) . . . . .	8
3.2.1. Megaco issues . . . . .	9
3.3. Third Part Call Control (3pcc) . . . . .	9
3.3.1. 3pcc issues . . . . .	10
4. Distributed Call Control with Actions . . . . .	11
4.1. Answering an A/V call using Two Separate Devices example . . . . .	12
4.1.1. Discovery of Capabilities . . . . .	12
4.1.2. Answering Using PC . . . . .	13
4.1.3. Answering Using Desk Phone . . . . .	14
4.1.4. Departure of One Device . . . . .	15
5. Scenarios Not Covered by Existing Mechanisms . . . . .	17
6. Recommendations . . . . .	17
7. Security Considerations . . . . .	18
8. IANA Considerations . . . . .	18
9. Informational References . . . . .	18
Authors' Addresses . . . . .	18

## 1. Introduction

Disaggregated media refers to the ability for a user to create a multimedia session combining different media streams, coming from different devices under his or her control, so that they are treated by the far end of the session as a single media session.

The SIP specification [RFC3261] defines a multimedia session as "an exchange of data between an association of participants". SDP (Session Description Protocol) is the default session description format in SIP. The SDP (Session Description Protocol) specification [RFC4566] defines a multimedia session as "a set of multimedia senders and receivers and the data streams flowing from senders to receivers".

Generally, a given participant uses a single device to establish (or participate in) a given multimedia session. Consequently, the SIP signaling to manage the multimedia session and the actual media streams are typically co-located in the same device. In scenarios involving disaggregated media, a user wants to establish a single multimedia session combining different media streams coming from different devices under his or her control. This creates a need to coordinate the exchange of the those media streams within the media session.

The remainder of this document is organized as follows. Section 2 contains use cases where different media streams, coming from different devices, are combined to establish a multimedia session. Section 3 describes what types of disaggregated media can be implemented using existing protocol mechanisms, and the pros and cons of using each of those mechanisms. Section 4 describes scenarios that are not covered by current mechanisms and proposes new IETF work to cover them.

## 2. Use Cases

This section lists several use cases where users participate in a multimedia session using multiple devices. That is, either the user initiating the session uses several devices in parallel during the session, or the user receiving the session invitation uses several devices in parallel during the session, or both.

### 2.1. Using Two Separate devices to Start a Conversation

Laura is at her office. On her desk, she has a PC with a soft client and a desk phone. The PC has a low-quality built-in microphone and is connected to high-quality speakers.

Laura wants to establish a voice session with Bob using the desk phone as the microphone and the soft client as the speaker.

Laura wants Bob to treat the send-only audio stream from her deskphone and the receive-only audio stream from her softclient as part of the same communication space. That is, Laura wants Bob to treat both streams as belonging to the same multimedia session.

## 2.2. Showing a Pre-recorded Video During a Conversation

Bob has a voice-only phone and an IP-TV device. Laura has an integrated advanced multimedia phone with camera.

Bob is talking on his voice-only phone with Laura, who is on her multimedia phone.

Bob wants to show Laura part of the TV show he recorded last night. Bob interacts, using his voice-only phone, with his IP-TV device and sends a video stream to Laura's device.

Bob talks about the show with Laura on his voice-only phone while Laura watches the show.

Bob wants Laura to treat the video stream from his IP-TV device and the voice stream from his voice-only phone as part of the same communication space. That is, Bob wants Laura to treat both streams as belonging to the same multimedia session.

## 2.3. Sending a File from a PC During a Conversation

Bob has a voice-only phone and a PC with a soft client. Laura has an integrated advanced multimedia phone with support for file transfers.

Bob wants to send a file to Laura from his PC during his conversation with Laura on his voice-only phone.

Bob interacts, using his voice-only phone, with his PC and starts a file transfer session to Laura's multimedia phone.

Bob wants Laura to treat the file transfer from his PC and the voice stream from his voice-only phone as part of the same communication space. That is, Bob wants Laura to treat both streams as belonging to the same multimedia session.

## 2.4. Including Live Video in a Conversation

Bob has a voice-only phone and a PC which has a soft client, a Webcam, and a low-quality built-in microphone. Laura has an

integrated advanced multimedia phone with camera.

Bob wants to send a live video to Laura from his PC during his conversation with Laura.

Bob interacts, using his voice-only phone, with his PC and starts live video stream to Laura's multimedia phone.

Bob wants Laura to treat the live video stream from his PC and the voice stream from his voice-only phone as part of the same communication space. That is, Bob wants Laura to treat both streams as belonging to the same multimedia session.

#### 2.5. Including Remote Live Video in a Conversation

Bob, who is at his office, has a multimedia phone. At his summer cottage, Bob has a webcam-phone (e.g. a video-surveillance system). Laura has an integrated advanced multimedia phone with a camera.

During his conversation with Laura, Bob wants to show her the new summer cottage he just bought. Bob interacts, using his multimedia phone, with his webcam phone at this summer cottage and start live video stream to Laura's multimedia phone.

Bob wants Laura to treat the live video stream from his webcam-phone and the voice stream from his voice-only phone as part of the same communication space. That is, Bob wants Laura to treat both streams as belonging to the same multimedia session.

#### 2.6. Answering a call using Two Separate Devices

Laura has a PC with a softclient and a desk phone. Bob has an integrated advanced multimedia phone with camera.

Laura receives on her desk phone an incoming voice-video call from Bob.

Laura decides to answer Bob's session invitation by establishing a voice session with Bob using the desk phone and the video session using her multimedia phone. Two SIP dialogs need to be established: one between Bob's device and Laura's desk phone and one between Bob's device and Laura's multimedia phone.

Laura wants Bob to treat the audio stream from her deskphone and the video stream from her softclient as part of the same communication space. That is, Laura wants Bob to treat both streams as belonging to the same multimedia session.

## 2.7. Other possible use cases

This section just enumerates, for sake of completeness, other possible use cases, similar to the one elaborated in the previous sections.

A user wants to start or answer a call combining:

- o Voice and video streams from a deskphone and application sharing from a computer.
- o Voice stream from a deskphone and video stream to/from a TV attached to a set top box with a camera built in.
- o The User Interface (UI) for the call on a mobile phone and the audio streaming coming in/out of a speakerphone that is in the same room where he is.

## 3. Existing Mechanisms to Implement Disaggregated Media

Figure 1 shows the media flow in the most generic scenario where both the Caller and the Callee use disaggregated media, involving in the multimedia session different devices under their control.

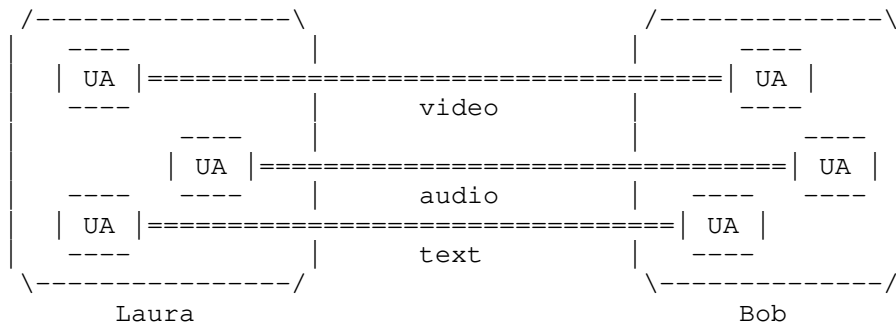


Figure 1: Media Flows in Disaggregated Media

All existing mechanisms to implement disaggregated media in SIP use a centralized approach whereby the far end of the session receives the same SIP signaling flow that it would receive if all the media streams came from a single device. This makes it transparent to the far end of the session the fact that the caller is using separate devices for different media.

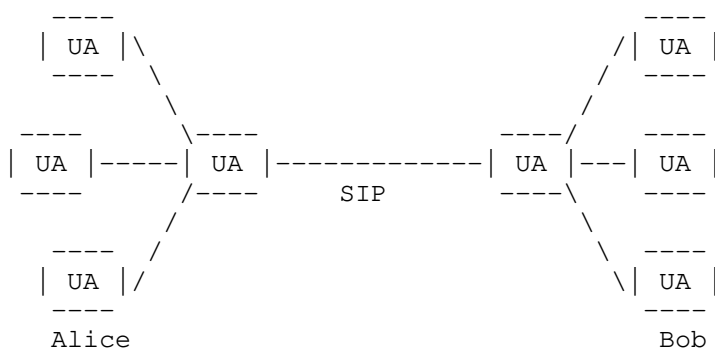


Figure 2: Centralized scenario

Figure 2 shows the generic signaling flow common to all centralized solutions, where a Central Node is able to manage all signaling messages needed to coordinate the different user's devices and hide from the far end of the session all the mechanisms used to distribute the media among different devices.

Section 3.1, Section 3.2 and Section 3.3 analyze how different existing mechanisms can be used to implement disaggregated media in a centralized way.

### 3.1. Message Bus (Mbus)

The Message Bus (Mbus) [RFC3259] is a light-weight local coordination protocol for developing component-based distributed applications. Mbus provides a simple and flexible message oriented communication channel for a group of components that may be distributed on multiple hosts in a local network. The transport services include useful features such as peer location, point-to-point and group communication and security.

In a disaggregated media scenario a user can use Mbus to coordinate the different devices under his control in a loosely coupled conference control model and so involve them in the call. The different devices can communicate with one another using Mbus messages, and then let only one device, a call control engine, to initiate, manage and terminate call control relations to other SIP endpoints. In this case the fact that the caller is using separate devices for different media becomes transparent to the callee.

Figure 3 shows an example of the relation between a call control engine in an Mbus enabled conferencing system.

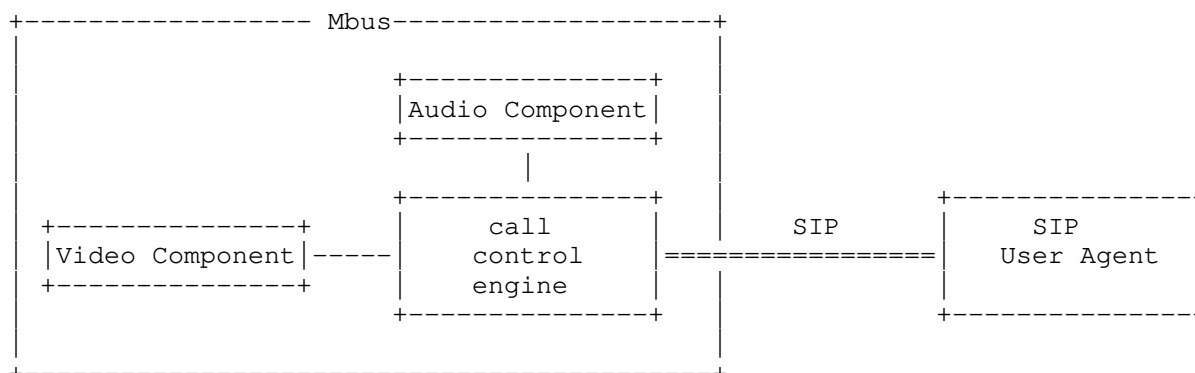


Figure 3: Mbus Architecture

#### 3.1.1. Mbus issues

The Mbus protocol introduces the following issues.

**Mbus support:** All the different user's devices need to support the Mbus protocol.

**Central point:** Because the call control engine has a complete control over the call, it needs to be involved during the whole duration of the session. It cannot leave the session before the whole session ends (unless it transfers the controller role to one of the other user's devices).

**Local network:** Mbus uses multicast with a limited scope for message transport. The multicast limits the coordination mechanism only to groups of devices that are connected to a local network. So Mbus can be used in a disaggregated media scenario only if all the different devices under the user control, or at least the one the user wants to involve in the communication, are attached to the same local network.

#### 3.2. Megaco (H.248)

The Megaco [RFC3525] protocol is used to establish, and terminate calls across terminations (end points) connected to Media Gateways (MGs). Megaco instructs Media Gateways (MG) to connect streams coming from outside a packet network on to a packet stream such as RTP. Master-MGC issues commands to send and receive media from addresses, generate tones, and to modify configuration. The architecture requires a Media Gateway Controller (MGC) controlling the Media Gateway(s). However it does not constitute a complete system. To establish communication between MGC(s) is necessary a session initiation protocol. SIP is the protocol normally used to



establish calls across domains or across MGCs.

Megaco can be used in a disaggregated media scenario to let one of the user's devices act as a Media Gateway Controller, coordinating all the other devices under the user's control, which in this case will act as Media Gateways. In this case the fact that the caller is using separate devices for different media becomes transparent to the callee.

### 3.2.1. Megaco issues

The Megaco protocol introduces the following issues.

Megaco support: All the different user's devices need to support the Megaco protocol.

Central point: Because the Media Gateway Controller has a complete control over the call, it needs to be involved for all the session time. It can not leave the session before the whole session ends.

### 3.3. Third Part Call Control (3pcc)

3pcc [RFC3725] allows one entity (called the Controller) to setup and manage a communications relationship between two or more other parties using SIP.

In a disaggregated media scenario, a user can use 3pcc mechanism only if at least one among the different devices under his control supports this mechanism and is able to become a Controller for the other in the call. In this case become transparent for the callee the fact that the caller is using separate devices for different media. In fact the Controller is a central point for the signaling on the caller side, and has complete control over the call, making everything in one dialog for the callee.

The call flow for disaggregated media using 3pcc is shown in Figure 4. It is based on Flow IV documented in Section 4.4 of [RFC3725] and recommended for calls to unknown entities, or to entities known to represent people. The flow requires some SDP manipulation by the Controller to convert offer2 to offer2' and offer2'', and then to convert answer2' and answer2'' to answer2.

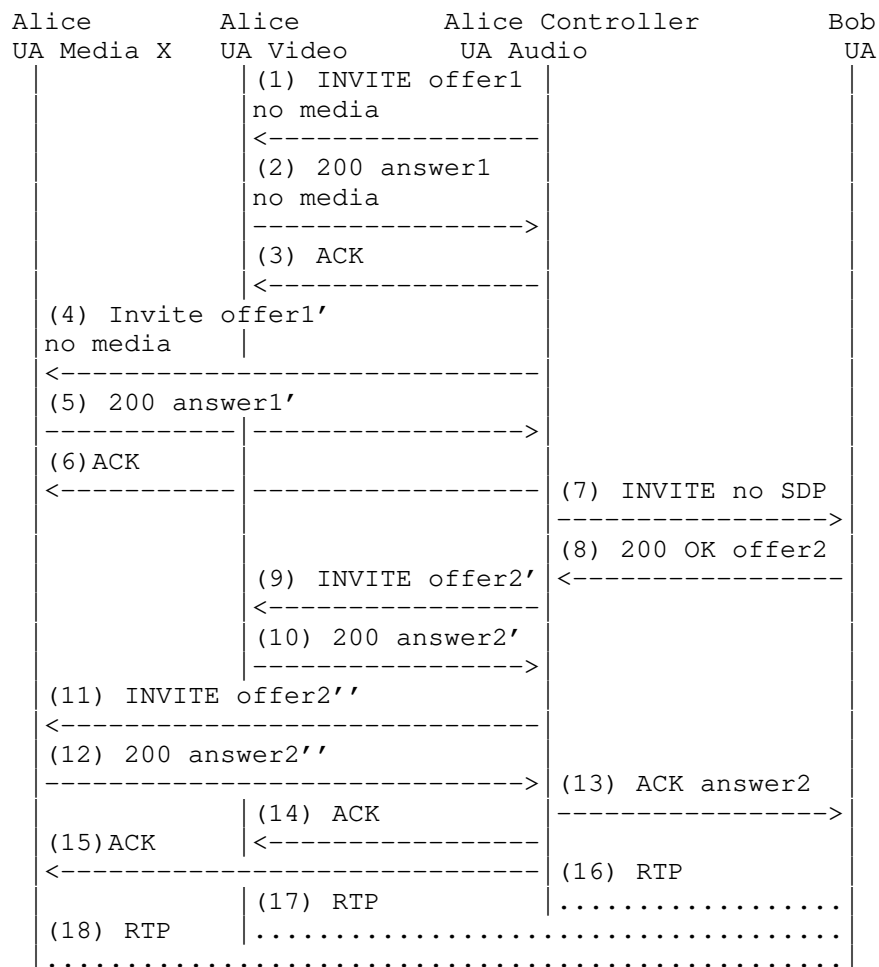


Figure 4: 3pcc call flow

### 3.3.1. 3pcc issues

The 3pcc mechanism introduces the following issues.

**Complexity:** Third party call control only uses protocol mechanism specified in [RFC3261]. However the usage of third party call control becomes more complex when aspects of the call utilize SIP extensions or optional features of SIP like resource reservation.

Central point: Because the Controller has a complete control over the call, it needs to be involved during the whole duration of the session. It cannot leave the session before the whole session ends (unless it transfers the controller role to one of the other user's devices).

User experience: 3ppc results in a suboptimal user experience because the slave phones are not aware that they are involved in a disaggregated media call scenario. Indeed, the slave phones behave as they were just involved in a normal call with the Controller. Moreover the slave phones will be alerted without any media having been established yet.

SDP manipulation: the Controller cannot "proxy" the SIP messages received from one of the parties. In many cases, it is required to modify the SDP exchanged between the participants in order to affect the changes.

#### 4. Distributed Call Control with Actions

The loosely coupled scenarios in this section describe the discovery of capabilities and interaction between the federated devices that allows these devices to present themselves to the remote party as one entity.

The federated devices utilize the Actions mechanism to allow a UA to invoke an action on another UA.

3PCC allows the Controller to setup and manage a communications relationship between two or more other parties using SIP. When 3PCC is coupled with the Actions mechanism, the Controller role becomes flexible enough to allow any of the federated parties to take that role at any time, depending on the specifics of the use case.

All federated devices are expected to be aware of each others capabilities and all available actions, and they are also expected to be aware of all the dialogs of each others by subscribing to the 'dialog' event package, and have the intelligence to utilize all this knowledge to provide solutions to a wide range of use cases, without requiring the remote party to change.

Because there is no one entity that takes the Controller role all the time, in the event of a sudden departure of one of the federated devices, one of the remaining devices can take control of the communication with the remote party and update it, based on the capabilities of the remaining federated parties. In the case of the departure of the Controller device and with multiple federated devices still alive, the rest of the federated devices can utilize the Actions mechanism to select the new Controller.

#### 4.1. Answering an A/V call using Two Separate Devices example

This section describes the example use case of answering an incoming A/V call using two separate devices, using the mechanism described in the section above. In the following example flows the Actions mechanism used is the one provided by the proposed new SIP INVOKE method. The use case has Alice with a PC with a Video SIP UA and Audio SIP Desk Phone while Bob has an integrated SIP Device with A/V.

##### 4.1.1. Discovery of Capabilities

Both Alice's devices subscribe to the 'reg' event package of each other, which allows each device to discover the capabilities of the other device based on the feature tags provided by each device. The Desk Phone knows that the PC supports Video, while the PC knows that the Desk Phone only supports audio. The two devices also subscribe to the 'dialog' event package of each other.

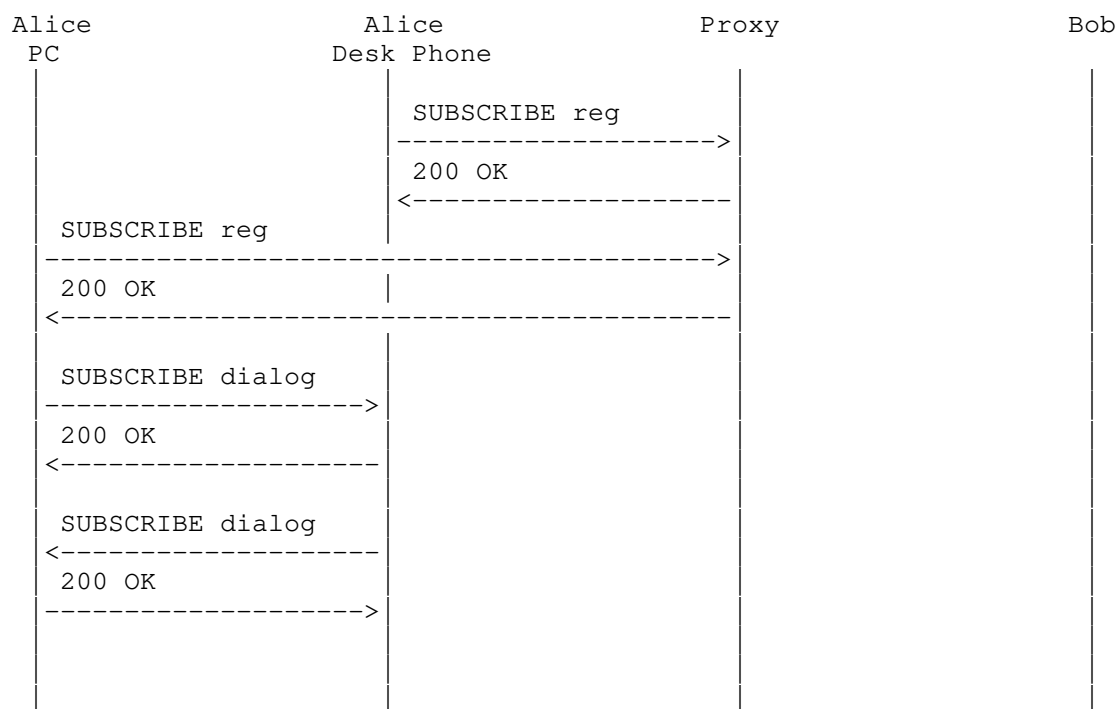
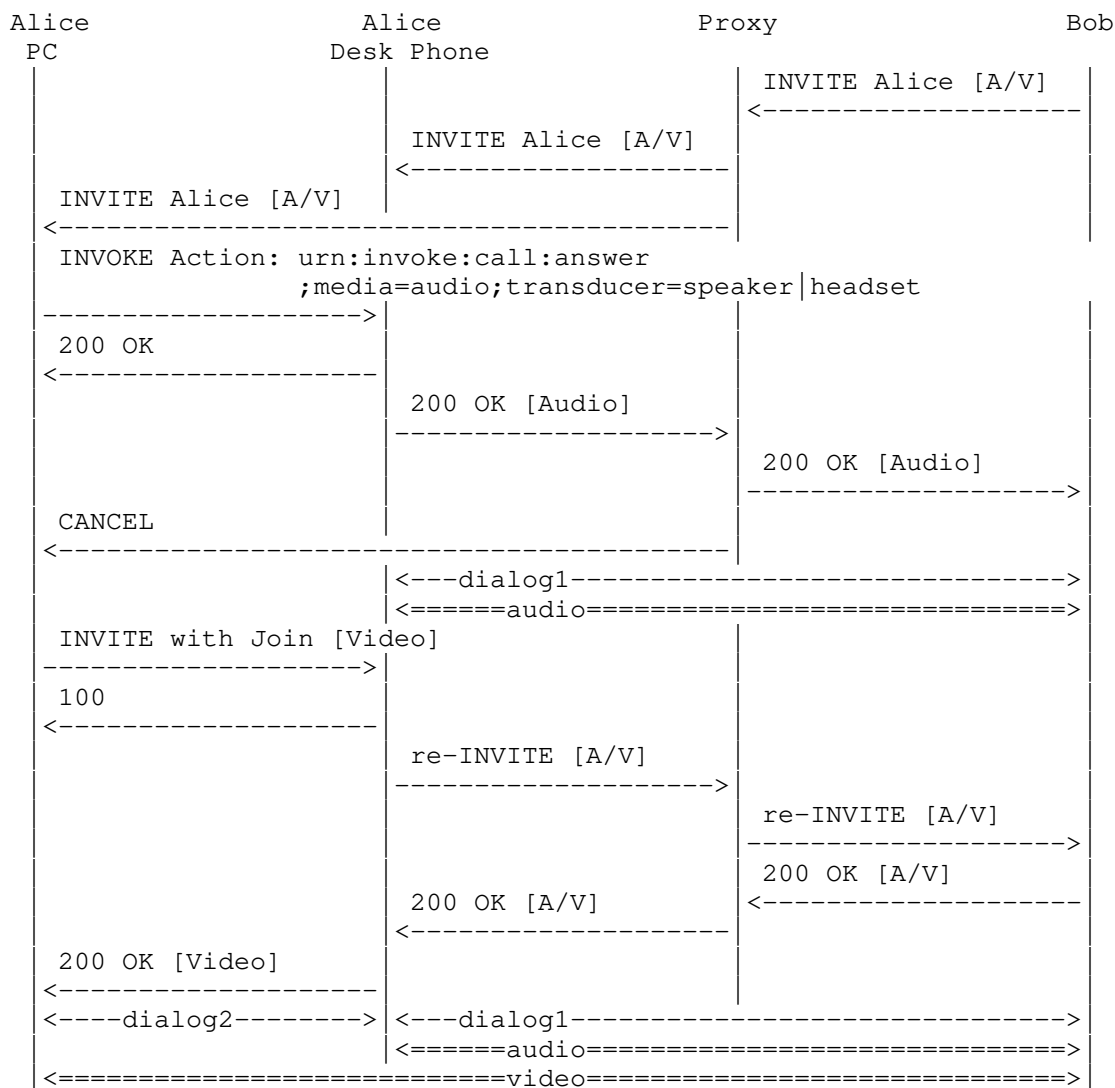


Figure 5

## 4.1.2. Answering Using PC

Bob makes an A/V call to Alice, which rings both of Alice's devices. Alice answered the call using her PC. The PC instructs the phone to answer the call with audio only, and then the PC joins the existing call with video, by sending an INVITE with Join to the phone, which will then take the Video offer from the PC and add its Audio offer and send a re-INVITE with an SDP with one audio media line and one video media line.



## 4.1.3. Answering Using Desk Phone

The phone answers the audio call and then instructs the PC to initiate a video call towards the phone, which the phone uses to join the existing audio call.

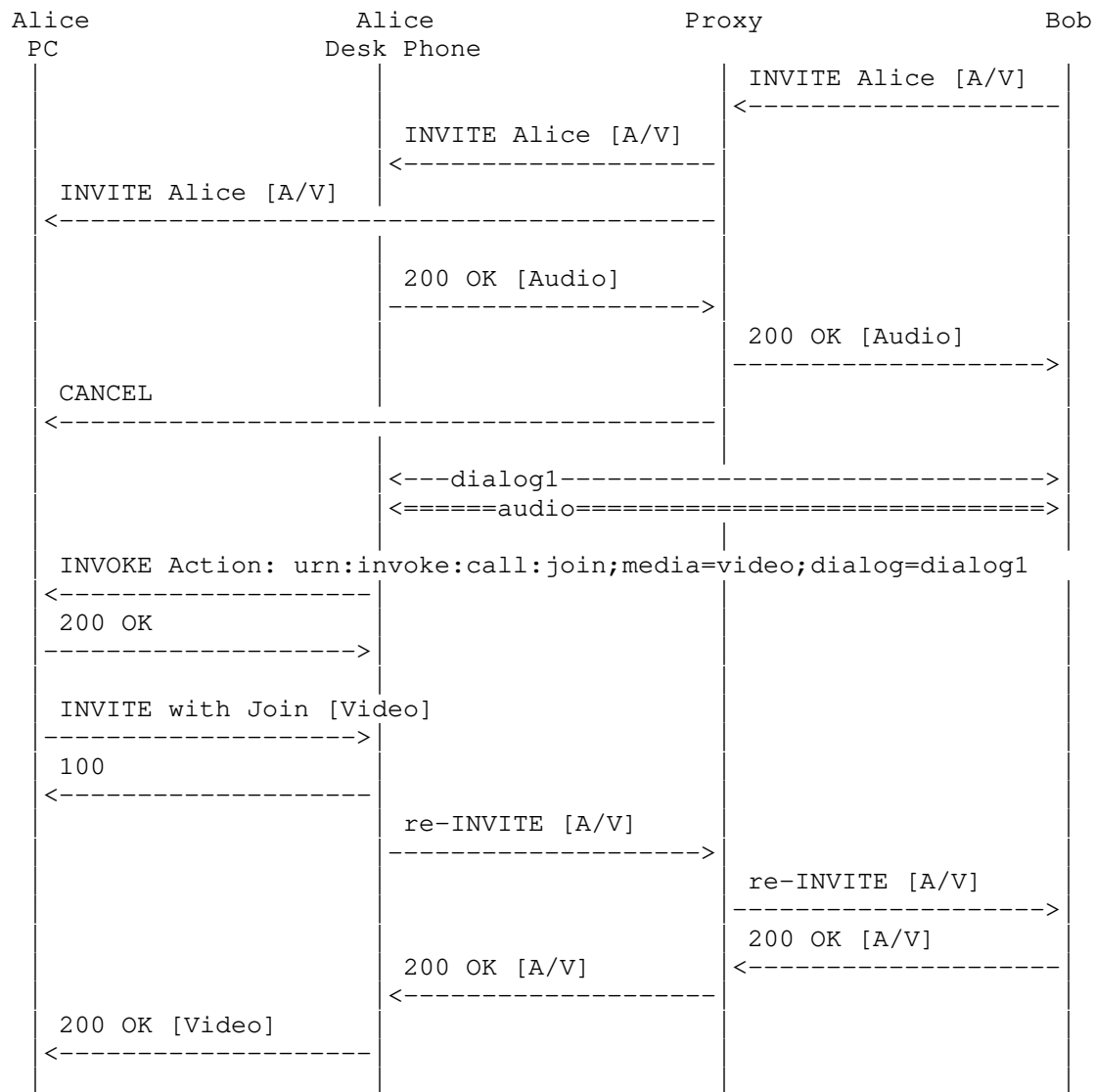


Figure 7

## 4.1.4. Departure of One Device

In the case of a sudden departure of one of the devices, the other device can take control over the communication with the remote party.

In the flow below, Alice's devices have a keep-alive mechanism that allows each one of them to discover if the other is gone (the mechanism is TBD). Let us assume that the PC discovers that the phone is gone. The PC is aware of the dialog with Bob because of the subscription to the dialog event package. The PC then clears dialog2, and sends an INVITE with Replaces to Bob's phone to replace the dialog with the phone.

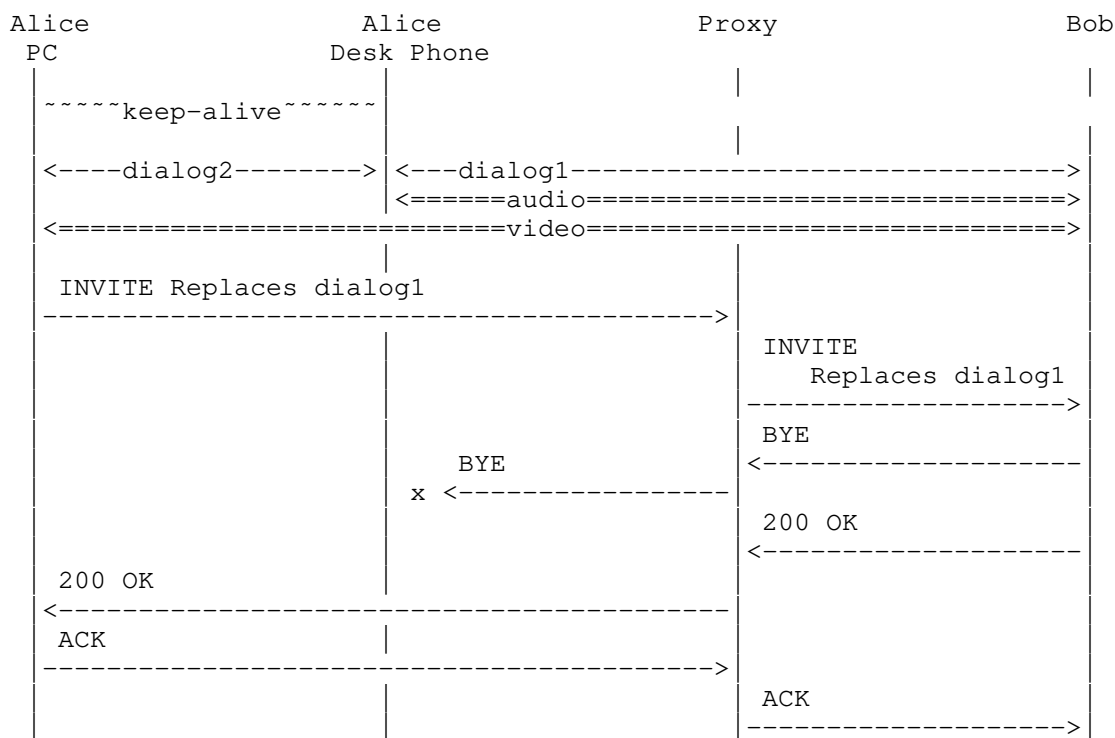


Figure 8

The above mechanism has its limitation, as the PC, in this case, must be able to discover that the other federated device is gone and replace the dialog between the Desk Phone and the remote phone before Bob's phone tears down the dialog with the Desk Phone (dialog1).

## 5. Scenarios Not Covered by Existing Mechanisms

As discussed previously, all existing mechanisms implement disaggregated media using a centralized approach. Scenarios not covered by existing mechanisms include those where none of the nodes can act as a controller because it does not support the necessary functionality or because it will not participate in the whole session (transferring the SIP dialog from a controller to a new one using REFER and Replaces is complex and requires support from the far end). These scenarios are better implemented using a more distributed approach.

In a distributed scenario, the far end of the session receives directly signaling messages from each of the devices involved in the multimedia session. That means that the receiver needs to treat all the signaling and media coming from different devices of the same user as part of the same media session.

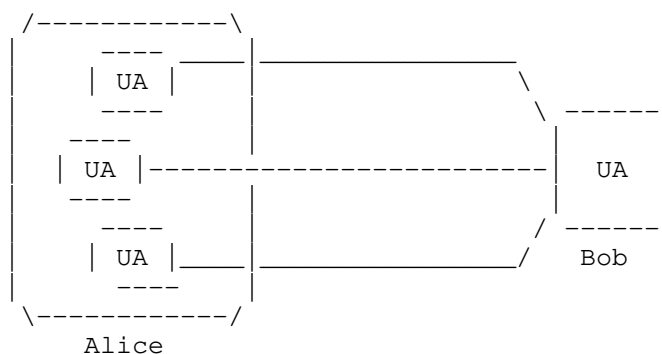


Figure 9

Figure 5 shows the generic signaling flow in a Distributed Scenario, where all the signaling messages go from the different user's devices to the far end of the session.

## 6. Recommendations

To maximize the possibility of the adoption of this mechanism, we recommend that the WG take the Distributed Call Control with Actions approach defined in section 4 above.

A future work may learn from the experience we gain with the Distributed Call Control with Actions approach and later try to standardize the approach described in section 5 above.



## 7. Security Considerations

To be done.

## 8. IANA Considerations

This document does not require any actions by the IANA.

## 9. Informational References

- [RFC3087] Campbell, B. and R. Sparks, "Control of Service Context using SIP Request-URI", RFC 3087, April 2001.
- [RFC3259] Ott, J., Perkins, C., and D. Kutscher, "A Message Bus for Local Coordination", RFC 3259, April 2002.
- [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.
- [RFC3525] Groves, C., Pantaleo, M., Anderson, T., and T. Taylor, "Gateway Control Protocol Version 1", RFC 3525, June 2003.
- [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.
- [RFC3911] Mahy, R. and D. Petrie, "The Session Initiation Protocol (SIP) "Join" Header", RFC 3911, October 2004.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

Authors' Addresses

Gonzalo Camarillo  
Ericsson  
Hirsalantie 11  
Jorvas 02420  
Finland

Email: Gonzalo.Camarillo@ericsson.com

Salvatore Loreto  
Ericsson  
Hirsalantie 11  
Jorvas 02420  
Finland

Email: salvatore.loreto@ericsson.com

Rifaat Shekh-Yusef  
Avaya  
250 Sidney Street  
Belleville, Ontario K8N 5B7  
Canada

Email: rifatyu@avaya.com

INTERNET-DRAFT  
Intended Status: Standards Track  
Expires: December 12, 2011

R. Shekh-Yusef  
Avaya  
C. Jennings  
Cisco Systems  
A. Johnston  
Avaya  
F. Audet  
Skype  
June 10, 2011

The Session Initiation Protocol (SIP) INVOKE Method  
draft-yusef-splices-invoke-01

Abstract

This document defines the SIP INVOKE method, which describes a mechanism by which a UA can invoke a well-defined action on a remote UA. These actions are represented by well-defined URNs that must be registered with IANA.

This document also defines a new event package 'invoke', and the Action and Action-Progress request headers.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at  
<http://www.ietf.org/lid-abstracts.html>

The list of Internet-Draft Shadow Directories can be accessed at  
<http://www.ietf.org/shadow.html>

## Copyright and License Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Terminology . . . . .	4
2. Background . . . . .	5
3. The INVOKE Method . . . . .	7
3.1. The Action Header Field . . . . .	7
3.2. URN Structure . . . . .	7
3.3. URN Categories . . . . .	8
3.4. URN Actions . . . . .	8
3.5. URN Action Parameters . . . . .	8
3.6. Message Body Inclusion . . . . .	9
4. Event Package . . . . .	10
4.1. Subscription . . . . .	10
4.2. Progress Indication . . . . .	10
4.3. Subscription Termination . . . . .	10
4.4. The Body of the NOTIFY . . . . .	10
5. User Agent Behavior . . . . .	11
5.1. Forming an INVOKE request . . . . .	11
5.2. Processing an INVOKE request . . . . .	11
5.3. Request Authorization . . . . .	11
5.4. Action Progress Indication . . . . .	11
6. Control Channel . . . . .	12
7. Capabilities Indications . . . . .	12
8. Security Considerations . . . . .	12
9. Example Flows . . . . .	13
10. IANA Considerations . . . . .	16
11. Acknowledgments . . . . .	16
12. References . . . . .	16
13. Author's Addresses . . . . .	17

## 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 RFC 2119 [RFC2119].

To simplify discussions of the INVOKE method and its extensions, the two terms below are being used throughout the document:

- o INVOKE-Issuer: the UA issuing the INVOKE request
- o INVOKE-Recipient: the UA receiving the INVOKE request

## 2. Background

The Session Initiation Protocol (SIP) [RFC3261] provides users with the ability to initiate, manage and terminate multimedia sessions. Many deployments have SIP applications in the SIP signaling path that get involved in the setup and management of these multimedia sessions.

A SIP application is a program running on a SIP network element that provides some value-added function to a user. Some of these applications need mechanisms to monitor or/and control a SIP User Agent (UA).

SIP already provides an extensible framework, SIP-Specific Event Notification [RFC3265], by which SIP UAs can monitor remote UAs and get indications that certain events have occurred. For example, the following are widely used event packages:

- o Dialog package - call states
- o Registration package - phone status
- o Conference package - conference status

SIP also provides a method for requesting UAs to perform certain task, i.e., REFER [RFC3515]. The REFER method has many limitations that prevents it from being the ideal method for application level interaction:

- o The REFER method is overloaded with five distinct meanings as described in [draft-worley-sip-many-refers-00.txt]
- o The body of the NOTIFY is always message/sipfrag and any application data must be delivered in the body of the sipfrag message.
- o The referral progress indication is inside the body of the NOTIFY method, instead of headers in the NOTIFY method.
- o Progress indications for accessing non-SIP resources are not clearly defined and use SIP progress indications.
- o Implicit subscription is used, but explicit subscription is not allowed

- o There is no way for the REFER-Issuer to ask the REFER-Recipient to keep the dialog alive after the referral.

This document defines the INVOKE method, which describes a mechanism by which a UA can invoke a well-defined action on a remote UA. These actions are represented by well-defined URNs that must be registered with IANA. It also provides a mechanism that allows the INVOKE issuer to be notified of the outcome of the invoked action. Furthermore, It allows the INVOKE issuer to keep the dialog established with the INVOKE recipient open, to allow both sides to exchange application specific information.

This document also defines the 'invoke' event package and the Action, Subscribe-Type and Action-Progress request headers.



### 3. The INVOKE Method

INVOKE is a SIP method as defined by [RFC3261]. The INVOKE method indicates that the INVOKE-Recipient should invoke the action specified in the request.

An INVOKE request MAY be placed inside or outside the scope of a dialog created with a SUBSCRIBE request.

#### 3.1. The Action Header Field

The Action header is a request header field as defined by [RFC3261]. It only appears in an INVOKE request or a SUBSCRIBE request.

The INVOKE method uses the Action header to hold a URN that represents the action to be invoked by the INVOKE-Recipient, e.g. urn:invoke:call:answer.

The SUBSCRIBE method uses the Action header to hold a URN that represents the action or category of actions to be monitored by the INVOKE-Recipient.

#### 3.2. URN Structure

The Namespace Identifier (NID) of the URN is intended to be in the formal space and assigned by IANA, as per the procedures of [RFC3406]. This document defines the 'invoke' namespace.

The Namespace Specific String (NSS) of the URN includes the action name, and may be followed by a semi-colon and additional action-specific parameters.

The action identifier has a hierarchical structure of categories separated by colon, and the right-most label is the action name.

The reason behind the above structure is to avoid the creation of a namespace with a very long list of unrelated actions.

### 3.3. URN Categories

This document defines the following categories as part of the NSS of the URN:

- o call: to allow access to call actions available on a SIP UA, e.g. answer a call, decline a call, ...
- o conference: to allow access to conference actions available on a SIP UA, e.g. add, remove, ...

### 3.4. URN Actions

This document defines the following actions:

- o Answer call           - urn:invoke:call:answer
- o Terminate call       - urn:invoke:call:terminate
- o Decline call         - urn:invoke:call:decline
- o Ignore call          - urn:invoke:call:ignore
- o Send call to VM      - urn:invoke:call:sendvm
- o Hold call            - urn:invoke:call:hold
- o Unhold call          - urn:invoke:call:unhold
- o Mute call            - urn:invoke:call:mute
- o Unmute call          - urn:invoke:call:unmute
- o Conference           - urn:invoke:conference:add
- urn:invoke:conference:remove

Note that the very important "Make call" CTI primitive does not require a action URN since it is accomplished by sending a REFER with a URI identifying the resource (e.g., a sip, sips or tel URI).

### 3.5. URN Action Parameters

An action can be followed by a semi-colon and additional action-specific parameters.

For example:

```
urn:invoke:call:answer;media=audio;transducer=speaker|headset
```

This action indicates to the UA to answer the call and provide an audio answer and use either the speaker or headset for the incoming media.

The following is a list of example action parameters:

- o media=audio|video|audvid
- o transducer=speaker|headset
- o target=<AOR>
- o direction=recvonly|sendonly|sendrecv
- o abort

### 3.6. Message Body Inclusion

An INVOKE method MAY contain a body. This specification assigns no meaning to such a body. A receiving agent may choose to process the body according to its Content-Type or based on the action that the request invokes.

#### OPEN ISSUE: Header vs. Body?

There has been some discussion on the list on the question of header field vs message body. This is not the real issue, as either could conceivably be used. Instead, the main issue is what is being invoked: is it a named feature/action/event, or is it a script. The authors strongly believe that the invocation of a script by the method would be a mistake. This would introduce all kinds of security issues and vulnerabilities. This mechanism is purposely defined using URNs - invoking a named feature/action/event. While this might seem to limit the flexibility, it allows a UA to understand exactly what operation is being requested, and apply appropriate policy. If a given feature, action, or event can not be carried out simply by referencing its name and perhaps a few parameters, then this means that it is not suitable for this mechanism. If a true script execution is needed, existing scripting approaches such as CPL (Call Processing Language) [RFC3880] should be considered.

Since the invocation is a simple named feature and not a generic script, the question of whether a header field or message body is appropriate. Since it is just a name with at most a few parameters, the authors feel that a header field is sufficient to carry this, and a body is overkill and inefficient.

#### 4. Event Package

This specification defines the new event package 'invoke', which enables one UA to monitor another UA for the invocation of a specific URN.

##### 4.1. Subscription

The UA issuing the subscribe request can monitor either a specific action or a category of actions as specified in the Action header field. For example, a subscription to urn:invoke:call:answer would result in NOTIFYs being sent when this specific URN is invoked, while a subscription to urn:invoke:call would result in NOTIFYs being sent when any URN in this category has been invoked.

##### 4.2. Progress Indication

The Action-Progress header is used to allow the INVOKE-Recipient to report on the progress of the invoked action. The Action-Progress header **MUST** be added to the NOTIFY requests sent to the INVOKE-Issuer.

OPEN ISSUE: Some of these actions are not SIP actions. Should a new list of generic response codes be defined for this purpose?

##### 4.3. Subscription Termination

Either side can terminate the subscription using the normal [RFC3265] procedures. The SUBSCRIBE-Issuer, by sending a SUBSCRIBE request with expires 0, when it is no longer interested in receiving notification about a specific action or category of actions. The SUBSCRIBE-Recipient, by sending a NOTIFY with Subscription-Status: terminated, when it no longer wants to allow the SUBSCRIBE-Issuer to monitor a specific action or a category of actions.

##### 4.4. The Body of the NOTIFY

A NOTIFY method **MAY** contain a body that is specific to the action invoked by the INVOKE request. A receiving agent **MAY** choose to process the body according to its Content-Type or based on the invoked action.

## 5. User Agent Behavior

### 5.1. Forming an INVOKE request

INVOKE is a SIP request and is constructed as defined in [RFC3261]. An INVOKE request MUST contain exactly one Action header field value. It MAY contain Target-Dialog header [RFC4538] to associate the action with an existing dialog.

### 5.2. Processing an INVOKE request

A UA accepting a well-formed INVOKE request MUST invoke the action identified by the URN in the Action header field value.

An agent responding to an INVOKE method MUST return a 400 (Bad Request) if the request contained zero or more than one Action header field values.

### 5.3. Request Authorization

When a UA receives a request to invoke an action, it needs to authorize that request. Some requests can be authorized based on the identity of the sender, the request method, local policy, etc.

The Target-Dialog mechanism [RFC4538] MAY be used when a user agent authorization depends on whether the sender of the request is currently in a dialog with that user agent, or whether the sender of the request is aware of a dialog the user agent has with another entity.

In most cases, the same user will be logged in to the different devices using the same credentials provided in the REGISTER requests. In this case, all INVOKE requests MUST be challenged using the digest authentication mechanism described by [RFC3261].

### 5.4. Action Progress Indication

The NOTIFY mechanism defined in [RFC3265] MUST be used to inform the INVOKE-Issuer of the status of the action.

Each NOTIFY MUST contain an Event header field with a value of 'invoke'. Each NOTIFY MUST contain an Action-Progress header field. The Action-Progress header allows the INVOKE-Recipient to report on the progress of the invoked action.

## 6. Control Channel

The INVOKE-Issuer MAY establish a Control Channel with the INVOKE-Recipient, using the subscription mechanism defined in [RFC3265], to allow both sides to exchange application specific information related to the invoked action.

The INVOKE-Issuer MAY use INVOKE in the context of the Control Channel dialog to send application specific updates to the INVOKE-Recipient.

The INVOKE-Recipient MUST use NOTIFY to send application specific updates to the INVOKE-Issuer.

## 7. Capabilities Indications

A UA compliant to this specification MUST indicate its support by including the option tag 'invoke' in the Supported header of all requests it sends.

A new feature tag is defined to allow a User Agent to indicate which categories it supports. The new feature tag is described below:

Feature tag name: invoke

Each value of the invoke tag indicates a URN category supported by a SIP UA. The values for this tag equal the URN category names that are supported by the UA.

For example, a UA that supports the 'call' and 'conference' categories would look as follows:

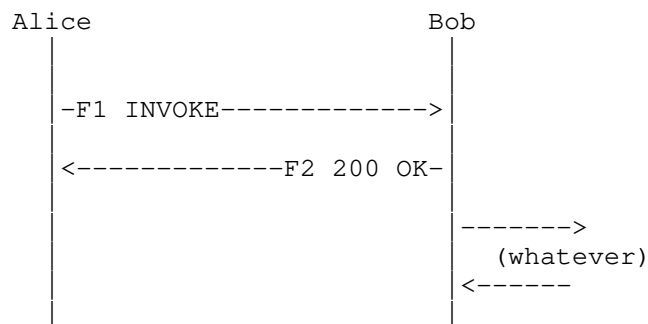
```
invoke="call,conference"
```

## 8. Security Considerations

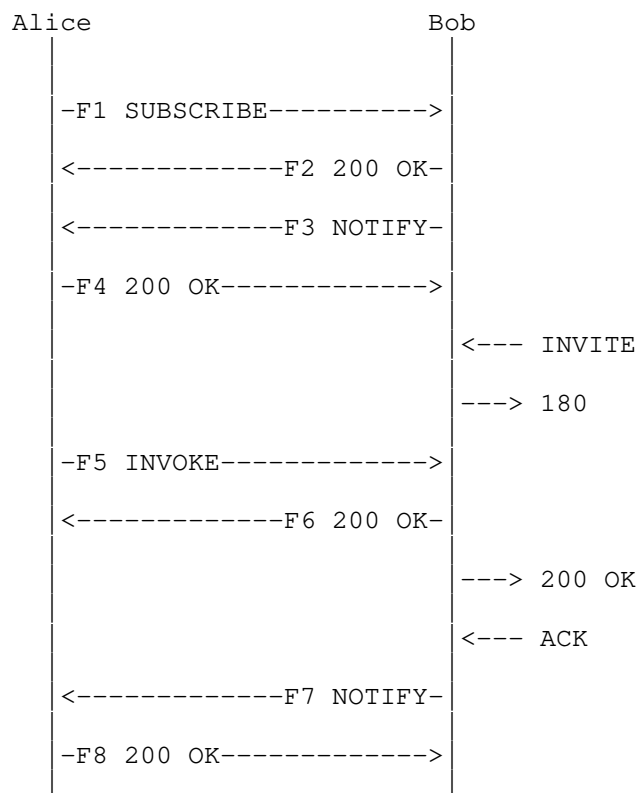
The functionality described in this document allows an authorized party to manipulate SIP sessions and dialogs in arbitrary ways. Any user agent that accepts these types of requests needs to be very careful in who it authorizes to send these types of requests. The same security considerations as [RFC3515] apply.

## 9. Example Flows

## INVOKE



## SUBSCRIBE and INVOKE



F1 SUBSCRIBE sip:bob@example.com SIP/2.0  
Via: SIP/2.0/UDP host.example.com;branch=z9hG4bKnashds7  
To: <sip:bob@example.com>  
From: <sip:alice@example.com>;tag=12341234  
Call-ID: 12345678@host.example.com  
CSeq: 1 SUBSCRIBE  
Max-Forwards: 70  
Expires: whatever  
Event: invoke  
Action: urn:invoke:call  
Contact: sip:alice@host.example.com  
Content-Length: 0

F2 SIP/2.0 200 OK  
Via: SIP/2.0/UDP host.example.com;branch=z9hG4bKnashds7  
To: <sip:bob@example.com>;tag=abcd1234  
From: <sip:alice@example.com>;tag=12341234  
Call-ID: 12345678@host.example.com  
CSeq: 1 SUBSCRIBE  
Contact: sip:bob@host.example.com  
Expires: whatever  
Content-Length: 0

F3 NOTIFY sip:alice@host.example.com SIP/2.0  
Via: SIP/2.0/UDP host.example.com;branch=z9hG4bK8sdf2  
To: <sip:alice@example.com>;tag=12341234  
From: <sip:bob@example.com>;tag=abcd1234  
Call-ID: 12345678@host.example.com  
CSeq: 1 NOTIFY  
Max-Forwards: 70  
Event: invoke  
Action: urn:invoke:call  
Action-Progress: 100 Trying  
Subscription-State: active; expires=whatever  
Contact: sip:bob@host.example.com  
Content-Length: 0

F4 SIP/2.0 200 OK  
Via: SIP/2.0/UDP host.example.com;branch=z9hG4bK8sdf2  
To: <sip:alice@example.com>;tag=12341234  
From: <sip:bob@example.com>;tag=abcd1234  
Call-ID: 12345678@host.example.com  
CSeq: 1 NOTIFY



F5 INVOKE sip:bob@example.com SIP/2.0  
Via: SIP/2.0/UDP host.example.com;branch=z9hG4bKnashds8  
To: <sip:bob@example.com>;tag=abcd1234  
From: <sip:alice@example.com>;tag=12341234  
Call-ID: 12345678@host.example.com  
CSeq: 1 INVOKE  
Max-Forwards: 70  
Action: urn:invoke:call:answer  
Contact: sip:alice@host.example.com  
Content-Length: 0

F6 SIP/2.0 200 OK  
Via: SIP/2.0/UDP host.example.com;branch=z9hG4bKnashds8  
To: <sip:bob@example.com>;tag=abcd1234  
From: <sip:alice@example.com>;tag=12341234  
Call-ID: 12345678@host.example.com  
CSeq: 1 INVOKE  
Contact: sip:bob@host.example.com  
Content-Length: 0

F7 NOTIFY sip:alice@host.example.com SIP/2.0  
Via: SIP/2.0/UDP host.example.com;branch=z9hG4bK4cd42a  
To: <sip:alice@example.com>;tag=12341234  
From: <sip:bob@example.com>;tag=abcd1234  
Call-ID: 12345678@host.example.com  
CSeq: 2 NOTIFY  
Max-Forwards: 70  
Event: invoke  
Action: urn:invoke:call:answer  
Action-Progress: 200 OK  
Subscription-State: active; expires=whatever  
Contact: sip:bob@host.example.com  
Content-Length: 0

F8 SIP/2.0 200 OK  
Via: SIP/2.0/UDP host.example.com;branch=z9hG4bK4cd42a  
To: <sip:alice@example.com>;tag=12341234  
From: <sip:bob@example.com>;tag=abcd1234  
Call-ID: 12345678@host.example.com  
CSeq: 2 NOTIFY  
Content-Length: 0

## 10. IANA Considerations

This specification defines the list of actions in section 3.4 as an initial set of URNs, to be registered with IETF, for use with this specification.

In order to add any new action URN to the list above, it must go through the "IETF review" process as defined in [RFC5226].

## 11. Acknowledgments

TO-DO

## 12. 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.

[RFC3265]Roach, A., "Session Initiation Protocol (SIP)-Specific Event Notification", RFC 3265, June 2002.

[RFC3515]Sparks, R., "The Session Initiation Protocol (SIP) Refer Method", RFC 3515, April 2003.

[RFC3406]Daigle, L., van Gulik, D., Iannella, R., and P. Faltstrom, "Uniform Resource Names (URN) Namespace Definition Mechanisms", BCP 66, RFC 3406, October 2002.

[RFC5226]Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC3880]Lennox, J., Wu, X., and H. Schulzrinne, "Call Processing Language (CPL): A Language for User Control of Internet Telephony Services", RFC 3880, October 2004.

[RFC4538]Rosenberg, J., "Request Authorization through Dialog Identification in the Session Initiation Protocol (SIP)", RFC 4538, June 2006.

## 13. Author's Addresses

Rifaat Shekh-Yusef  
Avaya  
250 Sidney Street  
Belleville, Ontario K8N 5B7  
Canada

Phone: +1-613-967-5267  
Email: rifatyu@avaya.com

Cullen Jennings  
Cisco Systems  
170 West Tasman Drive  
Mailstop SJC-21/2  
San Jose, CA 95134  
USA

Phone: +1-408-902-3341  
Email: fluffy@cisco.com

Alan Johnston  
Avaya  
St. Louis, MO 63124  
USA

Email: alan.b.johnston@gmail.com

Francois Audet  
Skype  
USA

Email: francois.audet@skype.net