

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/1id-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](http://trustee.ietf.org/bcp78) 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

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



## **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:

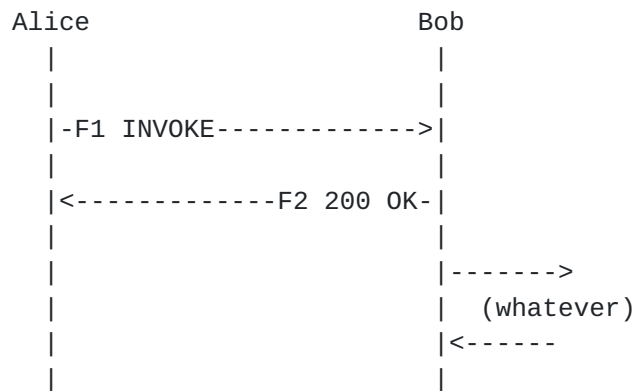
```
invoke="call,conference"
```

## **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.



INVOKE



```

Alice                                     Bob
|                                         |
|                                         |
| -F1 SUBSCRIBE----->|
|                                         |
| <-----F2 200 OK-|
|                                         |
| <-----F3 NOTIFY-|
|                                         |
| -F4 200 OK----->|
|                                         |
|                                         | <--- INVITE
|                                         |
|                                         | ---> 180
|                                         |
| -F5 INVOKE----->|
|                                         |
| <-----F6 200 OK-|
|                                         |
|                                         | ---> 200 OK
|                                         |
|                                         | <--- ACK
|                                         |
| <-----F7 NOTIFY-|
|                                         |
| -F8 200 OK----->|
|                                         |

```





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

