220 Perry Parkway

Gaithersburg, MD 20877

Tel (301) 977-9440

Fax (301) 977-8846

BROADWORKS SIP ACCESS SIDE EXTENSIONS INTERFACE SPECIFICATIONS

RELEASE 13.0

Version 1

www.broadsoft.com

BroadWorks® Guide

Copyright Notice

Trademarks

Table of Contents

Figures

1 Introduction

This document describes extensions to the Session Initiation Protocol (SIP) that can be used to enable traditional voice applications, including but not limited to: key system emulation, executive-admin station emulation, push to talk, click to dial, and other remote control CTI applications. These extensions are typically employed between a line side “application server” and an end-user agent, such as an IP phone or soft client. Application servers employing these extensions can provide a tightly integrated end-user experience and a rich service offering without being tightly coupled to the access device.

Note that these extensions compliment existing standards and draft extensions aimed at integrating end-user access equipment with operator-hosted service platforms.

These extensions have been adopted and implemented by many equipment vendors in the industry, including IP PBX vendors, IP phone vendors, access gateway vendors, and application server vendors.

2 Info-Parameter Extensions to “Call-Info”

2.1 Overview

In many traditional voice applications, it is important for the endpoint and the service delivery platform to maintain consistent “presentation” information. By this we mean, the relative order of call appearances on a line and the current state of the call appearances. This enables call control clients, attendant consoles, and other applications to maintain a synchronized view of call appearance information, so that end users can move from one endpoint or one interface to another, without being confused.

The *Call-Info* header is specified in *RFC 3261*. It is used to provide additional information about the calling or called party, depending on whether it is found in a request or a response. Here we use this header to send “presentation” information about the call appearances associated with a given address of record (that is, a line). An excerpt from the augmented BNF in *RFC 3261* follows for the *Call-Info* header:

```
Call-Info   =  "Call-Info" HCOLON info *(COMMA info)
info        =  LAQUOT absoluteURI RAQUOT *( SEMI info-param)
info-param  =  ( "purpose" EQUAL ( "icon" / "info"
               / "card" / token)) / generic-param
```

This shows that the *Call-Info* header allows for comma delimited information elements. Each element must have an absolute URI – and each element can optionally have an arbitrary number of information parameters.

2.2 Appearance-index, Appearance-state, and Appearance-URI

For this extension, the absolute URI should be a simple hostname SIP URI with no user portion. The host name should be set to the provisioned proxy server or in this case, the Application Server. Three new generic parameters are defined for indicating call information element parameters:

```
“appearance-index” EQUAL ( (1*DIGIT) / STAR ) )
“appearance-state” EQUAL ( “idle” / “seized” / “progressing” / “alerting”
/ “active” / “held” / “held-private” )
“appearance-uri” EQUAL quoted-string
```

For usability reasons, it is important to preserve the relative order of these call appearance across endpoints. The appearance-index parameter is used to qualify the call information elements, by identifying a relative index of the call appearance on the line. So if the appearance-index is 1, it indicates the first appearance on the line. If the appearance-index is 2, it indicates the second appearance on the line, and so on, up to the maximum number of appearances allowed on the line.

NOTE: An appearance-index of “*” is a shorthand form, indicating that the information element is applicable to the rest of the call appearances on the line.

The appearance-state is used to further qualify the call appearance with an actual visual state that can be used to light a lamp, or display an LED or bitmap on an LCD display. The defined states are:

State	Description
Idle	This appearance on the line is available for use.
Seized	This appearance has been seized by one of the endpoints in the SCA group.
Progressing	This appearance on the line is currently making an outgoing call.
Alerting	This appearance is receiving an incoming call.
Active	This appearance is actively involved in a call.
Held	This appearance has a call in the held state.
Held-private	This appearance has a call in the held state and only the endpoint which held the call, may retrieve it.

The appearance-URI is used to further qualify the call appearance with the remote call party identification, which devices can use to augment the display, if they support this capability on the device. The value of the appearance-URI parameter is a SIP quoted-string that contains a SIP name-addr (that is, the contents are “name-addr”). Since the name-addr is within a quoted-string, it is escaped according to the SIP escaping rules. The appearance-URI parameter is not included when the call appearance’s state is idle or seized, since there is no call involved with the call appearance in these states.

2.2.1 In INVITE requests

A UAC may include a *Call-Info* header when sending an INVITE request. If it does, it must contain at most one call appearance information element. It may contain other information elements for different purposes, but only one call appearance information element should be present. The call appearance information element must contain exactly one appearance-index parameter with a numeric value. The numeric value indicates the call appearance index where the call is being presented.

So, for example, assume a phone has one line with four separate call appearances, and the user interface on the phone allows the end user to arbitrarily select any one of the call appearances when originating a call. If the user chose the third call appearance, then the INVITE might look something like the following:

```
INVITE sip:5551212@broadworks.net …
From:  <sip:5551000@broadworks.net>…
To: <sip:5551212@broadworks.net>…
Call-Info:  <sip:broadworks.net>;appearance-index=3
…
```

Effectively the phone is saying: “The user has selected the third call appearance of line 5551000 to call 5551212”.

Note that the *Call-Info* header is not required to be present in a Re-INVITE, but if present, the Re-INVITE must use the same appearance-index (that is, a Re-INVITE is not allowed to change the appearance-index being used).

When BroadWorks is acting as the UAC, the device must honor the appearance-index included in an INVITE. In a race condition where the device initiates a line-seize for an appearance-index at the same time as BroadWorks sends an INVITE to the device for the same appearance-index, the device should allow the INVITE for the incoming call to proceed. A failure response to the line-seize SUBSCRIBE will be received by the device shortly afterwards.

When the device is acting as the UAC and includes the appearance-index in an INVITE, BroadWorks may choose to override the appearance-index in the response, as described in Section 2.2.2.

An endpoint designates a call as privately held, by sending a re-INVITE with hold SDP including a *call-info* header with an appearance-state of held-private. BroadWorks, upon receipt of the re-INVITE for hold, inspects the *call-info* header and updates the state for the call appearance to held-private. All shared call appearances with subscriptions to the call-info package in the group are updated via NOTIFYs with the held-private call appearance state.

Following is an example of the re-INVITE with hold SDP, which is privately held:

```
INVITE sip:5551212@broadworks.net …
From:  <sip:5551000@broadworks.net>…
To: <sip:5551212@broadworks.net>…
Call-Info:  <sip:broadworks.net>;appearance-state=held-private
…
```

Following is the resulting NOTIFYs sent to the shared call appearances in the group:

```
NOTIFY sip:contact@endpoint …
From:  <sip:5551000@broadworks.net>…
To: <sip:5551000@broadworks.net>…
Call-Info:  <sip:broadworks.net>;appearance-
index=1;appearance-state=held-private;appearance-
index=*;appearance-state=idle
```

When an endpoint in the shared call appearance group attempts to retrieve a privately held call appearance which it did not put on hold, BroadWorks rejects the re-INVITE retrieval attempt with a 403 Forbidden.

BroadSoft recommends that devices use the same lamp state for held-private calls that they use for active calls. The recommended active lamp state for all of the other endpoints which are not actively involved in the call is a solid red indicator. However, the held-private state is propagated to all of the endpoints in the shared call appearance group, such that an endpoint may choose a different visual indication for privately held calls.

2.2.2 In responses

Similarly, a UAS may include a *Call-Info* header when responding to an INVITE request. *Call-Info* headers with call appearance information elements may appear in any provisional or final response. They follow similar rules for Call-Infos sent in INVITE requests by UACs. So in the above example, assume the endpoint that receives the call for the address of record 5551212@broadworks.net allows up to six call appearances. At the time that this INVITE arrives at the endpoint, assume there are three calls present on the first three call appearances. So the phone chooses the next available call appearance (4) and sends back a provisional response:

```
SIP/2.0 180 Ringing …
From:  <sip:5551000@broadworks.net>…
To: <sip:5551212@broadworks.net>…
Call-Info:  <sip:broadworks.net>;appearance-index=4
```

Effectively the phone is saying: "The incoming call from 5551000 is ringing the user on the fourth call appearance of the line 5551212".

Once the call is answered, then the 200 OK final response from the UAS may also contain a *Call-Info* header. It is possible that the call may have moved relative appearances by the time it is answer. This is totally dependant on the user interface of the endpoint.

```
SIP/2.0 200 OK …
From:  <sip:5551000@broadworks.net>…
To: <sip:5551212@broadworks.net>…
Call-Info:  <sip:broadworks.net>;appearance-index=3
```

Effectively the phone is saying: "The incoming call from 5551000 has been answered by the user on the third call appearance of line 5551212".

If the user interface on the endpoint does not choose a particular call appearance until it answered, then it is possible that the 180 Ringing does not have a *Call-Info* header and the 200 OK has a *Call-Info* header.

Note that the *Call-Info* header is not required to be present in responses to a Re-INVITE, but if present, the responses must use the same appearance-index (that is, a Re-INVITE response is not allowed to change the appearance-index being used).

When BroadWorks is acting as the UAS, the device must honor the appearance-index included in INVITE responses. In some scenarios (for example, when the device does not perform a line-seize prior to the INVITE and the requested appearance-index is already in use by BroadWorks), BroadWorks will override the requested appearance-index when it responds to the INVITE. In these scenarios, the device should move the call to the appearance-index specified in the response.

When the device is acting as the UAS and it receives an INVITE with a requested appearance-index, it must use the same appearance-index for all responses (that is, it is not allowed to change the appearance-index in a response). In a race condition where the device initiates a line-seize for an appearance-index at the same time as BroadWorks sends an INVITE to the device for the same appearance-index, the device should allow the INVITE for the incoming call to proceed. A failure response to the line-seize SUBSCRIBE will be received by the device shortly afterwards in NOTIFY requests.

Finally, a notifier can include a *Call-Info* header when sending a NOTIFY request to a subscriber of the *Call-Info* event package (description follows). *The Call-Info* header of a NOTIFY request must contain a complete set of call appearance information elements – one for each potential call-appearance on the address of record. Each call appearance information element must contain one appearance-index and one appearance-state parameter. Optionally, each call appearance information element can contain one appearance-URI parameter. So the subscriber receives complete call appearance state information for the address of record in each NOTIFY.

For example, if the address of record 5551000@broadworks.net has four call appearances allowed, and there is one call active on the first call appearance and one call held on the second call appearance, and the rest of the call appearances are idle, then the following NOTIFY is sent to each subscriber of 5551000@broadworks.net.

```
NOTIFY sip:contact@endpoint …
From:  <sip:5551000@broadworks.net>…
To: <sip:5551000@broadworks.net>…
Call-Info:  <sip:broadworks.net>;appearance-index=1;appearance-
state=active;appearance-uri="\"George Smith\"
<sip:3015551234@broadworks.net>",
            <sip:broadworks.net>;appearance-index=2;appearance-
state=held,
            <sip:broadworks.net>;appearance-index=3;appearance-
state=idle,
            <sip:broadworks.net>;appearance-index=4;appearance-state=idle
```

We can use the "*" as a short-cut to indicate "the rest of the call appearances", which is useful when initializing the call-appearances for an address of record to "idle":

```
NOTIFY sip:contact@endpoint …
From:  <sip:5551000@broadworks.net>…
To: <sip:5551000@broadworks.net>…
Call-Info:  <sip:broadworks.net>;appearance-index=*;appearance-state=idle
```

is semantically equivalent to:

```
NOTIFY sip:contact@endpoint …
From:  <sip:5551000@broadworks.net>…
To: <sip:5551000@broadworks.net>…
Call-Info:  <sip:broadworks.net>;appearance-index=1;appearance-
state=idle,
            <sip:broadworks.net>;appearance-index=2;appearance-
state=idle,
            <sip:broadworks.net>;appearance-index=3;appearance-
state=idle,
            <sip:broadworks.net>;appearance-index=4;appearance-state=idle
```

2.2.3 In SUBSCRIBE requests

A subscriber MUST include a *Call-Info* header with an appearance-index information element in a SUBSCRIBE request to the *line-seize* event package. The Application Server handling the subscription uses the appearance-index in the *Call-Info* header to identify which call-appearance the line-seize subscription is being applied to.

2.3 Call-Info "Answer-After" Info-parameter

2.3.1 Overview

To support click to dial and push to talk applications, an application platform must be able to originate calls that force the endpoint off-hook, without end-user intervention.

Here we introduce a generic parameter called "answer-after". When present in the *Call-Info* header of an incoming INVITE request, it indicates how many alert cycles should be applied by the UAS before the call is automatically answered. For example, the following *Call-Info* header indicates that the call should be automatically answered after one alert cycle.

```
INVITE sip:123456789@broadworks.net  SIP/2.0
From:  <sip:jamie@broadworks.net>; tag=1
To:  <sip:foo@broadworks.net>
Call-Info:  <sip:broadworks.net>; answer-after=1
```

If the “answer-after” value is 0, then the call should be automatically answered without applying any alert tones, as shown in the following example.

```
INVITE sip:123456789@broadworks.net  SIP/2.0
From:  <sip:jamie@broadworks.net>; tag=1
To:  <sip:foo@broadworks.net>
Call-Info:  <sip:broadworks.net>; answer-after=0
```

In all cases, the *Call-Info* header is just a suggestion to the UAS. Depending on the UAS capabilities or local policy, the UAS may elect to ignore the “answer-after” parameter. For instance, if the UAS is a phone that does not support speakerphone, then it cannot automatically answer the call, and must wait for the user to actively answer the call. Similarly, the phone may have a speakerphone, but the user may have enabled a local privacy policy that rejects or ignores all “answer-after” requests. A UAC should be prepared to receive an 18x provisional response, even when it has requested that the UAS automatically answer the call without alerting (meaning answer-after=0).

In all cases, the UAS should attempt to authorize the incoming request. At minimum, the UAS should perform access control. Ideally, the UAS should challenge the incoming INVITE for credentials that authorize the request to perform an automatic answer.

2.3.2 Example Message Flow

The following diagram shows the message flow as a result of the users performing a click-to-dial operation from their thin client interface.

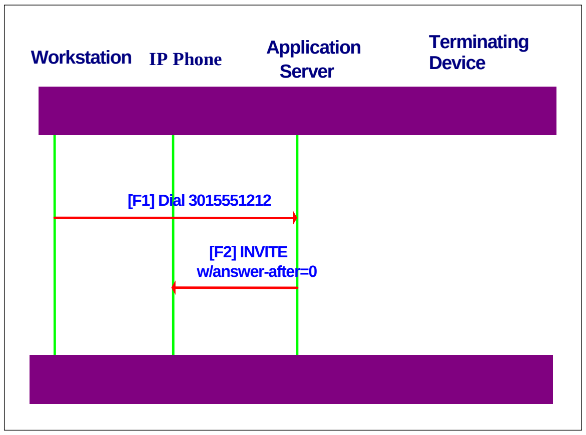

Figure 1 Message Flow as a Result of Click-To-Dial Operation

[F1] starts the flow with the user initiating a click-to-dial operation from the thin call client on the workstation. The Application Server processes the click-to-dial event. The Application Server sends an INVITE to the associated endpoint in [F2], as follows:

```
[F2] INVITE Application Server -> IP phone
INVITE sip:3015551000@ipphone.com SIP/2.0
Via: SIP/2.0/UDP a1.foo.com:5060;branch=m9hG4bK74bf9
Max-Forwards: 70
From: “Joe” <sip:3015551212@a1.foo.com>;tag=5fxced76sl
To: “Jane” <sip:3015551000@a1.foo.com>
Call-ID: 9848276298220188511@a1.foo.com
```

```
Call-Info:  <sip:as.foo.com>;appearance-index=1; answer-after=0
CSeq: 43234 INVITE
Contact: <sip:a1.foo.com:5060>
Allow-Events: conference
Allow: ACK, BYE, CANCEL, INFO, INVITE, OPTIONS, PRACK, REFER
Supported:  100rel, timer
Content-Length: 0
```

3 Call-Info Event Package

3.1 Overview

The ability to support shared call appearances (SCA) is a fundamental building block for a variety of enhanced telephony services. Features like attendant console, line extensions, and key system emulation cannot be delivered without some mechanism for sharing call appearances across access devices. Although SIP (*RFC 3261*) by itself offers no inherent semantics for supporting SCA features, when coupled with an appropriate instantiation of the "SIP Specific Event Notification" framework (*RFC 3265*), these services can be deployed quite easily in a distributed network.

A shared line is an address of record managed by central controlling element, such as an application server. The application server allows multiple endpoints to register locations against the address of record. The application server is responsible for policing who can register and who cannot register against the shared line.

For incoming call appearances, the application server sends separate INVITE requests to each of the registered endpoints. As the incoming call appearance progresses to the answered state on one of the endpoints, the INVITE requests to the other endpoints in the shared call appearance group are cancelled. For outgoing call appearances, the application server presents any call from the registered endpoints as originating from the same address of record. Effectively, the line is shared because any endpoint in the shared call appearance group can originate and terminate calls on behalf of the same address of record.

The following sections describe how the *Call-Info* header can be used to suggest the presentation state of the call appearances on a shared line. A *Call-Info* event package can then be used to broadcast this presentation state to endpoints that are not actively involved in the call appearances.

The *Call-Info* event package is an instantiation of the SIP Specific Event Notification Framework (as defined in *RFC 3265*). For the applications described earlier, the Application Server acts as the "notifier", while the IP phones configured with shared lines act as the "subscribers".

Following is a description of the event package details, as per *RFC 3265*.

3.2 Example Message Flow

3.2.1 Shared Call Appearance Client Subscription

This process allows the endpoint to SUBSCRIBE to the call appearance state of a given line. The Request-URI for these transactions is the same as the Request-URI used in the registration process for the address of record (in this case, a "line").

Note that the call flow does not show the SUBSCRIBE being challenged. The application server may choose to use access control (using the IP address authenticated in the REGISTER transaction), or optionally it could challenge the transaction. If the transaction is challenged, then the IP phone should use the same credentials it used in the corresponding registration process for that call appearance. Throughout the rest of this example, it is assumed that standard access control is being applied to all transactions.

After accepting the SUBSCRIBE, the application server initializes the phone to the appropriate call status by sending a NOTIFY with a call state document. This NOTIFY event is sent even if the call state is idle, and serves as a synchronization event between the application server and the IP phone.

```
[F1] SUBSCRIBE A-1 -> Application Server
SUBSCRIBE sip:shared-a@as.foo.com SIP/2.0
Via: SIP/2.0/UDP a1.foo.com:5060;branch=dsdf32nashds7
Max-Forwards: 70
From: “Shared-A” <sip:shared-a@as.foo.com>;tag=dsa3dszlfl
To: “Shared-A” <sip:shared-a@as.foo.com>
Call-ID: 5j9FpLxk3uxtm8to@a1.foo.com
CSeq: 6 SUBSCRIBE
Event: call-info
Expires: 3600
Contact: <sip:shared-a@a1.foo.com>
Content-Length: 0

[F2] 200 OK Application Server → A-1
SIP/2.0 200 OK
Via: SIP/2.0/UDP a1.foo.com:5060;branch=dsdf32nashds7
From: “Shared-A” <sip:shared-a@as.foo.com>;tag=dsa3dszlfl
To: “Shared-A” <sip:shared-a@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 6 SUBSCRIBE
Event:  call-info
Contact: sip:shared-a@as.foo.com
Expires: 3600
Content-Length: 0

[F3] NOTIFY Application Server -> A-1
NOTIFY sip:shared-a@a1.foo.com SIP/2.0
Via: SIP/2.0/UDP as.foo.com:5060;branch=gsdf32nashds7
Max-Forwards: 70
From: “Shared-A” <sip:shared-a@as.foo.com>;tag=dsa3dszlfl
To: “Shared-A” <sip:shared-a@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 1344 NOTIFY
Event: call-info
Subscription-State: active; expires=3599
Contact:  sip:shared-a@as.foo.com
Call-Info:  <sip:as.foo.com>;appearance-index=*;appearance-state=idle
Content-Length: 0

[F4] 200 OK A-1 → Application Server
SIP/2.0 200 OK
Via: SIP/2.0/UDP as.foo.com:5060;branch=gsdf32nashds7
From: “Shared-A” <sip:shared-a@as.foo.com>;tag=dsa3dszlfl
To: “Shared-A” <sip:shared-a@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 1344 NOTIFY
Content-Length: 0
```

3.2.2 Outgoing Call

The following figure shows the call flow as a result of endpoint A-1 originating a call to B using A's shared line. It is assumed that A-1 has successfully seized the line.

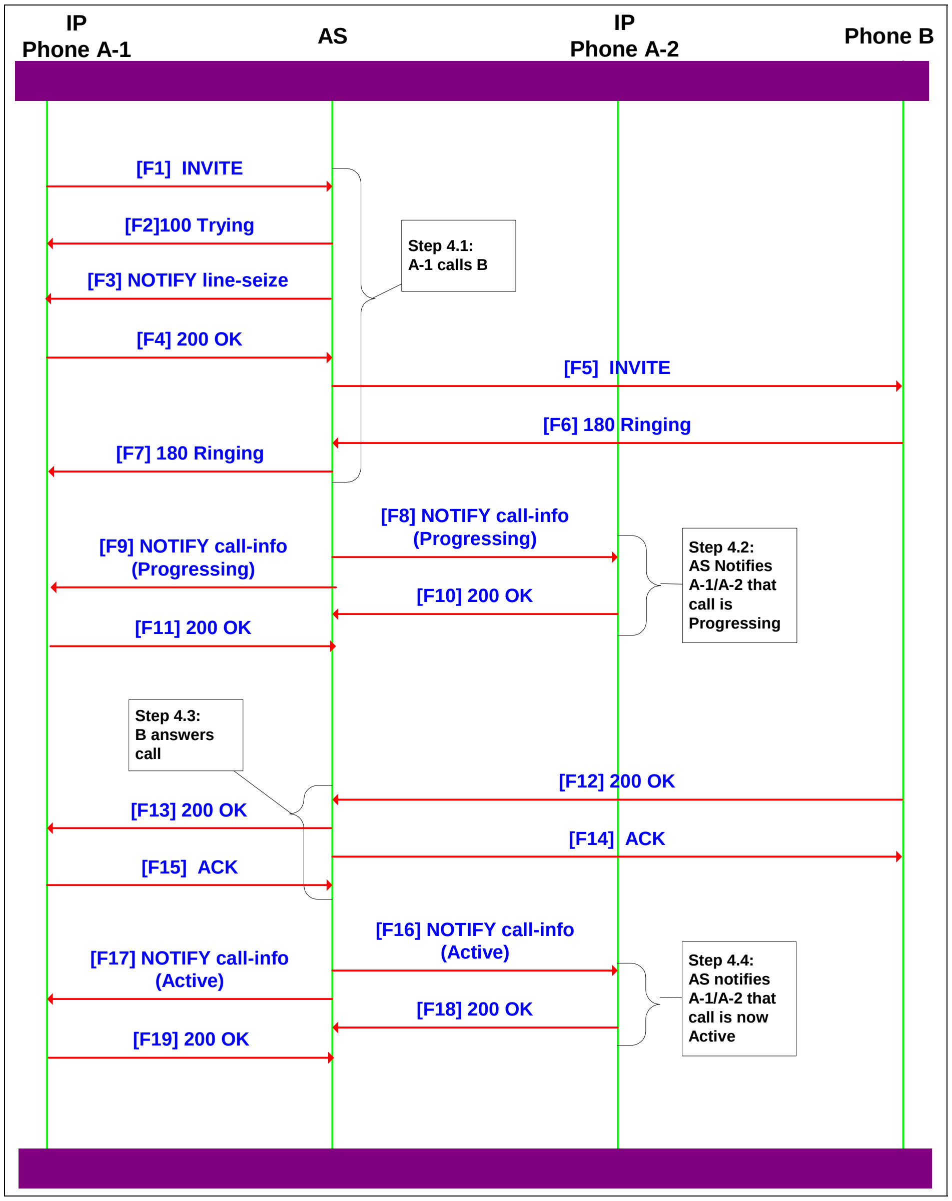

Figure 2 Call Flow as a Result of Endpoint A-1 Originating a Call to B using A's Shared Line

3.2.2.1 A-1 Calls B

In [F1 to F7], the user on IP phone A-1 has entered digits that originates a call to B. Because the endpoint has already seized the line, the Application Server accepts the INVITE and simply treats the call like a standard outbound call to an off network device. Note that in [F3], the Application Server sends a NOTIFY-request to A-1, indicating that the line-seize subscription has been terminated. This is normal once an INVITE has been received from the endpoint that has seized the call appearance on the shared line.

```
[F1] INVITE A-1 → Application Server

INVITE sip:B@as.foo.com SIP/2.0
Via: SIP/2.0/UDP a1.foo.com:5060;branch=m9hG4bK74bf9
Max-Forwards: 70
From: “Shared-A” <sip:shared-a@as.foo.com>;tag=5fxced76sl
To: “B” <sip:B@as.foo.com>
Call-ID: 98482762982201885110a1.foo.com
Call-Info:  <sip:as.foo.com>;appearance-index=1
CSeq: 43234 INVITE
Contact: <sip:shared-a@a1.foo.com>
Content-Type: application/sdp
Content-Length: 143

v=0
o=UserA 2890844526 2890844526 IN IP4 a1.foo.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000

[F2] 100 Trying Application Server → A-1

SIP/2.0 100 Trying
Via: SIP/2.0/UDP a1.foo.com:5060;branch=m9hG4bK74bf9
From: “Shared-A” <sip:shared-a@as.foo.com>;tag=5fxced76sl
To: “B” <sip:B@foo.com>; tag=4323z39
Call-ID: 9848276298220188511@a1.foo.com
Call-Info:  <sip:as.foo.com>;appearance-index=1
CSeq: 43234 INVITE
Contact: <sip:shared-a@as.foo.com>
Content-Length: 0

[F3] NOTIFY Application Server -> A-1
NOTIFY sip:shared-a@a1.foo.com SIP/2.0
Via: SIP/2.0/UDP as.foo.com:5060; branch=gsdf32nashds7
Max-Forwards: 70
From: “Shared-A” <sip:shared-a@as.foo.com>;tag=gsa3dszlfl
To: “Shared-A” <sip:shared-a@as.foo.com>;tag=423423233
Call-ID: 6j9FpLxk3uxtm8to@a1.foo.com
CSeq: 24 NOTIFY
Event: line-seize
Subscription-State: terminated; expires=0
Contact: sip:shared-a@as.foo.com
Content-Length: 0

[F4] 200 OK A-1 → Application Server
SIP/2.0 200 OK
Via: SIP/2.0/UDP a1.foo.com:5060;branch=gsdf32nashds7
From: “Shared-A” <sip:shared-a@as.foo.com>;tag=gsa3dszlfl
To: “Shared-A” <sip:shared-a@as.foo.com>;tag=423423233
```

```
Call-ID: 6j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 24 NOTIFY
Content-Length: 0


[F5] INVITE Application Server → B

INVITE sip:B@b.foo.com SIP/2.0
Via: SIP/2.0/UDP as.foo.com:5060;branch=234adfrjksd
Max-Forwards: 70
From: “Shared-A” <sip:shared-a@as.foo.com>;tag=234wdkfj
To: “B” <sip:B@as.foo.com>
Call-ID: asdf2220188511@as.foo.com
CSeq: 12345 INVITE
Contact: <sip:shared-a@as.foo.com>
Content-Type: application/sdp
Content-Length: 143

v=0
o=UserA 2890844526 2890844526 IN IP4 a1.foo.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000

[F6] 180 Ringing B → Application Server

SIP/2.0 180 Ringing
Via: SIP/2.0/UDP as.foo.com:5060;branch=234adfrjksd
From: “Shared-A” <sip:shared-a@as.foo.com>;tag=234wdkfj
To: “B” <sip:B@as.foo.com>;tag=1234dsf2
Call-ID: asdf2220188511@as.foo.com
CSeq: 12345 INVITE
Contact: <sip:B@b.foo.com>
Content-Length: 0

[F7] 180 Ringing Application Server → A-1

SIP/2.0 180 Ringing
Via: SIP/2.0/UDP a1.foo.com:5060;branch=m9hG4bK74bf9
From: “Shared-A” <sip:shared-a@as.foo.com>;tag=5fxced76sl
To: “B” <sip:B@as.foo.com>;tag=4323z39
Call-ID: 98482762982220188511@a1.foo.com
Call-Info:  <sip:as.foo.com>;appearance-index=1
CSeq: 43234 INVITE
Contact: <sip:B@as.foo.com>
Content-Length: 0
```

3.2.2.2 Application Server Notifies A-1 and A-2 that Call is Progressing

As the call progresses, the Application Server sends Call-Info NOTIFY-requests to all endpoints that have subscribed to the *Call-Info* event package on the shared line. In this example, IP phones A-1 and A-2 have both subscribed to the *Call-Info* event package, so the Application Server sends a NOTIFY-request to both A-1 and A-2, with a *Call-Info* header indicating a state of “Progressing” (events [F8-F11]).

```
[F8] NOTIFY Application Server -> A-1
NOTIFY sip:shared-a@a1.foo.com SIP/2.0
Via: SIP/2.0/UDP as.foo.com:5060;branch=gsdf32nashds7
Max-Forwards: 70
From: “Shared-A” <sip:shared-a@as.foo.com>;tag=dsa3dszlfl
To: “Shared-A” <sip:shared-a@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 1344 NOTIFY
Event: call-info
Subscription-State: active; expires=3200
Call-Info:  <sip:as.foo.com>;appearance-index=1;appearance-
state=progressing,
            <sip:as.foo.com>;appearance-index=*;appearance-state=idle
Content-Length: 0

[F9] NOTIFY Application Server -> A-2
NOTIFY sip:shared-a@a2.foo.com SIP/2.0
Via: SIP/2.0/UDP as.foo.com:5060;branch=gsdf32nashds7
Max-Forwards: 70
From: “Shared-A” <sip:shared-a@as.foo.com>;tag=ggsa3dszlfl
To: “Shared-A” <sip:shared-a@as.foo.com>;tag=123456
Call-ID: 9k9FpLxk3uxtm8tn@a2.foo.com
CSeq: 1345 NOTIFY
Event: call-info
Subscription-State: active; expires=3100
Call-Info:  <sip:as.foo.com>;appearance-index=1;appearance-
state=progressing,
            <sip:as.foo.com>;appearance-index=*;appearance-state=idle
Content-Length: 0

[F10] 200 OK A-1 → Application Server
SIP/2.0 200 OK
Via: SIP/2.0/UDP as.foo.com:5060;branch=gsdf32nashds7
From: “Shared-A” <sip:shared-a@as.foo.com>;tag=dsa3dszlfl
To: “Shared-A” <sip:shared-a@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 1344 NOTIFY
Content-Length: 0

[F11] 200 OK A-2 → Application Server
SIP/2.0 200 OK
Via: SIP/2.0/UDP as.foo.com:5060;branch=gsdf32nashds7
From: “Shared-A” <sip:shared-a@as.foo.com>;tag=ggsa3dszlfl
To: “Shared-A” <sip:shared-a@as.foo.com>;tag=123456
Call-ID: 9k9FpLxk3uxtm8tn@a2.foo.com
CSeq: 1345 NOTIFY
Content-Length: 0
```

3.2.2.3 B Answers Call

[F12 to F15] shows B answering the call using standard back-to-back user agent call flow. A-1 and B are now active in a call.

```
[F12] 200 OK B → Application Server

SIP/2.0 200 OK
Via: SIP/2.0/UDP as.foo.com:5060;branch=234adfrjksd
From: "Shared-A" <sip:shared-a@as.foo.com>;tag=234wdkfj
To: "B" <sip:B@as.foo.com>;tag=1234dsf2
Call-ID: asdf2220188511@as.foo.com
CSeq: 12345 INVITE
Contact: <sip:B@b.foo.com>
Content-Length: 0

[F13] 200 OK Application Server → A-1

SIP/2.0 200 OK
Via: SIP/2.0/UDP a1.foo.com:5060;branch=m9hG4bK74bf9
From: "Shared-A" <sip:shared-a@as.foo.com>;tag=5fxced76sl
To: "B" <sip:B@as.foo.com>; tag=4323z39
Call-ID: 98482762982201885ll@a1.foo.com
Call-Info:  <sip:as.foo.com>;appearance-index=1
CSeq: 43234 INVITE
Contact: <sip:B@as.foo.com>
Content-Length: 0

[F14] ACK A-1 → Application Server

ACK sip:B@as.foo.com SIP/2.0
Via: SIP/2.0/UDP a1.foo.com:5060;branch=345rbK74bf9
Max-Forwards: 70
From: "Shared-A" <sip:shared-a@as.foo.com>;tag=234wdkfj
To: "B" <sip:B@as.foo.com>;tag=1234dsf2
Call-ID: asdf2220188511@foo.com
CSeq: 1093 ACK
Contact: <sip:shared-a@a1.foo.com>
Content-Length: 0

[F15] ACK Application Server → B

ACK sip:B@b.foo.com SIP/2.0
Via: SIP/2.0/UDP as.foo.com:5060;branch=23dsdf2ksd
Max-Forwards: 70
From: "Shared-A" <sip:shared-a@as.foo.com>;tag=234wdkfj
To: "B" <sip:B@as.foo.com>;tag=1234dsf2
Call-ID: asdf2220188511@foo.com
CSeq: 12345 ACK
Contact: <sip:shared-a@as.foo.com>
Content-Length: 0
```

3.2.2.4 Application Server Notifies A-1 and A-2 that Call is Now Active

In [F16 to F19] the Application Server sends a NOTIFY-request to IP phones A-1 and A-2 with a *Call-Info* header, indicating a call appearance state of “Active” for the first call appearance on the shared line.

```
[F16] NOTIFY Application Server -> A-1
NOTIFY sip:shared-a@a1.foo.com SIP/2.0
Via: SIP/2.0/UDP as.foo.com:5060;branch=gsdf32nashds7
Max-Forwards: 70
From: “Shared-A” <sip:shared-a@as.foo.com>;tag=dsa3dszlfl
To: “Shared-A” <sip:shared-a@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 1344 NOTIFY
Event: call-info
Subscription-State: active; expires=3000
Call-Info:  <sip:as.foo.com>;appearance-index=1;appearance-state=active,
            <sip:as.foo.com>;appearance-index=*;appearance-state=idle
Content-Length: 0

[F17] NOTIFY Application Server -> A-2
NOTIFY sip:shared-a@a2.foo.com SIP/2.0
Via: SIP/2.0/UDP as.foo.com:5060;branch=gsdf32nashds7
Max-Forwards: 70
From: “Shared-A” <sip:shared-a@as.foo.com>;tag=ggsa3dszlfl
To: “Shared-A” <sip:shared-a@as.foo.com>;tag=123456
Call-ID: 9k9FpLxk3uxtm8tn@a2.foo.com
CSeq: 1345 NOTIFY
Event: call-info
Subscription-State: active; expires=2999
Call-Info:  <sip:as.foo.com>;appearance-index=1;appearance-state=active,
            <sip:as.foo.com>;appearance-index=*;appearance-state=idle
Content-Length: 0

[F18] 200 OK A-1 → Application Server
SIP/2.0 200 OK
Via: SIP/2.0/UDP as.foo.com:5060;branch=gsdf32nashds7
From: “Shared-A” <sip:shared-a@as.foo.com>;tag=dsa3dszlfl
To: “Shared-A” <sip:shared-a@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 1344 NOTIFY
Content-Length: 0

[F19] 200 OK A-2 → Application Server
SIP/2.0 200 OK
Via: SIP/2.0/UDP as.foo.com:5060;branch=gsdf32nashds7
From: “Shared-A” <sip:shared-a@as.foo.com>;tag=ggsa3dszlfl
To: “Shared-A” <sip:shared-a@as.foo.com>;tag=123456
Call-ID: 9k9FpLxk3uxtm8tn@a2.foo.com
CSeq: 1345 NOTIFY
Content-Length: 0
```

3.3 Event Package Name

The name of this event is “Call-Info”. It is carried in the *Event and Allow-Events* header as defined in *RFC 3265*.

3.4 Event Package Parameters

This package does not define any event package parameters.

3.5 SUBSCRIBE Bodies

A SUBSCRIBE for a *Call-Info* package must not contain a body. A body is not required for this application of the *Call-Info* event package – any body sent in a SUBSCRIBE is ignored by the notifier.

3.6 Subscription Duration

Subscribers SHOULD specify the duration of a subscription with an *Expires* header in the SUBSCRIBE request. It is recommended that the subscription use the same refresh period as the REGISTER event. If the notifier changes the expiration period to a lower value (via an *Expires* header in the final response), the subscriber should honor it.

If the subscriber does not specify the duration of a subscription, then the notifier chooses a default expiration. The recommended default is 1800 seconds.

3.7 NOTIFY Bodies

A NOTIFY for a *Call-Info* package MUST NOT contain a body. No body is required for the *Call-Info* event package as all state is reflected through the *Call-Info* header.

3.8 Subscriber Generation of SUBSCRIBE Requests

Subscriber generation of SUBSCRIBE requests MUST follow all rules with respect to subscriber behavior as described in *RFC 3265*. For this application, SIP phones that are configured with shared lines should generate SUBSCRIBE requests immediately after successfully registering a location against the provisioned address of record. The Request-URI, *From*, and *To* header must be the same as the address of record of the shared line.

3.9 Notifier Processing of SUBSCRIBE Requests

Notifier processing of SUBSCRIBE requests MUST follow all rules with respect to notifier behavior as described in *RFC 3265*. The *Call-Info* package contains potentially sensitive information. Subscriptions SHOULD be authorized by the notifier. The notifier MAY perform implicit authorization by looking at the source IP address of the SUBSCRIBE event and matching it to any endpoints that have successfully been authenticated through the registration process. The notifier MAY choose to explicitly challenge the subscriber for authentication by using a “401 Unauthorized” response. If the subscriber has also registered a location for this address of record, then it SHOULD use the same credentials to answer the subscription challenge.

3.10 Notifier Generation of NOTIFY Requests

Notifier generation of NOTIFY requests MUST follow all rules with respect to notifier behavior as described in *RFC 3265*. Notifications are generated for the *Call-Info* package whenever the supplementary call information associated with an address of record (in this case a line) changes. As described in RFC-3261, supplementary call information can be almost anything. For the purposes of these applications, the call information includes a complete list of all the call appearances allowed on the address of record and the state of each call appearance. The notifier sends this information in a NOTIFY request by explicitly including a *Call-Info* header in addition to the required *NOTIFY request* headers.

See the sections above for details about the content of the *Call-Info* header and how it should be used.

3.11 Subscriber Processing of NOTIFY Requests

In general, subscriber processing of NOTIFY requests MUST follow all rules with respect to subscriber behavior, as described in *RFC 3265*. The SIP Specific Event Notification Framework expects packages to specify how a subscriber processes NOTIFY requests, and in particular, how it uses the NOTIFY requests to construct a coherent view of the state of the subscribed resource.

To be specific, the *Call-Info* package itself has no state, but in this application, the state of the call appearances of a given address of record can be derived from the content of the *Call-Info* header that is delivered in the NOTIFY request. This state is described in detailed in the following sections. Because these applications use the content of the NOTIFY's *Call-Info* header to visually present the call appearance state to the user, the subscriber (that is, the phone) SHOULD perform some level of authorization before processing NOTIFY's. The subscriber MAY perform implicit authorization by looking at the source IP address of the NOTIFY request event and matching it to any one of the IP addresses of the notifier, as resolved through DNS. The subscriber MAY perform explicit authentication by challenging the notifier, by using a "401 Unauthorized" response. If the subscriber has also registered a location for this address of record, then the notifier SHOULD use those same credentials to answer the notification challenge.

3.12 Handling of Forked Requests

Although it is theoretically possible for a SUBSCRIBE request to fork, this is not likely for the applications described in this document. Robust subscriber implementations SHOULD be prepared to install multiple Call-Info subscriptions, but practically speaking, this never happens.

3.13 Rate of Notifications

A very active phone line may result in many notifications sent to every endpoint in the shared call appearance group. Endpoints should be prepared to accept NOTIFY requests as frequently as a few times per second in burst scenarios (which should be infrequent).

4 Line-Seize Event Package

4.1 Overview

SIP has been employed as a line side access protocol in voice networks. When compared to other line side access protocols like SCCP or MGCP, SIP is considered a "peer to peer" protocol. Because of this, SIP-UAs embedded in IP phones typically provide local dial tone, digit collection, and essentially perform their own line side call setup. This works well for lines that are private to the endpoint. But this behavior can cause race conditions when emulating shared line functionality across multiple endpoints. Emulating shared line functionality requires some central line controller to arbitrate which endpoint has "seized" the line. The purpose of the "line-seize" event package is to support the concept of a line-seizure between an endpoint and line-controller, as found in a typical "shared line" solution.

The approach defined here requires that all endpoints that participate in a shared line obtain a subscription to the *line-seize* package for the shared line before presenting the user with dial tone or collecting digits. This means the endpoints must send a SUBSCRIBE-request for the *line-seize* event package whenever a user attempts to take the shared line off hook. The "line controller" guarantees that only one subscription to the *line-seize* event package can exist for a given shared line resource. When the line controller grants a subscription to the *line-seize* package, it responds to the SUBSCRIBE with a 200 OK response and generates a NOTIFY, indicating that the subscription state is now "active". This is an indicator to the endpoint that it can safely play dial tone to the user and collect digits. Once the digits have been collected, then the endpoint can send an INVITE-request. While one endpoint has obtained the *line-seize* subscription, any requests from other endpoints to use that shared line are rejected with a 480 Temporarily Unavailable response.

The *line-seize* event package is an instantiation of the SIP Specific Event Notification Framework (as defined in *RFC 3265*). Following is a description of the event package details, as per *RFC 3265*.

To avoid scenarios where the endpoint seizes the line, but never actually uses it (that is, it never sends an INVITE-request) and effectively hangs the line - the *line-seize* package must have a suitable duration associated with it. Fortunately, the SIP specific event notification framework, as defined in *RFC 3265,* explicitly describes a mechanism for negotiating subscription duration between the subscriber (the endpoint) and the notifier (the line controller). The SUBSCRIBE-request to the *line-seize* package contains an expiration time as offered by the endpoint – and in response the line-controller can choose to set a smaller expiration time. If the endpoint has not collected digits and sent an INVITE-request before the subscription has expired, then it must refresh the subscription or risk losing it to another endpoint. Endpoints should only collect digits for a suitable length of time before they let the subscription expire naturally. This avoids scenarios where one endpoint in the shared call appearance group goes off hook accidentally and effectively ties up the line. After a reasonable amount of time, the endpoint should stop refreshing the subscription, let it expire naturally, and then apply an appropriate local treatment (re-order tone), so the end user is aware that digits can no longer being collected.

If the user goes on hook before digit collection is complete – effectively terminating the line seizure without originating a call – then the endpoint must clear the line seizure by explicitly canceling the subscription. This allows the line to be immediately seized by another endpoint. As per RFC 3261, this is effectively achieved by refreshing the subscription with an expiration of "0".

4.2 Example Message Flows

The following figure shows the call flow as a result of a user at A-1 taking the shared line off hook. The line controller in this scenario is provided by a central Application Server (AS).

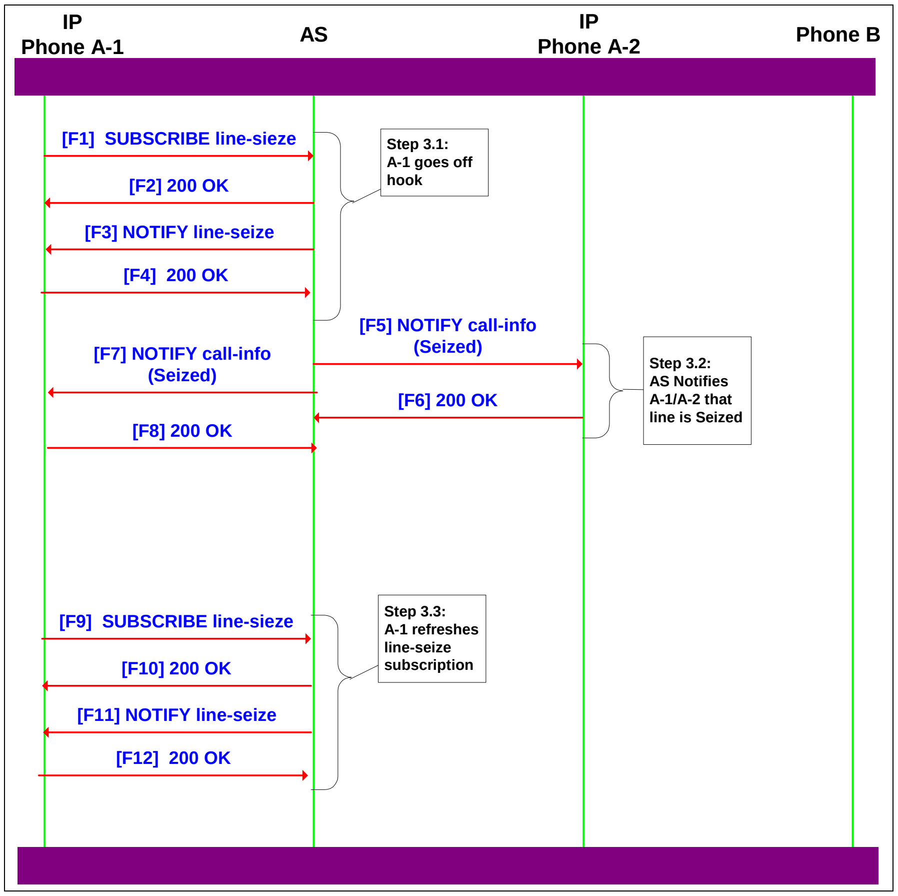
IP Phone A-1
AS
IP Phone A-2
Phone B
[F1] SUBSCRIBE line-sieze
[F2] 200 OK
[F3] NOTIFY line-seize
[F4] 200 OK
Step 3.1: A-1 goes off hook
[F5] NOTIFY call-info (Seized)
[F6] 200 OK
[F7] NOTIFY call-info (Seized)
[F8] 200 OK
Step 3.2: AS Notifies A-1/A-2 that line is Seized
[F9] SUBSCRIBE line-sieze
[F10] 200 OK
[F11] NOTIFY line-seize
[F12] 200 OK
Step 3.3: A-1 refreshes line-seize subscription

Figure 3 Call Flow as a Result of a User at A-1 Taking the Shared Line Off Hook

In [F1 to F4], IP-phone A-1 must be granted a subscription to the *line-seize* package. Only after the subscription has been granted, should the IP phone play dial tone to the user.

```
[F1] SUBSCRIBE A-1 -> Application Server
SUBSCRIBE sip:shared-a@as.foo.com SIP/2.0
Via: SIP/2.0/UDP a1.foo.com:5060;branch=gsdf32nashds7
Max-Forwards: 70
From: “Shared-A” <sip:shared-a@as.foo.com>;tag=gsa3dszlfl
To: “Shared-A” <sip:shared-a@as.foo.com>
Call-ID: 6j9FpLxk3uxtm8to@a1.foo.com
Call-Info:  <sip:as.foo.com>; appearance-index=1
CSeq: 7 SUBSCRIBE
Event: line-seize
Expires: 15
Contact: <sip:shared-a@a1.foo.com>
Content-Length: 0

[F2] 200 OK Application Server → A-1
SIP/2.0 200 OK
Via: SIP/2.0/UDP a1.foo.com:5060;branch=gsdf32nashds7
```

```
From: “Shared-A” <sip:shared-a@as.foo.com>;tag=gsa3dszlfl
To: “Shared-A” <sip:shared-a@as.foo.com>;tag=423423233
Call-ID: 6j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 7 SUBSCRIBE
Contact: sip:shared-a@as.foo.com
Event: line-seize
Expires=15
Content-Length: 0

[F3] NOTIFY Application Server -> A-1
NOTIFY sip:shared-a@a1.foo.com SIP/2.0
Via: SIP/2.0/UDP as.foo.com:5060; branch=gfasdf237
Max-Forwards: 70
From: “Shared-A” <sip:shared-a@as.foo.com>;tag=gsa3dszlfl
To: “Shared-A” <sip:shared-a@as.foo.com>;tag=423423233
Call-ID: 6j9FpLxk3uxtm8to@a1.foo.com
Call-Info:  <sip:as.foo.com>; appearance-index=1
CSeq: 9 NOTIFY
Event: line-seize
Subscription-State: active; 14
Contact: sip:shared-a@as.foo.com
Content-Length: 0

[F4] 200 OK A-1 → Application Server
SIP/2.0 200 OK
Via: SIP/2.0/UDP a1.foo.com:5060; branch=gfasdf237
From: “Shared-A” <sip:shared-a@as.foo.com>;tag=gsa3dszlfl
To: “Shared-A” <sip:shared-a@as.foo.com>;tag=423423233
Call-ID: 6j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 9 NOTIFY
Content-Length: 0
```

4.3 Event Package Name

The name of this event package is “line-seize”. It is carried in the *Event and Allow-Events* header as defined in *RFC 3265*.

4.4 Event Package Parameters

This package does not define any event package parameters.

4.5 SUBSCRIBE Bodies

A SUBSCRIBE for a *line-seize* package MUST NOT contain a body. No body is required for this application of the *line-seize* event package – any body sent in a SUBSCRIBE is ignored by the notifier.

4.6 Subscription Duration

Subscribers SHOULD specify the duration of a subscription with an *Expires* header in the SUBSCRIBE request. It is recommended that the subscription use a refresh period, equal roughly to how long it takes for a typical end user to dial a full E.164 number. If the notifier changes the expiration period to a lower value (via an *Expires* header in the final response), the subscriber should honor it.

If the subscriber does not specify the duration of a subscription, then the notifier chooses a default expiration value. The recommended default is 15 seconds.

4.7 NOTIFY Bodies

A NOTIFY for a *line-seize* package MUST NOT contain a body. No body is required for the *line-seize* event package, as it uses a very simple state machine of either active or terminated – which is easily reflected through the *Subscription-State* header. A body contained in a NOTIFY-request should be ignored by the subscriber.

4.8 Subscriber Generation of SUBSCRIBE Requests

Subscriber generation of SUBSCRIBE requests MUST follow all rules with respect to subscriber behavior, as described in *RFC 3265*. Typically, SIP phones that are configured with shared lines should generate SUBSCRIBE requests immediately after a user's attempts to go off hook on an idle shared line. The Request-URI, *From*, and *To* header must be the same as the address of record of the shared line.

4.9 Notifier Processing of SUBSCRIBE Requests

Notifier processing of SUBSCRIBE requests MUST follow all rules with respect to notifier behavior, as described in *RFC 3265*. The *line-seize* package contains potentially sensitive information. Subscriptions SHOULD be authorized by the notifier. The notifier MAY perform implicit authorization by looking at the source IP address of the SUBSCRIBE event and matching it to any endpoints that have successfully been authenticated through the registration process. The notifier MAY choose to explicitly challenge the subscriber for authentication, by using a "401 Unauthorized" response. If the subscriber has also registered a location for this address of record, then it SHOULD use the same credentials to answer the subscription challenge.

4.10 Notifier Generation of NOTIFY Requests

Notifier generation of NOTIFY requests MUST follow all rules with respect to notifier behavior, as described in *RFC 3265*. Notifications are generated for the *line-seize* package immediately after a subscription has been accepted.

4.11 Subscriber Processing of NOTIFY Requests

In general, subscriber processing of NOTIFY requests MUST follow all rules with respect to subscriber behavior, as described in *RFC 3265*. The SIP Specific Event Notification framework expects packages to specify how a subscriber processes NOTIFY requests, and in particular, how it uses the NOTIFY requests to construct a coherent view of the state of the subscribed resource.

The *line-seize* package has binary state. Either the "line-seize" subscription is active or it's terminated. This state is reflected in the *Subscription-State* header. The endpoint SHOULD perform some level of authorization before processing NOTIFYs. The subscriber MAY perform implicit authorization by looking at the source IP address of the NOTIFY request event and matching it to any one of the known IP addresses of the notifier (in this case the Application Server), as resolved through DNS. The subscriber MAY perform explicit authentication by challenging the notifier, by using a "401 Unauthorized" response. If the subscriber has also registered a location for this address of record, then the notifier SHOULD use those same credentials to answer the notification challenge.

4.12 Handling of Forked Requests

Although it is theoretically possible for a SUBSCRIBE request to fork, this is not likely for typical applications that use the *line-seize* event package. Since the nature of the subscription guarantees that only one subscription exist at a time, then the first forked request that arrives at the Application Server is granted a subscription, while the rest of the SUBSCRIBE-requests are rejected.

4.13 Rate of Notifications

Notifications are only sent when the endpoint explicitly refreshes the subscription (about every 15 seconds), when the subscription expires (also after about 15 seconds), or when the subscription is terminated.

5 Remote Control Talk Event Package

5.1 Overview

When providing CTI based applications such as desktop call control clients, a service delivery platform must be able to request that an endpoint answer an incoming call. This extension provides the ability for an Application Server to send an asynchronous NOTIFY event to an endpoint, using an existing INVITE dialog. When received, the endpoint can elect to honor the request and answer the call, without local user intervention.

This extension can also be used to support remotely retrieving a call that has been previously held. When the endpoint receives the NOTIFY event and the associated dialog is in the held state, then the endpoint can elect to honor the request and retrieve the call from the held state.

The “talk” event package is an instantiation of the SIP Specific Event Notification Framework (as defined in *RFC 3265*). Following are details of the event package, as per *RFC 3265*.

5.2 Example Message Flow

The following diagram shows the call flow as a result of a user performing a click-to-answer function on an incoming call from the PSTN.

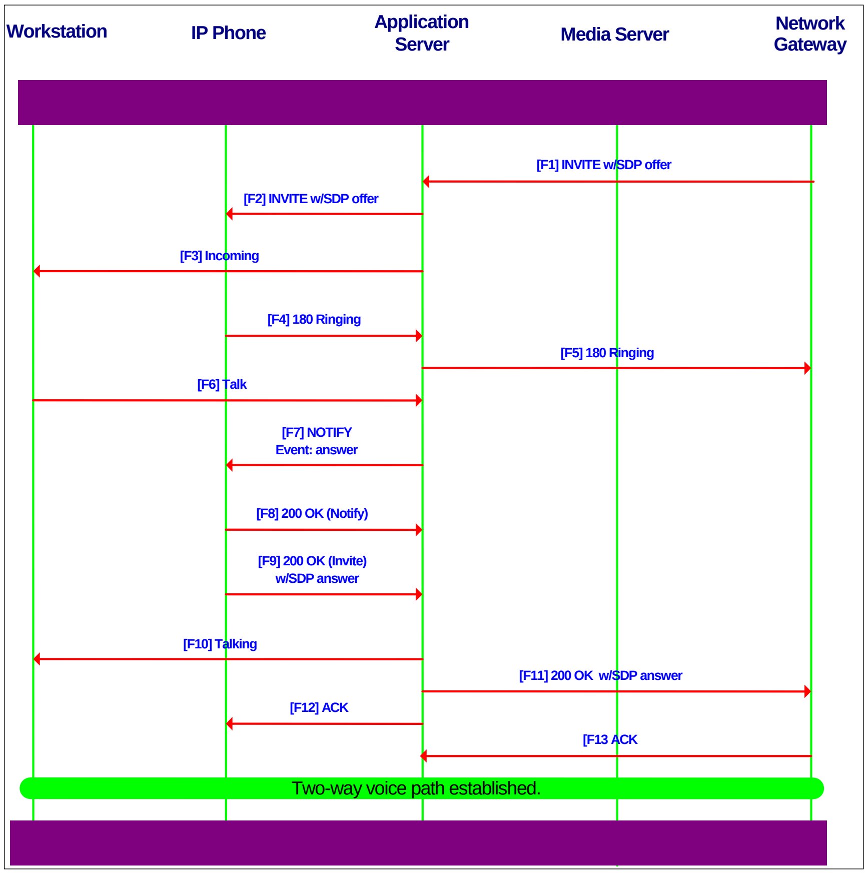

Figure 4 Click-To-Answer Call Flow

The call arrives at the Application Server in [F1] as usual. The Application Server sends an INVITE to the IP phone registered against the user profile in [F2].

```
[F2] INVITE Application Server -> IP phone
INVITE sip:3015551000@ipphone.com SIP/2.0
Via: SIP/2.0/UDP a1.foo.com:5060;branch=m9hG4bK74bf9
Max-Forwards: 70
From: “Joe” <sip:3015551212@a1.foo.com>;tag=5fxced76s2
To: “Jane” <sip:3015551000@a1.foo.com>
Call-ID: 98482762982201885l2@a1.foo.com
Call-Info:  <sip:as.foo.com>;appearance-index=1
CSeq: 43234 INVITE
Contact: <sip:a1.foo.com:5060>
Allow: ACK, BYE, CANCEL, INFO, INVITE, OPTIONS, PRACK, REFER
Supported:  100rel, timer
Content-Type:  application/sdp
Content-Length: 192

v=0
o=foo-UA 5184 14642 IN IP4 networkgateway.com
s=SIP Call
c=IN IP4 networkgateway.com
t=0 0
m=audio 27192 RTP/AVP 0 8 18
a=rtpmap :0 PCMU/8000
a=rtpmap :8 PCMA/8000
a=rtpmap:10 G729/0000
```

At about the same time, the Application Server sends a notification [F3] to the call client on the workstation, indicating that a call has arrived. The call client presents this to the end user. At about the same time as the user sees the call client present the incoming call on the desktop, the phone on his/her desk starts to alert and the phone responds to the INVITE with a 180 Ringing in [F4].

```
[F4] SIP/2.0 180 Ringing
Via: SIP/2.0/UDP a1.foo.com:5060;branch=m9hG4bK74bf9
From: “Joe” <sip:3015551212@a1.foo.com>;tag=5fxced76s2
To: “Jane” <sip:3015551000@a1.foo.com>;tag=763jxd879sa
Call-ID: 98482762982201885l2@a1.foo.com
Call-Info:  <sip:as.foo.com>;appearance-index=1
CSeq: 43234 INVITE
Contact: <sip: 3015551000@ipphone.com:5060>
Allow-Events:hold,talk
Content-Length: 0
```

The 180 Ringing progress is passed on to the network gateway in [F5]. The *Allow-Events* header in the 180 Ringing indicates to the Application Server that this endpoint supports remote call control primitives. When the user selects the incoming call and clicks on “talk” – an event is sent from the call client to the Application Server in [F6], indicating that the user is requesting that the incoming call be answered. The Application Server reacts by sending a NOTIFY to the SIP phone in [F7].

```
[F7] NOTIFY Application Server -> IP Phone
NOTIFY sip:3015551000@ipphone.com SIP/2.0
Via: SIP/2.0/UDP a1.foo.com:5060;branch=m9hG4bK74bf9
From: “Joe” <sip:3015551212@a1.foo.com>;tag=5fxced76s2
To: “Jane” <sip:3015551000@a1.foo.com> ; tag=432d33
Call-ID: 98482762982201885l2@a1.foo.com
CSeq: 434 NOTIFY
Contact: <sip:a1.foo.com:5060>
Event: talk
```

```
Subscription-State: active
Contact: <sip:a1.foo.com:5060>
Content-Length: 0
```

The IP phone indicates that it honors the request by responding to the NOTIFY with a 200 OK.

NOTE: For brevity, the authorization procedure is not shown in the figure. The phone should always challenge an incoming NOTIFY for credentials that authorize the notifier to perform remote call control.

```
[F8] 200 OK IP Phone -> Application Server
SIP/2.0 200 OK
Via: SIP/2.0/UDP a1.foo.com:5060;branch=m9hG4bK74bf9
From: “Joe” <sip:3015551212@a1.foo.com>;tag=5fxced76s2
To: “Jane” <sip:3015551000@a1.foo.com> ; tag=432d33
Call-ID: 9848276298220188512@a1.foo.com
CSeq: 434 NOTIFY
Contact: <sip:a1.foo.com:5060>
Content-Length: 0
```

The phone then automatically answers the incoming call by forcing off-hook and activating the speaker.

5.3 Event Package Names

The event package name is called “talk”. It is carried in the *Event and Allow-Events* header as defined in *RFC 3265*.

5.4 Event Package Parameters

This event package does not define any event package parameters.

5.5 SUBSCRIBE Bodies

The *talk* event package does not support dynamic subscriptions through SUBSCRIBE requests.

5.6 Subscription Duration

Subscriptions are implied by sessions established through INVITE requests and last throughout the duration of the session. When a UAC sends an INVITE to a UAS, it adds an *Allow-Events* header to the request, indicating all of the event packages it supports. This implicitly creates the subscription between the UAC and the UAS. When a UAS responds to the INVITE with an 18x provisional response or a 200 OK response, it adds an *Allow-Events* header indicating all of the event packages it supports. This implicitly creates the subscription between the UAS and the UAC. These subscriptions last until the session has been terminated with a BYE or CANCEL transaction.

5.7 NOTIFY Bodies

A NOTIFY for a *talk* package MUST NOT contain a body. No body is required for the *talk* event package as they have trivial state.

5.8 Subscriber Generation of SUBSCRIBE Requests

Subscribers should not generate SUBSCRIBE requests. Subscriptions are granted to subscribers implicitly by the establishment of sessions between two user agents.

5.9 Notifier Processing of SUBSCRIBE Requests

Notifiers should not process SUBSCRIBE requests. If received, they should be rejected.

5.10 Subscriber Processing of NOTIFY Requests

In general, subscriber processing of NOTIFY requests MUST follow all rules with respect to subscriber behavior, as described in RFC 3265. The SIP Specific Event Notification framework expects packages to specify how a subscriber processes NOTIFY requests and, in particular, how it uses the NOTIFY requests to construct a coherent view of the state of the subscribed resource.

Because these event notifications have the ability to remotely control the user's handset behavior, the subscriber (for example, the phone) SHOULD perform some level of authorization before processing NOTIFYs. The subscriber MAY perform implicit authorization by looking at the source IP address of the NOTIFY request event and matching it to any one of the IP addresses of the notifier (in this case, the Application Server) as resolved through DNS. The subscriber MAY perform explicit authentication by challenging the notifier, by using a "401 Unauthorized" response. If the subscriber has also registered a location for this address of record, then the notifier SHOULD use those same credentials to answer the notification challenge.

When a subscriber receives a talk event notification for a particular dialog, it must apply the "talk" action to the local session in progress. If the session is in the alerting state, then the subscriber should attempt to answer the session, by forcing the call off hook and turning on the speakerphone (if supported). If the session is in the held state, then the subscriber should attempt to retrieve the call. If the subscriber receives a talk notification for a session that is not either in the alerting state or the held state, then the talk notification should be rejected.

Note that the subscriber should always respond to the NOTIFY transaction if it was received and before the action is complete. A 200 OK response to a NOTIFY transaction only indicates that the notification has been received, and not that any action associated with the notification was successfully completed.

5.11 Handling of Forked Requests

NOTIFYs should not be forked.

5.12 Rate of Notifications

The rate of notifications is dictated by the end-user interface on the call control client and can vary from application to application. Endpoints should be prepared to accept NOTIFY requests as frequently as once every two or three seconds.

5.13 Notifier Generation of NOTIFY Requests

Notifications are generated for the *talk* package whenever a remote call control application requests that an alerting session be answered, or a held session be retrieved. Notifiers should not generate a talk event notification for a session that is already active and streaming media.

6 Remote Control Hold Event Package

6.1 Overview

When providing CTI based applications such as desktop call control clients, a service delivery platform must be able to remotely request that an endpoint put a call on hold. This extension provides the ability for an Application Server to send an asynchronous NOTIFY event to an endpoint using an existing INVITE dialog. When received, the endpoint can elect to honor the request and put a call on hold, without local user intervention.

The "hold" event package is an instantiation of the SIP Specific Event Notification Framework (as defined in *RFC 3265*). Following is a description of the event package, as per *RFC 3265*.

6.2 Example Message Flow

The following diagram shows the call flow as a result of a user performing a click-to-hold function on an active call from the PSTN.

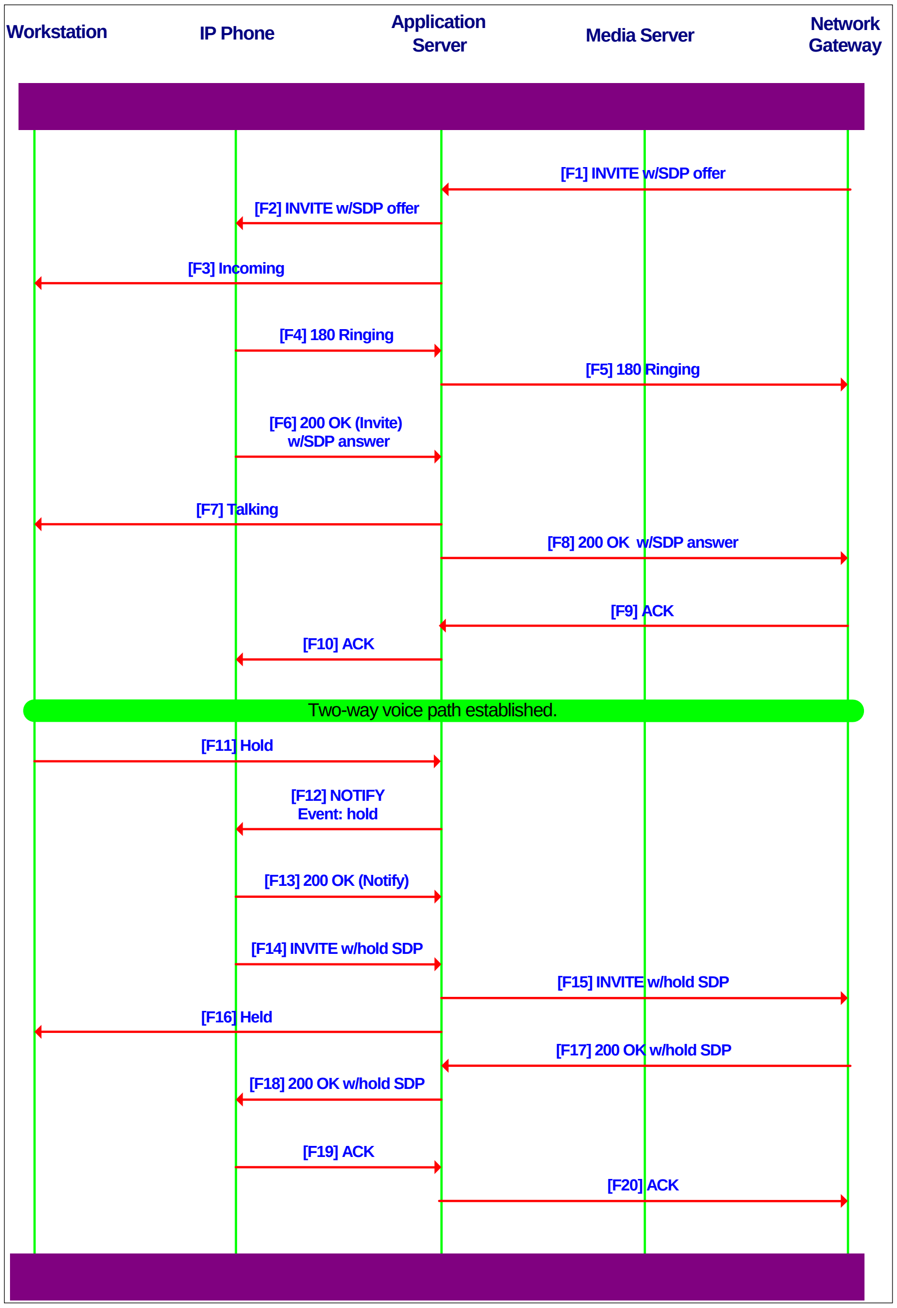

Figure 5 Click-To-Hold Call Flow

The call is set up as usual in [F1-F10] and a two-way audio is established. Now in [F11] the user at the call client decides to remotely hold the call and sends a "hold" request to the Application Server. The Application Server discovered that the endpoint supported the *hold* event package through an *Allow-Events* header in the 180 Ringing provisional response at [F4]. The Application Server reacts by sending a NOTIFY with a hold event in [F12]. Note that this NOTIFY is sent using the same dialog as the session it is acting on.

```
[F12] NOTIFY Application Server -> IP phone
NOTIFY sip:3015551000@ipphone.com SIP/2.0
Via: SIP/2.0/UDP a1.foo.com:5060;branch=m9hG4bK74bf9
From: “Joe” <sip:3015551212@a1.foo.com>;tag=5fxced76s2
```

```
To: “Jane” <sip:3015551000@a1.foo.com> ; tag=432d33
Call-ID: 9848276298220188512@a1.foo.com
CSeq: 444 NOTIFY
Contact: <sip:a1.foo.com:5060>
Event: hold
Subscription-State: active
Contact: <sip:a1.foo.com:5060>
Content-Length: 0
```

The IP phone indicates that it honors the request by responding to the NOTIFY with a 200 OK.

NOTE: For brevity, the authorization procedure is not shown in the diagram. The phone should always challenge an incoming NOTIFY for credentials that authorize the notifier to perform remote call control.

```
[F13] 200 OK IP Phone -> Application Server
SIP/2.0 200 OK
Via: SIP/2.0/UDP a1.foo.com:5060;branch=m9hG4bK74bf9
From: “Joe” <sip:3015551212@a1.foo.com>;tag=5fxced76s2
To: “Jane” <sip:3015551000@a1.foo.com> ; tag=432d33
Call-ID: 9848276298220188512@a1.foo.com
CSeq: 444 NOTIFY
Contact: <sip:a1.foo.com:5060>
Content-Length: 0
```

The phone then performs the call-hold action.

6.3 Event Package Names

The event package name is called “hold”. It is carried in the *Event and Allow-Events* header, as defined in *RFC 3265*.

6.4 Event Package Parameters

This event package does not define any event package parameters.

6.5 SUBSCRIBE Bodies

The *hold* event packages does not support dynamic subscriptions through SUBSCRIBE requests.

6.6 Subscription Duration

Subscriptions are implied by sessions established through INVITE requests and last throughout the duration of the session. When a UAC sends an INVITE to a UAS, it adds an *Allow-Events* header to the request, indicating all of the event packages it supports. This implicitly creates the subscription between the UAC and the UAS. When a UAS responds to the INVITE with an 18x provisional response or 200 OK response, it adds an *Allow-Events* header, indicating all of the event packages it supports. This implicitly creates the subscription between the UAS and the UAC. These subscriptions last until the session has been terminated with a BYE or CANCEL transaction.

6.7 NOTIFY Bodies

A NOTIFY for a *hold* package MUST NOT contain a body. No body is required as they have trivial state.

6.8 Subscriber Generation of SUBSCRIBE Requests

Subscribers should not generate SUBSCRIBE requests. Subscriptions are granted to subscribers implicitly, by the establishment of sessions between two user agents.

6.9 Notifier Processing of SUBSCRIBE Requests

Notifiers should not process SUBSCRIBE requests. If received, they should be rejected.

6.10 Subscriber Processing of NOTIFY Requests

In general, subscriber processing of NOTIFY requests MUST follow all rules with respect to subscriber behavior, as described in RFC 3265. The SIP Specific Event Notification framework expects packages to specify how a subscriber processes NOTIFY requests and, in particular, how it uses the NOTIFY requests to construct a coherent view of the state of the subscribed resource.

Because these event notifications have the ability to remotely control the user's handset behavior, the subscriber (for example, the phone) SHOULD perform some level of authorization before processing NOTIFYs. The subscriber MAY perform implicit authorization, by looking at the source IP address of the NOTIFY request event and matching it to any one of the IP addresses of the notifier (in this case, the Application Server) as resolved through DNS. The subscriber MAY perform explicit authentication by challenging the notifier, by using a "401 Unauthorized" response. If the subscriber has also registered a location for this address of record, then the notifier SHOULD use those same credentials to answer the notification challenge.

When a subscriber receives a hold event notification for a particular dialog, it must apply the "hold" action to the local session in progress. If the session is in the active state, then it should be held. If it is in any other state, the request should be rejected.

Note that the subscriber should always respond to the NOTIFY transaction if it was received and before the action is complete. A 200 OK response to a NOTIFY transaction only indicates that the notification has been received, and not that any action associated with the notification was successfully completed.

6.11 Handling of Forked Requests

SUBSCRIBEs and NOTIFYs should not be forked.

6.12 Rate of Notifications

The rate of notifications is dictated by the end-user interface on the call control client and can vary from application to application. Endpoints should be prepared to accept NOTIFY requests as frequently as once every two or three seconds.

6.13 Notifier Generation of NOTIFY Requests

Notifications are generated for the *hold* package whenever a remote call control application requests that an active session be held. Notifiers should not generate a hold event notification for a session that is already in the held state.

7 Index