# Cisco BroadWorks

# Shared Call Appearance

## Interface Specification

Release 23.0

Document Version 2

## Notification

The BroadSoft BroadWorks has been renamed to Cisco BroadWorks. Beginning in September 2018, you will begin to see the Cisco name and company logo, along with the new product name on the software, documentation, and packaging. During this transition process, you may see both BroadSoft and Cisco brands and former product names. These products meet the same high standards and quality that both BroadSoft and Cisco are known for in the industry.

## Copyright Notice



## Trademarks

## Document Revision History

| Release | Version | Reason for Change | Date | Author |
|---|---|---|---|---|
| 14.0 | 1 | Created document. | October 31, 2007 | Martin Perron |
| 14.0 | 1 | Put new document in template. | November 28, 2007 | Roberta Boyle |
| 14.0 | 1 | Edited and published document. | December 3, 2007 | Andrea Fitzwilliam |
| 15.0 | 1 | Updated document for Releases 14.sp1 through 14.sp6 and Release 15. | June 22, 2008 | Martin Perron |
| 15.0 | 1 | Edited and published document. | June 22, 2008 | Patricia Renaud |
| 16.0 | 1 | Updated document for Release 16.0. Introduced dialog event package and *Join/Replaces* headers. | April 21, 2009 | Jocelyn Lepage |
| 16.0 | 1 | Edited changes and published document. | July 14, 2009 | Andrea Fitzwilliam |
| 17.0 | 1 | Updated document for Release 17.0. | April 21, 2009 | Martin Perron |
| 17.0 | 1 | Edited changes and published document. | April 1, 2010 | Andrea Fitzwilliam |
| 18.0 | 1 | Updated document for Release 18.0. | November 4, 2011 | Martin Chabbert |
| 18.0 | 1 | Edited changes and published document. | November 25, 2011 | Jessica Boyle |
| 19.0 | 1 | Updated document for Release 19.0. | October 2, 2012 | Martin Chabbert |
| 19.0 | 1 | Updated section *6.1.4 Race Condition on SUBSCRIBE* for EV 175112. | October 4, 2012 | Veit Gentry |
| 19.0 | 1 | Edited changes and published document. | October 16, 2012 | Patricia Renaud |
| 19.0 | 2 | Corrected call flow for Shared Call Appearance (SCA) Retrieve using *Replaces* header for EV 184045. | July 23, 2013 | Doug Sauder |
| 19.0 | 2 | Edited changes and published document. | November 1, 2013 | Joan Renaud |
| 20.0 | 1 | Updated document for Release 20.0. | October 11, 2013 | Martin Chabbert |
| 20.0 | 1 | Edited changes and published document. | November 4, 2013 | Joan Renaud |
| 21.0 | 1 | Updated document for Release 21.0. | November 24, 2014 | Michael Boyle |
| 21.0 | 1 | Updated the BroadSoft legal notice and edited changes. | November 26, 2014 | Joan Renaud |
| 21.0 | 1 | Rebranded and published document. | December 8, 2014 | Joan Renaud |
| 21.0 | 2 | Updated server icons and published document. | March 3, 2015 | Joan Renaud |
| 22.0 | 1 | Updated document for Release 22.0. | December 15, 2016 | Joan Renaud |
| 22.0 | 1 | Edited changes and published document. | December 15, 2016 | Joan Renaud |
| 23.0 | 1 | Updated document for Release 23.0. | September 10, 2018 | Eric Bernier |
| 23.0 | 1 | Rebranded document for Cisco.<br>Added *Acronyms and Abbreviations*.<br>Edited changes and published document. | November 15, 2018 | Joan Renaud |
| 23.0 | 2 | Completed rebranding for Cisco and republished document. | March 8, 2019 | Jessica Boyle |

## Table of Contents

## Table of Figures

# 1 Interface Changes

## 1.1 Changes for Release 23.0, Document Version 2

Completed rebranding for Cisco.

## 1.2 Changes for Release 23.0, Document Version 1

There are no interface modifications for Release 23.0.

## 1.3 Changes for Release 22.0, Document Version 1

There are no interface modifications for Release 22.0.

## 1.4 Changes for Release 21.0, Document Version 2

Added rebranded server icons.

## 1.5 Changes for Release 21.0, Document Version 1

There are no interface modifications for Release 21.0.

## 1.6 Changes for Release 20.0, Document Version 1

Updated sections *6.4 Functional Overview* and *7.3 Functional Overview* to add the description for Silent Monitoring capabilities.

## 1.7 Changes for Release 19.0, Document Version 2

The following changes were made in this document version:

- Corrected the call flow for Call Retrieve using *Replaces* header (section *7.3.7 A-2 Retrieves B from Hold*).

## 1.8 Changes for Release 19.0, Document Version 1

There are no interface modifications for Release 19.0.

## 1.9 Changes for Release 18.0

In this release, a new type of content is added for both dialog and call-info event packages, to indicate the parked call state.

## 1.10 Changes for Release 17.0

There are no interface modifications for Release 17.0.

## 1.11 Changes for Release 16.0

This version of the document introduces the dialog event package (*RFC 4235*), the Session Initiation Protocol (SIP) *join* header (*RFC 3911*), and the SIP *Replaces* header (*RFC 3891*) as an alternative to the Call-Info event package, line-seize event package, and the *Call-Info SIP* header.

## 1.12 Changes for Release 15.0

There are no interface modifications for Releases 14.sp1 through 14.sp6 and for Release 15.0.

## 1.13 Changes for Release 14.0

The following interface modifications occurred for Release 14.0:

- Cisco BroadWorks added support for a station to barge in to a call in progress.

## 1.14 Changes from Previous Releases

The following interface modifications occurred in previous releases:

- Cisco BroadWorks added support for a call on a shared line to be privately held by one device, such that it cannot be retrieved by another device via a shared line.
- Cisco BroadWorks added Calling Line ID (CLID) support for call appearances.
- Cisco BroadWorks added support for multiple call appearances on a single shared line.

# 2 Introduction

The ability to support SCAs is a fundamental building block for a variety of enhanced telephony services. Features such as 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* [1]) by itself offers no inherent semantics for supporting Shared Call Appearance features, when coupled with an appropriate instantiation of the "SIP-specific Event Notification" framework (*RFC 3265* [3]), these services can be deployed quite easily in a distributed network.

The purpose of this document is to present general requirements for access devices to support Shared Call Appearance-enabled features hosted on a Cisco BroadWorks Application Server using SIP as the line-side access protocol. Note that the Cisco BroadWorks Application Server uses a SIP back-to-back user agent model for service delivery, and this model is assumed in the Shared Call Appearance mechanism proposed. The Cisco BroadWorks Application Server is the customer-facing and endpoint-hosting component of the overall Cisco BroadWorks service delivery platform as shown in the following figure.

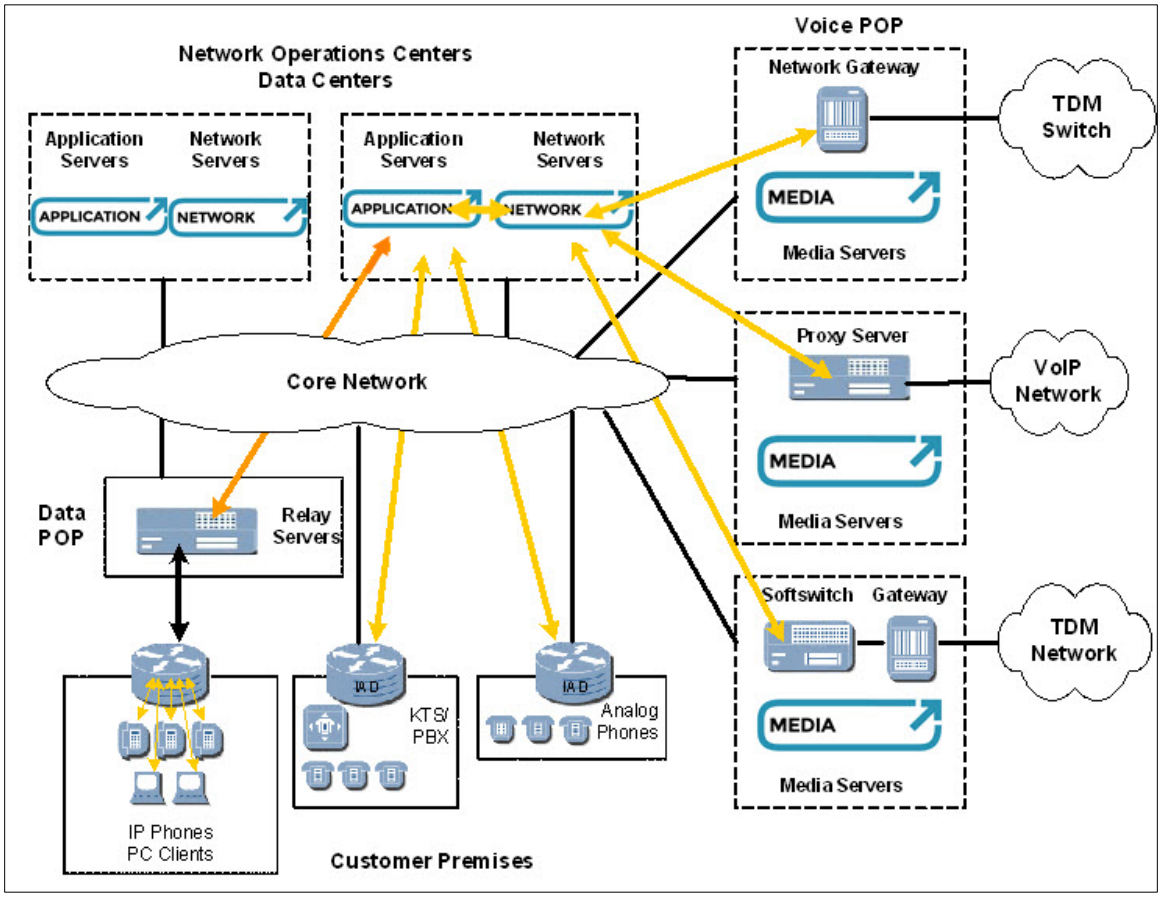


Figure 1 Cisco BroadWorks Service Delivery Platform

This document provides an overview of the two target applications that require Shared Call Appearance functionality. These applications are part of an upcoming release of BroadWorks, but they require corresponding Internet Protocol (IP) phone support before they can be delivered. The rest of this document serves to outline these requirements on IP phones and describe how they must interoperate with BroadWorks to enable shared call appearances.

# 3 Application Overview

There are two primary applications that drive the requirements for Shared Call Appearance functionality on Cisco BroadWorks: Key System Emulation and Attendant Console. Delivering these features is primarily the responsibility of the Cisco BroadWorks Application Server and the detailed requirements are specified in corresponding application requirements documents. These applications are summarized in the following sections to aid in the understanding of the shared call appearance building block requirements.

## 3.1 Key System Emulation

Cisco BroadWorks offers key system emulation similar to that found in legacy time division multiplexing (TDM)-based key systems. In this application, one or more lines are shared across many "stations". The classic deployment model is found in the retail market. A department store may have a number of phone lines brought into a customer premises equipment (CPE)-based key system. Attached to the key system are handsets. Usually there is a many-to-one ratio of handsets to phone lines.

*Figure 2 Legacy Key System* shows a legacy key system with three lines and six station handsets. When calls arrive on any one of the three lines, all stations are alerted. Each handset has a separate line key for each of the available lines. The line key is used to access the shared call appearance on the corresponding line. The call appearance is "shared" because it is available on all handsets.

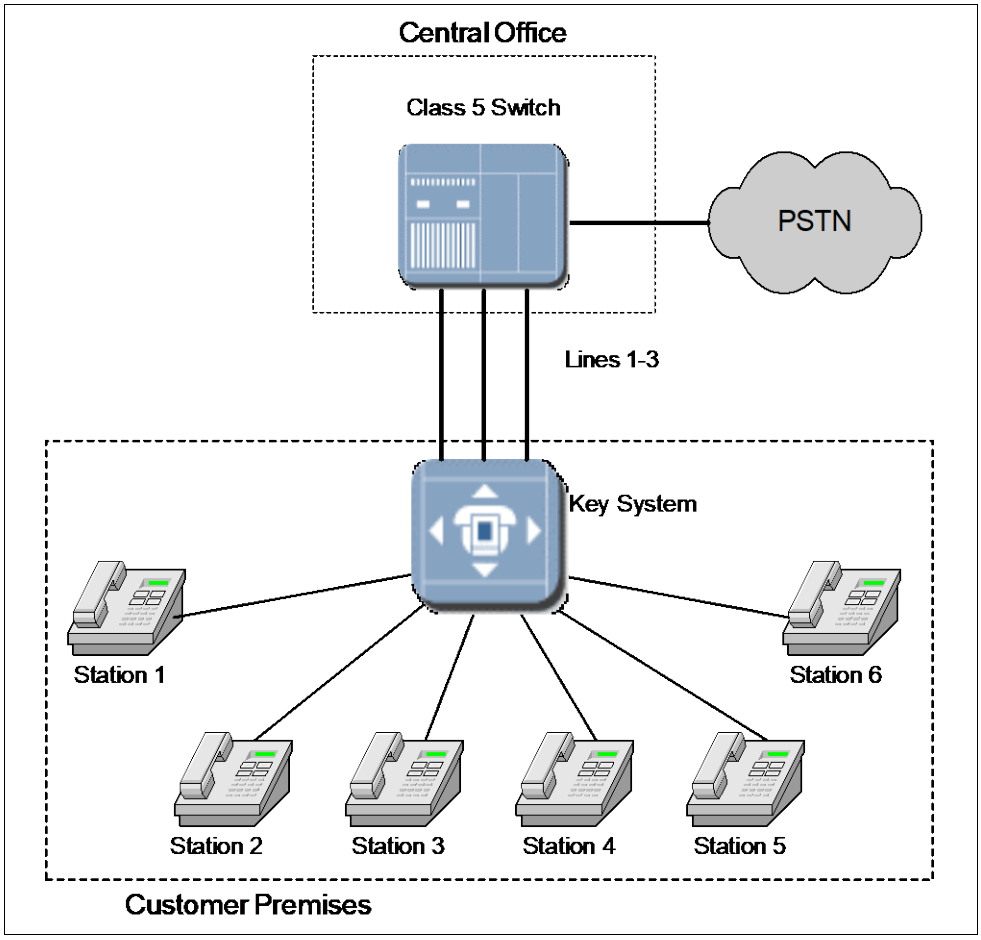


Figure 2 Legacy Key System

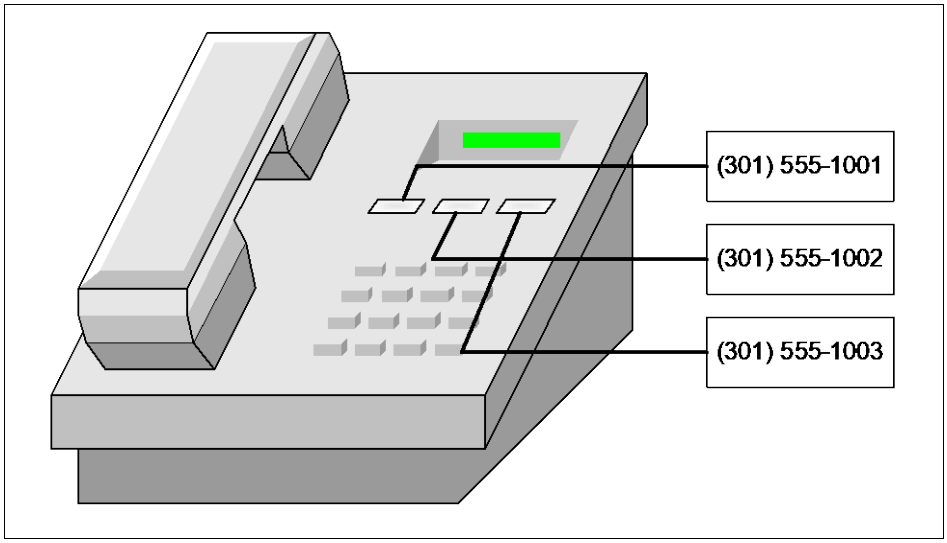


Figure 3 Station Handset with Line Keys

Some common features found on legacy key systems include the following:

- Users can:
  - Place and receive call appearances from any station on any one of the three lines.
  - Monitor the “availability” of each line from any one of the stations.
  - Place call appearances on hold from one station and retrieve them from another station.
- Lines can be configured to “roll over” in sequence when a line is busy.
- Privacy release allows stations to “barge in” on calls in progress.
- Station-to-station intercom allows stations to be used like intercoms without consuming any available lines.
- Paging allows stations to be used to broadcast announcements to all other stations.

Many of the features and attributes of legacy key systems are artifacts of the restrictions of the circuit-switched technology available when the systems were developed. The concept of a line for instance, is not as important in a packet-based Voice over IP (VoIP) network. However, since the user base has become accustomed to these concepts and trained to use this “line-based” model, it is important to emulate as much as possible the same user interactions when deploying next generation networks.

*Figure 4* shows a Cisco BroadWorks network with key system emulation. Key System Emulation in Cisco BroadWorks uses SIP IP phones as station handsets. These are analogous to legacy key system stations. The concept of a line is modeled in software on the Cisco BroadWorks Application Server. In *Figure 4*, an enterprise is assigned the Key System service. The service is then provisioned with three lines and six handsets. The Application Server handles all policies associated with delivering calls to each of the handsets through each of the corresponding lines. The end result to the user is a similar interaction experience found on a Legacy Key System.

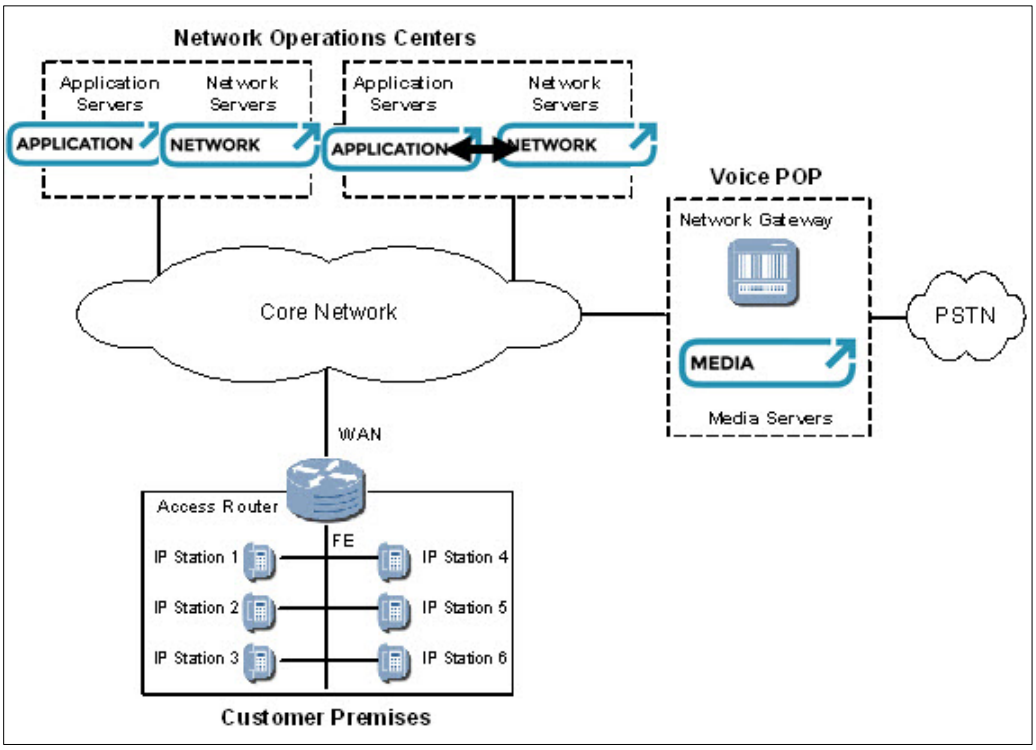


Figure 4 Cisco BroadWorks Network with Key System Emulation

## 3.2 Attendant Console

The Attendant Console application allows a station in the network to monitor the call state of other stations in the network. Conventionally, an executive assistant or “front desk” operator uses this application. The operator is equipped with an enhanced station that offers enough line keys to adequately monitor a large set of lines in the network.

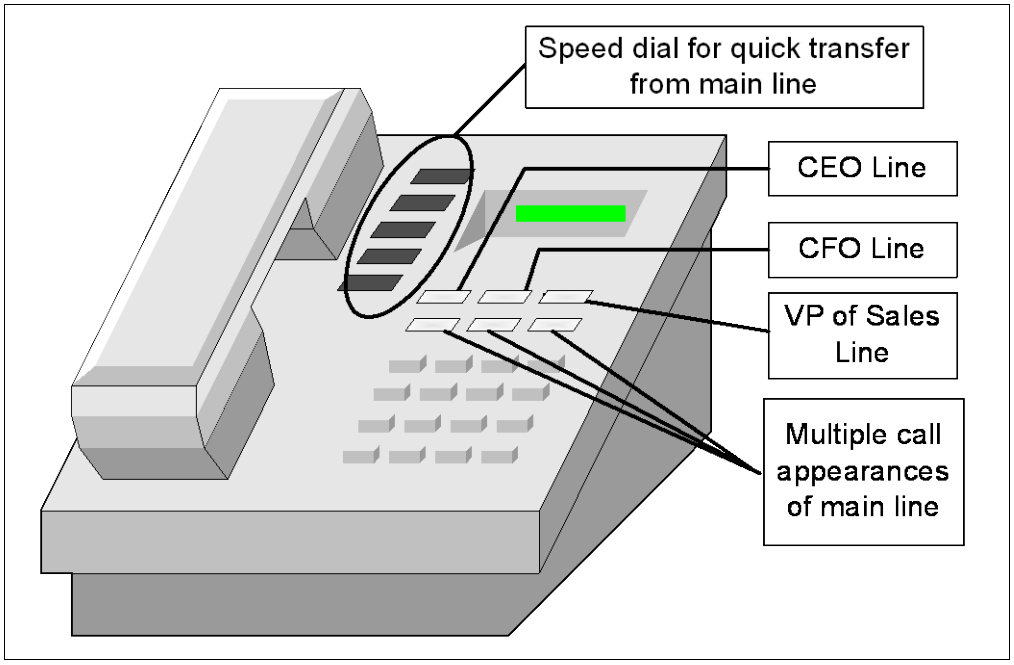


Figure 5 Attendant Console

An Attendant Console can have one or more lines that are configured to roll over from one to another. The purpose is to use these lines for all incoming calls that are managed by the administrator. Calls can be transferred from one of the main lines to the destination extension in the enterprise. Speed dial buttons can be configured to facilitate transfer interactions. When calls arrive for a user who has a line that is being "monitored" by the Attendant Console, the operator can easily determine if the user is busy (by looking at the lamp associated with the line key of that user) and make appropriate call routing decisions.

An Attendant Console can also host expansion modules with additional keys. Keys on an expansion module are typically configured for speed dial and allow the attendant to monitor a large number of users.

In legacy systems, Attendant Consoles are typically attached to a CPE-based private branch exchange (PBX) that is connected to all the other stations in the enterprise. The Attendant Console cannot monitor stations that are not directly attached to the same PBX. Using the hosted features in Cisco BroadWorks, it is possible to not only offer the classic Attendant Console feature set, but also to offer them across geographically dispersed networks with heterogeneous access devices.

# 4 IP Phone

*Figure 6* provides a hypothetical layout of an IP phone. It is not intended to represent a specific make or model, but rather a generic IP phone that incorporates the variety of features found on emerging IP phones in the industry. The definitions of the various keys and displays are provided following *Figure 6* and these terms are used throughout this document.

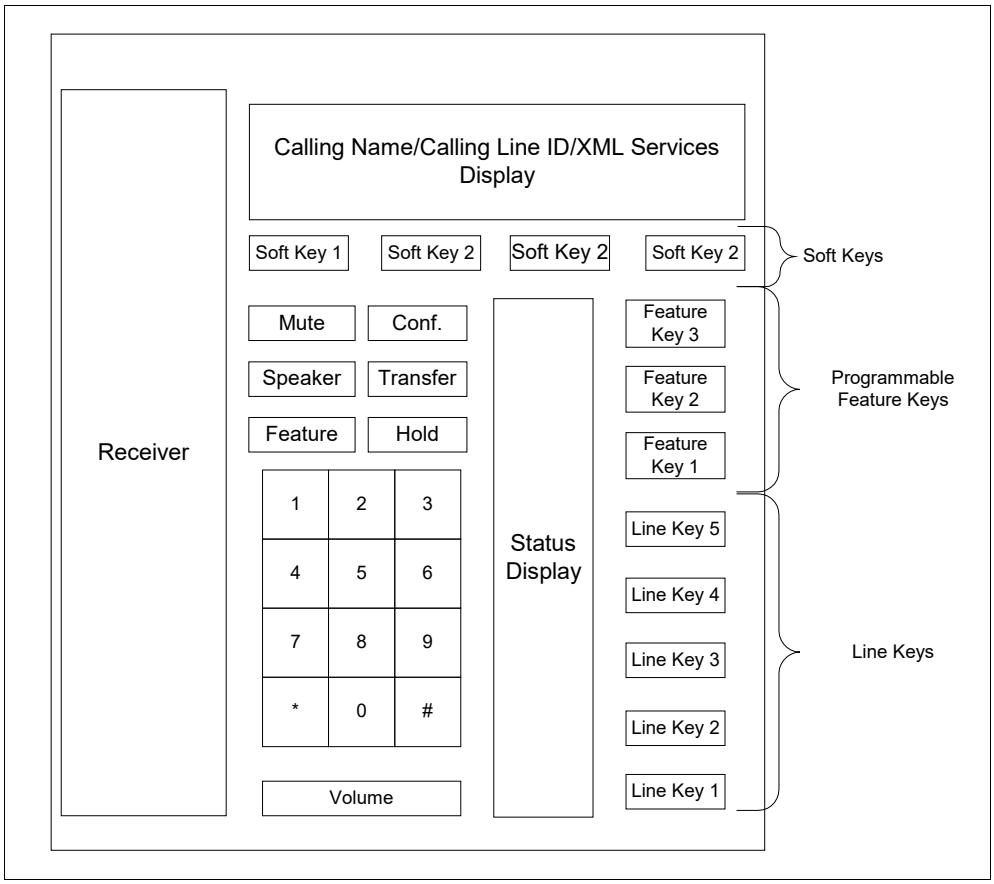


Figure 6 Layout of IP Phone

**Built-In Feature Keys:** IP phones are smart endpoints with a variety of local features that can be invoked directly on the phone through “built-in feature keys”. The most common built-in features are “Hold”, “Transfer”, “Mute” and “Speaker” (or “Hands-free”). These features are said to be “built-in” because the service function itself is implemented locally on the phone. Standard mechanisms for supporting these types of services are well-defined in the SIP community, and are not described in this document.

**Programmable Feature Key:** Some IP phones have "programmable feature keys", which allow the user to either invoke a service, or toggle it on and off. Because these keys are programmable, they have removable caps or labels that can be changed to reflect the associated service. Feature keys also have a "status indicator" that provides a visual queue of the state of the corresponding service. The status indicator is either a lamp or backlighting against the key itself, or a line on a liquid crystal display (LCD) adjacent to the key. Some IP phones allow feature keys to be programmed with services implemented locally on the phone or with remote services implemented in the network. Mechanisms for supporting programmable feature keys are outside of the scope of this document.

**Soft Key:** Almost all IP phones have some form of "soft key". Soft keys are like programmable feature keys, but instead of having labels or caps that must be physically replaced on the phone, the keys are unmarked, and a corresponding label is dynamically displayed on an adjacent LCD based on the context of the user interaction with the phone. Soft keys are tightly integrated with an eXtensible Markup Language (XML) presentation language used to manage the contents of the LCD. The mechanisms for supporting soft keys and the corresponding XML presentation language are described in the *BroadWorks Busy Lamp Field Interface Specification* [10].

**Line:** The concept of a "line" originates from circuit-switched networks. Lines do not really exist in packet-switched networks, but the concept is still useful when describing phones. For the purpose of this document, one address-of-record as defined in *RFC 3261* [1] represents one line. When a user registers with a SIP network, the *To* header containing their address-of-record identifies their line. Lines can have multiple call appearances. Lines can be qualified as being either "private" or "shared".

**Call Appearance:** Every call is associated with a specific line. The presentation of a call on a line is a "call appearance".

**Line Key:** This is a bit of a misnomer, but it is commonly accepted that a line key is any key on an IP phone that is used to present and manage call appearances for a specific line or address-of-record. Some phones can have multiple line keys per line, making management of multiple call appearances intuitive, as each new call on the line has a separate line key associated with it. Some phones use only one line key per line and use soft keys to manage multiple call appearances. Line keys can also have a status indicator that provides a visual queue of the state of the call appearance on the given line. The status indicator can be a lamp, a light emitting diode (LED), or backlighting against the key. Alternatively, an LCD adjacent to the key is used.

**Private Call Appearance:** A private call appearance is any call appearance that is only visible and accessible through the original endpoints involved in setting up the call.

**Private Line:** A private line is a line that is only presented with private call appearances.

**Shared Call Appearance:** A shared call appearance is any call appearance that is visible and optionally accessible through the original endpoints in the call as well as an authorized set of other endpoints in the network. This document describes mechanisms for supporting shared call appearances in SIP.

**Shared Line:** A shared line is a line that is only presented with shared call appearances. This document describes mechanisms for delivering shared call appearances on a shared line in SIP.

# 5 General Protocol Requirements

## 5.1 Assumptions

The mechanism described assumes that all endpoints in the shared call appearance group are hosted by an Application Server acting as a third-party call control element. "Hosted" means that:

- All endpoints register their locations with the third-party controlling element (directly or indirectly).
- All outbound calls traverse the third-party controlling element immediately after leaving the endpoint.
- All inbound calls traverse the third-party controlling element before being delivered to the endpoint.

The general approach is focused on keeping the protocol elements and the semantics for using them simple. SIP endpoints tend to be constrained by memory and processing power. Therefore, the introduction of unnecessary signaling requirements should be avoided. Another primary objective is to keep the approach as standardized as possible.

To implement Shared Call Appearance, Cisco BroadWorks provides two approaches, using distinct protocol elements. The approaches are:

- *Call-Info*: Provides lamp management, call state management, dial-tone control, hold/retrieve from distinct locations, barge-in (bridging), and private hold functions. It relies on proprietary extensions to the standard *Call-Info* header (as defined in *RFC 3261* [1]) and instantiates two SIP-specific event packages (as defined in *RFC 3265* [3]), namely "Call-Info" and "line-seize". For information on the "Call-Info" and "line-seize" event packages, see the *BroadWorks SIP Access Side Extensions Interface Specification* [11].
- *Dialog*: Provides the same function as Call-Info, with the exception of dial-tone management and with more a limited line/lamp and call state management capabilities. However, the protocol elements used by this approach are completely defined by standard-track RFCs, that is, the SIP Dialog event package (as defined in *RFC 4235* [5]), the SIP *Join* header (as defined in *RFC 3911* [6]), and the SIP *Replaces* header (as defined in *RFC 3891* [7]).

Neither one of the approaches introduces new methods, headers, or document types. The following section describes the two solutions separately.

Devices MUST support either the Call-Info or the Dialog approach. It is possible to have deployments where some devices implement the Call-Info approach while other devices implement the Dialog approach. Although not expected, it is also possible that devices combine parts of both approaches. For example, it would be perfectly acceptable for a device to implement the line-seize event package and *Call-Info* header for SIP INVITE dialogs, but it also implements the dialog event package and supports the *Join/Replaces* headers.

## 5.2 Shared Line Concept

A shared line is a concept whereby multiple endpoints are associated in a shared call appearance group for the purpose of handling incoming and outgoing calls.

For incoming call appearances, an Application Server presents the call to each endpoint in the shared call appearance group in the form of an INVITE request. 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 shared call appearance group.

For incoming calls appearances, a shared line implies forking the call request to multiple endpoints. An Application Server may implement forking at the protocol level or at the application level. With protocol-level forking, all endpoints register with the same address-of-record and the INVITE request is forked to all registered endpoints according to *RFC 3261* [1]. With application-level forking, all endpoints register with a different address-of-record. Each registered endpoint receives a unique INVITE request.

Cisco BroadWorks implements application-level forking. Having a different address-of-record for each endpoint allows the Application Server to provide device/endpoint-specific behavior as each endpoint is associated with a signaling profile on Cisco BroadWorks. This also allows a mixture of devices to be included in the forking service, allowing the Application Server to tailor the protocol messaging towards each device, as applicable.

In the context of the applications described in this guide, application-level forking also allows the endpoints to synchronize with the appearance state of *Stable*, and progressing calls on other endpoints in the group. Using the Call-Info solution, each endpoint individually subscribes to the “Call-Info” and “line-seize” event packages. Alternatively, with the Dialog solution, each endpoint subscribes to the “Dialog” event package.

# 6 “Call-Info” Approach

This section describes the “Call-Info” approach. It provides the following functions:

- Lamp management
- Call state management
- Dial-tone control
- Hold/Retrieve from distinct locations
- Barge-in (bridging)
- Private Hold

Lamp management, Call state management, and Dial-tone control use the line seizure concept, defined in section *6.1 Line Seizure*. There is no equivalent for line seizure in the Dialog approach.

## 6.1 Line Seizure

SIP endpoints are smart endpoints when compared to devices that use other line-side access protocols, such as Simple Conference Control Protocol (SCCP) or Media Gateway Control Protocol (MGCP). They provide local dial tone, digit collection, and essentially perform their own line-side call setup. This works for lines that are private to the endpoint. However, this behavior can cause “race” conditions when emulating shared line functionality across multiple endpoints.

### 6.1.1 Race Condition on INVITE

If three phones are participating in a shared line, then it is possible for a user at each phone to go off hook and dial digits simultaneously.

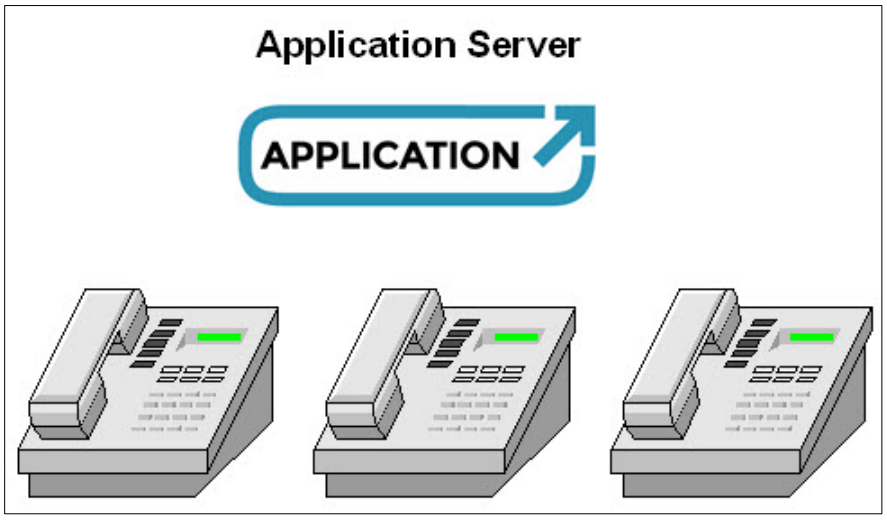


Figure 7 Three Phones Participating in a Shared Line

Each of the users thinks they have seized the line, since they have dial tone and can enter digits. As a result, three INVITE-requests arrive at the Application Server at relatively the same time. However, because they are all contending for the same resource (a call appearance on the shared line), only one of the INVITE-requests is allowed to progress. The other two INVITE-requests are rejected with a "480 Temporarily Unavailable" response. As a result, one phone receives progress indication and the other two phones receive a busy treatment.

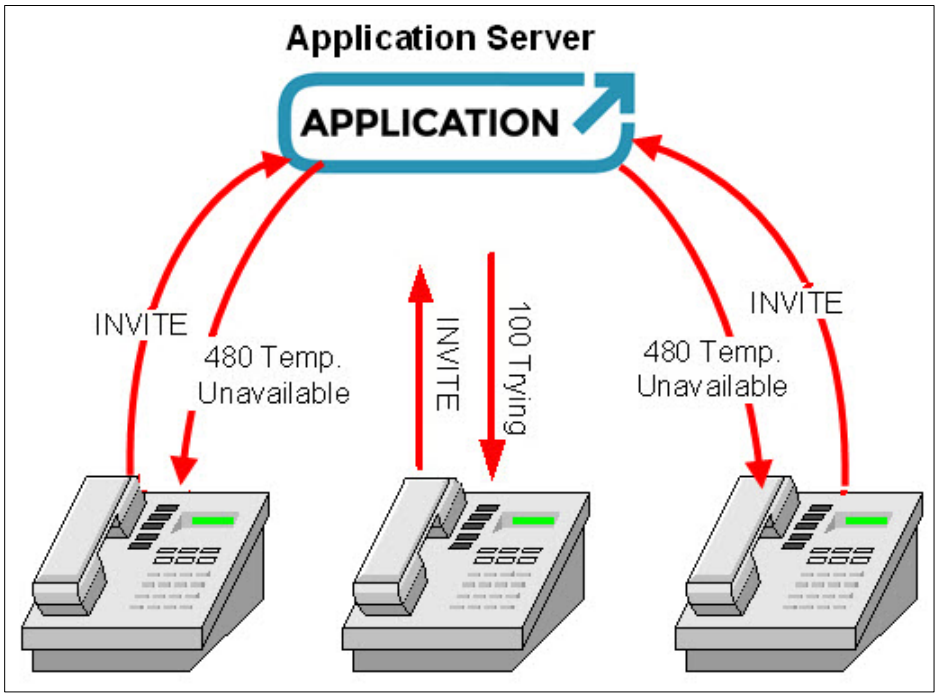


Figure 8 Race Condition on INVITE

This user experience is not positive – it is falsely perceived as a far-end busy. To resolve this, a mechanism for supporting a distributed line seizure is required. The approach defined here requires all endpoints that participate in a shared line to 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 Application Server guarantees that only one subscription to the "line-seize" event package can exist for a given shared line resource. When the Application Server grants a subscription to the *line-seize* package, it generates a NOTIFY, indicating that the subscription state is now "active". This is an indicator to the endpoint that it can safely play a dial tone to the user and collect digits. Once the digits have been collected, then the endpoint can send an INVITE-request to the Application Server. While one endpoint has obtained the *line-seize* subscription, any requests from other endpoints to use that shared line are rejected with an appropriate response.

### 6.1.2 Avoid Hung Line Seizures

To avoid scenarios in which 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* [3] explicitly describes a mechanism for negotiating subscription duration between the subscriber (the endpoint) and the notifier (the Application Server). The SUBSCRIBE-request to the "line-seize" package contains an expiration time as offered by the endpoint, and in response, the Application Server can choose to set a shorter expiration time. If the endpoint has not collected digits or 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 in which 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 be collected.

### 6.1.3 Clear Line Seizure

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 cancelling the subscription. This allows the line to be immediately seized by another endpoint. According to *RFC 3261* [1], this is effectively achieved by refreshing the subscription with an expiration of "0".

### 6.1.4 Race Condition on SUBSCRIBE

Arguably, there is still a potential race condition as this mechanism is still based on a distributed peer-to-peer model. For instance, if three phones go off hook at the same time, they send SUBSCRIBE-requests to the Application Server at roughly the same time. Note however, that the race condition is less likely, as the window of opportunity is now much smaller. The user feedback can also be delivered before dial tone is provided.

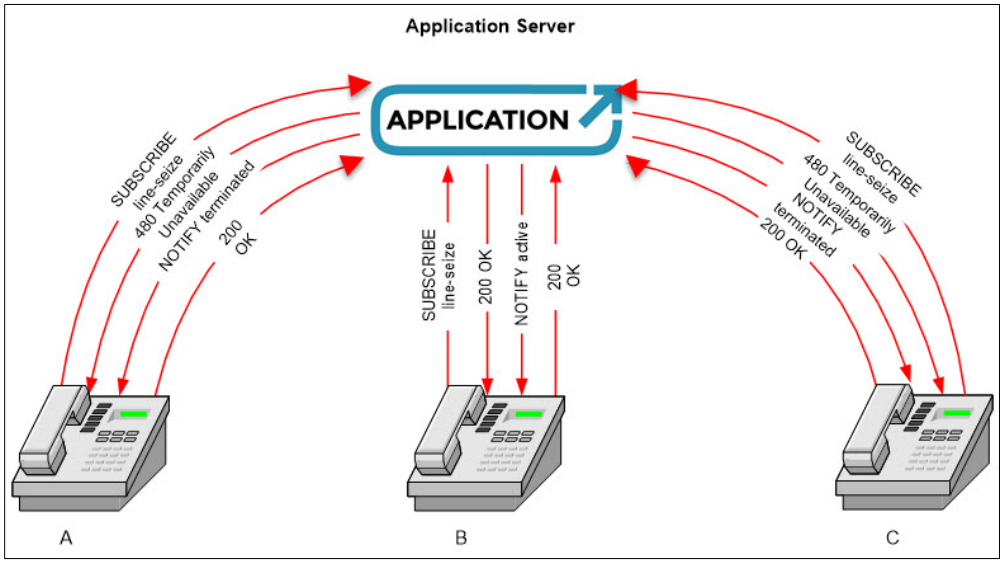


Figure 9 Race Condition on SUBSCRIBE

In the example above, B goes off hook just slightly before A and C. The SUBSCRIBE request from B is processed by that Application Server first and receives a *200 OK* response. At this point B is not only certain that the subscription request has been accepted, but that the subscription has actually been granted (indicated by the *200 OK* response). This means B can safely play dial tone to the user and start collecting digits. The other two endpoints (A and C) receive a 480 Temporarily Unavailable response to their subscription. This indicates that the subscription request has been accepted for processing, but it has not been granted. Phones A and C must wait for a corresponding NOTIFY event to determine the disposition of their subscription. While waiting for the NOTIFY, the endpoints should not present a dial tone to the user. In the example above, the NOTIFY event arrives with a subscription-state of *terminated*. This indicates that the subscription has not been granted. Endpoints A and C should then present corresponding feedback to the user, indicating that the line could not be seized. The feedback could take the form of a treatment, for example, re-order, or it could be a message presented on the LCD.

### 6.1.5 Relationship to Call-Info Event Package

Note that line seizure also represents a state change in a call appearance on the shared line, so a corresponding “Call-Info” NOTIFY-request is sent to all endpoints that participate in the shared line (as described above). The appearance-state is *seized*. These endpoints should then present the line as active and busy to the end user.

### 6.1.6 Relationship to Call-Info Header

Note that the SUBSCRIBE-request SHOULD also contain a *Call-Info* header in the same fashion as an INVITE-request (described above). The *Call-Info* header contains an “appearance-index”, which indicates the call appearance that the endpoint would present to the user if the subscription is granted. The Application Server may choose to manage multiple subscriptions on a given shared line – one for each call appearance. This is a matter of local policy as dictated by the Application Server and the profile of the shared line. For example, if policy permits, it may be possible for one endpoint to seize the first call appearance on a shared line, while another endpoint seizes the second call appearance. If the SUBSCRIBE-request contains a *Call-Info* header with an “appearance-index” information element, then all NOTIFY-requests associated with the subscription contains a corresponding *Call-Info* header.

## 6.2 Application Server Requirements

### 6.2.1 Configuration

The Application Server configuration requirements are as follows:

- The Application Server must allow an administrator to configure which users are configured with shared lines and which are configured with private lines.
- The Application Server must statically understand or be able to dynamically discover the line key and call appearance capabilities of the IP phone associated with a given user, and adjust service provisioning options to match the capabilities.
- The Application Server must generate phone configuration information based on the capabilities of the IP phone and the services provisioned against the user.

### 6.2.2 Execution

The Application Server execution requirements are as follows:

- The Application Server must allow multiple endpoints to register locations against a shared line.
- The Application Server must determine the endpoint registering against a shared line and which policies apply to it.
- The Application Server must allow multiple endpoints to SUBSCRIBE to the "Call-Info" event package on a shared line.
- The Application Server must allow only one endpoint at a time to SUBSCRIBE to the "line-seize" event package on a shared line.
- If the Application Server is configured to challenge INVITE requests from endpoints, then it must also challenge SUBSCRIBE requests from endpoints.
- If the Application Server is configured to perform access control on INVITE requests from endpoints, then it must also perform access control on SUBSCRIBE requests from endpoints.
- After granting a subscription to a "Call-Info" event package, the Application Server must initialize the IP phone to the current shared line call appearance state by sending a NOTIFY request with corresponding call appearance status in the *Call-Info* header. This applies even if all call appearances on the shared line are idle. The *Call-Info* header contains information elements that reflect the current state of all call appearances on a shared line.
- When a call arrives for a shared line, the Application Server must attempt to terminate the call on every endpoint that has successfully registered against the shared line.
- The Application Server must send NOTIFY requests to all subscribed endpoints whenever a change in call appearance state occurs for a shared line. The *Call-Info* header contains information elements that reflect the current state of all call appearances on a shared line (not just the call appearance that has a changed state).
- When one of the registered endpoints answers the call, the Application Server must CANCEL calls to all other registered endpoints. As this reflects a call appearance state change, the Application Server must then send a NOTIFY request to all subscribed endpoints with a *Call-Info* header, indicating the complete state of all call appearances on the line (not just the call appearance that has a changed state).
- When an INVITE-request arrives from an endpoint that is participating in a shared line, the Application Server must only originate the call if the INVITE-request is authorized to make a call (that is, via access control list (ACL) or an explicit INVITE challenge), and:
  - The line appearance indicated in the *Call-Info* header is currently idle.

    -or-

  - The line appearance indicated in the *Call-Info* header is currently seized and the endpoint originating the call has been granted the "line-seize" subscription for the call appearance on that shared line.

  Otherwise, the Application Server must reject the call with a "480 Temporarily Unavailable" response.
- When originating calls, the Application Server sends a NOTIFY request to all subscribed endpoints with a *Call-Info* header, indicating the complete state of all call

appearances on the shared line (not just the call appearance that has a changed state).

- When a call appearance transitions to the *seized*, *progressing*, *alerting*, or *active* state, the Application Server must send a NOTIFY-request to all endpoints that have subscribed to the "Call-Info" event package. The NOTIFY request MUST contain a *Call-Info* header, indicating the complete state of all call appearances on the line (not just the call appearance that has a changed state).
- When a call appearance is released, the Application Server must send a NOTIFY request to all subscribed endpoints with a *Call-Info* header, indicating the complete state of all call appearances on the line (not just the call appearance that has a changed state).

## 6.3 Device Requirements

### 6.3.1 Configuration

The device configuration requirements are as follows:

- The device MUST securely retrieve configuration information from a server in the hosted provider network at startup time.
- The device configuration information MUST be able to identify how many private lines and how many shared lines are to be configured on the phone.
- The device MUST allow for a separate display name, address-of-record, and credentials (user name and password) for each line (shared or private).
- The device configuration information SHOULD be able identify how many simultaneous call appearances should be accepted for each line before the phone sends back busy.

### 6.3.2 Execution

The device execution requirements are as follows:

- The device must send a separate REGISTER request for each configured line (private or shared) according to *RFC 3261* [1].
- The device must also send a separate SUBSCRIBE request for each configured shared line, according to *RFC 3265* [3]. The subscription is for the "Call-Info" event package that provides appearance state associated with the shared line.
- The device must continually refresh the registrations before they expire, according to *RFC 3261* [1].
- The device must continually refresh the "Call-Info" subscriptions before they expire, according to *RFC 3265* [3].
- The device must be prepared to provide corresponding credentials for authentication challenges for any REGISTER, INVITE, or SUBSCRIBE transactions.
- When a call arrives for one of the lines (private or shared), the phone must alert the user of the new call appearance and identify which line it is associated with.
- If the maximum number of simultaneous call appearances has been reached for a particular line, then the phone must return busy for any new call appearances that arrive for the line.
- IP phones must map the following states to a suitable appearances on the phone user interface:

- Idle
- Seized
- Progressing (outgoing call)
- Alerting (incoming call)
- Active
- Held

- Due to the distributed nature of the model, devices should assume that a transition from any to state to any other state is possible. That is, devices should not rely on any particular state machine, and always honor the state transitions indicated by NOTIFY-requests.
- When a NOTIFY arrives for one of the shared lines, the phone must perform access control and only accept the request if the source IP address corresponds to one of the IP addresses of the Application Server.
- After a NOTIFY request has been accepted, the device should modify the call appearance state of the associated line based on the body of the NOTIFY request.
- When originating new calls for private lines or shared lines, devices SHOULD include a *Call-Info* header with a corresponding appearance-index information element in the INVITE-request describing which call appearance of the line the user has chosen to originate the call on.
- When terminating calls for private lines or shared lines, devices SHOULD use the appearance-index information element from the *Call-Info* header of the incoming INVITE (if present) to decide which call appearance of the line it should use to present the call to the end user. If a *Call-Info* header is not present in the incoming INVITE, then the phone should choose the next available call appearance.
- When sending progress information (18x responses), devices SHOULD include a *Call-Info* header with a corresponding appearance-index, indicating which call appearance was used to present the alerting call to the user. This is applicable for both private and shared lines.
- When answering (200 responses) an incoming call, devices SHOULD include a *Call-Info* header with a corresponding appearance-index, indicating which call appearance was ultimately used to answer the call. This is applicable for both private and shared lines.
- The device SHOULD use the following guidelines for managing the state of each call appearance on a shared line:

**Idle:**

- When there is no active call for a specific appearance-index on a shared line, then the call appearance on that shared line is considered idle.
- If applicable, the LED for the call appearance SHOULD be OFF.
- If the user presses the line key for the call appearance, then a SUBSCRIBE-request for the *line-seize* event package should be sent to the Application Server. The SUBSCRIBE-request should have a *Call-Info* header, indicating the chosen appearance-index. If the *line-seize* subscription is granted, then the phone should play a dial tone and start collecting digits.

**Seized:**

- If one endpoint in the shared line group has been granted a *line-seize* subscription, then the corresponding appearance on the shared line is considered seized.
- If applicable, the LED of the call appearance on the endpoint that has been granted the *line-seize* subscription SHOULD be solid green, indicating that the call appearance is in use locally. The LED for the same call appearance on all other endpoints in the shared line group SHOULD be solid red, indicating that the call appearance is in use remotely.
- If the user on the endpoint that has been granted the *line-seize* subscription puts the phone back on hook, then the endpoint should cancel the line-seize subscription.
- If the user on the endpoint that has been granted the *line-seize* subscription presses the line key again, it should be treated as an "on hook", and the endpoint should cancel the *line-seize* subscription.
- If a user at one of the other endpoints (that is, endpoints that do not currently have the line seized) presses the corresponding line key, then it SHOULD either be ignored or treated as if the user wants to barge in on the call.

**Alerting (incoming call):**

- When a new call appearance arrives for a shared line, the call appearance is considered to be in the *Alerting* state.
- The phone should indicate to the user that a new call is arriving (either visually or audibly), just as it does for a private line.
- If the INVITE for the call appearance includes a *Call-Info* header with an appearance-index information element, then the device SHOULD attempt to present the call on the same relative appearance index on the phone user interface.
- If the INVITE for the call appearance includes an *Alert-Info* header, then it should be honored.
- If applicable, the LED for the call appearance SHOULD be blinking green fast.
- A device SHOULD provide a *Call-Info* header in the *18x* progress message to identify where the alerting call is being presented on the phone user interface. If this cannot be determined, then the *Call-Info* header can be omitted.
- If the user presses the corresponding line key, then the device SHOULD attempt to answer the call just as it does for a private line.
- A device SHOULD provide a *Call-Info* header in the *200 OK* answer message to identify where the call has been answered on the phone user interface.

**Active:**

- When a user answers an incoming call appearance, then it is said to be in the *Active* state.
- IP phones should indicate that the line is active visually. If applicable, the LED of the call appearance on the endpoint that has explicitly answered the call should be solid green, indicating that the call has been answered and is active locally. The LED of the call appearance on all other endpoints should be solid red, indicating that the call is active on a remote endpoint.

- If the user on the endpoint that has answered the call puts the phone back on hook, then the endpoint should release the call.
- If the user on the endpoint that has answered the call presses the line key again, it SHOULD be treated as either:
  - A request to put the call on hold if the endpoint has a separate “release” key.

  -or-

  - A request to release the call if the endpoint does not have a separate “release” key.
- If a user at one of the other endpoints (that is, endpoints that are not currently active in the call) presses the corresponding line key, then it SHOULD either be ignored or treated like the user.

**Held:**

- If a user puts the call on hold from the phone, then the call appearance state is indicated as *Held*.
- IP phones should indicate that the line is held visually. If applicable, the LED of the call appearance on the endpoint that actively held the call appearance should be blinking green slowly, indicating that the call has been held locally. The LED of the call appearance on all other endpoints should be blinking red slowly, indicating that the call has been held on another endpoint in the shared line group.
- If the user on the endpoint that actively held the call retrieves the held call, then the call appearance should return to the *Active* state, as it does with a private line.
- If a user at one of the other endpoints (that is, endpoints that are not currently active in the call) presses the corresponding line key, then it SHOULD either be ignored or treated as if the user wants to retrieve the held call from the remote endpoint.

**Held-private:**

- If a user puts the call on private-hold from the phone, then the call appearance state is indicated as *Held-private*. A private hold is desirable in cases in which the user is in the process of transferring a call or starting a conference. The device indicates a private-hold by including a *Call-Info* header in the re-INVITE request that triggers the hold. The *Call-Info* header contains the appearance-state parameter set to “held-private”.
- IP phones should indicate that the line is held visually. If applicable, the LED of the call appearance on the endpoint that actively held the call appearance should be blinking green slowly, indicating that the call has been held locally. The LED of the call appearance on all other endpoints should remain solid red, as if the call appearance was still active.
- If the user on the endpoint that actively held the call retrieves the held call, then the call appearance should return to the *Active* state, as it does with a private line.
- If a user at one of the other endpoints (that is, endpoints that are not currently active in the call) presses the corresponding line key, then it SHOULD be ignored.

**Bridge-active:**

- If a user barges-in to an active call appearance, then the call appearance state is indicated as *Bridge-active*.

- IP phones should indicate that the line is active visually. If applicable, the LED of the call appearance on the endpoint that has explicitly answered or barged-in the call should be solid green, indicating that the call has been answered and is active locally. The LED of the call appearance on all other endpoints should be solid red, indicating that the call is active on a remote endpoint.
- If the user puts the phone back on hook, then the endpoint releases its call leg for the call appearance. The call appearance stays in the *Bridge-active* state if there is more than one endpoint that remains bridged to the call appearance. If only one endpoint remains, then the call appearance state is indicated as *Active*.
- If the user on the endpoint that has answered the call presses the line key again, it SHOULD be treated as either:
  - A request to put the call leg on hold if the endpoint has a separate "release" key.

  -or-

  - A request to release the call leg if the endpoint does not have a separate "release" key.
- If a user at one of the other endpoints (that is, endpoints that are not currently active in the call) presses the corresponding line key, then it SHOULD either be ignored or treated as if the user wants to barge in on the call.

**Bridge-held:**

- A call appearance that is in the *Bridge-active* state may be held from a call client. In this case, the call appearance state is indicated as *Bridge-held*. This appearance state value is purely informative and should be handled similar to the "bridge-active" value.
- IP phones should indicate that the line is active visually. If applicable, the LED of the call appearance on the endpoint that has explicitly answered or barged-in the call should be solid green, indicating that the call has been answered and is active locally. The LED of the call appearance on all other endpoints should be solid red, indicating that the call is active on a remote endpoint. The IP phone may implement an additional visual indicator to show that the call is Bridge-held. When a call appearance is Bridge-held, the active endpoints can communicate with one another; however, the media stream to the remote party is inactive.
- If the user puts the phone back on hook, then the endpoint releases its call leg for the call appearance. The call appearance stays in the *Bridge-held* state if there is more than one endpoint that remains bridged to the call appearance. If only one endpoint remains, then the call appearance state is indicated as *Held*.
- If the user on the endpoint that has answered the call presses the line key again, it SHOULD be treated as either:
  - A request to put the call leg on hold if the endpoint has a separate "release" key.

  -or-

  - A request to release the call leg if the endpoint does not have a separate "release" key.
- If a user at one of the other endpoints (that is, endpoints that are not currently active in the call) presses the corresponding line key, then it SHOULD either be ignored or treated as if the user wants to barge in on the call.

- Some devices can only visually represent one call appearance per line. If this is the case, then the presented call appearance should always be the first non-idle call appearance on the line.

## 6.4 Functional Overview

This section presents a functional overview of how the building blocks previously specified work to deliver Shared Call Appearance Functionality in a hosted-communications environment. The intent is to show an application-neutral set of call flows that cover the most common interactions that can be expected between the SIP IP phones and the Application Server. For simplicity, the Shared Call Appearance is limited to two endpoints; however, the basic principles can be directly extrapolated to any number of endpoints.

*Figure 10 Functional Overview* shows the network configuration for the examples that follow. IP phone A-1 (a1.foo.com) and IP phone A-2 (a2.foo.com) are monitoring the same shared line (sip: shared-a@as.foo.com) hosted on the Application Server (as.foo.com).

Phone B is behind a SIP network gateway and represents the off-network party in the call flows. Note that each phone also has a respective private line: private-sip:a1@a1.foo.com and sip:private-a2@a2.foo.com.

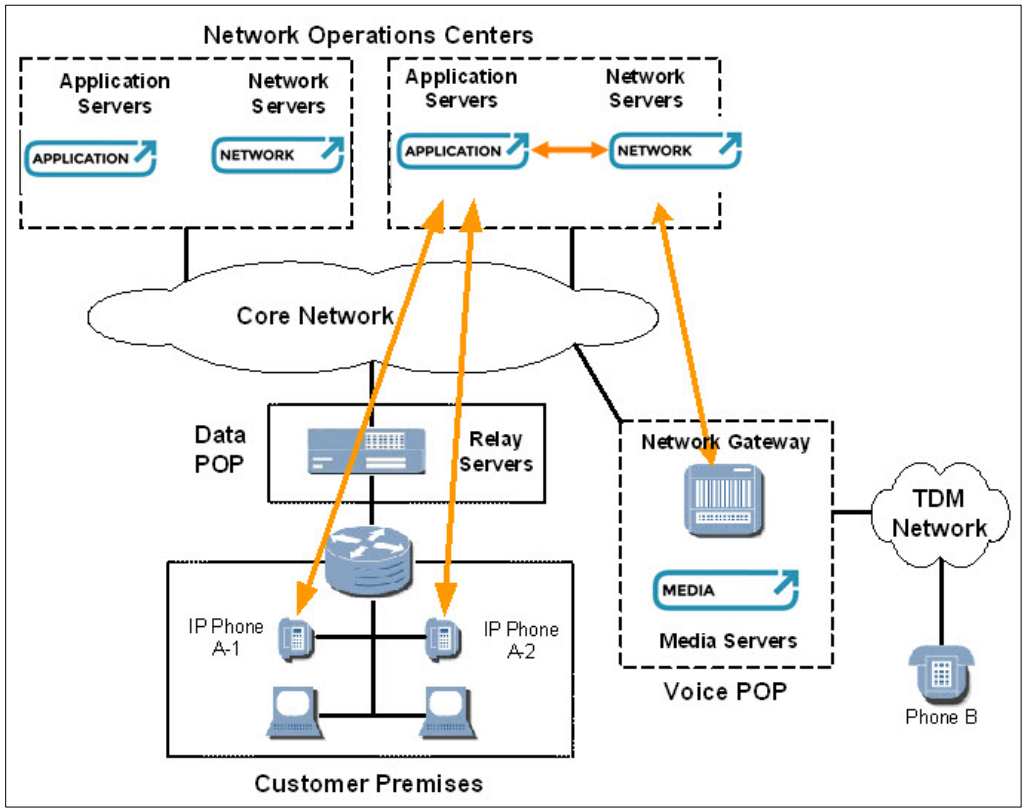


Figure 10 Functional Overview

### 6.4.1 IP Phone Power Up

*Figure 11 Phone A-1 Powers Up* shows the message flow as a result of powering up IP phone A-1.

#### 6.4.1.1 Step 1.1: A-1 Phone Configuration

The phones must first retrieve network topology and device configuration information. This is done through a "Configuration Request" in [F1]. The transport for the configuration request can be any number of options: File Transfer Protocol (FTP), Trivial File Transfer Protocol (TFTP), Hypertext Transfer Protocol (HTTP), or Hypertext Transfer Protocol Secure Sockets (HTTPS). HTTPS is preferred as it offers confidential transport of potentially sensitive configuration information. The flow shows the Application Server (AS) as the target of the configuration request. In practice this can be any centralized server on the network, and does not have to be the Application Server. The "Configuration Response" in [F2] provides any number of device-specific configuration parameters, as well as the following critical information:

- Which line keys are *shared* call appearances
- Which line keys are *local* call appearances
- The display name for each line
- The address of record for each line
- The credentials for each line
- The registrar for each line
- The proxy for each line

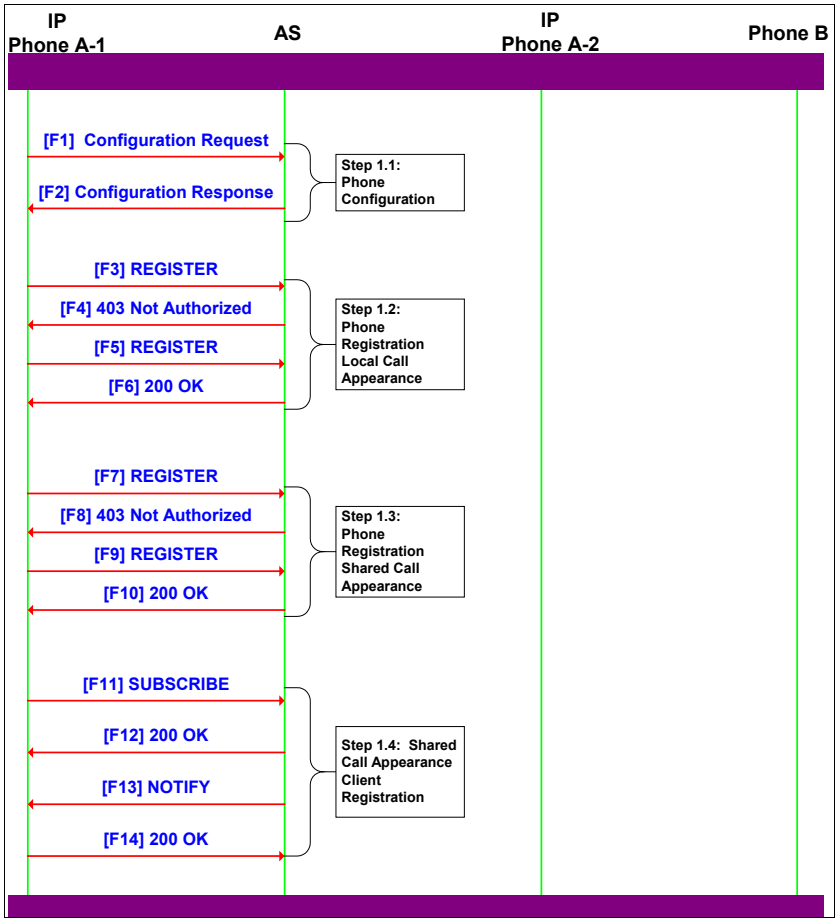


Figure 11 Phone A-1 Powers Up

### 6.4.1.2 Step 1.2: A-1 Phone Registration for Local Private Lines

[F3] to [F6] is performed for each unique private line. This is the standard registration process as described in *RFC 3261* [1]. SIP MD5 digest authentication is used to authenticate the endpoint.

```
[F3] REGISTER A-1 -> Application Server
REGISTER sip:as.foo.com SIP/2.0
Via: SIP/2.0/UDP a1.foo.com:5060;branch=z9hG4bKnashds7
Max-Forwards: 70
From: "Private-A1" <sip:private-a1@as.foo.com>;tag=a73kszlfl
To: "Private-A1" <sip:private-a1@as.foo.com>
Call-ID: 1j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 1 REGISTER
Contact: <sip:private-a1@a1.foo.com>
Content-Length: 0

[F4] 401 Unauthorized Application Server → A-1
SIP/2.0 401 Unauthorized
Via: SIP/2.0/UDP a1.foo.com:5060;branch=z9hG4bKnashds7
From: "Private-A1" <sip:private-a1@as.foo.com>;tag=a73kszlfl
To: "Private-A1" <sip:private-a1@as.foo.com>;tag=1410948204
Call-ID: 1j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 1 REGISTER
```

```
WWW-Authenticate: Digest realm="foo.com", qop="auth",
nonce="ea9c8e88df84f1cec4341ae6cbe5a359", opaque="", stale=FALSE,
algorithm=MD5
Content-Length: 0

[F5] REGISTER A-1 -> Application Server
REGISTER sip:as.foo.com SIP/2.0
Via: SIP/2.0/UDP a1.foo.com:5060;branch=asdf32nashds7
Max-Forwards: 70
From: "Private-A1" <sip:private-a1@as.foo.com>;tag=asa3dszlfl
To: "Private-A1" <sip:private-a1@as.foo.com>
Call-ID: 1j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 2 REGISTER
Contact: <sip:private-a1@a1.foo.com>
Authorization: Digest username="private-a1", realm="foo.com"
nonce="ea9c8e88df84f1cec4341ae6cbe5a359", opaque="",
uri="sip:as.foo.com", response="dfe56131d1958046689d83306477ecc"
Content-Length: 0

[F6] 200 OK Application Server → A-1
SIP/2.0 200 OK
Via: SIP/2.0/UDP a1.foo.com:5060;branch=asdf32nashds7
From: "Private-A1" <sip:private-a1@as.foo.com>;tag= asa3dszlfl
To: "Private-A1" <sip:private-a1@as.foo.com>;tag=323423233
Call-ID: 1j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 2 REGISTER
Contact: <sip:private-a1@a1.foo.com>;expires=3600
Content-Length: 0
```

#### 6.4.1.3 Step 1.3: A-1 Phone Registration for Shared Call Appearances

[F7] to [F10] is performed for each unique shared call appearance. Again, this is the standard registration process as described in *RFC 3261* [1]. SIP MD5 digest authentication is used to authenticate the endpoint. There is no fundamental difference between this registration and the registration described in sections *6.4.1.1 Step 1.1: A-1 Phone Configuration* and *6.4.1.2 Step 1.2: A-1 Phone Registration for Local Private Lines*.

```
[F7] REGISTER A-1 -> Application Server
REGISTER sip:as.foo.com SIP/2.0
Via: SIP/2.0/UDP a1.foo.com:5060;branch=x9hG4bKnashds7
Max-Forwards: 70
From: "Shared-A" <sip:shared-a1@as.foo.com>;tag=b73kszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>
Call-ID: 1j9FpLxk3uxtm8to@a1.foo.com
CSeq: 3 REGISTER
Contact: <sip:shared-a1@a1.foo.com>
Content-Length: 0

[F8] 401 Unauthorized Application Server → A-1
SIP/2.0 401 Unauthorized
Via: SIP/2.0/UDP a1.foo.com:5060;branch=x9hG4bKnashds7
From: "Shared-A" <sip:shared-a1@as.foo.com>;tag=b73kszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=2410948204
Call-ID: 1j9FpLxk3uxtm8to@a1.foo.com
CSeq: 3 REGISTER
WWW-Authenticate: Digest realm="foo.com", qop="auth",
nonce="fa9c8e88df84f1cec4341ae6cbe5a359", opaque="", stale=FALSE,
algorithm=MD5
Content-Length: 0
```

```
[F9] REGISTER A-1 -> Application Server
REGISTER sip:as.foo.com SIP/2.0
Via: SIP/2.0/UDP a1.foo.com:5060;branch=bsdf32nashds7
Max-Forwards: 70
From: "Shared-A" <sip:shared-a1@as.foo.com>;tag=bsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>
Call-ID: 1j9FpLxk3uxtm8to@a1.foo.com
CSeq: 4 REGISTER
Contact: <sip:shared-a1@a1.foo.com>
Authorization: Digest username="shared-a1", realm="foo.com"
nonce="fa9c8e88df84f1cec4341ae6cbe5a359", opaque="",
uri="sip:as.foo.com", response="efe56131d1958046689d83306477ecc"
Content-Length: 0

[F10] 200 OK Application Server → A-1
SIP/2.0 401 Unauthorized
Via: SIP/2.0/UDP a1.foo.com:5060;branch=bsdf32nashds7
From: "Shared-A" <sip:shared-a1@as.foo.com>;tag= bsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=323423233
Call-ID: 1j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 4 REGISTER
Contact: <sip:shared-a1@a1.foo.com>;expires=3600
Content-Length: 0
```

#### 6.4.1.4 Step 1.4: A-1 Shared Call Appearance Client Subscription

[F11] to [F14] is performed for each unique shared call appearance. 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 in steps 1.3 and 2.3 in sections *6.4.1.3 Step 1.3: A-1 Phone Registration for Shared Call Appearances* and *6.4.1.7 Step 2.3: A-2 Phone Registration for Shared Call Appearances*.

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.

```
[F11] SUBSCRIBE A-1 -> Application Server
SUBSCRIBE sip:shared-a1@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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>
Call-ID: 5j9FpLxk3uxtm8to@a1.foo.com
CSeq: 6 SUBSCRIBE
Event: call-info
Expires: 3600
Contact: <sip:shared-a1@a1.foo.com>
Content-Length: 0

[F12] 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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 6 SUBSCRIBE
Event:  call-info
Contact: sip:shared-a1@as.foo.com
Expires: 3600
Content-Length: 0

[F13] NOTIFY Application Server -> A-1
NOTIFY sip:shared-a1@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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@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-a1@as.foo.com
Call-Info:  <sip:as.foo.com>;appearance-index=*;appearance-state=idle
Content-Length: 0

[F14] 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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 1344 NOTIFY
Content-Length: 0
```

#### 6.4.1.5 Step 2.1: A-2 Phone Configuration

The same steps described in sections *6.4.1.1 Step 1.1: A-1 Phone Configuration, 6.4.1.2 Step 1.2: A-1 Phone Registration for Local Private Lines, 6.4.1.3 Step 1.3: A-1 Phone Registration for Shared Call Appearances,* and *6.4.1.4 Step 1.4: A-1 Shared Call Appearance Client Subscription* are repeated for the phone A-2.

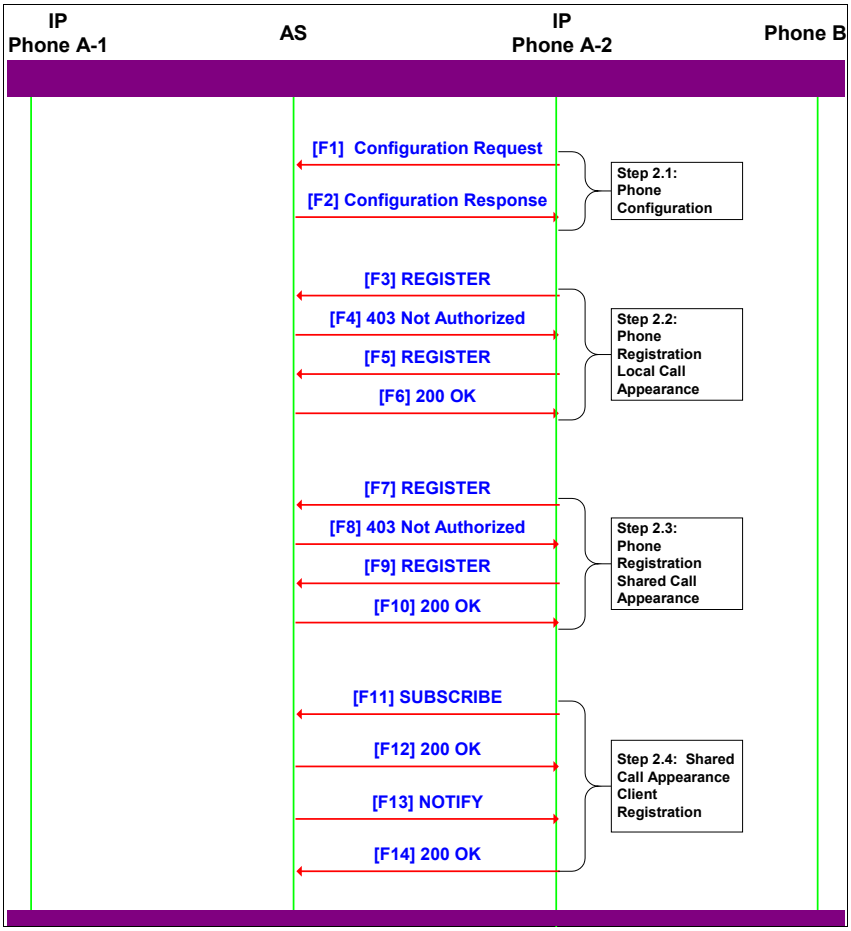


Figure 12 Phone A-2 Powers Up

### 6.4.1.6 Step 2.2: A-2 Phone Registration for Local Private Lines

```
[F3] REGISTER A-2 -> Application Server
REGISTER sip:as.foo.com SIP/2.0
Via: SIP/2.0/UDP a2.foo.com:5060;branch=z9hG4bKnashds7
Max-Forwards: 70
From: "Private-A2" <sip:private-a2@as.foo.com>;tag=a73kszlfl
To: "Private-A2" <sip:private-a2@as.foo.com>
Call-ID: 1j9FpLxk3uxtm8tn@a2.foo.com
CSeq: 1 REGISTER
Contact: <sip:private-a2@a2.foo.com>
Content-Length: 0

[F4] 401 Unauthorized Application Server → A-2
SIP/2.0 401 Unauthorized
Via: SIP/2.0/UDP a2.foo.com:5060;branch=z9hG4bKnashds7
From: "Private-A2" <sip:private-a2@as.foo.com>;tag=a73kszlfl
To: "Private-A2" <sip:private-a2@as.foo.com>;tag=1410948204
Call-ID: 1j9FpLxk3uxtm8tn@a2.foo.com
CSeq: 1 REGISTER
WWW-Authenticate: Digest realm="foo.com", qop="auth",
nonce="ea9c8e88df84f1cec4341ae6cbe5a359", opaque="", stale=FALSE,
algorithm=MD5
Content-Length: 0

[F5] REGISTER A-2 -> Application Server
```

```
REGISTER sip:as.foo.com SIP/2.0
Via: SIP/2.0/UDP a2.foo.com:5060;branch=asdf32nashds7
Max-Forwards: 70
From: "Private-A2" <sip:private-a2@as.foo.com>;tag=asa3dszlfl
To: "Private-A2" <sip:private-a2@as.foo.com>
Call-ID: 2j9FpLxk3uxtm8tn@a2.foo.com
CSeq: 2 REGISTER
Contact: <sip:private-a2@a2.foo.com>
Authorization: Digest username="private-a2", realm="foo.com"
nonce="ea9c8e88df84f1cec4341ae6cbe5a359", opaque="",
uri="sip:as.foo.com", response="dfe56131d1958046689d83306477ecc"
Content-Length: 0

[F6] 200 OK Application Server → A-1
SIP/2.0 200 OK
Via: SIP/2.0/UDP a2.foo.com:5060;branch=asdf32nashds7
From: "Private-A2" <sip:private-a2@as.foo.com>;tag= asa3dszlfl
To: "Private-A2" <sip:private-a2@as.foo.com>;tag=323423233
Call-ID: 2j9FpLxk3uxtm8tn@a2.foo.com
CSeq: 2 REGISTER
Contact: <sip:private-a2@a1.foo.com>;expires=3600
Content-Length: 0
```

#### 6.4.1.7 Step 2.3: A-2 Phone Registration for Shared Call Appearances

```
[F7] REGISTER A-2 -> Application Server
REGISTER sip:as.foo.com SIP/2.0
Via: SIP/2.0/UDP a2.foo.com:5060;branch=x9hG4bKnashds7
Max-Forwards: 70
From: "Shared-A" <sip:shared-a2@as.foo.com>;tag=b73kszlfl
To: "Shared-A" <sip:shared-a2@as.foo.com>
Call-ID: 3j9FpLxk3uxtm8to@a2.foo.com
CSeq: 3 REGISTER
Contact: <sip:shared-a2@a1.foo.com>
Content-Length: 0

[F8] 401 Unauthorized Application Server → A-2
SIP/2.0 401 Unauthorized
Via: SIP/2.0/UDP a2.foo.com:5060;branch=x9hG4bKnashds7
From: "Shared-A" <sip:shared-a2@as.foo.com>;tag=b73kszlfl
To: "Shared-A" <sip:shared-a2@as.foo.com>;tag=2410948204
Call-ID: 3j9FpLxk3uxtm8to@a2.foo.com
CSeq: 3 REGISTER
WWW-Authenticate: Digest realm="foo.com", qop="auth",
nonce="fa9c8e88df84f1cec4341ae6cbe5a359", opaque="", stale=FALSE,
algorithm=MD5
Content-Length: 0

[F9] REGISTER A-2 -> Application Server
REGISTER sip:as.foo.com SIP/2.0
Via: SIP/2.0/UDP a2.foo.com:5060;branch=bsdf32nashds7
Max-Forwards: 70
From: "Shared-A" <sip:shared-a2@as.foo.com>;tag=bsa3dszlfl
To: "Shared-A" <sip:shared-a2@as.foo.com>
Call-ID: 4j9FpLxk3uxtm8to@a2.foo.com
CSeq: 4 REGISTER
Contact: <sip:shared-a2@a2.foo.com>
Authorization: Digest username="shared-a2", realm="foo.com"
nonce="fa9c8e88df84f1cec4341ae6cbe5a359", opaque="",
uri="sip:as.foo.com", response="efe56131d1958046689d83306477ecc"
```

```
Content-Length: 0

[F10] 200 OK Application Server → A-2
SIP/2.0 401 Unauthorized
Via: SIP/2.0/UDP a1.foo.com:5060;branch=asdf32nashds7
From: "Shared-A" <sip:shared-a2@as.foo.com>;tag=bsa3dszlfl
To: "Shared-A" <sip:shared-a2@as.foo.com>;tag=323423233
Call-ID: 4j9FpLxk3uxtm8tn@foo.com
CSeq: 4 REGISTER
Contact: <sip:shared-a2@a2.foo.com>;expires=3600
Content-Length: 0
```

#### 6.4.1.8 Step 2.4: A-2 Shared Call Appearance Client Subscription

```
[F11] SUBSCRIBE A-2 -> Application Server
SUBSCRIBE sip:shared-a2@as.foo.com SIP/2.0
Via: SIP/2.0/UDP a2.foo.com:5060;branch=ggsdf32nashds7
Max-Forwards: 70
From: "Shared-A" <sip:shared-a2@as.foo.com>;tag=ggsa3dszlfl
To: "Shared-A" <sip:shared-a2@as.foo.com>
Call-ID: 9k9FpLxk3uxtm8to@a2.foo.com
CSeq: 6 SUBSCRIBE
Event: call-info
Expires: 3600
Contact: <sip:shared-a2@a2.foo.com>
Content-Length: 0

[F12] 200 OK Application Server → A-2
SIP/2.0 200 OK
Via: SIP/2.0/UDP a2.foo.com:5060;branch=ggsdf32nashds7
From: "Shared-A" <sip:shared-a2@as.foo.com>;tag=ggsa3dszlfl
To: "Shared-A" <sip:shared-a2@as.foo.com>;tag=123456
Call-ID: 9k9FpLxk3uxtm8tn@a2.foo.com
CSeq: 6 SUBSCRIBE
Event: call-info
Contact:  sip:shared-a2@as.foo.com
Expires: 3600
Content-Length: 0

[F13] NOTIFY Application Server -> A-2
NOTIFY sip:shared-a2@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-a2@as.foo.com>;tag=ggsa3dszlfl
To: "Shared-A" <sip:shared-a2@as.foo.com>;tag=123456
Call-ID: 9k9FpLxk3uxtm8tn@a2.foo.com
CSeq: 9 NOTIFY
Event: call-info
Contact:  sip:shared-a2@as.foo.com
Subscription-State: active; expires=3599
Call-Info:  <sip:as.foo.com>;appearance-index=*;appearance-state=idle
Content-Length: 0

[F14] 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-a2@as.foo.com>;tag=ggsa3dszlfl
To: "Shared-A" <sip:shared-a2@as.foo.com>;tag=123456
Call-ID: 9k9FpLxk3uxtm8tn@a2.foo.com
CSeq: 9 NOTIFY
Content-Length: 0
```

### 6.4.2 A-1 Takes Shared Line Off Hook

*Figure 13 A-1 Goes Off Hook* shows the call flow as a result of a user at A-1 taking the shared line off hook.

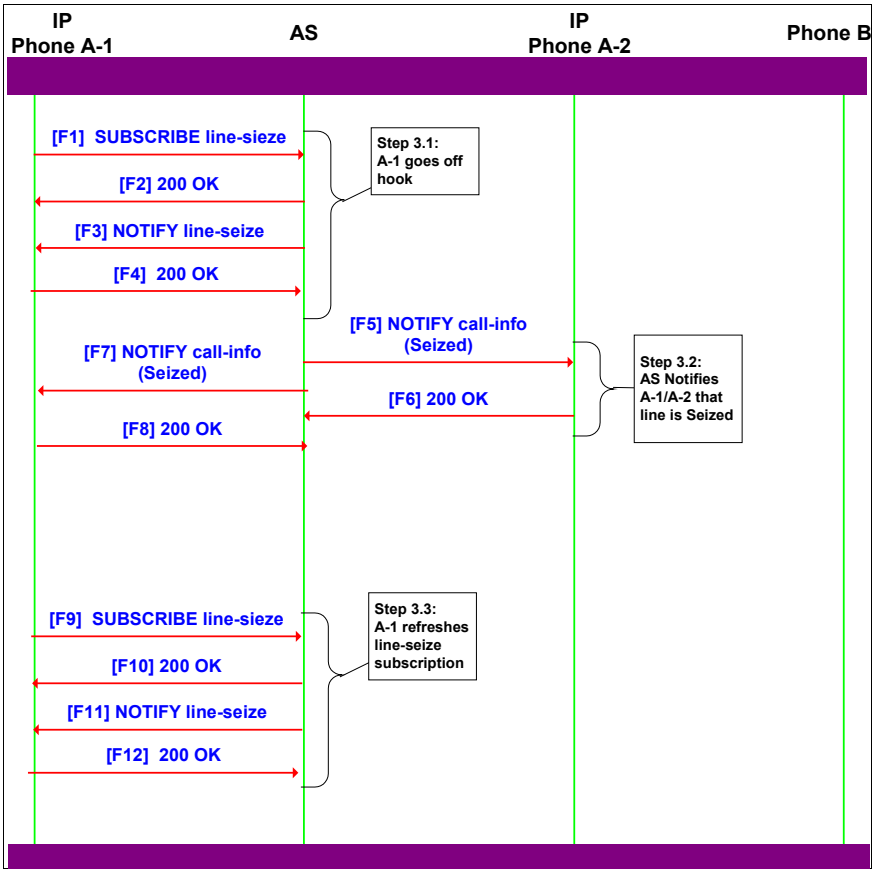


Figure 13 A-1 Goes Off Hook

#### 6.4.2.1 Step 3.1: A-1 Requests Line-Seize Subscription

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 a dial tone to the user.

```
[F1] SUBSCRIBE A-1 -> Application Server
SUBSCRIBE sip:shared-a1@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-a1@as.foo.com>;tag=gsa3dszlfl
To: "Shared-A" <sip:shared-a1@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-a1@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-a1@as.foo.com>;tag=gsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=423423233
Call-ID: 6j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 7 SUBSCRIBE
Contact: sip:shared-a1@as.foo.com
Event: line-seize
Expires=15
Content-Length: 0

[F3] NOTIFY Application Server -> A-1
NOTIFY sip:shared-a1@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-a1@as.foo.com>;tag=gsa3dszlfl
To: "Shared-A" <sip:shared-a1@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-a1@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-a1@as.foo.com>;tag=gsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=423423233
Call-ID: 6j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 9 NOTIFY
Content-Length: 0
```

#### 6.4.2.2 Step 3.2: Application Server Sends Call-Info NOTIFY-Requests

In [F5 to F8], the Application Server sends a corresponding Call-Info NOTIFY-request to A-1 and A-2 because the line seizure has caused a state change on the shared line. The *Call-Info* header in the NOTIFY-request indicates the current state of the shared line, which includes the seized state on the first call appearance.

```
[F5] NOTIFY Application Server -> A-1
NOTIFY sip:shared-a1@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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 9 NOTIFY
Event: call-info
Subscription-State: active; expires=3400
Call-Info:  <sip:as.foo.com>;appearance-index=1;appearance-state=seized,
            <sip:as.foo.com>;appearance-index=*;appearance-state=idle
Content-Length: 0

[F6] 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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
```

```
CSeq: 9 NOTIFY
Content-Length: 0

[F7] NOTIFY Application Server -> A-2
NOTIFY sip:shared-a1@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-a2@as.foo.com>;tag=ggsa3dszlfl
To: "Shared-A" <sip:shared-a2@as.foo.com>;tag=123456
Call-ID: 9k9FpLxk3uxtm8tn@a2.foo.com
CSeq: 9 NOTIFY
Event: call-info
Subscription-State: active; expires=3230
Call-Info:  <sip:as.foo.com>;appearance-index=1;appearance-state=seized,
            <sip:as.foo.com>;appearance-index=*;appearance-state=idle
Content-Length: 0

[F8] 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-a2@as.foo.com>;tag=ggsa3dszlfl
To: "Shared-A" <sip:shared-a2@as.foo.com>;tag=123456
Call-ID: 9k9FpLxk3uxtm8tn@a2.foo.com
CSeq: 9 NOTIFY
Content-Length: 0
```

#### 6.4.2.3 Step 3.3: A-1 Refreshes Line-Seize Subscription

In [F9 to F12], the user at A-1 has taken longer than 15 seconds to dial the number, but the inter-digit time-out has not expired, so the user is still entering digits. To continue collecting digits, the phone must refresh the *line-seize* subscription.

```
[F9] SUBSCRIBE A-1 -> Application Server
SUBSCRIBE sip:shared-a1@as.foo.com SIP/2.0
Via: SIP/2.0/UDP a1.foo.com:5060;branch=gsdf32nashds8
Max-Forwards: 70
From: "Shared-A" <sip:shared-a1@as.foo.com>;tag=gsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=423423233
Call-ID: 6j9FpLxk3uxtm8to@a1.foo.com
Call-Info:  <sip:as.foo.com>; appearance-index=1
CSeq: 12 SUBSCRIBE
Event: line-seize
Expires: 15
Contact: <sip:shared-a1@a1.foo.com>
Content-Length: 0

[F10] 200 OK Application Server → A-1
SIP/2.0 200 OK
Via: SIP/2.0/UDP a1.foo.com:5060;branch=gsdf32nashds8
From: "Shared-A" <sip:shared-a1@as.foo.com>;tag=gsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=423423233
Call-ID: 6j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 12 SUBSCRIBE
Event: line-seize
Contact: sip:shared-a1@as.foo.com
Expires=15
Content-Length: 0

[F11] NOTIFY Application Server -> A-1
NOTIFY sip:shared-a1@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-a1@as.foo.com>;tag=gsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=423423233
Call-ID: 6j9FpLxk3uxtm8to@a1.foo.com
Call-Info:  <sip:as.foo.com>; appearance-index=1
CSeq: 11 NOTIFY
Event: line-seize
Subscription-State: active; expires=14
Contact: <sip:shared-a1@as.foo.com>
Content-Length: 0
```

***[F12] 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-a1@as.foo.com>;tag=gsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=423423233
Call-ID: 6j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 11 NOTIFY
Content-Length: 0
```

### 6.4.3 A-1 Calls B

*Figure 14* shows the call flow as a result of A-1 originating a call to B using the shared line. It is assumed that A-1 has successfully seized the line as shown in the previous section.

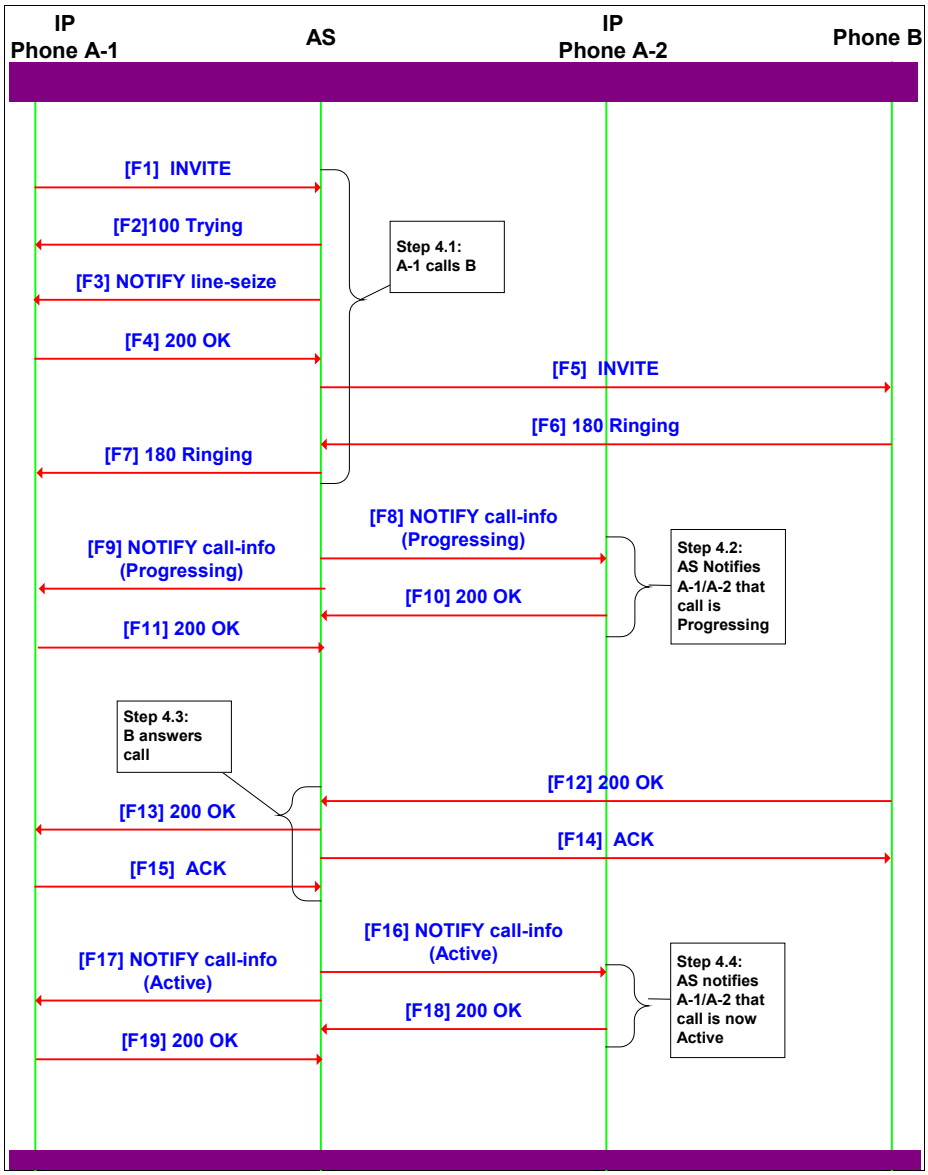


Figure 14 A-1 Calls B

#### 6.4.3.1 Step 4.1: 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 usual 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-a1@as.foo.com>;tag=5fxced76sl
To: "B" <sip:B@as.foo.com>
Call-ID: 98482762982201885l1@a1.foo.com
Call-Info:  <sip:as.foo.com>;appearance-index=1
CSeq: 43234 INVITE
Contact: <sip: shared-a1@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-a1@as.foo.com>;tag=5fxced76sl
To: "B" <sip:B@foo.com>; tag=4323z39
Call-ID: 98482762982201885l1@a1.foo.com
Call-Info:  <sip:as.foo.com>;appearance-index=1
CSeq: 43234 INVITE
Contact: <sip:as.foo.com>
Content-Length: 0
```

***[F3] NOTIFY Application Server -> A-1***

```
NOTIFY sip:shared-a1@a1.foo.com SIP/2.0
Via: SIP/2.0/UDP as.foo.com:5060; branch=gsdf32nashds7
Max-Forwards: 70
From: "Shared-A" <sip:@as.foo.com>;tag=gsa3dszlfl
To: "Shared-A" <sip:@as.foo.com>;tag=423423233
Call-ID: 6j9FpLxk3uxtm8to@a1.foo.com
CSeq: 24 NOTIFY
Event: line-seize
Subscription-State: terminated; expires=0
Contact: sip: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-a1@as.foo.com>;tag=gsa3dszlfl
To: "Shared-A" <sip:shared-a1@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-a1@as.foo.com>;tag=5fxced76sl
To: "B" <sip:B@as.foo.com>;tag=4323z39
Call-ID: 9848276298220188511@a1.foo.com
Call-Info:  <sip:as.foo.com>;appearance-index=1
CSeq: 43234 INVITE
Contact: <sip:B@as.foo.com>
Content-Length: 0
```

### 6.4.3.2 Step 4.2: 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-a1@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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@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-uri="\"B. Foo\"<sip:
B@as.foo.com;user=phone>";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-a2@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-a2@as.foo.com>;tag=ggsa3dszlfl
To: "Shared-A" <sip:shared-a2@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-uri="\"B. Foo\"<sip:
B@as.foo.com;user=phone>";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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@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-a2@as.foo.com>;tag=ggsa3dszlfl
To: "Shared-A" <sip:shared-a2@as.foo.com>;tag=123456
Call-ID: 9k9FpLxk3uxtm8tn@a2.foo.com
CSeq: 1345 NOTIFY
Content-Length: 0
```

#### 6.4.3.3 Step 4.3: 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-a1@as.foo.com>;tag=5fxced76sl
To: "B" <sip:B@as.foo.com>; tag=4323z39
Call-ID: 9848276298220188511@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-a1@as.foo.com>;tag=234wdkfj
To: "B" <sip:B@as.foo.com>;tag=1234dsf2
Call-ID: 9848276298220188511@a1.foo.com
```

```
CSeq: 1093 ACK
Contact: <sip:shared-a1 @a2.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@as.foo.com
CSeq: 12345 ACK
Contact: <sip:shared-a1@as.foo.com>
Content-Length: 0
```

#### 6.4.3.4 Step 4.4: 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-a1@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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@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-uri="\"B. Foo\"<sip:
B@as.foo.com;user=phone>";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-a2@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-a2@as.foo.com>;tag=ggsa3dszlfl
To: "Shared-A" <sip:shared-a2@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-uri="\"B. Foo\"<sip:
B@as.foo.com;user=phone>";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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@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-a2@as.foo.com>;tag=ggsa3dszlfl
To: "Shared-A" <sip:shared-a2@as.foo.com>;tag=123456
Call-ID: 9k9FpLxk3uxtm8tn@a2.foo.com
CSeq: 1345 NOTIFY
Content-Length: 0
```

### 6.4.4 Phone A-2 Error Scenario

*Figure 15* shows an error case that could result if IP phone A-2 becomes out of synchronization with the Application Server and attempts to originate a call on the shared call appearance while a call is currently active.

Note that this scenario can also occur if A-2 attempts to barge into a held call. In addition, if A-2 attempts to retrieve a call that is privately held, then the Application Server sends a 403 response and does not send NOTIFY requests to synchronize the IP phone states.

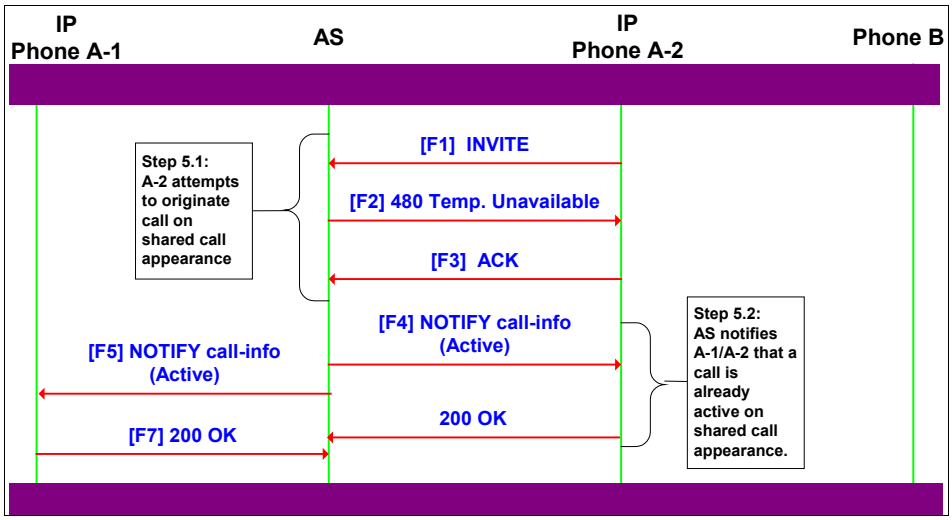


Figure 15 Phone A-2 Error Scenario

#### 6.4.4.1 Step 5.1: A-2 Attempts to Originate Call on Shared Call Appearance

For whatever reason, IP phone A-2 has sent an INVITE to the Application Server on the shared call appearance, even though the call appearance is currently *Active* (with a call between A-1 and B). The Application Server detects this contention and rejects the INVITE in [F1] with a 480 Temp Unavailable in [F2].

```
[F1] INVITE A-2 → Application Server
INVITE sip:X@as.foo.com SIP/2.0
Via: SIP/2.0/UDP a2.foo.com:5060;branch=q9hG4bK74bf9
Max-Forwards: 70
From: "Shared-A" <sip:shared-a2@as.foo.com>;tag=ffxced76sl
To: "X" <sip:X@as.foo.com>
Call-ID: 100827629822018851l@a2.foo.com
Call-Info:  <sip:as.foo.com>;appearance-index=1
CSeq: 43236 INVITE
Contact: <sip:shared-a2@a2.foo.com>
Content-Type: application/sdp
Content-Length: 143

v=0
```

```
o=UserA 2890844526 2890844526 IN IP4 a2.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] 480 Temporarily Unavailable Application Server → A-2
SIP/2.0 480 Temporarily Unavailable
Via: SIP/2.0/UDP a2.foo.com:5060;branch=q9hG4bK74bf9
From: "Shared-A" <sip:shared-a2@as.foo.com>;tag=ffxced76sl
To: "X" <sip:X@foo.com>; tag=4323z44
Call-ID: 10082762982201885l1@a2.foo.com
Call-Info:  <sip:as.foo.com>;appearance-index=1
CSeq: 43236 INVITE
Contact: <sip:X@as.foo.com>
Content-Length: 0

[F3] ACK A-2 → Application Server
ACK sip:X@as.foo.com SIP/2.0
Via: SIP/2.0/UDP a2.foo.com:5060;branch=q9hG4bK74bf9
Max-Forwards: 70
From: "Shared-A" <sip:shared-a2@as.foo.com>;tag=ffxced76sl
To: "X" <sip:X@as.foo.com>; tag=4323z44
Call-ID: 10082762982201885l1@a2.foo.com
CSeq: 43232 ACK
Contact: <sip:shared-a2@a2.foo.com>
Content-Length: 0
```

#### 6.4.4.2 Step 5.2: Application Server Notifies A-1 and A-2 that Call is Already Active on Shared Call Appearance

To make sure A-1 and A-2 are synchronized, the Application Server sends a NOTIFY with the current call state [F4 to F7].

```
[F4] NOTIFY Application Server -> A-2
NOTIFY sip:shared-a2@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-a2@as.foo.com>;tag=ggsa3dszlfl
To: "Shared-A" <sip:shared-a2@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-uri="\"B. Foo\"<sip:
B@as.foo.com;user=phone>";appearance-index=1;appearance-state=active,
            <sip:as.foo.com>;appearance-index=*;appearance-state=idle
Content-Length: 0

[F5] NOTIFY Application Server -> A-1
NOTIFY sip:shared-a1@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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@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-uri="\"B. Foo\"<sip:
B@as.foo.com;user=phone>";appearance-index=1;appearance-state=active,
            <sip:as.foo.com>;appearance-index=*;appearance-state=idle
Content-Length: 0

[F6] 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-a2@as.foo.com>;tag=ggsa3dszlfl
To: "Shared-A" <sip:shared-a2@as.foo.com>;tag=123456
Call-ID: 9k9FpLxk3uxtm8tn@a2.foo.com
CSeq: 1345 NOTIFY
Content-Length: 0

[F7] 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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 1344 NOTIFY
Content-Length: 0
```

### 6.4.5 A-1 Places B on Hold

*Figure 16* shows the call flow as a result of A-1 placing B on hold.

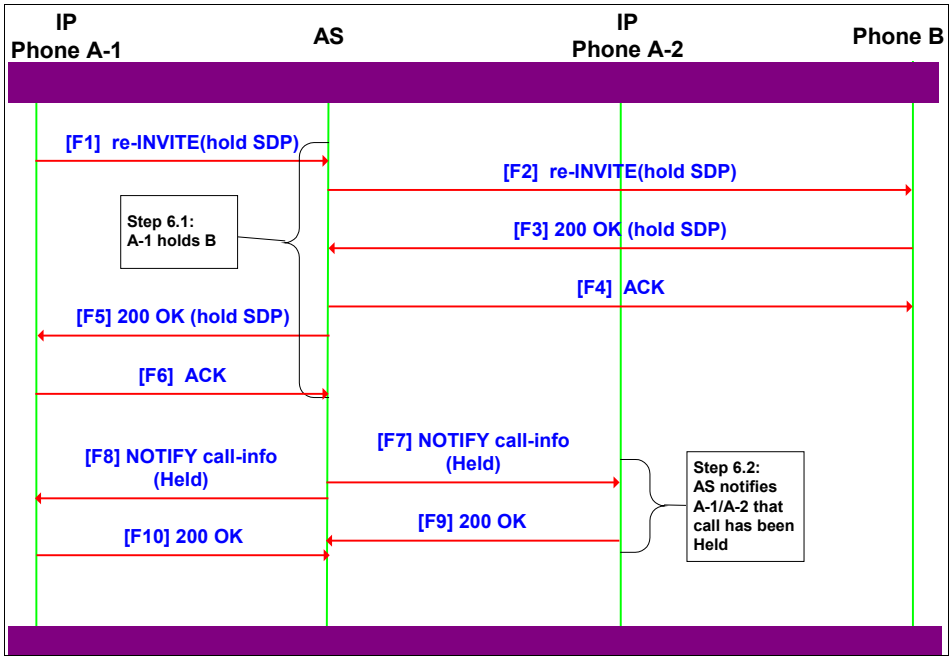


Figure 16 A-1 Places B on Hold

#### 6.4.5.1 Step 6.1: A-1 Holds B

[F1 to F6] shows the normal call flow as a result of A-1 placing B on hold. A-1 changes the call appearance interface to reflect the "held" state.

#### 6.4.5.2 Step 6.2: Application Server Notifies A-1 and A-2 that Call has been Held

The Application Server sends a NOTIFY to A-1 and A-2 [F7 to F10] to indicate that the call has been "Held". A-2's phone changes the call appearance interface to reflect the *Held* state.

### 6.4.6 A-1 Retrieves Call from Hold

*Figure 17* shows the call flow as a result of A-1 retrieving B from hold.

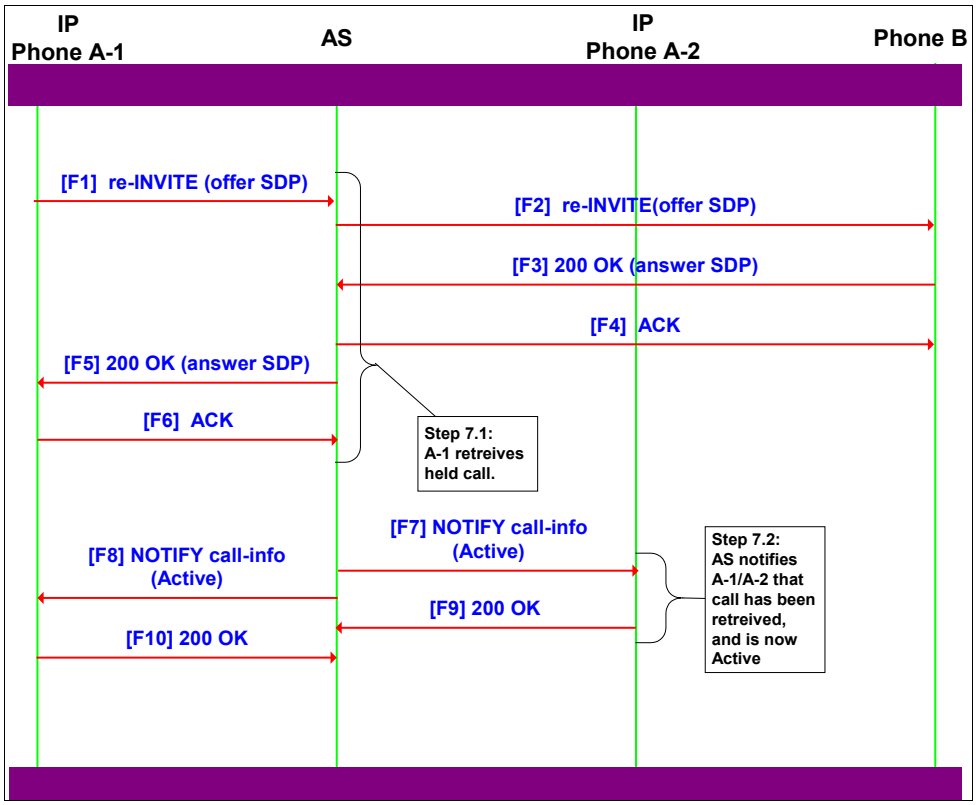


Figure 17 A-1 Retrieves B from Hold

#### 6.4.6.1 Step 7.1: A-1 Retrieves B

[F1 to F6] shows the normal call flow as a result of A-1 retrieving B from hold. A-1 changes the call appearance interface to reflect the *Active* state.

#### 6.4.6.2 Step 7.2: Application Server Notifies A-1 and A-2 that Call is Now Active

The Application Server sends a NOTIFY to A-1 and A-2 [F7-F10] to indicate that the call is now *Active*. A-2's phone changes the call appearance interface to reflect the *Active* state.

### 6.4.7 A-1 Places B on Private-Hold

*Figure 18* shows the call flow as a result of A-1 placing B on private-hold.

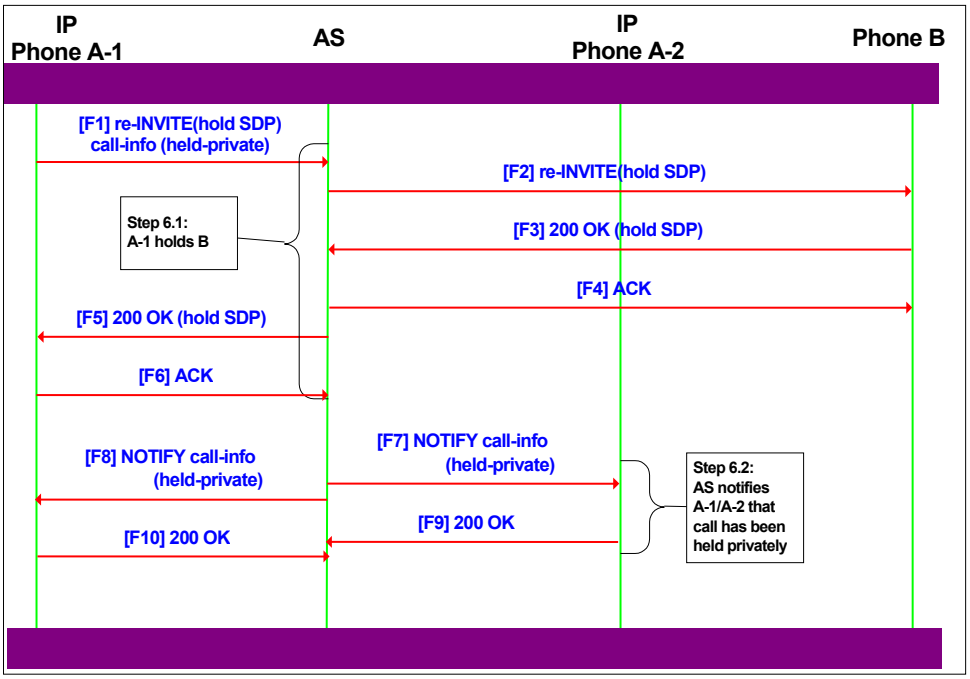


Figure 18 A-1 Places on Hold

#### 6.4.7.1 Step 6.1: A-1 Holds B

[F1 to F6] shows the normal call flow as a result of A-1 placing B on hold. A-1 changes the call appearance interface to reflect the *Held-private* state.

#### 6.4.7.2 Step 6.2: Application Server Notifies A-1 and A-2 that Call has been Held Privately

The Application Server sends a NOTIFY to A-1 and A-2 [F7 to F10] to indicate that the call has been *Held-private*. A-2's phone changes the call appearance interface to reflect the *Held-private* state.

### 6.4.8 A-1 Releases Call

*Figure 19* shows the call flow as a result of A-1 releasing the call with B.

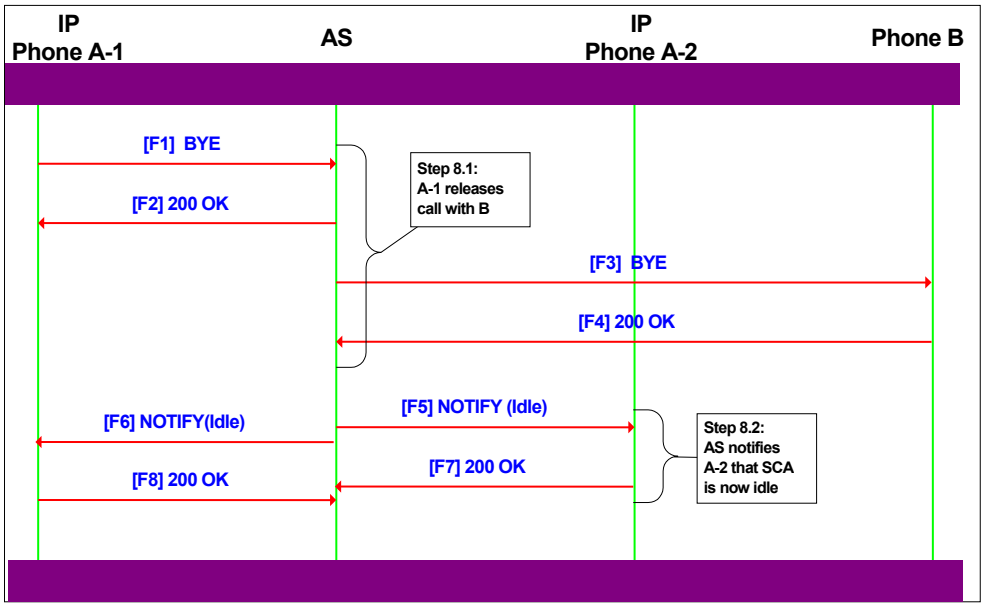


Figure 19 A-1 Releases Call

#### 6.4.8.1 Step 8.1: Application Server Releases Call with B

The user at A-1 hangs up and the phone sends a BYE to the Application Server [F1]. The Application Server sends a corresponding BYE to B [F3]. The call is effectively released.

#### 6.4.8.2 Step 8.2: Application Server Notifies A-1 and A-2 that Shared Call Appearance is Now Idle

The Application Server sends a NOTIFY to A-2 [F5-F8] with an event of "Idle" and an empty call dialog document. IP phone A-2 changes the status of the corresponding line key.

### 6.4.9 B Calls A

*Figure 20* shows the call flow as a result of B calling A. Note that B cannot explicitly call A-1 or A-2. Instead it can only address the number of the shared line for A. The Application Server then deals with delivering the call simultaneously to the endpoints at A-1 and A-2.

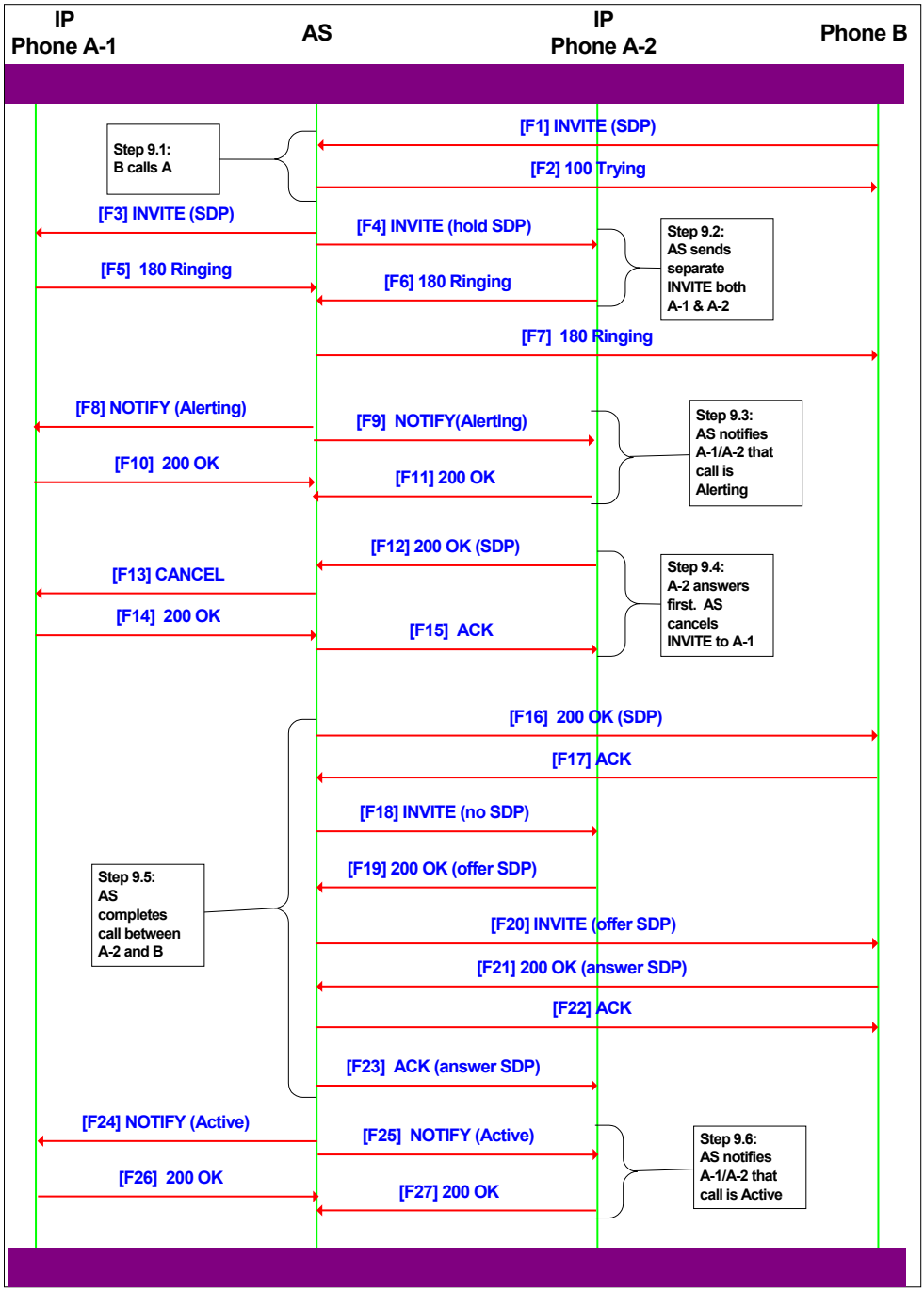


Figure 20 B Calls A

#### 6.4.9.1 Step 9.1: B Calls A

B delivers an INVITE to the Application Server [F1] addressing the logical user A.

#### 6.4.9.2 Step 9.2: Application Server Sends Separate INVITE to both A-1 and A-2

[F3 to F6] because A actually represents a shared call appearance, the Application Server sends a separate INVITE to each registered endpoint. Note that the "primary" endpoint receives the original Session Description Protocol (SDP) offered from B. The other endpoints (called alternate endpoints) receive a hold SDP.

#### 6.4.9.3 Step 9.3: Application Server Notifies A-1 and A-2 that Call is Alerting

[F7 to F11] once the Application Server receives indication that the incoming call is alerting, NOTIFY-requests are sent to A-1 and A-2, indicating that the call is now alerting.

#### 6.4.9.4 Step 9.3: A-2 Answers First, Application Server Cancels INVITE to A-1

A-2 (an alternate endpoint) answers the incoming call first [F12]. The Application Server then sends a CANCEL to A-1, to make it stop alerting [F13 – F14].

#### 6.4.9.5 Step 9.4: Application Server Completes Call between A-2 and B

In [F16 to F23] the Application Server performs some offer/answer processing to make sure the SDP between A-2 and B align properly.

#### 6.4.9.6 Step 9.5: Application Server Notifies A-1 and A-2 that Call is Active

In [F24 – F27], the Application Server sends a NOTIFY to A-1 and A-2, indicating that a call is currently *Active*. The corresponding call dialog details are sent in the body of the NOTIFY in case A-1 must interact with the *Active* shared call appearance.

### 6.4.10 A-2 Retrieves B from Hold

*Figure 21* provides the same call flow as described in section *6.4.5 A-1 Places B on Hold*. At the end of this scenario, A-1 has placed B on hold. For information on this call flow, see section *6.4.5 A-1 Places B on Hold*.

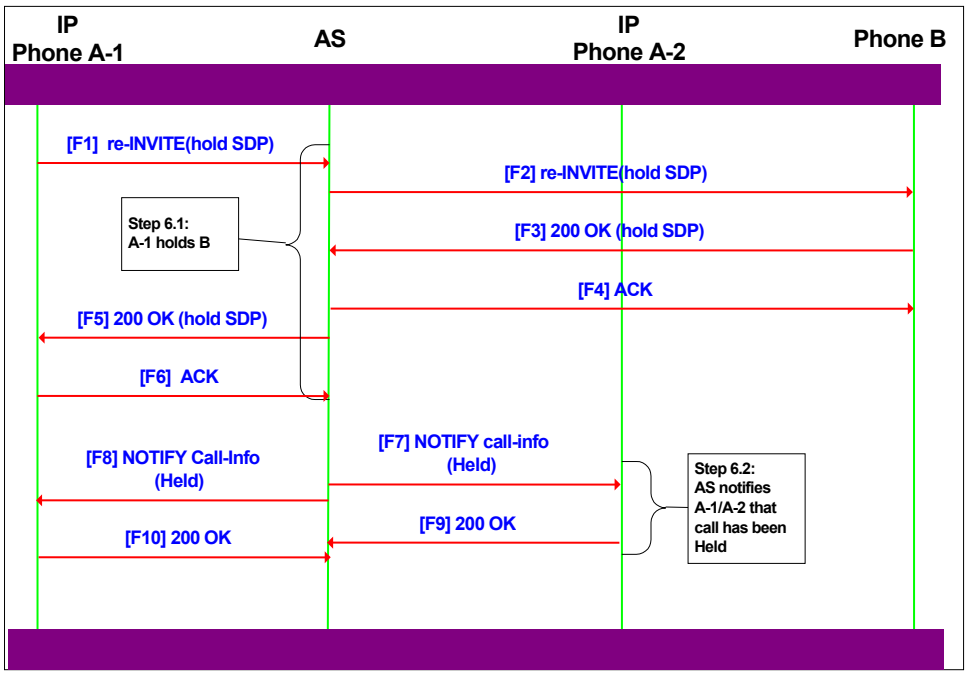


Figure 21 A-2 Retrieves B from Hold

Now the user at IP phone A-2 can see that there is a call held remotely on the shared line. The user at IP phone A-2 wishes to retrieve the held call and pushes the associated line key.

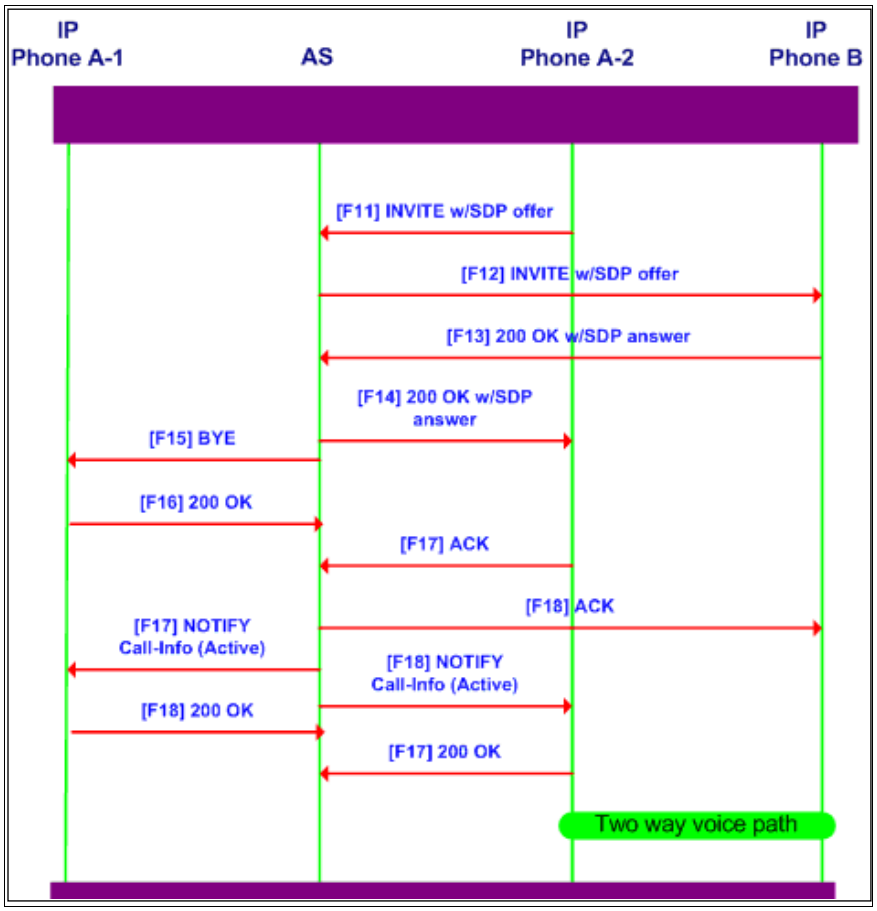


Figure 22 User at IP Phone A-2 Retrieves Held Call

In [F11] the IP phone A-2 initiates an INVITE to retrieve the held call, as follows.

```
[F1] INVITE A-2 → Application Server
INVITE sip:as.foo.com SIP/2.0
Via: SIP/2.0/UDP a2.foo.com:5060;branch=m9hG4bK74bf9
Max-Forwards: 70
From: "Shared-A" <sip:shared-a2@as.foo.com>;tag=5fxced76s3
To: "Shared-A" <sip:shared-a2@as.foo.com>
Call-ID: 98482762982201885l3@a2.foo.com
Call-Info:  <sip:as.foo.com>;appearance-index=1
CSeq: 43294 INVITE
Contact: <sip:shared-a2@a2.foo.com>
Content-Type: application/sdp
Content-Length: 143

v=0
o=UserA 2890844526 2890844526 IN IP4 a2.foo.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

The Application Server looks at the *Request-URI*, the *From* and *To* headers, and the *Call-Info* header to determine that A-2 is attempting to retrieve the held call on A-1. The Application Server applies local policy and determines that A-2 is authorized to retrieve the call. It then proceeds to re-INVITE B to the SDP offered by A-2 [F12-F14]. Once it is clear that the re-INVITE has succeeded, then the original held call with A-2 is released, and a NOTIFY with Call-Info event state is broadcast to all the endpoints in the shared group.

### 6.4.11 A-2 Barges into Call between A-1 and B

*Figure 23* provides the same call flow as described in section *6.4.3 A-1 Calls B*. At the end of this scenario, A-1 has placed B on hold. For information on this call flow, see section *6.4.3 A-1 Calls B*.

*Figure 23* shows the call flow as a result of A-2 barging into an existing call between A-1 and B.

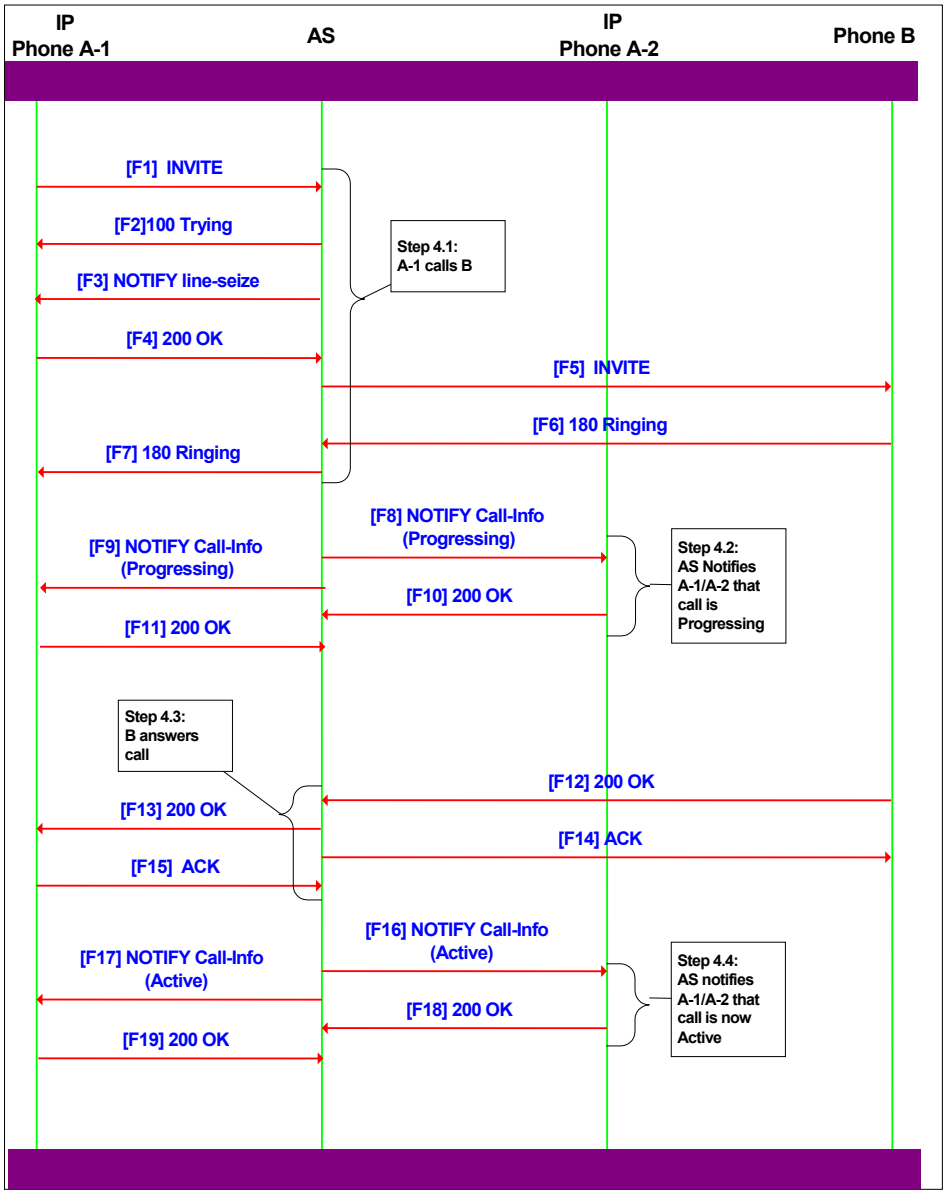


Figure 23 A-1 Calls B

Now the user at IP phone A-2 can see that there is an active call on the shared line. The user at IP phone A-2 wishes to barge in to the active call and pushes the associated line key.

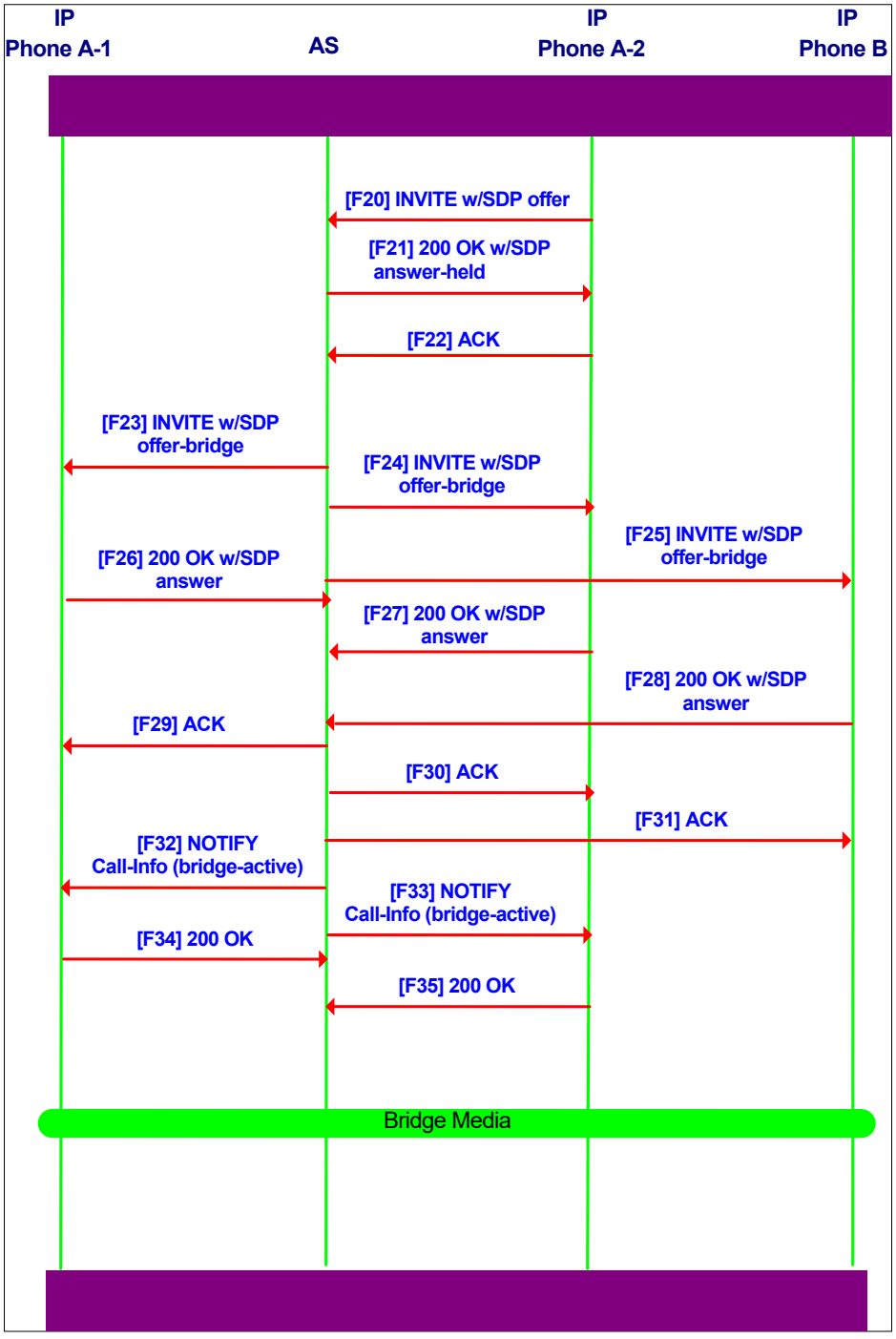


Figure 24 User at IP Phone A-2 Barges into Active Call

In [F20] the IP phone A-2 initiates an INVITE to barge in to the active call, as follows.

```
[F20] INVITE A-2 → Application Server
INVITE sip:as.foo.com SIP/2.0
Via: SIP/2.0/UDP a2.foo.com:5060;branch=m9hG4bK74bf9
Max-Forwards: 70
From: "Shared-A" <sip:shared-a2@as.foo.com>;tag=5fxced76s3
To: "Shared-A" <sip:shared-a2@as.foo.com>
Call-ID: 984827629822018851 3@a2.foo.com
Call-Info:  <sip:as.foo.com>;appearance-index=1
CSeq: 43294 INVITE
Contact: <sip:shared-a2@a2.foo.com>
Content-Type: application/sdp
Content-Length: 143

v=0
o=UserA 2890844526 2890844526 IN IP4 a2.foo.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

The Application Server looks at the *Request-URI*, the *From* and *To* headers, and the *Call-Info* header to determine that A-2 is attempting to barge in to the active call on A-1. The Application Server applies local policy and determines that A-2 is authorized to barge in to the call. It acknowledges the INVITE request with a 200 response with a held SDP, and proceeds with the creation of a conference bridge. A-1, A-2, and B are then individually re-invited to the bridge [F23-F31]. A NOTIFY with Call-Info event state is broadcast to all the endpoints in the shared group.

#### 6.4.11.1 Silent Monitoring

Shared Call Appearance also supports the ability to invoke Silent Monitor Bridging. Silent Monitor Bridging provides the ability for the user to bridge in to a call in silent monitoring mode where the monitoring (invoking) user can hear the other parties, but the other parties cannot hear the monitoring user.

Silent Monitor Bridging is invoked as an option on the above SIP INVITE with *Call-Info* [F20] where the *silent-monitor* parameter in added to the header.

The following is an example of the *Call-Info* header with the *silent-monitor* parameter.

```
[F20] INVITE A-2 → Application Server
INVITE sip:as.foo.com SIP/2.0
Via: SIP/2.0/UDP a2.foo.com:5060;branch=m9hG4bK74bf9
Max-Forwards: 70
From: "Shared-A" <sip:shared-a2@as.foo.com>;tag=5fxced76s3
To: "Shared-A" <sip:shared-a2@as.foo.com>
Call-ID: 984827629822018851 3@a2.foo.com
Call-Info:  <sip:as.foo.com>;appearance-index=1;silent-monitor
CSeq: 43294 INVITE
Contact: <sip:shared-a2@a2.foo.com>
Content-Type: application/sdp
Content-Length: 143

v=0
o=UserA 2890844526 2890844526 IN IP4 a2.foo.com
s=-
c=IN IP4 192.0.2.101
```

```
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Note that if the SIP INVITE contains a *Call-Info* header with the *silent-monitor* parameter but the INVITE is not treated as a bridging INVITE (that is, it is treated as a new call or a call retrieval instead), then the *silent-monitor* parameter is ignored.

This behavior is controlled by the *enableSilentMonitoring* system parameter from *AS_CLI/Applications/ExecutionAndProvisioning/LawfulIntercept*. When this attribute is set to “false”, the user can still trigger bridging but they are not muted.

### 6.4.12 Call Park

Call Park is a service provided by Cisco BroadWorks to keep calls on hold and be able to retrieve them at a later time, potentially from another user. A call is parked on a specific target user and can then be retrieved by specifying the extension of that user.

A parameter in the Shared Call Appearance service allows enabling the transmission of parked call information via the existing *Call-Info* subscription dialog. If the parameter is enabled and the *Accept* header in the SUBSCRIBE request contains *application/x-broadworks-callpark-info+xml*, then the following behavior to applies (otherwise the regular behavior applies, as described in the previous sections).

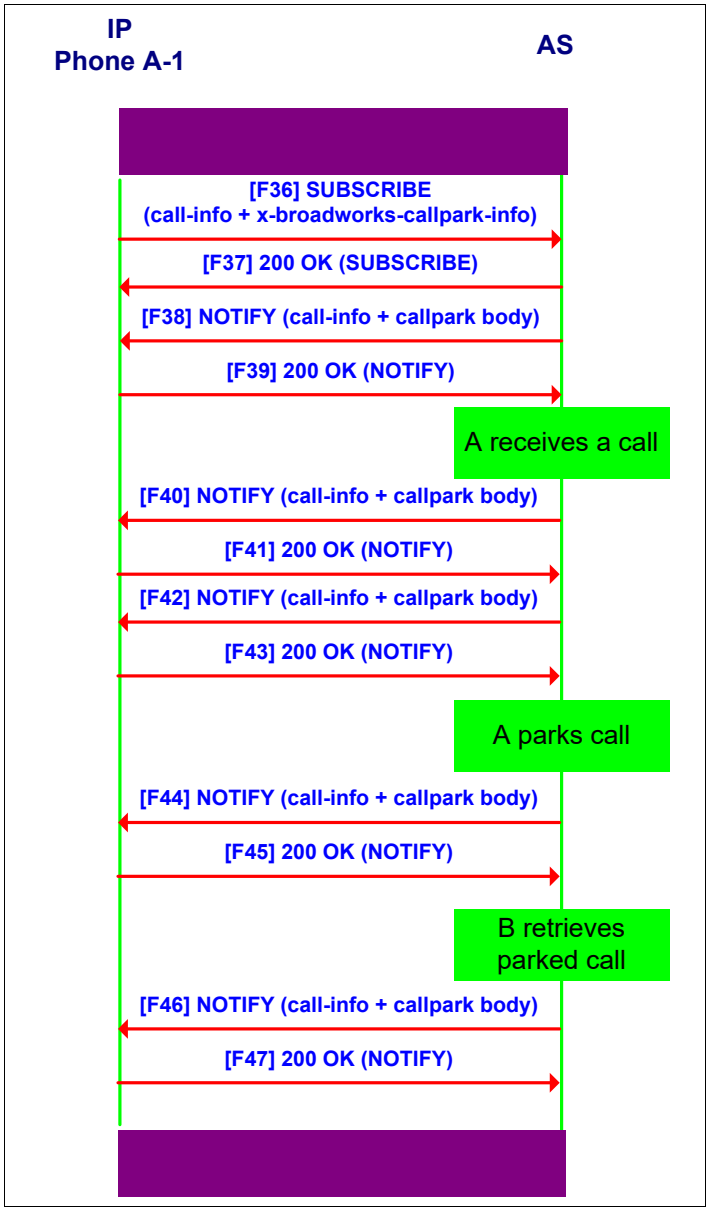


Figure 25 User at IP Phone A-1 has Call Park Notification Enabled

### Initial Subscription

```
[F36] SUBSCRIBE A-1 -> Application Server
SUBSCRIBE sip:shared-a1@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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>
Call-ID: 5j9FpLxk3uxtm8to@a1.foo.com
CSeq: 6 SUBSCRIBE
Event: call-info
Expires: 3600
Accept: application/x-broadworks-callpark-info+xml
Contact: <sip:shared-a1@a1.foo.com>
Content-Length: 0
```

```
[F37] 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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 6 SUBSCRIBE
Event:  call-info
Contact: sip:shared-a1@as.foo.com
Expires: 3600
Content-Length: 0

[F38] NOTIFY Application Server -> A-1
NOTIFY sip:shared-a1@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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@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-a1@as.foo.com
Call-Info:  <sip:as.foo.com>;appearance-index=*;appearance-state=idle
Content-Type:application/x-broadworks-callpark-info+xml
Content-Length:149

<?xml version="1.0" encoding="UTF-8"?>
<x-broadworks-callpark-info
  xmlns="http://schema.broadsoft.com/callpark">
 <callpark/>
</x-broadworks-callpark-info>

[F39] 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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 1344 NOTIFY
Content-Length: 0
```

### User A receives a regular call

```
[F40] NOTIFY Application Server -> A-1
NOTIFY sip:shared-a1@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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 1489 NOTIFY
Event: call-info
Subscription-State: active; expires=3599
Contact:  sip:shared-a1@as.foo.com
Call-Info: <sip:as.foo.com>;appearance-uri="\"B. Foo\"<sip:
B@as.foo.com;user=phone>";appearance-index=1;appearance-state=alerting,
           <sip:as.foo.com>;appearance-index=*;appearance-state=idle
Content-Type:application/x-broadworks-callpark-info+xml
Content-Length:149

<?xml version="1.0" encoding="UTF-8"?>
```

```
<x-broadworks-callpark-info
  xmlns="http://schema.broadsoft.com/callpark">
 <callpark/>
</x-broadworks-callpark-info>
```

***[F41] 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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 1489 NOTIFY
Content-Length: 0
```

***[F42] NOTIFY Application Server -> A-1***

```
NOTIFY sip:shared-a1@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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 1495 NOTIFY
Event: call-info
Subscription-State: active; expires=3599
Contact:  sip:shared-a1@as.foo.com
Call-Info:  <sip:as.foo.com>;appearance-uri="\"B. Foo\"<sip:
B@as.foo.com;user=phone>";appearance-index=1;appearance-state=active,
            <sip:as.foo.com>;appearance-index=*;appearance-state=idle
Content-Type:application/x-broadworks-callpark-info+xml
Content-Length:149

<?xml version="1.0" encoding="UTF-8"?>
<x-broadworks-callpark-info
  xmlns="http://schema.broadsoft.com/callpark">
 <callpark/>
</x-broadworks-callpark-info>
```

***[F43] 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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 1495 NOTIFY
Content-Length: 0
```

### A call is parked on User A

***[F44] NOTIFY Application Server -> A-1***

```
NOTIFY sip:shared-a1@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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 1534 NOTIFY
Event: call-info
Subscription-State: active; expires=3599
Contact:  sip:shared-a1@as.foo.com
Call-Info:  <sip:as.foo.com>;appearance-uri="\"B. Foo\"<sip:
B@as.foo.com;user=phone>";appearance-index=1;appearance-state=active,
            <sip:as.foo.com>;appearance-index=*;appearance-state=idle
```

```
Content-Type:application/x-broadworks-callpark-info+xml
Content-Length:333

<?xml version="1.0" encoding="UTF-8"?>
<x-broadworks-callpark-info
  xmlns=http://schema.broadsoft.com/callpark>
 <callpark>
  <parked>
   <identity display="C. Parked">
    sip:C@as.foo.com;user=phone
   </identity>
  </parked>
 </callpark>
</x-broadworks-callpark-info>
```

***[F45] 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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 1534 NOTIFY
Content-Length: 0
```

**Parked call is removed from User A**

***[F46] NOTIFY Application Server -> A-1***

```
NOTIFY sip:shared-a1@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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 1577 NOTIFY
Event: call-info
Subscription-State: active; expires=3599
Contact:  sip:shared-a1@as.foo.com
Call-Info:  <sip:as.foo.com>;appearance-uri="\"B. Foo\"<sip:
B@as.foo.com;user=phone>";appearance-index=1;appearance-state=active,
           <sip:as.foo.com>;appearance-index=*;appearance-state=idle
Content-Type:application/x-broadworks-callpark-info+xml
Content-Length:149

<?xml version="1.0" encoding="UTF-8"?>
<x-broadworks-callpark-info
  xmlns="http://schema.broadsoft.com/callpark">
 <callpark/>
</x-broadworks-callpark-info>
```

***[F47] 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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 1577 NOTIFY
Content-Length: 0
```

# 7 “Dialog” Approach

This section describes the “Dialog” approach. It provides the following functions:

- Hold/Retrieve from distinct locations
- Barge-in (bridging)
- Line/Lamp management
- Call state management

This approach relies solely on standard-track Request for Comments (RFCs), thus expected to be interoperable with a broader range of SIP devices.

However, there is no equivalent to the “line seizure” concept with the Dialog approach. (The *draft-ietf-bliss-shared-appearances-02* [8] proposes some mechanisms similar to line-seize; however, such mechanisms have not been implemented by Cisco BroadWorks as of the current release.) As such, appearance is always assigned dynamically by the Application Server during the INVITE SIP dialog setup. (Line-seizure reports events *ahead* of the INVITE SIP dialog setup). A consequence of this is when a shared line device is originating or receiving a call it cannot assign it to a “fixed” appearance and must select one dynamically; appearance display consistency across shared line devices cannot be guaranteed.

Even when implementing the “Dialog” approach, SIP devices still receive the *Call-Info* header in SIP INVITE requests and responses from the Application Server. The appearance information provided in the *Call-Info* header is always consistent with the SIP dialog information provided in NOTIFY requests.

## 7.1 Application Server Requirements

### 7.1.1 Configuration

The Application Server configuration requirements are as follows:

- The Application Server must allow an administrator to configure which users are configured with shared lines and which are configured with private lines.
- The Application Server must statically understand or be able to dynamically discover the line key and call appearance capabilities of the IP phone associated with a given user, and adjust service provisioning options to match the capabilities.
- The Application Server must generate phone configuration information based on the capabilities of the IP phone and the services provisioned against the user.

### 7.1.2 Execution

The Application Server execution requirements are as follows:

- The Application Server must allow multiple endpoints to register locations against a shared line.
- The Application Server must determine the endpoint registering against a shared line and which policies apply to it.
- The Application Server must allow multiple endpoints to SUBSCRIBE to the “Dialog” event package on a shared line.
- If the Application Server is configured to challenge INVITE requests from endpoints, then it must also challenge SUBSCRIBE requests from endpoints.

- If the Application Server is configured to perform access control on INVITE requests from endpoints, then it must also perform access control on SUBSCRIBE requests from endpoints.
- After granting a subscription to a "Dialog" event package, the Application Server must initialize the IP phone to the current shared line call appearance state by sending a NOTIFY request with SIP dialog status in an application/dialog-info+xml body. Each ongoing dialog is associated with an appearance number that is also included in the dialog-info+xml body, for each <dialog> element. The NOTIFY request is sent even if all call appearances on the shared line are idle (that is, there is no INVITE SIP dialogs established for all endpoints), in which case the application/dialog-info+xml body contains a <dialog-info> element with no children. This notification is a "full" dialog notification, as defined in *RFC 4235* [5].
- The Application Server MUST send "full" dialog information state in a NOTIFY request each time a device is subscribing or refreshing an existing subscription to the "Dialog" event package.
- To minimize bandwidth usage, the Application Server MUST send "partial" dialog information in NOTIFY requests (at the exception of NOTIFY requests following an initial SUBSCRIBE or a SUBSCRIBE-refresh).
- When a call is originated or terminated on a shared line, the Application Server MUST dynamically assign an appearance index to the call. Shared line devices "know" about assigned appearances through dialog event package notifications.
- When a call arrives for a shared line, the Application Server must attempt to terminate the call on every endpoint that has successfully registered against the shared line.
- The Application Server must send NOTIFY requests to all subscribed endpoints whenever a change occurs in a SIP dialog associated with the shared line. This includes dialog state change as well as media rendering changes (media rendering is used to describe hold condition by the shared line device). The NOTIFY request only contains the changed SIP dialog. Note that Cisco BroadWorks supports a subset of all dialog states defined in *RFC 4235* [5]*: proceeding* (which encompasses the *trying, early* states), *confirmed, and terminated*.
  - The Application Server MUST provide SIP dialog identifiers (*call-id, from-tag*, and *to-tag*) for *confirmed* dialogs.
  - The *sip.rendering* parameter is used to describe active/held dialogs. The *sip.rendering* parameter is only provided for the local target, that is, the target associated with the shared line device. The *Holding* state from the remote party is not reported in dialog notifications.
- When one of the registered endpoints answers the call, the Application Server must CANCEL calls to all other registered endpoints. As this implies SIP dialog state changes (that is, one confirmed SIP dialog and others being terminated), the Application Server must then send a NOTIFY request to all subscribed endpoints with an application/dialog-info+xml body containing information about all changed SIP dialogs (that is, it sends a "partial" dialog notification).
- When an INVITE request arrives from an endpoint that is participating in a shared line, the Application Server must only originate the call if the INVITE request is authorized to make a call (that is, via ACL or an explicit INVITE challenge). Otherwise, the Application Server must reject the call with a *480 Temporarily Unavailable* response. If the call is originated, the call appearance is dynamically assigned by the Application Server.

- When originating calls, the Application Server sends a NOTIFY request to all subscribed endpoints with an application/dialog-info+xml body containing information about the newly created SIP dialog (that is, it sends a "partial" dialog notification).
- When a call is terminated to a shared line subscriber, all SIP phones registered for the shared line are alerted; the Application Server sends a dialog NOTIFY request to all subscribed endpoints containing all corresponding *proceeding* SIP dialogs. When the call is answered at one of the phones, the Application Server sends a NOTIFY request containing a *confirmed* dialog matching the answering endpoint and *terminated* dialogs for all other alerted endpoints.

## 7.2 Device Requirements

### 7.2.1 Configuration

The device configuration requirements are as follows:

- The device MUST securely retrieve configuration information from a server in the hosted provider network at startup time.
- The device configuration information MUST be able to identify how many private lines and how many shared lines are to be configured on the phone.
- The device MUST allow for a separate display name, address-of-record, and credentials (user name and password) for each line (shared or private).
- The device configuration information SHOULD be able identify how many simultaneous call appearances should be accepted for each line before the phone sends back busy.

### 7.2.2 Execution

The device execution requirements are as follows:

- The device must send a separate REGISTER request for each configured line (private or shared) according to *RFC 3261* [1].
- The device must also send a separate SUBSCRIBE request for each configured shared line, according to *RFC 3265* [3]. The subscription is for the "Dialog" event package documented in *RFC 4235* [5]. The device must use its address of record in the SUBSCRIBE's *Request-URI* (for fresh SUBSCRIBEs)[1].
- The device must continually refresh the registrations before they expire, according to *RFC 3261* [1].
- The device must continually refresh the "dialog" subscriptions before they expire, according to *RFC 3265* [3] and *RFC 4235* [5].
- The device must be prepared to provide corresponding credentials for authentication challenges for any REGISTER, INVITE, or SUBSCRIBE transactions.
- When a call arrives for one of the lines (private or shared), the phone must alert the user of the new call appearance and identify which line it is associated with.
- If the maximum number of simultaneous call appearances has been reached for a particular line, then the phone must return busy for any new call appearances that arrive for the line.

[1] The Application Server uses the *Request-URI* to discriminate between dialog subscriptions for Busy Lamp Field and dialog subscriptions for Shared Call Appearance. For more information on dialog subscriptions to Busy Lamp Field lists, see the *BroadWorks Busy Lamp Field Interface Specification* [10].

- Due to the distributed nature of the model, devices should assume that a transition from any state to any other state is possible. That is, devices should not rely on any particular state machine. When receiving a "partial" notification, the device must update the state of dialogs reported in the application/dialog-info+xml body only and assume that any other ongoing SIP dialogs remained unchanged. When receiving a "full" notification, it MUST completely reset all dialogs state to those specified in the application/dialog-info+xml body. Dialogs that are no longer specified in a "full" notification must be considered as "terminated". For example due to network outage the device might have missed a NOTIFY indicating that a SIP dialog has terminated.
- When a NOTIFY arrives for one of the shared lines, the phone must perform access control and only accept the request if the source IP address corresponds to one of the IP addresses of the Application Server.
- IP phones must map the following dialog states to suitable appearances on the phone user interface:
  - *terminated*: A SIP dialog is terminated, so it no longer uses the appearance.
  - *proceeding*: A call setup is in progress, either originating or terminating.
  - *confirmed, not rendering media (held)*: A call exists, but is being held by the shared line device.
  - *confirmed*: An active call exists on the shared line device.
- After a NOTIFY request has been accepted, the device should modify the call appearance state of the associated line based on the body of the NOTIFY request. In case multiple SIP dialogs are associated with the same appearance (that is, Barge-in/Bridging), the IP phones should use the *most confirmed* dialog state among all SIP dialogs associated with the appearance to render the appearance state. The *most confirmed* dialog state is defined as following:
  - If there is at least one "confirmed" dialog, the displayed appearance state should be *confirmed* (no matter whether held or active).
  - If there is no "confirmed" dialog but there is at least a "proceeding" dialog associated with the appearance, the displayed appearance state should be *proceeding*.
  - In all other cases, the line appearance is displayed as "Idle".
- The device SHOULD use the following guidelines for mapping the dialog state received in NOTIFY requests to the state of each call appearance on a shared line:

**"terminated" or no dialog for the appearance:**

- When there is no SIP dialog associated with an appearance or when a SIP dialog is reported as "terminated", then the call appearance on that shared line is considered idle.
- If applicable, the LED for the call appearance SHOULD be OFF.

**"proceeding":**

- This indicates that a SIP dialog to or from a shared line device is being set up on a specific appearance.
- In termination cases, endpoints receiving the INVITE from the Application Server MUST alert the user with an audible/visible signal like a basic SIP call termination. If the IP phone understands the *Call-Info* header, it may correlate the terminating call with the relevant appearance LED.

- In origination cases, the endpoint initiating the SIP INVITE dialog must perform call setup like a normal basic SIP call origination. If the IP phone understands the *Call-Info* header, it may correlate the originating call with the relevant appearance LED when receiving response messages from the Application Server.
- Depending on the application, the IP phone not involved in "proceeding" dialogs may or may not provide an indicator[2] when receiving NOTIFY requests indicating such dialogs.

**"confirmed":**

- This indicates that a SIP dialog to or from a shared line device is now completely set up. For confirmed dialogs, SIP dialog identifiers (*call-id, local-tag, remote-tag*) are specified so SIP phones can correlate active dialogs with their own dialogs, (thus rendering the user the interface differently).
- There may be one or more dialogs associated with a single appearance; this indicates a barge-in (bridging) situation.
- "confirmed" dialogs are by default active unless the dialog's <local> element contains the <param pname="+sip.rendering" pval="no"/> child element, in which case the dialog is held by the device. If all dialogs associated with the appearance are held, this means the remote party is held.
- IP phones should indicate that the line is active visually. If applicable, the LED of the call appearance from the endpoint that has explicitly answered the call should be solid green, indicating that the call has been answered and is active locally.
- The LED of the call appearance on all other endpoints should be solid red, indicating that the call is active on another endpoint on the shared line.
- IP phones involved in "confirmed" SIP dialogs for a call appearance should present the user with the option to hold the appearance (or retrieve the appearance if already locally held).
- IP phones NOT involved in "confirmed" SIP dialogs for a specific call appearance should present to the user the options to barge-in (bridge) on the call appearance or to retrieve the call appearance[3]. If there is more than one SIP dialog associated with a given appearance, only the barge-in (bridge) option shall be presented.
- If the user selects the *barge-in* option, the IP phone must initiate a new SIP INVITE dialog containing the *Join* header populated with the SIP dialog information for one of the dialogs associated with the appearance. The user is then bridged to all other endpoints already associated with the call appearance.
- If the user selects the *retrieve* option, the IP phone must initiate a new SIP INVITE dialog containing the *Replaces* header populated with the SIP dialog information for one of the dialogs associated with the appearance. This connects the endpoint to the remote party. All other SIP dialogs associated with the call appearance are released by the Application Server.

[2] Dialog notifications do not report full SIP dialog information (*call-id, local-tag*, and *remote-tag*) for dialogs in the *proceeding* state, since the SIP dialog may not be fully specified at that time. As such, shared line devices may not be able to correlate the proceeding dialogs with the dialogs they are currently involved in. It might then not be desirable to provide any indicator for SIP dialogs in the *proceeding* state. For applications where such correlation is not necessary, the device can show such SIP dialog state indication.

[3] This is different from retrieving a SIP dialog held locally, which consists simply of sending a re-INVITE with an active SDP. In this case, the Application Server does not release the other SIP dialogs associated with the same call appearance.

– IP phones may render different appearances involved with a single SIP dialog or with multiple dialogs. (When there is more than one SIP dialog, this indicates a barge-in.)

## 7.3 Functional Overview

This section is similar to section *6.4 Functional Overview*, but adapted for the "dialog" approach. Again, the intent is to show an application-neutral set of call flows covering the most common interactions between SIP IP phones and the Application Server.

The example setup described in section *6.4 Functional Overview* is used here as well. The setup description was copied for ease of reading.

Figure 26 shows the network configuration for the examples that follow. IP phone A-1 (a1.foo.com) and IP phone A-2 (a2.foo.com) are monitoring the same shared line (sip: shared-a@as.foo.com) hosted on the Application Server (as.foo.com).

Phone B is behind a SIP network gateway and represents the off-network party in the call flows. Note that each phone also has a respective private line: sip:private-a1@a1.foo.com and sip:private-a2@a2.foo.com.

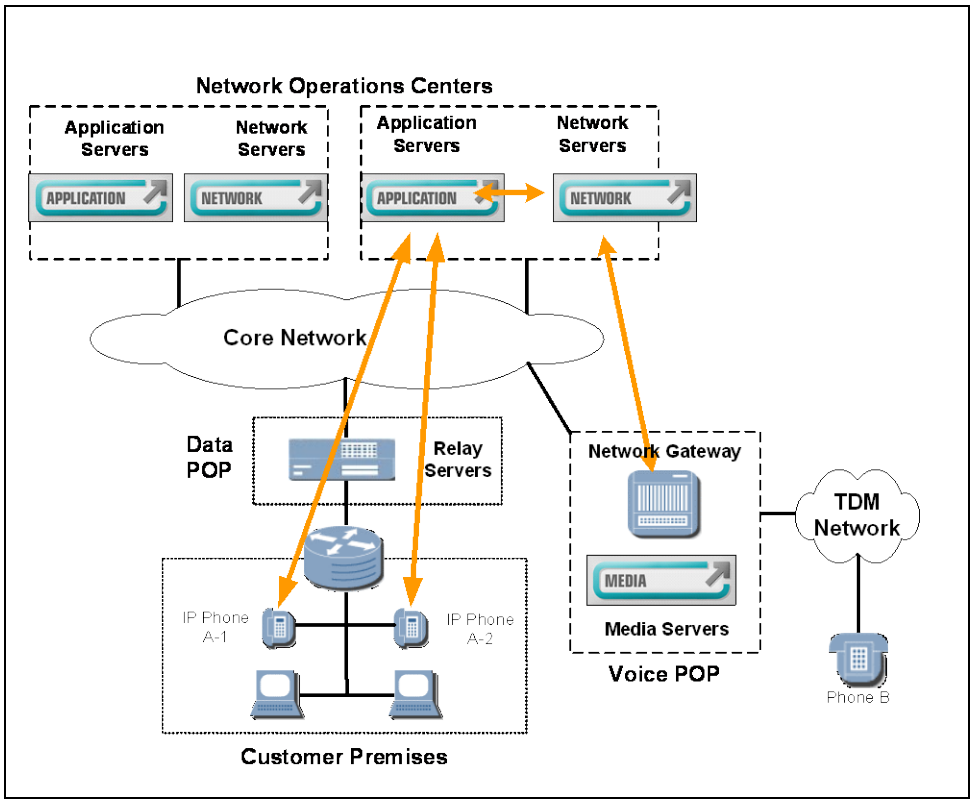


Figure 26 Functional Overview

### 7.3.1 IP Phone Power Up

*Figure 11 Phone A-1 Powers Up* shows the message flow as a result of powering up IP phone A-1.

#### 7.3.1.1 Step 1.1: A-1 Phone Configuration

The phones must first retrieve network topology and device configuration information. This is done through a "Configuration Request" in [F1]. The transport for the configuration request can be any number of options: FTP, TFTP, Hypertext Transfer Protocol (HTTP), or Hypertext Transfer Protocol Secure Sockets (HTTPS). HTTPS is preferred as it offers confidential transport of potentially sensitive configuration information. The flow shows the Application Server (AS) as the target of the configuration request. In practice, this can be any centralized server on the network and does not have to be the Application Server. The "Configuration Response" in [F2] provides any number of device-specific configuration parameters, as well as the following critical information:

- Which line keys are *shared* call appearances
- Which line keys are *local* call appearances
- The display name for each line
- The address of record for each line
- The credentials for each line
- The registrar for each line
- The proxy for each line

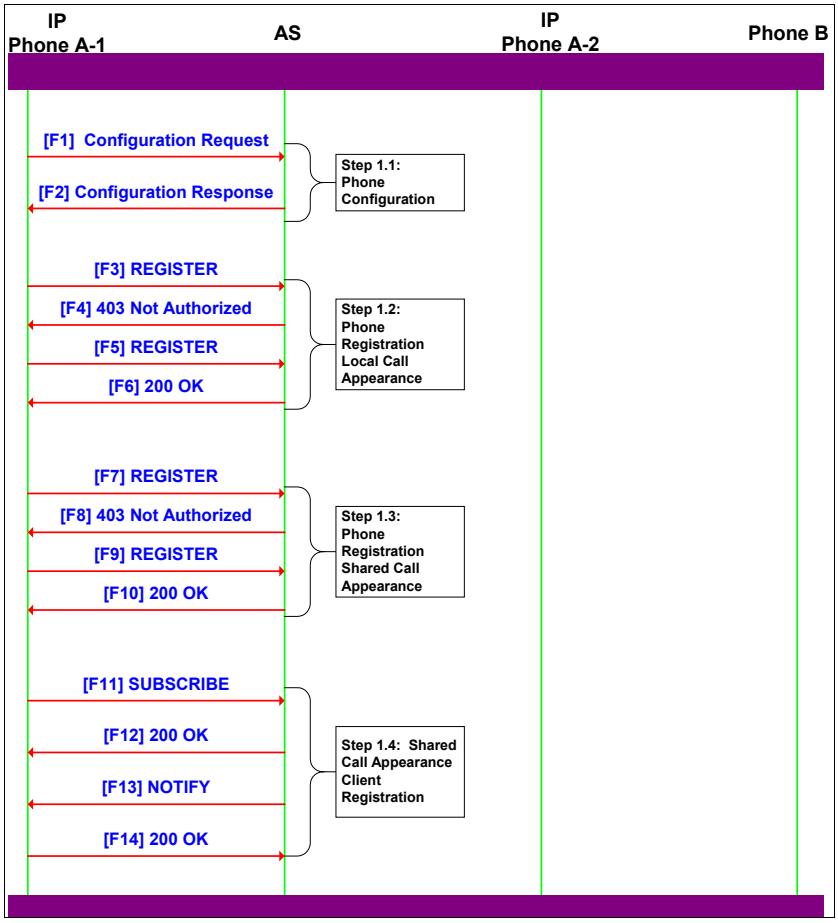


Figure 27 Phone A-1 Powers Up

### 7.3.1.2 Step 1.2: A-1 Phone Registration for Local Private Lines

[F3] to [F6] is performed for each unique private line. This is the standard registration process described in *RFC 3261* [1]. SIP MD-5 digest authentication is used to authenticate the endpoint.

```
[F3] REGISTER A-1 -> Application Server
REGISTER sip:as.foo.com SIP/2.0
Via: SIP/2.0/UDP a1.foo.com:5060;branch=z9hG4bKnashds7
Max-Forwards: 70
From: "Private-A1" <sip:private-a1@as.foo.com>;tag=a73kszlfl
To: "Private-A1" <sip:private-a1@as.foo.com>
Call-ID: 1j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 1 REGISTER
Contact: <sip:private-a1@a1.foo.com>
Content-Length: 0

[F4] 401 Unauthorized Application Server → A-1
SIP/2.0 401 Unauthorized
Via: SIP/2.0/UDP a1.foo.com:5060;branch=z9hG4bKnashds7
From: "Private-A1" <sip:private-a1@as.foo.com>;tag=a73kszlfl
To: "Private-A1" <sip:private-a1@as.foo.com>;tag=1410948204
Call-ID: 1j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 1 REGISTER
```

```
WWW-Authenticate: Digest realm="foo.com", qop="auth",
nonce="ea9c8e88df84f1cec4341ae6cbe5a359", opaque="", stale=FALSE,
algorithm=MD5
Content-Length: 0

[F5] REGISTER A-1 -> Application Server
REGISTER sip:as.foo.com SIP/2.0
Via: SIP/2.0/UDP a1.foo.com:5060;branch=asdf32nashds7
Max-Forwards: 70
From: "Private-A1" <sip:private-a1@as.foo.com>;tag=asa3dszlfl
To: "Private-A1" <sip:private-a1@as.foo.com>
Call-ID: 1j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 2 REGISTER
Contact: <sip:private-a1@a1.foo.com>
Authorization: Digest username="private-a1", realm="foo.com"
nonce="ea9c8e88df84f1cec4341ae6cbe5a359", opaque="",
uri="sip:as.foo.com", response="dfe56131d1958046689d83306477ecc"
Content-Length: 0

[F6] 200 OK Application Server → A-1
SIP/2.0 200 OK
Via: SIP/2.0/UDP a1.foo.com:5060;branch=asdf32nashds7
From: "Private-A1" <sip:private-a1@as.foo.com>;tag= asa3dszlfl
To: "Private-A1" <sip:private-a1@as.foo.com>;tag=323423233
Call-ID: 1j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 2 REGISTER
Contact: <sip:private-a1@a1.foo.com>;expires=3600
Content-Length: 0
```

#### 7.3.1.3 Step 1.3: A-1 Phone Registration for Shared Call Appearances

[F7] to [F10] is performed for each unique shared call appearance. Again, this is the standard registration process described in *RFC 3261* [1]. SIP MD5 digest authentication is used to authenticate the endpoint. There is no fundamental difference between this registration and the registration described in sections *7.3.1.1 Step 1.1: A-1 Phone Configuration* and *7.3.1.2 Step 1.2: A-1 Phone Registration for Local Private Lines*.

```
[F7] REGISTER A-1 -> Application Server
REGISTER sip:as.foo.com SIP/2.0
Via: SIP/2.0/UDP a1.foo.com:5060;branch=x9hG4bKnashds7
Max-Forwards: 70
From: "Shared-A" <sip:shared-a1@as.foo.com>;tag=b73kszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>
Call-ID: 1j9FpLxk3uxtm8to@a1.foo.com
CSeq: 3 REGISTER
Contact: <sip:shared-a1@a1.foo.com>
Content-Length: 0

[F8] 401 Unauthorized Application Server → A-1
SIP/2.0 401 Unauthorized
Via: SIP/2.0/UDP a1.foo.com:5060;branch=x9hG4bKnashds7
From: "Shared-A" <sip:shared-a1@as.foo.com>;tag=b73kszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=2410948204
Call-ID: 1j9FpLxk3uxtm8to@a1.foo.com
CSeq: 3 REGISTER
WWW-Authenticate: Digest realm="foo.com", qop="auth",
nonce="fa9c8e88df84f1cec4341ae6cbe5a359", opaque="", stale=FALSE,
algorithm=MD5
Content-Length: 0
```

```
[F9] REGISTER A-1 -> Application Server
REGISTER sip:as.foo.com SIP/2.0
Via: SIP/2.0/UDP a1.foo.com:5060;branch=bsdf32nashds7
Max-Forwards: 70
From: "Shared-A" <sip:shared-a1@as.foo.com>;tag=bsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>
Call-ID: 1j9FpLxk3uxtm8to@a1.foo.com
CSeq: 4 REGISTER
Contact: <sip:shared-a1@a1.foo.com>
Authorization: Digest username="shared-a1", realm="foo.com"
nonce="fa9c8e88df84f1cec4341ae6cbe5a359", opaque="",
uri="sip:as.foo.com", response="efe56131d1958046689d83306477ecc"
Content-Length: 0

[F10] 200 OK Application Server → A-1
SIP/2.0 401 Unauthorized
Via: SIP/2.0/UDP a1.foo.com:5060;branch=bsdf32nashds7
From: "Shared-A" <sip:shared-a1@as.foo.com>;tag= bsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=323423233
Call-ID: 1j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 4 REGISTER
Contact: <sip:shared-a1@a1.foo.com>;expires=3600
Content-Length: 0
```

#### 7.3.1.4 Step 1.4: A-1 Shared Call Appearance Client Subscription

[F11] to [F14] is performed by each device participating in a shared line. This process allows the endpoint to SUBSCRIBE to the dialog state of a given line. The *Request-URI* for these transactions is the same as the *Request-URI* used in the registration process in steps 1.3 and 2.3 in sections *7.3.1.3 Step 1.3: A-1 Phone Registration for Shared Call Appearances* and *7.3.1.7 Step 2.3: A-2 Phone Registration for Shared Call Appearances*.

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 dialog status by sending a NOTIFY with an application/dialog-info+xml body. This NOTIFY event is sent even if there is no ongoing SIP INVITE dialogs between the Application Server and any IP phone participating in the shared line. The application/dialog-info+xml contains a "full" dialog information state and serves as a synchronization event between the Application Server and the IP phone[4].

```
[F11] SUBSCRIBE A-1 -> Application Server
SUBSCRIBE sip:shared-a1@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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>
Call-ID: 5j9FpLxk3uxtm8to@a1.foo.com
CSeq: 6 SUBSCRIBE
Event: dialog
Expires: 3600
Accept: application/dialog-info+xml
Contact: <sip:shared-a1@a1.foo.com>
```

[4] In all examples, application/dialog-info+xml bodies have been reformatted for ease of reading.

```
Content-Length: 0

[F12] 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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 6 SUBSCRIBE
Event:  dialog
Contact: sip:shared-a1@as.foo.com
Expires: 3600
Content-Length: 0

[F13] NOTIFY Application Server -> A-1
NOTIFY sip:shared-a1@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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 1344 NOTIFY
Event: dialog
Subscription-State: active; expires=3599
Contact:  sip:shared-a1@as.foo.com
Content-Type:application/dialog-info+xml
Content-Length:230

<?xml version="1.0" encoding="UTF-8"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
   xmlns:sa="urn:ietf:params:xml:ns:sa-dialog-info"
   version="0" state="full"
   entity="sip:shared-a@as.foo.com"></dialog-info>

[F14] 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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 1344 NOTIFY
Content-Length: 0
```

#### 7.3.1.5 Step 2.1: A-2 Phone Configuration

The same steps described in sections *7.3.1.1 Step 1.1: A-1 Phone Configuration*, *7.3.1.2 Step 1.2: A-1 Phone Registration for Local Private Lines*, *7.3.1.3 Step 1.3: A-1 Phone Registration for Shared Call Appearances*, and *7.3.1.4 Step 1.4: A-1 Shared Call Appearance Client Subscription* are repeated for the phone A-2.

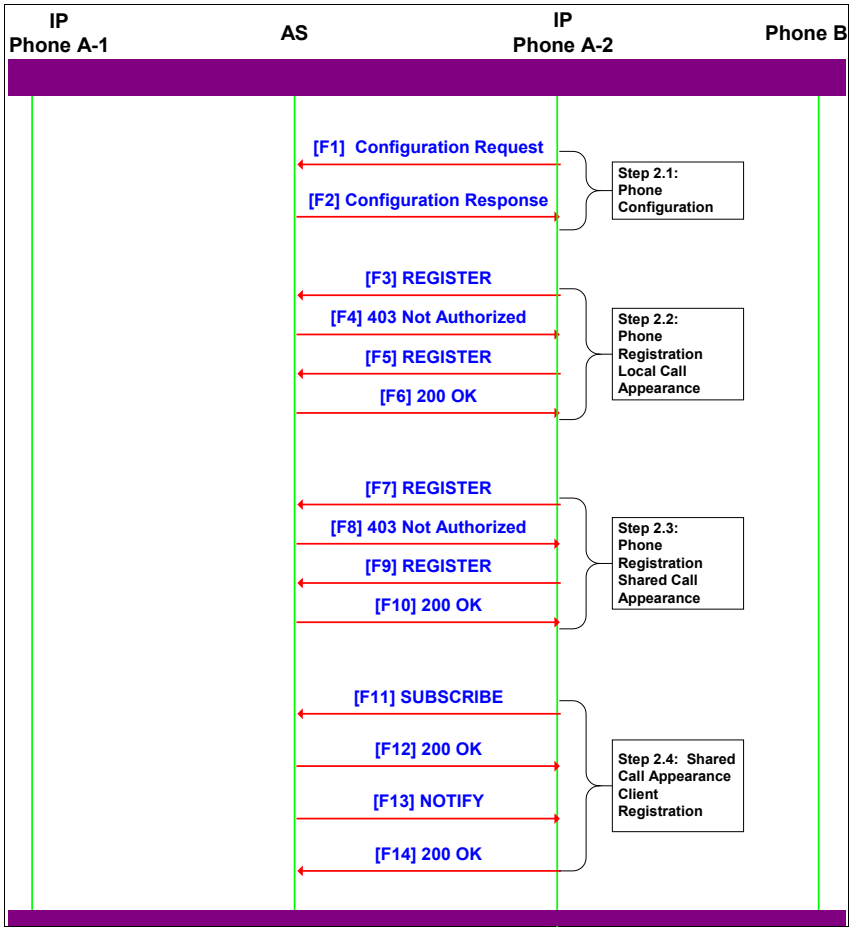


Figure 28 Phone A-2 Powers Up

#### 7.3.1.6 Step 2.2: A-2 Phone Registration for Local Private Lines

```
[F3] REGISTER A-2 -> Application Server
REGISTER sip:as.foo.com SIP/2.0
Via: SIP/2.0/UDP a2.foo.com:5060;branch=z9hG4bKnashds7
Max-Forwards: 70
From: "Private-A2" <sip:private-a2@as.foo.com>;tag=a73kszlfl
To: "Private-A2" <sip:private-a2@as.foo.com>
Call-ID: 1j9FpLxk3uxtm8tn@a2.foo.com
CSeq: 1 REGISTER
Contact: <sip:private-a2@a2.foo.com>
Content-Length: 0

[F4] 401 Unauthorized Application Server → A-2
SIP/2.0 401 Unauthorized
Via: SIP/2.0/UDP a2.foo.com:5060;branch=z9hG4bKnashds7
From: "Private-A2" <sip:private-a2@as.foo.com>;tag=a73kszlfl
To: "Private-A2" <sip:private-a2@as.foo.com>;tag=1410948204
Call-ID: 1j9FpLxk3uxtm8tn@a2.foo.com
```

```
CSeq: 1 REGISTER
WWW-Authenticate: Digest realm="foo.com", qop="auth",
nonce="ea9c8e88df84f1cec4341ae6cbe5a359", opaque="", stale=FALSE,
algorithm=MD5
Content-Length: 0

[F5] REGISTER A-2 -> Application Server
REGISTER sip:as.foo.com SIP/2.0
Via: SIP/2.0/UDP a2.foo.com:5060;branch=asdf32nashds7
Max-Forwards: 70
From: "Private-A2" <sip:private-a2@as.foo.com>;tag=asa3dszlfl
To: "Private-A2" <sip:private-a2@as.foo.com>
Call-ID: 2j9FpLxk3uxtm8tn@a2.foo.com
CSeq: 2 REGISTER
Contact: <sip:private-a2@a2.foo.com>
Authorization: Digest username="private-a2", realm="foo.com"
nonce="ea9c8e88df84f1cec4341ae6cbe5a359", opaque="",
uri="sip:as.foo.com", response="dfe56131d1958046689d83306477ecc"
Content-Length: 0

[F6] 200 OK Application Server → A-1
SIP/2.0 200 OK
Via: SIP/2.0/UDP a2.foo.com:5060;branch=asdf32nashds7
From: "Private-A2" <sip:private-a2@as.foo.com>;tag= asa3dszlfl
To: "Private-A2" <sip:private-a2@as.foo.com>;tag=323423233
Call-ID: 2j9FpLxk3uxtm8tn@a2.foo.com
CSeq: 2 REGISTER
Contact: <sip:private-a2@a1.foo.com>;expires=3600
Content-Length: 0
```

#### 7.3.1.7 Step 2.3: A-2 Phone Registration for Shared Call Appearances

```
[F7] REGISTER A-2 -> Application Server
REGISTER sip:as.foo.com SIP/2.0
Via: SIP/2.0/UDP a2.foo.com:5060;branch=x9hG4bKnashds7
Max-Forwards: 70
From: "Shared-A" <sip:shared-a2@as.foo.com>;tag=b73kszlfl
To: "Shared-A" <sip:shared-a2@as.foo.com>
Call-ID: 3j9FpLxk3uxtm8to@a2.foo.com
CSeq: 3 REGISTER
Contact: <sip:shared-a2@a1.foo.com>
Content-Length: 0

[F8] 401 Unauthorized Application Server → A-2
SIP/2.0 401 Unauthorized
Via: SIP/2.0/UDP a2.foo.com:5060;branch=x9hG4bKnashds7
From: "Shared-A" <sip:shared-a2@as.foo.com>;tag=b73kszlfl
To: "Shared-A" <sip:shared-a2@as.foo.com>;tag=2410948204
Call-ID: 3j9FpLxk3uxtm8to@a2.foo.com
CSeq: 3 REGISTER
WWW-Authenticate: Digest realm="foo.com", qop="auth",
nonce="fa9c8e88df84f1cec4341ae6cbe5a359", opaque="", stale=FALSE,
algorithm=MD5
Content-Length: 0

[F9] REGISTER A-2 -> Application Server
REGISTER sip:as.foo.com SIP/2.0
Via: SIP/2.0/UDP a2.foo.com:5060;branch=bsdf32nashds7
Max-Forwards: 70
From: "Shared-A" <sip:shared-a2@as.foo.com>;tag=bsa3dszlfl
```

```
To: "Shared-A" <sip:shared-a2@as.foo.com>
Call-ID: 4j9FpLxk3uxtm8to@a2.foo.com
CSeq: 4 REGISTER
Contact: <sip:shared-a2@a2.foo.com>
Authorization: Digest username="shared-a2", realm="foo.com"
nonce="fa9c8e88df84f1cec4341ae6cbe5a359", opaque="",
uri="sip:as.foo.com", response="efe56131d1958046689d83306477ecc"
Content-Length: 0

[F10] 200 OK Application Server → A-2
SIP/2.0 401 Unauthorized
Via: SIP/2.0/UDP a1.foo.com:5060;branch=asdf32nashds7
From: "Shared-A" <sip:shared-a2@as.foo.com>;tag=bsa3dszlfl
To: "Shared-A" <sip:shared-a2@as.foo.com>;tag=323423233
Call-ID: 4j9FpLxk3uxtm8tn@foo.com
CSeq: 4 REGISTER
Contact: <sip:shared-a2@a2.foo.com>;expires=3600
Content-Length: 0
```

#### 7.3.1.8 Step 2.4: A-2 Shared Call Appearance Client Subscription

```
[F11] SUBSCRIBE A-2 -> Application Server
SUBSCRIBE sip:shared-a2@as.foo.com SIP/2.0
Via: SIP/2.0/UDP a2.foo.com:5060;branch=ggsdf32nashds7
Max-Forwards: 70
From: "Shared-A" <sip:shared-a2@as.foo.com>;tag=ggsa3dszlfl
To: "Shared-A" <sip:shared-a2@as.foo.com>
Call-ID: 9k9FpLxk3uxtm8to@a2.foo.com
CSeq: 6 SUBSCRIBE
Event: dialog
Expires: 3600
Accept: application/dialog-info+xml
Contact: <sip:shared-a2@a2.foo.com>
Content-Length: 0

[F12] 200 OK Application Server → A-2
SIP/2.0 200 OK
Via: SIP/2.0/UDP a2.foo.com:5060;branch=ggsdf32nashds7
From: "Shared-A" <sip:shared-a2@as.foo.com>;tag=ggsa3dszlfl
To: "Shared-A" <sip:shared-a2@as.foo.com>;tag=123456
Call-ID: 9k9FpLxk3uxtm8tn@a2.foo.com
CSeq: 6 SUBSCRIBE
Event: dialog
Contact:  sip:shared-a2@as.foo.com
Expires: 3600
Content-Length: 0

[F13] NOTIFY Application Server -> A-2
NOTIFY sip:shared-a2@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-a2@as.foo.com>;tag=ggsa3dszlfl
To: "Shared-A" <sip:shared-a2@as.foo.com>;tag=123456
Call-ID: 9k9FpLxk3uxtm8tn@a2.foo.com
CSeq: 9 NOTIFY
Event: dialog
Contact:  sip:shared-a2@as.foo.com
Subscription-State: active; expires=3599
Content-Type:application/dialog-info+xml
Content-Length:230
```

```
<?xml version="1.0" encoding="UTF-8"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
   xmlns:sa="urn:ietf:params:xml:ns:sa-dialog-info"
   version="0" state="full"
   entity="sip:shared-a@as.foo.com"></dialog-info>
```

***[F14] 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-a2@as.foo.com>;tag=ggsa3dszlfl
To: "Shared-A" <sip:shared-a2@as.foo.com>;tag=123456
Call-ID: 9k9FpLxk3uxtm8tn@a2.foo.com
CSeq: 9 NOTIFY
Content-Length: 0
```

### 7.3.2 A-1 Calls B

*Figure 29* shows the call flow as a result of A-1 originating a call to B using the shared line.

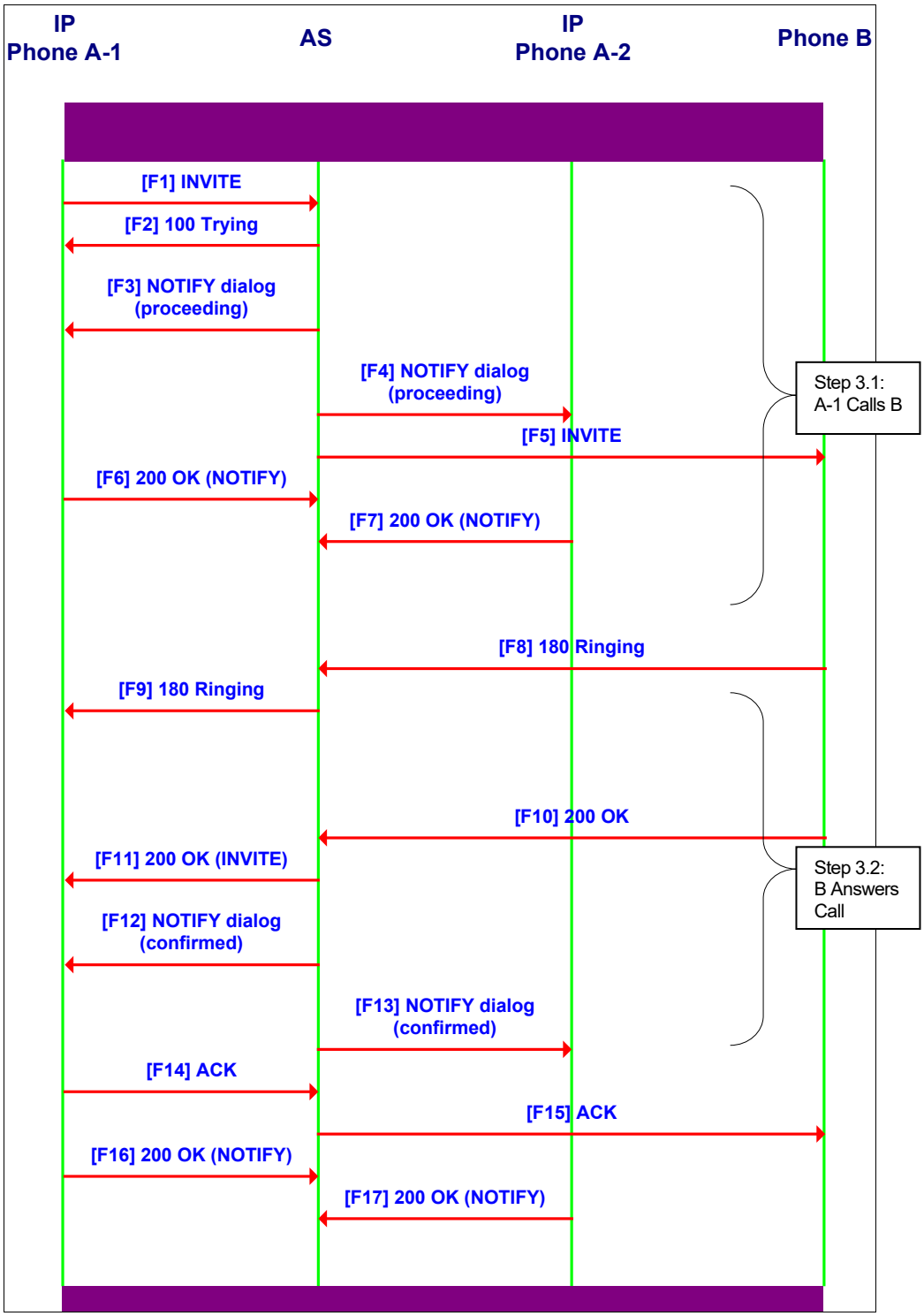


Figure 29 A-1 Calls B

#### 7.3.2.1 Step 3.1: A-1 Calls B

In [F1 to F9], the user on IP phone A-1 has entered digits for a call origination to B. The Application Server accepts the INVITE like a standard outbound call. As the call progresses, the Application Server sends dialog NOTIFY-requests to all endpoints having subscribed to the dialog event package on the shared line [5]. In this example, IP phones A-1 and A-2 have both subscribed to the dialog event package, so the Application Server sends a NOTIFY-request to both A-1 and A-2 with an application/dialog-info+xml body indicating a SIP dialog in the *proceeding* state.

```
[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-a1@as.foo.com>;tag=5fxced76sl
To: "B" <sip:B@as.foo.com>
Call-ID: 98482762982201885l1@a1.foo.com
CSeq: 43234 INVITE
Contact: <sip:shared-a1@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-a1@as.foo.com>;tag=5fxced76sl
To: "B" <sip:B@foo.com>; tag=4323z39
Call-ID: 98482762982201885l1@a1.foo.com
Call-Info:  <sip:as.foo.com>;appearance-index=1
CSeq: 43234 INVITE
Contact: <sip:@as.foo.com>
Content-Length: 0

[F3] NOTIFY Application Server -> A-1
NOTIFY sip:shared-a1@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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn @a2.foo.com
CSeq: 1344 NOTIFY
Event: dialog
Subscription-State: active; expires=3200
Content-Type:application/dialog-info+xml
Content-Length:xxx

<?xml version="1.0" encoding="UTF-8"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
  xmlns:sa="urn:ietf:params:xml:ns:sa-dialog-info" version="1"
  state="partial" entity="sip:shared-a@as.foo.com">
  <dialog id="bG9jYWxIb3N0MzIwOjA=" direction="initiator">
```

[5] The subscription must have been done using the endpoint's address of record as the SUBSCRIBE's *Request-URI*. Dialog subscriptions for Busy Lamp Field is are not covered in this document.

```
    <state>proceeding</state>
    <local>
      <identity display="Shared-A">sip:shared-a@as.foo.com
      </identity>
      <target uri="sip:shared-a1@a1.foo.com"></target>
    </local>
    <remote>
      <identity display="B">
        sip:B@foo.com</identity>
    </remote>
    <sa:appearance>1</sa:appearance>
  </dialog>
</dialog-info>
```

***[F4] NOTIFY Application Server -> A-2***

```
NOTIFY sip:shared-a2@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-a2@as.foo.com>;tag=ggsa3dszlfl
To: "Shared-A" <sip:shared-a2@as.foo.com>;tag=123456
Call-ID: 9k9FpLxk3uxtm8tn@a2.foo.com
CSeq: 1345 NOTIFY
Event: dialog
Subscription-State: active; expires=3100
Content-Type:application/dialog-info+xml
Content-Length:xxx

<?xml version="1.0" encoding="UTF-8"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
  xmlns:sa="urn:ietf:params:xml:ns:sa-dialog-info" version="1"
  state="partial" entity="sip:shared-a@as.foo.com">
  <dialog id="bG9jYWxIb3N0MzIwOjA=" direction="initiator">
    <state>proceeding</state>
    <local>
      <identity display="Shared-A">sip:shared-a@as.foo.com
      </identity>
      <target uri="sip:shared-a1@a1.foo.com"></target>
    </local>
    <remote>
      <identity display="B">
        sip:B@foo.com</identity>
    </remote>
    <sa:appearance>1</sa:appearance>
  </dialog>
</dialog-info>
```

***[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-a1@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] 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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn @a1.foo.com
CSeq: 1344 NOTIFY
Content-Length: 0

[F7] 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-a2@as.foo.com>;tag=ggsa3dszlfl
To: "Shared-A" <sip:shared-a2@as.foo.com>;tag=123456
Call-ID: 9k9FpLxk3uxtm8tn @a2.foo.com
CSeq: 1345 NOTIFY
Content-Length: 0

[F8] 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

[F9] 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-a1@as.foo.com>;tag=5fxced76sl
To: "B" <sip:B@as.foo.com>;tag=4323z39
Call-ID: 9848276298220188511@a1.foo.com
CSeq: 43234 INVITE
Contact: <sip:B@as.foo.com>
Content-Length: 0
```

#### 7.3.2.2 Step 3.2: B Answers Call

[F10 to F17] shows B answering the call using the standard back-to-back user agent call flow, but adding dialog notifications sent by the Application Server to A-1 and A-2. A-1 and B are now active in a call.

```
[F10] 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

[F11] 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-a1@as.foo.com>;tag=5fxced76sl
To: "B" <sip:B@as.foo.com>; tag=4323z39
Call-ID: 9848276298220188511 @a2.foo.com
Call-Info:  <sip:as.foo.com>;appearance-index=1
CSeq: 43234 INVITE
Contact: <sip:B@as.foo.com>
Content-Length: 0

[F12] NOTIFY Application Server → A-1
NOTIFY sip:shared-a1@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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 1344 NOTIFY
Event: dialog
Subscription-State: active; expires=3000
Content-Type:application/dialog-info+xml
Content-Length:xxx

<?xml version="1.0" encoding="UTF-8"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
  xmlns:sa="urn:ietf:params:xml:ns:sa-dialog-info" version="2"
  state="partial" entity="sip:shared-a@as.foo.com">
  <dialog id="bG9jYWxIb3N0MzIwOjA=" direction="initiator"
    call-id="5j9FpLxk3uxtm8tn@a1.foo.com" local-tag="5fxced76sl"
    remote-tag="4323z39">
    <state>confirmed</state>
    <local>
      <identity display="Shared-A">sip:shared-a@as.foo.com
      </identity>
      <target uri="sip:shared-a1@a1.foo.com"></target>
    </local>
    <remote>
      <identity display="B">sip:B@as.foo.com
      </identity>
    </remote>
    <sa:appearance>1</sa:appearance>
  </dialog>
</dialog-info>

[F13] NOTIFY Application Server -> A-2
NOTIFY sip:shared-a2@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-a2@as.foo.com>;tag=ggsa3dszlfl
To: "Shared-A" <sip:shared-a2@as.foo.com>;tag=123456
Call-ID: 9k9FpLxk3uxtm8tn@a2.foo.com
CSeq: 1345 NOTIFY
Event: dialog
Subscription-State: active; expires=2999
Content-Type:application/dialog-info+xml
Content-Length:xxx

<?xml version="1.0" encoding="UTF-8"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
  xmlns:sa="urn:ietf:params:xml:ns:sa-dialog-info" version="2"
  state="partial" entity="sip:shared-a@as.foo.com">
  <dialog id="bG9jYWxIb3N0MzIwOjA=" direction="initiator"
    call-id="5j9FpLxk3uxtm8tn@a1.foo.com" local-tag="5fxced76sl"
    remote-tag="4323z39">
```

```
    <state>confirmed</state>
    <local>
      <identity display="Shared-A">sip:shared-a@as.foo.com
      </identity>
      <target uri="sip:shared-a1@a1.foo.com"></target>
    </local>
    <remote>
      <identity display="B">sip:B@as.foo.com
      </identity>
    </remote>
    <sa:appearance>1</sa:appearance>
  </dialog>
</dialog-info>
```

***[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-a1@as.foo.com>;tag=5fxced76sl
To: "B" <sip:B@as.foo.com>;tag=4323z39
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 1093 ACK
Contact: <sip:shared-a1@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
```

***[F16] 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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 1344 NOTIFY
Content-Length: 0
```

***[F17] 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-a2@as.foo.com>;tag=ggsa3dszlfl
To: "Shared-A" <sip:shared-a2@as.foo.com>;tag=123456
Call-ID: 9k9FpLxk3uxtm8tn@a2.foo.com
CSeq: 1345 NOTIFY
Content-Length: 0
```

### 7.3.3 A-1 Places B on Hold

*Figure 30* shows the call flow as a result of A-1 placing B on hold.

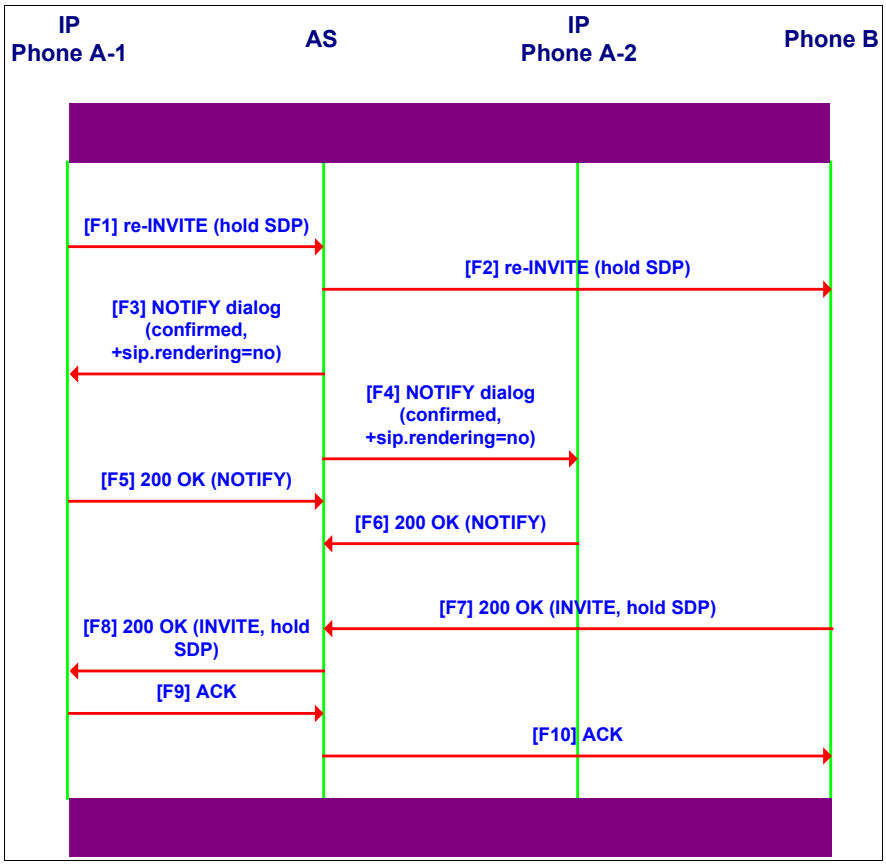


Figure 30 A-1 Places B on Hold

[F1 to F10] shows the normal call flow as a result of A-1 placing B on hold. A-1 changes the call appearance interface to reflect the *held* state. The Application Server sends a NOTIFY to A-1 and A-2 [F3 to F6] to indicate that the call has been "Held" by A. In the message examples that follow, only the NOTIFY is shown to illustrate how held calls are reporting in dialog notifications.

```
[F3] NOTIFY Application Server → A-1
NOTIFY sip:shared-a1@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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 1346 NOTIFY
Event: dialog
Subscription-State: active; expires=3000
Content-Type:application/dialog-info+xml
Content-Length:xxx

<?xml version="1.0" encoding="UTF-8"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
  xmlns:sa="urn:ietf:params:xml:ns:sa-dialog-info" version="3"
  state="partial" entity="sip:shared-a@as.foo.com">
```

```
  <dialog id="bG9jYWxIb3N0MzIwOjA=" direction="initiator"
    call-id="5j9FpLxk3uxtm8tn@a1.foo.com" local-tag="5fxced76sl"
    remote-tag="4323z39">
    <state>confirmed</state>
    <local>
      <identity display="Shared-A">sip:shared-a@as.foo.com
      </identity>
      <target uri="sip:shared-a1@a1.foo.com">
        <param pname="+sip.rendering" pval="no"/>
      </target>
    </local>
    <remote>
      <identity display="B">sip:B@as.foo.com
      </identity>
    </remote>
    <sa:appearance>1</sa:appearance>
  </dialog>
</dialog-info>
```

***[F4] NOTIFY Application Server -> A-2***

```
NOTIFY sip:shared-a2@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-a2@as.foo.com>;tag=ggsa3dszlfl
To: "Shared-A" <sip:shared-a2@as.foo.com>;tag=123456
Call-ID: 9k9FpLxk3uxtm8tn@a2.foo.com
CSeq: 1347 NOTIFY
Event: dialog
Subscription-State: active; expires=2999
Content-Type:application/dialog-info+xml
Content-Length:xxx

<?xml version="1.0" encoding="UTF-8"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
  xmlns:sa="urn:ietf:params:xml:ns:sa-dialog-info" version="3"
  state="partial" entity="sip:shared-a@as.foo.com">
  <dialog id="bG9jYWxIb3N0MzIwOjA=" direction="initiator"
    call-id="5j9FpLxk3uxtm8tn@a1.foo.com" local-tag="5fxced76sl"
    remote-tag="4323z39">
    <state>confirmed</state>
    <local>
      <identity display="Shared-A">sip:shared-a@as.foo.com
      </identity>
      <target uri="sip:shared-a1@a1.foo.com">
        <param pname="+sip.rendering" pval="no"/>
      </target>
    </local>
    <remote>
      <identity display="B">sip:B@as.foo.com
      </identity>
    </remote>
    <sa:appearance>1</sa:appearance>
  </dialog>
</dialog-info>
```

***[F5] 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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 1346 NOTIFY
```

```
Content-Length: 0

[F6] 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-a2@as.foo.com>;tag=ggsa3dszlfl
To: "Shared-A" <sip:shared-a2@as.foo.com>;tag=123456
Call-ID: 9k9FpLxk3uxtm8tn@a2.foo.com
CSeq: 1347 NOTIFY
Content-Length: 0
```

### 7.3.4 A-1 Retrieves Call from Hold

*Figure 31* shows the call flow as a result of A-1 retrieving B from hold.

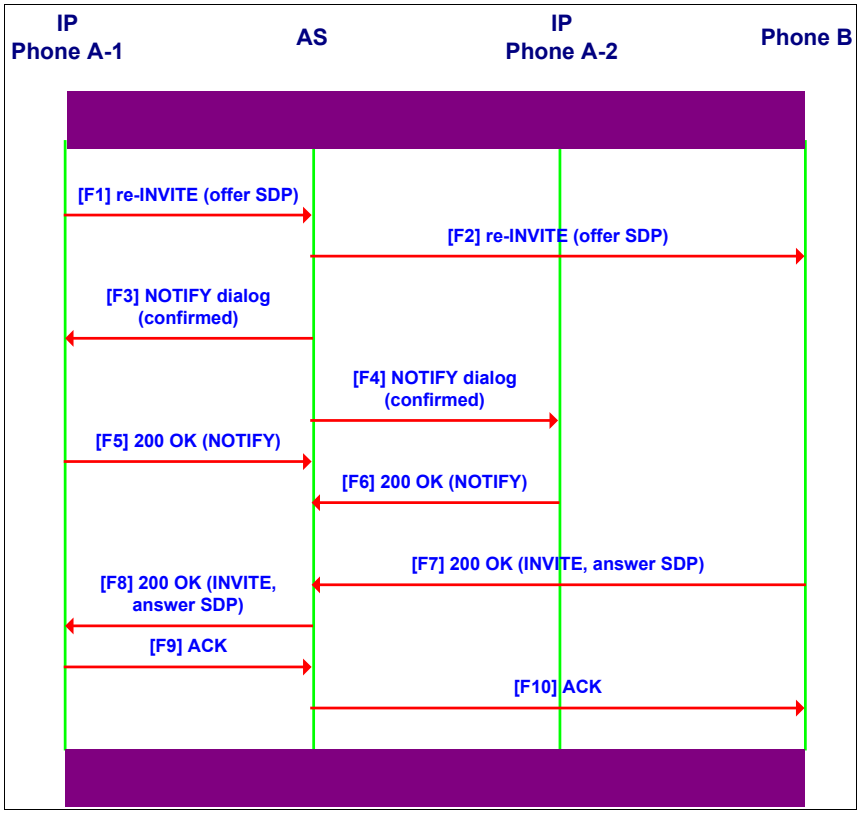


Figure 31 A-1 Retrieves B from Hold

[F1 to F0] shows the usual call flow as a result of A-1 retrieving B from hold. A-1 changes the call appearance interface to reflect the *Active* state. The Application Server sends a NOTIFY to A-1 and A-2 [F3- F6] to indicate that the call is now *Active*.

Following are the NOTIFY transactions [F3-F6].

```
[F3] NOTIFY Application Server → A-1
NOTIFY sip:shared-a1@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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
```

```
CSeq: 1348 NOTIFY
Event: dialog
Subscription-State: active; expires=3000
Content-Type:application/dialog-info+xml
Content-Length:xxx

<?xml version="1.0" encoding="UTF-8"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
  xmlns:sa="urn:ietf:params:xml:ns:sa-dialog-info" version="4"
  state="partial" entity="sip:shared-a@as.foo.com">
  <dialog id="bG9jYWxIb3N0MzIwOjA=" direction="initiator"
    call-id="5j9FpLxk3uxtm8tn@a1.foo.com" local-tag="5fxced76sl"
    remote-tag="4323z39">
    <state>confirmed</state>
    <local>
      <identity display="Shared-A">sip:shared-a@as.foo.com
      </identity>
      <target uri="sip:shared-a1@a1.foo.com"></target>
    </local>
    <remote>
      <identity display="B">sip:B@as.foo.com
      </identity>
    </remote>
    <sa:appearance>1</sa:appearance>
  </dialog>
</dialog-info>
```

***[F4] NOTIFY Application Server -> A-2***

```
NOTIFY sip:shared-a2@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-a2@as.foo.com>;tag=ggsa3dszlfl
To: "Shared-A" <sip:shared-a2@as.foo.com>;tag=123456
Call-ID: 9k9FpLxk3uxtm8tn@a2.foo.com
CSeq: 1349 NOTIFY
Event: dialog
Subscription-State: active; expires=2999
Content-Type:application/dialog-info+xml
Content-Length:xxx

<?xml version="1.0" encoding="UTF-8"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
  xmlns:sa="urn:ietf:params:xml:ns:sa-dialog-info" version="4"
  state="partial" entity="sip:shared-a@as.foo.com">
  <dialog id="bG9jYWxIb3N0MzIwOjA=" direction="initiator"
    call-id="5j9FpLxk3uxtm8tn@a1.foo.com" local-tag="5fxced76sl"
    remote-tag="4323z39">
    <state>confirmed</state>
    <local>
      <identity display="Shared-A">sip:shared-a@as.foo.com
      </identity>
      <target uri="sip:shared-a1@a1.foo.com"></target>
    </local>
    <remote>
      <identity display="B">sip:B@as.foo.com
      </identity>
    </remote>
    <sa:appearance>1</sa:appearance>
  </dialog>
</dialog-info>
```

***[F5] 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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 1348 NOTIFY
Content-Length: 0

[F6] 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-a2@as.foo.com>;tag=ggsa3dszlfl
To: "Shared-A" <sip:shared-a2@as.foo.com>;tag=123456
Call-ID: 9k9FpLxk3uxtm8tn@a2.foo.com
CSeq: 1349 NOTIFY
Content-Length: 0
```

### 7.3.5 A-1 Releases Call

*Figure 32* shows the call flow as a result of A-1 releasing the call with B.

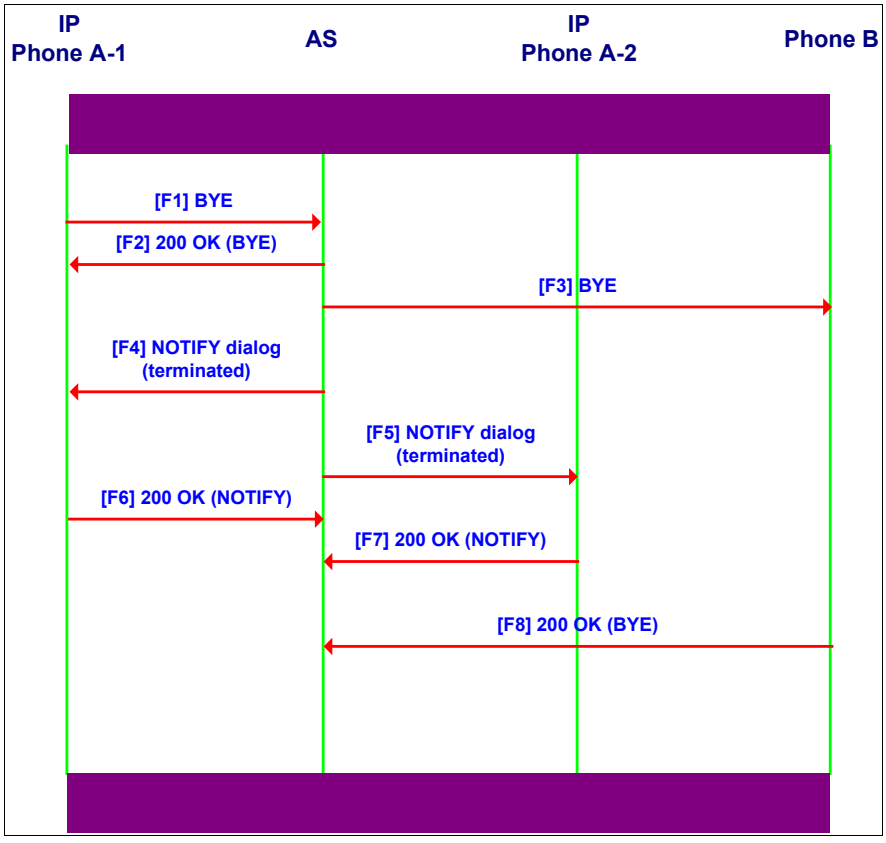


Figure 32 A-1 Releases Call

The user at A-1 hangs up and the phone sends a BYE to the Application Server [F1]. The Application Server sends a corresponding BYE to B [F3]. The call is effectively released. The Application Server sends a NOTIFY to A-1 and A-2 [F4-F7] indicating the terminated SIP dialog.

Following are the NOTIFY requests [F4-F7].

***[F4] NOTIFY Application Server → A-1***

```
NOTIFY sip:shared-a1@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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 1350 NOTIFY
Event: dialog
Subscription-State: active; expires=3000
Content-Type:application/dialog-info+xml
Content-Length:xxx

<?xml version="1.0" encoding="UTF-8"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
  xmlns:sa="urn:ietf:params:xml:ns:sa-dialog-info" version="5"
  state="partial" entity="sip:shared-a@as.foo.com">
  <dialog id="bG9jYWxIb3N0MzIwOjA=">
    <state>terminated</state>
    <local>
      <identity display="Shared-A">sip:shared-a@as.foo.com
      </identity>
    </local>
  </dialog>
</dialog-info>
```

***[F5] NOTIFY Application Server -> A-2***

```
NOTIFY sip:shared-a2@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-a2@as.foo.com>;tag=ggsa3dszlfl
To: "Shared-A" <sip:shared-a2@as.foo.com>;tag=123456
Call-ID: 9k9FpLxk3uxtm8tn@a2.foo.com
CSeq: 1351 NOTIFY
Event: dialog
Subscription-State: active; expires=2999
Content-Type:application/dialog-info+xml
Content-Length:xxx

<?xml version="1.0" encoding="UTF-8"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
  xmlns:sa="urn:ietf:params:xml:ns:sa-dialog-info" version="5"
  state="partial" entity="sip:shared-a@as.foo.com">
  <dialog id="bG9jYWxIb3N0MzIwOjA=">
    <state>terminated</state>
    <local>
      <identity display="Shared-A">sip:shared-a@as.foo.com
      </identity>
    </local>
  </dialog>
</dialog-info>
```

***[F6] 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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 1350 NOTIFY
Content-Length: 0
```

***[F7] 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-a2@as.foo.com>;tag=ggsa3dszlfl
To: "Shared-A" <sip:shared-a2@as.foo.com>;tag=123456
Call-ID: 9k9FpLxk3uxtm8tn@a2.foo.com
CSeq: 1351 NOTIFY
Content-Length: 0
```

### 7.3.6 B Calls A

*Figure 33* shows the call flow as a result of B calling A. Note that B cannot explicitly call A-1 or A-2. Instead, it can only address the number of the shared line for A. The Application Server then deals with delivering the call simultaneously to the endpoints at A-1 and A-2.

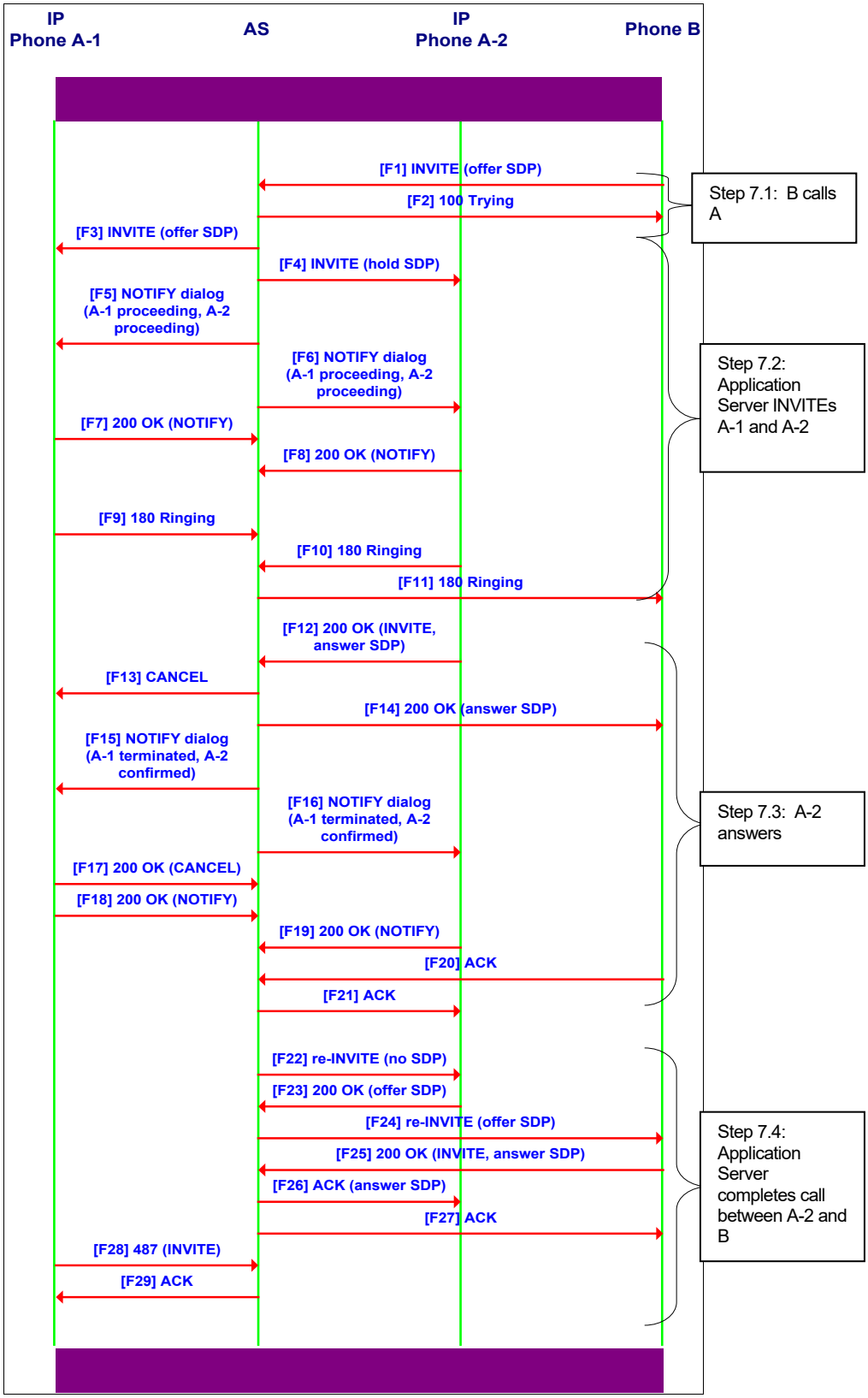


Figure 33 B Calls A

#### 7.3.6.1 Step 7.1: B Calls A

B delivers an INVITE to the Application Server [F1] addressing the logical user A.

#### 7.3.6.2 Step 7.2: Application Server Sends Separate INVITE to both A-1 and A-2 and Notifies A-1 and A-2 about Proceeding Dialogs

Because A actually represents a Shared Call Appearance, the Application Server sends a separate INVITE [F3-F4] to each registered endpoint. Note that the "primary" endpoint receives the original SDP offer from B. The other endpoints (called alternate endpoints) receive a hold SDP. At the same time, the Application Server notifies A-1 and A-2 of the proceeding SIP dialogs [F5-F6].

Both A-1 and A-2 answer with a *180*; a single *180* provisional response is then sent back to B.

The following shows the NOTIFY message sent to A-1 [F5].

```
[F5] NOTIFY Application Server → A-1
NOTIFY sip:shared-a1@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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 1352 NOTIFY
Event: dialog
Subscription-State: active; expires=3000
Content-Type:application/dialog-info+xml
Content-Length:xxx

<?xml version="1.0" encoding="UTF-8"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
  xmlns:sa="urn:ietf:params:xml:ns:sa-dialog-info" version="6"
  state="partial" entity="sip:shared-a@as.foo.com">
  <dialog id="bG9jYWxIb3N0Mzc6MQ==" direction="recipient">
    <state>proceeding</state>
    <local>
      <identity display="Shared-A">sip:shared-a@as.foo.com
      </identity>
    </local>
    <remote>
      <identity display="B">
        sip:B@as.foo.com;user=phone</identity>
    </remote>
    <sa:appearance>1</sa:appearance>
  </dialog>
  <dialog id="bG9jYWxIb3N0Mzc6MA==" direction="recipient">
    <state>proceeding</state>
    <local>
      <identity display="Shared-A">sip:shared-a@as.foo.com
      </identity>
    </local>
    <remote>
      <identity display="B">
        sip:B@as.foo.com;user=phone</identity>
    </remote>
    <sa:appearance>1</sa:appearance>
  </dialog>
</dialog-info>
```

#### 7.3.6.3 Step 7.3: A-2 Answers First, Application Server Cancels INVITE to A-1

A-2 (an alternate endpoint) answers the incoming call first [F12]. The Application Server then sends a CANCEL to A-1 (so it stops alerting) [F13].

The Application Server sends NOTIFY requests to A-1 and A-2, reporting the confirmed dialog for A-2 and the terminated dialog for A-1 [F15-F16].

The following shows the NOTIFY request sent to A-1 [F15].

```
[F5] NOTIFY Application Server → A-1
NOTIFY sip:shared-a1@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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 1354 NOTIFY
Event: dialog
Subscription-State: active; expires=3000
Content-Type:application/dialog-info+xml
Content-Length:xxx

<?xml version="1.0" encoding="UTF-8"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
  xmlns:sa="urn:ietf:params:xml:ns:sa-dialog-info" version="6"
  state="partial" entity="sip:shared-a@as.foo.com">
  <dialog id="bG9jYWxIb3N0Mzc6MQ==">
    <state>terminated</state>
    <local>
      <identity display="Shared-A">sip:shared-a@as.foo.com
      </identity>
    </local>
  </dialog>
  <dialog id="bG9jYWxIb3N0Mzc6MA==" direction="recipient"
    call-id="5j9FpLxk3uxtm8tn@a2.foo.com" local-tag="5fxced76sl"
    remote-tag="4323z39">
    <state>confirmed</state>
    <local>
      <identity display="Shared-A">sip:shared-a@as.foo.com
      </identity>
      <target uri="sip:shared-a2@a2.foo.com"></target>
    </local>
    <remote>
      <identity display="B">sip:B@as.foo.com
      </identity>
    </remote>
    <sa:appearance>1</sa:appearance>
  </dialog>
</dialog-info>
```

#### 7.3.6.4 Step 7.4: Application Server Completes Call between A-2 and B

In [F22 to F29] the Application Server performs some offer/answer processing to make sure the SDP between A-2 and B is an active one. The INVITE transaction to A-1 is also completed.

### 7.3.7 A-2 Retrieves B from Hold

*Figure 34* provides the same call flow as described in section *7.3.3 A-1 Places B on Hold*. At the end of this scenario, A-1 has placed B on hold. For information on this call flow, see section *7.3.3 A-1 Places B on Hold*.

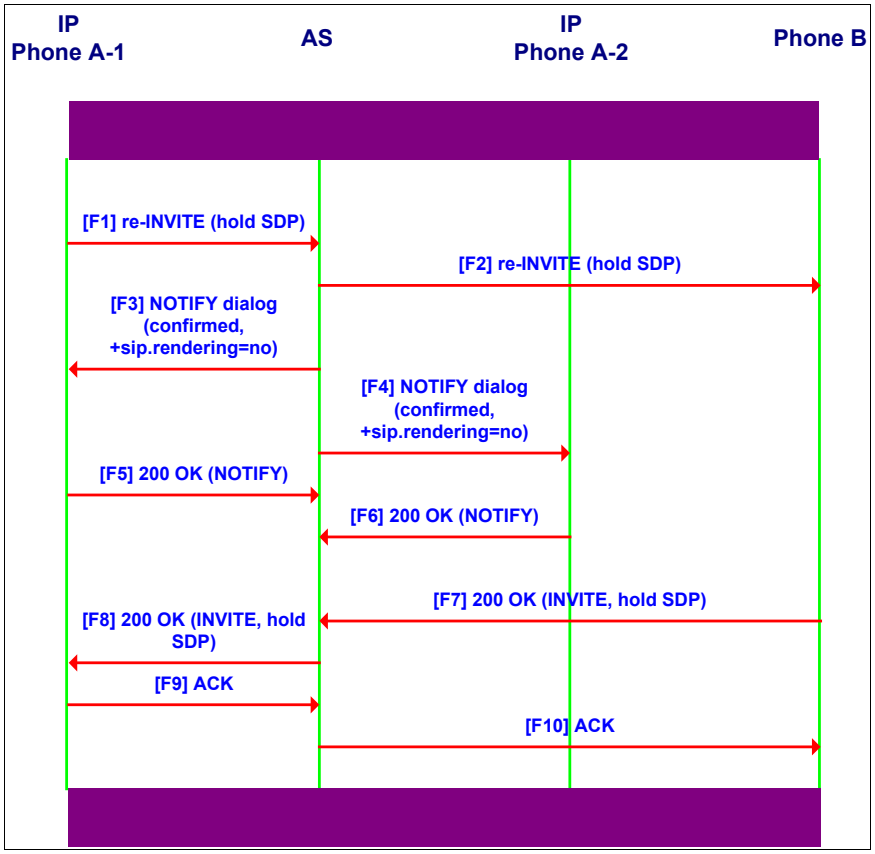


Figure 34 A-1 Places B on Hold

Now the user at IP phone A-2 can see that there is a call held remotely on the shared line. The user at IP phone A-2 wants to retrieve the held call and pushes the associated button (line key, and so on).

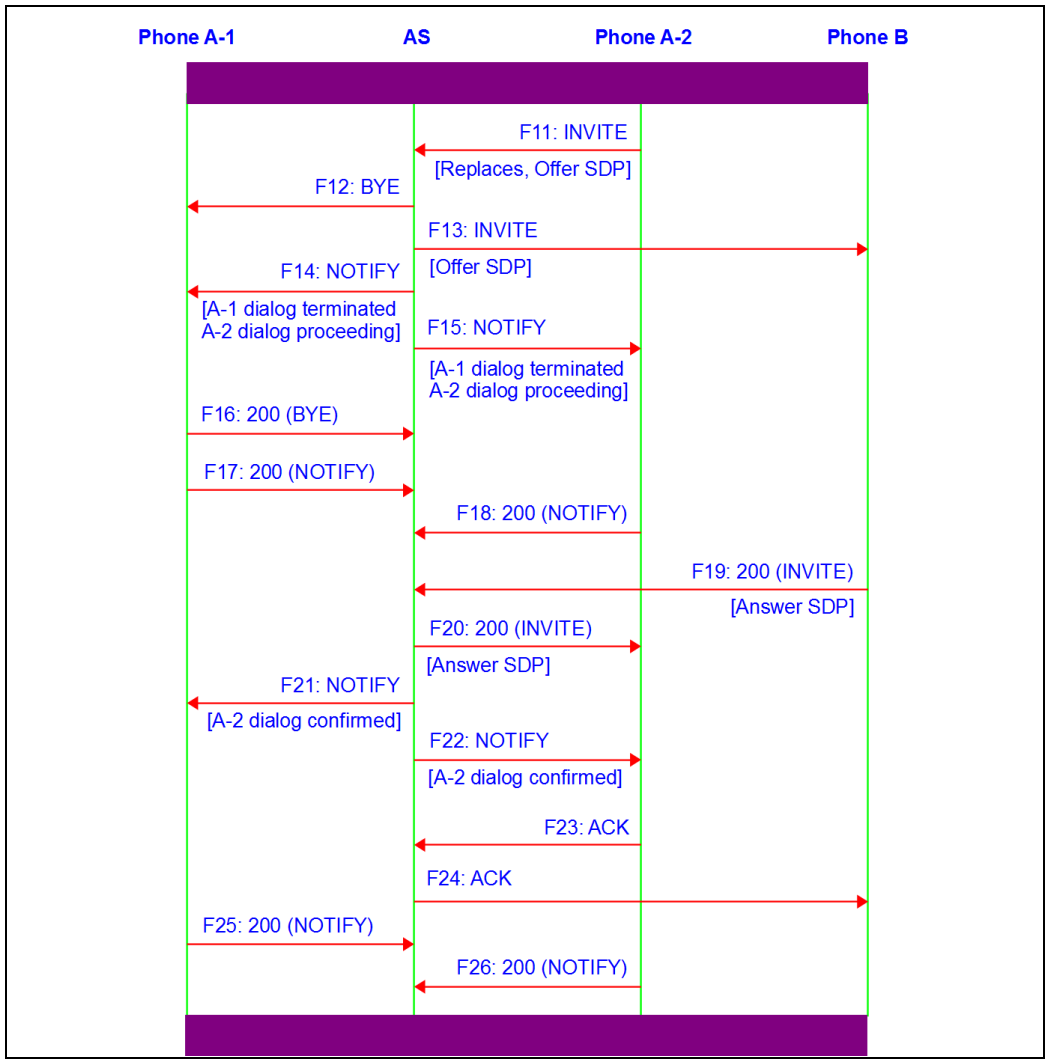


Figure 35 User at IP Phone A-2 Retrieves Held Call

In [F11] the IP phone A-2 initiates an INVITE to retrieve the held call, as follows:

```
[F11] INVITE A-2 → Application Server
INVITE sip:as.foo.com SIP/2.0
Via: SIP/2.0/UDP a2.foo.com:5060;branch=m9hG4bK74bf9
Max-Forwards: 70
From: "Shared-A" <sip:shared-a2@as.foo.com>;tag=5fxced76s3
To: "Shared-A" <sip:shared-a2@as.foo.com>
Call-ID: 98482762982201885l3@a2.foo.com
Replaces: 5j9FpLxk3uxtm8tn@a1.foo.com
  ;from-tag=323423233
  ;to-tag=dsa3dszlfl
CSeq: 43294 INVITE
Contact: <sip:shared-a2@a2.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
```

The Application Server looks at the *Request-URI*, the *From* and *To* headers, and the *Replaces* header to determine that A-2 is attempting to retrieve the call held by A-1. The Application Server applies local policy and determines that A-2 is authorized to retrieve the call. It proceeds by releasing the call to A-1 [F12] and sends a re-INVITE B with the SDP offered by A-2 [F13].

The Application Server sends NOTIFY requests to A-1 and A-2 to inform them about the terminated SIP dialog to A-1 and about the new partial SIP dialog to A-2.

The NOTIFY request [F14] is as follows. ([F15] has the same body except possibly for the version number, which is maintained independently for each endpoint subscription.)

```
[F14] NOTIFY Application Server → A-1
NOTIFY sip:shared-a1@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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 1348 NOTIFY
Event: dialog
Subscription-State: active; expires=3000
Content-Type:application/dialog-info+xml
Content-Length:xxx

<?xml version="1.0" encoding="UTF-8"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
  xmlns:sa="urn:ietf:params:xml:ns:sa-dialog-info" version="3"
  state="partial" entity="sip:shared-a@as.foo.com">
  <dialog id="bG9jYWxIb3N0MzIwOjA=">
    <state>terminated</state>
    <local>
      <identity display="Shared-A">sip:shared-a@as.foo.com</identity>
    </local>
  </dialog>
  <dialog id="bG9jYWxIb3N0MzIwOjB=" direction="initiator">
    <state>proceeding</state>
    <local>
      <identity display="Shared-A">sip:shared-a@as.foo.com</identity>
      <target uri="sip:shared-a2@a2.foo.com"></target>
    </local>
    <remote>
      <identity display="B">sip:B@as.foo.com</identity>
    </remote>
    <sa:appearance>1</sa:appearance>
  </dialog>
</dialog-info>
```

After the Application Server establishes the dialog with IP phone A-2, it sends NOTIFY requests ([F21] and [F22]) to A-1 and A-2 to inform them about the confirmed SIP dialog with A-2.

The NOTIFY request [F21] is as follows. ([F22] has the same body except possibly for the version number, which is maintained independently for each endpoint subscription.)

***[F21] NOTIFY Application Server → A-1***

```
NOTIFY sip:shared-a1@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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 1349 NOTIFY
Event: dialog
Subscription-State: active; expires=3000
Content-Type:application/dialog-info+xml
Content-Length:xxx

<?xml version="1.0" encoding="UTF-8"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
  xmlns:sa="urn:ietf:params:xml:ns:sa-dialog-info" version="4"
  state="partial" entity="sip:shared-a@as.foo.com">
  <dialog id="bG9jYWxIb3N0MzIwOjB=" direction="initiator"
    call-id="9848276298220188513@a2.foo.com" local-tag="5fxced76s3"
    remote-tag="4323z39">
    <state>confirmed</state>
    <local>
      <identity display="Shared-A">sip:shared-a@as.foo.com</identity>
      <target uri="sip:shared-a2@a2.foo.com"></target>
    </local>
    <remote>
      <identity display="B">sip:B@as.foo.com</identity>
    </remote>
    <sa:appearance>1</sa:appearance>
  </dialog>
</dialog-info>
```

### 7.3.8 A-2 Barges into Call between A-1 and B

*Figure 36* provides the same call flow as described in section *7.3.2 A-1 Calls B*. At the end of this scenario, A-1 has an active call with B.

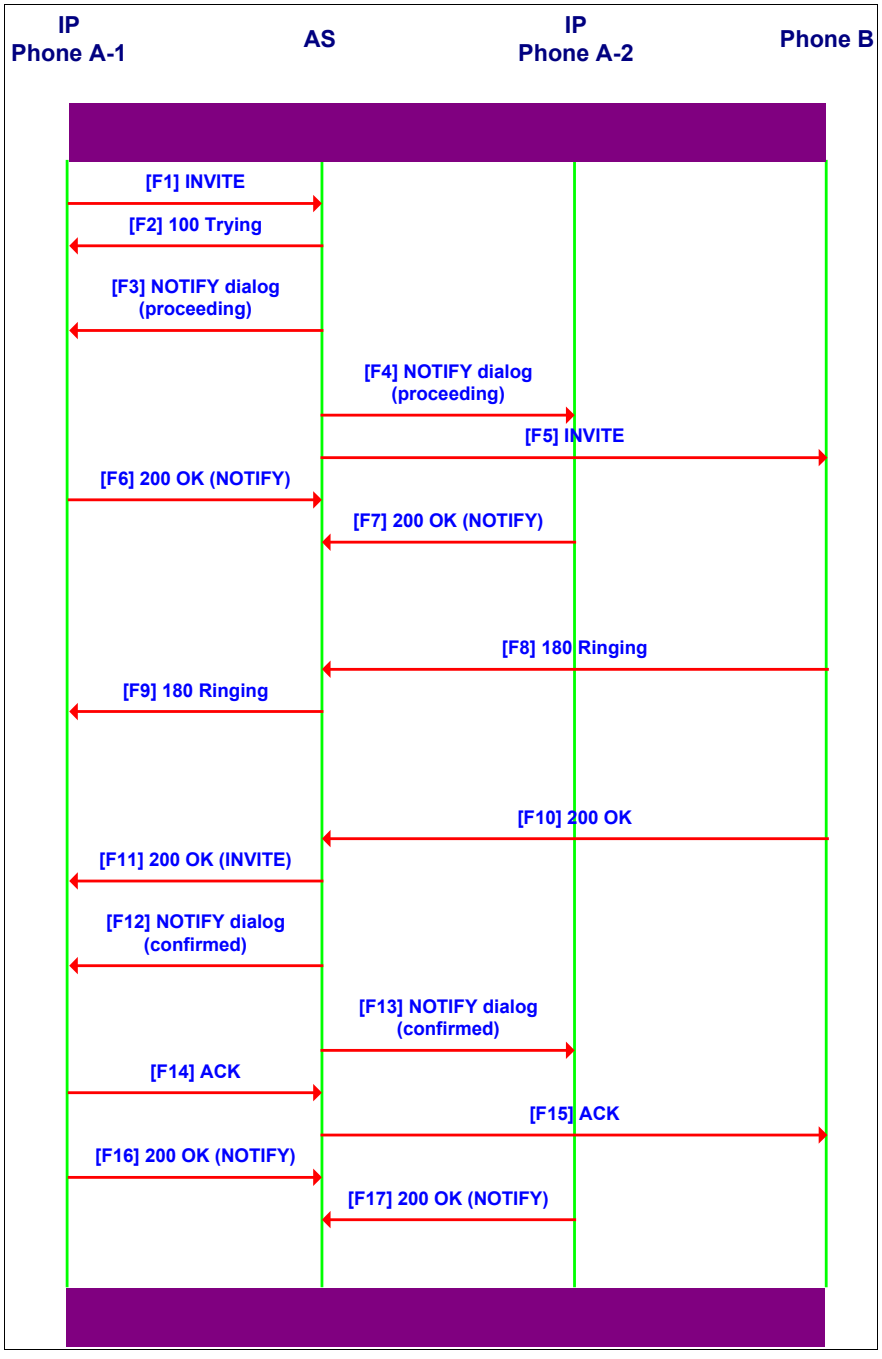


Figure 36 A-1 Calls B

Now the user at IP phone A-2 can see that there is an active call on the shared line. The user at IP phone A-2 wants to barge in to the active call, and pushes the associated button (line key, and so on).

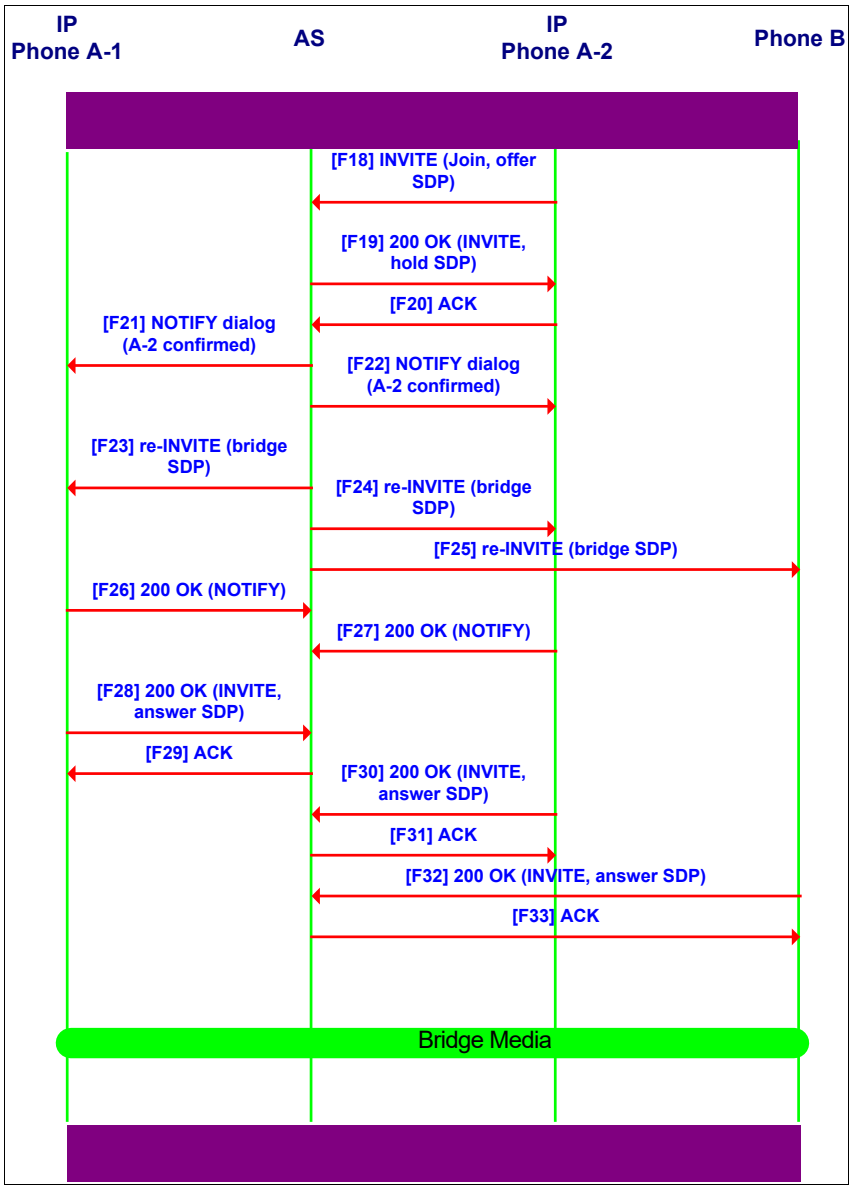


Figure 37 User at IP Phone A-2 Barges into Active Call

In [F18] the IP phone A-2 initiates an INVITE to barge in to the active call. The INVITE contains the *Join* header populated with A-1 SIP dialog information, as follows.

```
[F20] INVITE A-2 → Application Server
INVITE sip:as.foo.com SIP/2.0
Via: SIP/2.0/UDP a2.foo.com:5060;branch=m9hG4bK74bf9
Max-Forwards: 70
From: "Shared-A" <sip:shared-a2@as.foo.com>;tag=5fxced76s3
To: "Shared-A" <sip:shared-a2@as.foo.com>
Call-ID: 9848276298220188513@a2.foo.com
Join: 5j9FpLxk3uxtm8tn@a1.foo.com
  ;from-tag=323423233
```

```
  ;to-tag=dsa3dszlfl
CSeq: 43294 INVITE
Contact: <sip:shared-a2@a2.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
```

The Application Server looks at the *Request-URI*, the *From* and *To* headers, and the *Join* header to determine that A-2 is attempting to barge in to the active call on A-1. The Application Server applies local policy and determines that A-2 is authorized to barge in to the call. It acknowledges the INVITE request with a 200 response with a held SDP, and proceeds with the creation of a conference bridge. A-1, A-2, and B are then individually re-invited to the bridge [F23-F25]. A NOTIFY for the dialog event package containing the new confirmed SIP dialog between A-2 and the Application Server is also broadcast to all the endpoints in the shared group [F21-F22].

The NOTIFY request [F21] is as follows. ([F22] has the same body except for the version number, which is specified for each endpoint subscription.)

```
[F23] NOTIFY Application Server → A-1
NOTIFY sip:shared-a1@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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 1348 NOTIFY
Event: dialog
Subscription-State: active; expires=3000
Content-Type:application/dialog-info+xml
Content-Length:xxx

<?xml version="1.0" encoding="UTF-8"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
  xmlns:sa="urn:ietf:params:xml:ns:sa-dialog-info" version="3"
  state="partial" entity="sip:shared-a@as.foo.com">
  <dialog id="bG9jYWxIb3N0MzIwOjB=" direction="initiator"
    call-id="98482762982201885l3@a2.foo.com" local-tag="5fxced76s3"
    remote-tag="4323z39">
    <state>confirmed</state>
    <local>
      <identity display="Shared-A">sip:shared-a@as.foo.com
      </identity>
      <target uri="sip:shared-a2@a2.foo.com">
      </target>
    </local>
    <remote>
      <identity display="B">sip:B@as.foo.com
      </identity>
    </remote>
    <sa:appearance>1</sa:appearance>
  </dialog>
</dialog-info>
```

Note that the difference with the NOTIFY sent for call retrieval (section *7.3.7 A-2 Retrieves B from Hold*) is that the dialog for A-1 is not reported as terminated. Since this is a partial notification, this means two SIP dialogs are confirmed against the appearance 1, indicating that a bridge is set up.

#### 7.3.8.1 Silent Monitoring

Shared Call Appearance also supports the ability to invoke Silent Monitor Bridging. Silent Monitor Bridging provides the ability for the user to bridge in to a call in silent monitoring mode where the monitoring (invoking) user can hear the other parties, but the other parties cannot hear the monitoring user.

Silent Monitor Bridging is invoked as an option on the above SIP INVITE with Join [F20] where the *silent-monitor* parameter in added to the header.

The following is an example of the *Join* header with the *silent-monitor* parameter.

```
[F20] INVITE A-2 → Application Server
INVITE sip:as.foo.com SIP/2.0
Via: SIP/2.0/UDP a2.foo.com:5060;branch=m9hG4bK74bf9
Max-Forwards: 70
From: "Shared-A" <sip:shared-a2@as.foo.com>;tag=5fxced76s3
To: "Shared-A" <sip:shared-a2@as.foo.com>
Call-ID: 98482762982201885l3@a2.foo.com
Join: 5j9FpLxk3uxtm8tn@a1.foo.com
  ;from-tag=323423233
  ;to-tag=dsa3dszlfl;silent-monitor
CSeq: 43294 INVITE
Contact: <sip:shared-a2@a2.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
```

This behavior is controlled by the *enableSilentMonitoring* system parameter from *AS_CLI/Applications/ExecutionAndProvisioning/LawfulIntercept*. When this attribute is set to "false" the user can still trigger bridging but they are not muted.

### 7.3.9 Call Park

Call Park is a service provided by Cisco BroadWorks to keep calls on hold and be able to retrieve them at a later time, potentially from another user. A call is parked on a specific target user and can then be retrieved by specifying the extension of that user.

A parameter in the Shared Call Appearance service allows enabling the transmission of parked call information via the existing *Call-Info* subscription dialog. If the parameter is enabled the following behavior to applies (otherwise the regular behavior applies, as described in the previous sections).

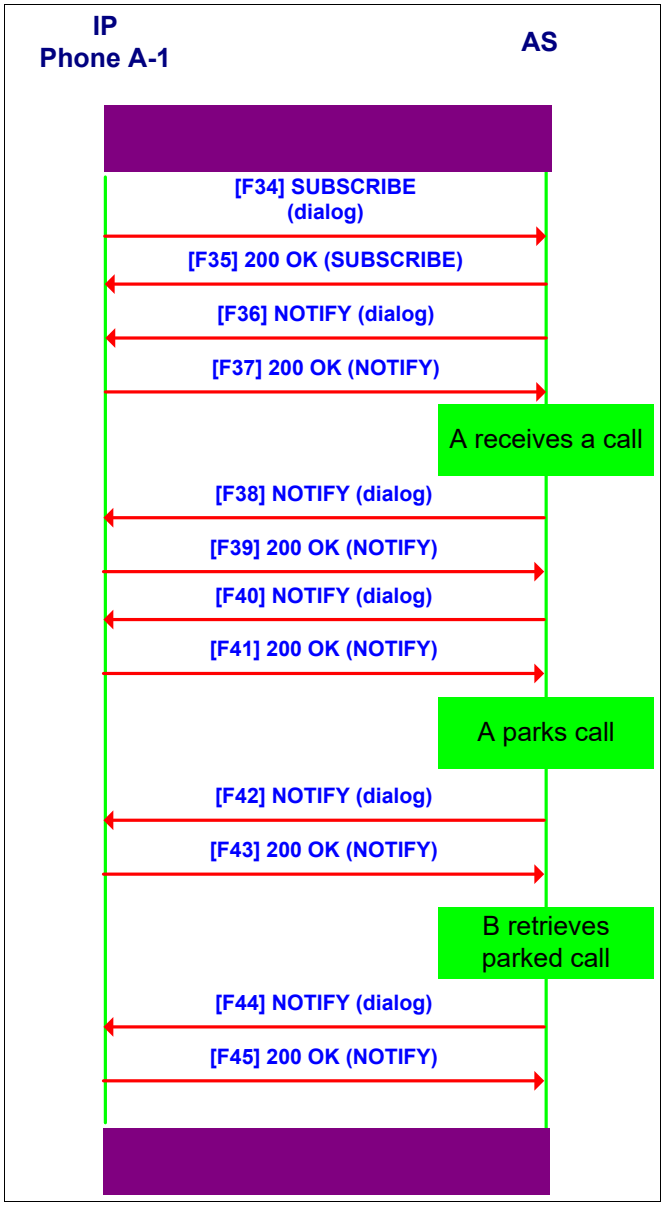


Figure 38 User at IP Phone A-1 has Call Park Notification Enabled

### Initial Subscription

Messages [F34] to [F37] are exactly the same as the regular subscription exchange in dialog mode. The callpark information does not apply at this time.

### User A receives regular call

Messages [F38] to [F41] are exactly the same as the regular subscription exchange in dialog mode. The callpark information does not apply at this time.

### Call is parked on User A

***[F42] NOTIFY Application Server -> A-1***

```
NOTIFY sip:shared-a1@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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 1556 NOTIFY
Event: dialog
Subscription-State: active; expires=3599
Contact:  sip:shared-a1@as.foo.com
Content-Type:application/dialog-info+xml
Content-Length:468

<?xml version="1.0" encoding="UTF-8"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
  xmlns:sa="urn:ietf:params:xml:ns:sa-dialog-info"
  xmlns:bw=http://schema.broadsoft.com/callpark
  version="3" state="partial"
  entity="sip:shared-a@as.foo.com">
 <dialog id="Y2FsbGhhbGYtMzM6MA==">
  <state>confirmed</state>
  <bw:callpark>
   <bw:parked>
    <identity display="C. Parked">
     sip:C@as.foo.com;user=phone
    </identity>
   </bw:parked>
  </bw:callpark>
 </dialog>
</dialog-info>
```

***[F43] 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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 1556 NOTIFY
Content-Length: 0
```

**Parked call is removed from User A**

```
[F44] NOTIFY Application Server -> A-1
NOTIFY sip:shared-a1@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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 1691 NOTIFY
Event: dialog
Subscription-State: active; expires=3599
Contact:  sip:shared-a1@as.foo.com
Content-Type:application/dialog-info+xml
Content-Length:468

<?xml version="1.0" encoding="UTF-8"?>
<dialog-info xmlns="urn:ietf:params:xml:ns:dialog-info"
  xmlns:sa="urn:ietf:params:xml:ns:sa-dialog-info"
  xmlns:bw=http://schema.broadsoft.com/callpark
  version="3" state="partial"
  entity="sip:shared-a@as.foo.com">
 <dialog id="Y2FsbGhhbGYtMzM6MA==">
  <state>terminated</state>
  <bw:callpark/>
</dialog>
</dialog-info>

[F45] 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-a1@as.foo.com>;tag=dsa3dszlfl
To: "Shared-A" <sip:shared-a1@as.foo.com>;tag=323423233
Call-ID: 5j9FpLxk3uxtm8tn@a1.foo.com
CSeq: 1691 NOTIFY
Content-Length: 0
```

## Acronyms and Abbreviations

| | |
|---|---|
| ACL | Access Control List |
| AS | Application Server |
| CLI | Command Line Interface |
| FTP | File Transfer Protocol |
| HTTPS | Hypertext Transfer Protocol Secure Sockets |
| IP | Internet Protocol |
| LCD | Liquid Crystal Display |
| LED | Light Emitting Diode |
| MD5 | Message Digest 5 algorithm |
| MGCP | Media Gateway Control Protocol |
| PBX | Private Branch Exchange |
| RFC | Request for Comments |
| SCA | Shared Call Appearance |
| SCCP | Simple Conference Control Protocol |
| SDP | Session Description Protocol |
| SIP | Session Initiation Protocol |
| TFTP | Trivial File Transfer Protocol |
| URI | Uniform Resource Identifier |
| VoIP | Voice over IP |

## References


[1] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J, Sparks R., Handley, M., and Schooler, E., "SIP: Session Initiation Protocol", RFC 3261, Internet Engineering Task Force, June 2002. Available from http://www.ietf.org/.

[2] Rosenberg, J., Peterson, J., Schulzrinne, H., Camarillo, G., "Best Current Practices for Third Party Call Control (3pcc) in the Session Initiation Protocol (SIP)", RFC 3725, Internet Engineering Task Force, April 2004. Available from http://www.ietf.org/.

[3] Roach, A., "Session Initiation Protocol (SIP)-Specific Event Notification", RFC 3265, Internet Engineering Task Force, June 2002. Available from http://www.ietf.org/.

[4] Rosenberg, J., Schulzrinne, H., "An Offer/Answer Model with the Session Description Protocol (SDP)", RFC 3264, Internet Engineering Task Force, June 2002. Available from http://www.ietf.org/.

[5] Rosenberg, J., Schulzrinne, H., Mahy, R., "An INVITE Initiated Dialog Event Package for the Session Initiation Protocol (SIP)", RFC 4235, Internet Engineering Task Force, November 2005. Available from http://www.ietf.org/.

[6] Mahy, R., Petrie, D., "The Session Initiation Protocol (SIP) "Join" Header", RFC 3911, Internet Engineering Task Force, October 2004. Available from http://www.ietf.org/.

[7] Mahy, R., Biggs, B., Dean, R., "The Session Initiation Protocol (SIP) "Replaces" Header", RFC 3891, September 2004. Available from http://www.ietf.org/.

[8] Johnston, A. Ed., Soroushnejad, M., Venkataramanan, V., "Shared Appearances of a Session Initiation Protocol (SIP) Address of Record (AOR)", draft-ietf-bliss-shared-appearances-02, Internet Engineering Task Force, March 9, 2009. Available from http://www.ietf.org/.

[9] BroadSoft Inc. 2016. *Cisco BroadWorks SIP Access Interface Interworking Guide Release 23.0.* Available from BroadSoft at xchange.broadsoft.com.

[10] BroadSoft Inc. 2016. *Cisco BroadWorks Busy Lamp Field Interface Specification, Release 23.0*. Available from BroadSoft at xchange.broadsoft.com.

[11] BroadSoft Inc. 2016. *Cisco BroadWorks SIP Access Side Extensions Interface Specification Release 23.0*. Available from BroadSoft at xchange.broadsoft.com.

# Index